UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER EN INVESTIGACIÓN EN INTELIGENCIA ARTIFICIAL

Trabajo de fin de máster

El proceso de separación del empleado en una organización: aplicación de diagramas de influencia y cadenas de Markov en la toma de decisiones de recursos humanos

Sergio Montes Vázquez

Dirigido por: Prof. Dr. Francisco Javier Díez Vegas

Curso: 2019-2020: 2ª Convocatoria

Agradecimientos

Primeramente, quiero agradecer a mi tutor Francisco Javier Díez Vegas; gracias él, a toda su experiencia, conocimiento y paciencia he logrado concluir este trabajo. No tengo palabras para poderle dar las gracias por todo el apoyo y conocimiento que me ha aportado. En segundo lugar, quiero expresar toda mi gratitud a Maricarmen Subirachs Cortés, la persona que me ha ayudado desde México, con su gran conocimiento, a entender los procesos de recursos humanos, área donde llevo trabajando más de quince años y aún me siento un novato; muchas gracias, Maricarmen. A la Universidad Nacional de Educación a Distancia (UNED) de España, a todos sus docentes y administrativos, que han permitido que una persona como yo logre concluir sus estudios de máster.

A mis padres Sergio y Carmen, que me han apoyado en esta aventura de venir a España y que desde muy niño me han inculcado el valor del ser humano y la nobleza del trabajo. A mis hermanos José y Laura Estela, que también me han apoyado en todo, me han aconsejado y animado en los momentos más difíciles, a ellos y sus parejas Erika y Salvador, a todos ellos que desde mi país de origen me han dado tanto cariño, gracias.

A mis amigos de México y Europa, en especial a los amigos de Madrid y el País Vasco que he conocido y me han hecho sentir parte de este país que tan bien me ha acogido; a los amigos de Polonia, de Inglaterra, de Francia, de Georgia y de Italia, que a pesar de la distancia siempre han estado a lado y más en estos tiempos tan difíciles. No quiero poner nombres porque tengo miedo de omitir alguno, a todos ellos gracias por acompañarme en este tramo de mi vida.

Finalmente quiero dedicar este trabajo a las heredaras de esta singularidad que estamos construyendo, mis sobrinas Beatriz, Regina y Adara.

De todo corazón, muchas gracias.

Resumen

En toda organización existe un área de recursos humanos encargada, entre otras muchas cosas, de procesar toda la información relativa a los empleados, y con esa información realizar los procesos propios de esta área. Uno de los procesos más incómodos que, tarde o temprano se tiene que realizar, es cuando al empleado se le separa de su puesto, o bien es el propio empleado el que desea abandonarlo su puesto. El problema radica en que no siempre la organización desea que el empleado cause baja; lo quiere retener, pues el costo de esta baja a veces es incalculable, por lo que es necesario crear una herramienta que sirva como un indicador para anticipase a ese momento y si es posible revertirlo. El objetivo de este proyecto es ofrecer una propuesta de cómo construir esta herramienta para la toma de decisiones.

Para crear esa herramienta se hará uso de diagramas de influencias, construyendo ese modelo a partir de la información con la que se gestiona al empleado. Entendiendo que el empleado pasa por diferentes estados en su desempeño y su valoración dentro de la organización hasta llegar a la baja. Estos estados se pueden modelar como un modelo de Markov que tiene que ser parte del diagrama de influencia que ayuda a determinar la situación en la que está el empleado.

Esta herramienta se construye explotando y procesando la base de datos de recursos humanos para generar ficheros XML que puede ser usado por la aplicación OpenMarkov; para ello se hará uso de algoritmos implementados con lenguajes de programación. Cada uno de estos ficheros es una configuración diferente del mismo modelo, por lo cual hay que probar y evaluar hasta obtener el mejor de ellos.

Una vez obtenido el mejor modelo, podemos ver que se puede aplicar para toma de decisiones en lo individual, para cada uno de los empleados, o en lo general, grupos enteros de empleados.

Palabras clave

Recursos humanos, separación del empleado, baja del empleado, abandono del puesto, cadenas de Markov, diagramas de influencia.

Abstract

In every organization there is a human resources area in charge, among many other things, gathering and controlling all the information processes related to employees and, with that information carrying out the processes of this area. One of the most uncomfortable processes that, sooner or later, has to be implemented, is the employee's separation from her or his position; sometimes it is the employee herself or himself who wishes to leave. The problem arises when the organization does not want the employee to leave but, on the contrary, it wants to retain him, because the cost of this leave is sometimes unmeasurable, ergo it is necessary to create a tool that serves as an indicator to anticipate that moment and, if possible, reverse it. The objective of this project is a proposal on how to build this tool for decision making.

To create this tool, use will be made of influence diagrams, building that model using the information with which the employee is managed in the understanding that the employees go through different stages in their performance and their assessment within the organization until the employee's leave arrives. These stages can be modeled as a Markov chain that has to be part of the Influence Diagram that helps determine the situation the employee is in.

This tool is built by exploiting and processing the human resources database to generate XML files that can be used by the Open Markov application. For this, algorithms implemented with programming languages will be used. Each of these files is a different configuration of the same model, so we have to test and evaluate them until getting the best of them.

Once the best model has been obtained, we can see that it can be applied to decision-making individually, for each of the employees, or in general, to entire groups of employees.

Keywords

Employee dismissal, employee leave, Abandonment of the position, Markov chains, Influence diagrams.

Índice general

Nomenclatura

ERP Enterprise Resource Planning, página 7

EVP Employee Value Proposition, página 14

MID Markov Influence Diagrams, página 3

RRHH Recursos Humanos, página 8

SBA Salario Bruto Anual, página 19

SCM Supply Chain Management, página 9

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general y objetivos

"*People don't leave jobs, they leave toxic work cultures.*"

Dr. Amima Aite-Selmi, consultora en RRHH

1.1. Motivación

Una anécdota personal, la motivación para realizar este trabajo.

Como cualquier persona, he empezado con ilusión y con muchas ganas casi todos mis trabajos, pero con el tiempo la luna de miel fue terminando; después vino un periodo al que yo llamo de "estabilidad" y finalmente el momento de abandonar; en algunos de esos momentos realmente me ha dado mucha pena dejar algunos de esos trabajos, pero era necesario para evolucionar como persona y como profesional. La otra cara es cuando el abandono no fue por una decisión propia sino por parte de la organización. Solo me ha ocurrido un par de veces en mi vida laboral: en mi primer trabajo y en el trabajo donde más tiempo he permanecido; esto último fue devastador, pero fue en el que más he reflexionado sobre distintos aspectos de mi vida secular y laboral. Medité mucho sobre esta ruptura, en especial por la parte que me correspondía, y lo cierto es que yo había cometido mis propios errores. Los dos principales fueron haber entrado en esa organización sabiendo que yo no compartía ni su cultura empresarial ni sus valores y segundo haber permanecido tanto tiempo: yo debí de haber abandonado esa empresa cuando me di cuenta de que no cubría mi expectativa laboral; así me hubiera ahorrado esos momentos tan incómodos. Después entendí que para tomar la decisión correcta me hacía falta información y —ahora lo puedo decir— hasta valor, pero este fracaso fue un gran maestro en lo laboral, académico y personal.

Hacía tiempo que había visto que el abandono y la rotación de personal, voluntario o involuntariamente, siempre tiene un costo. Esto lo percibí desde la primera vez que empecé a trabajar en el área de la automatización de los procesos de recursos humanos. Desde entonces,

hace más de 15 años, he visto cómo diferentes empresas, órganos de gobierno, hospitales y centros educativos de México y España gestionan su personal con herramientas de software, recolectan su información, hace evaluaciones y —algunas entidades mejor que otras— toman decisiones sobre la permanencia de cada uno de sus empleados. Pero desde mi experiencia, las decisiones sobre el abandono del puesto de trabajo no toman en cuenta la gran cantidad de información que día a día los sistemas de gestión van acumulando en la base de datos, la cuál podría ayudar a responder las siguientes cuestiones: ¿Cuándo retener o dejar ir a empleados? ¿Qué hacer para retener al empleado talentoso? ¿Cómo encaminar a los empleados dentro de la empresa?

Se me ocurren muchas preguntas sobre el abandono del puesto de trabajo, pero esta última es la que me hizo ver que tal vez a las organizaciones que ya cuentan con un sistema de información para la gestión de recursos humanos les hace falta un *sistema de toma de decisiones* para gestionar a su personal en situaciones como ésta. En el momento de escribir este trabajo, he visto cómo los medios de comunicación constantemente nos bombardean con noticias de que la inteligencia artificial va a sustituir a un porcentaje muy alto de empleados en los próximos años, pero no mencionan que también la inteligencia artificial puede ser una herramienta para gestionar a los empleados, y el mejor indicador de que se está gestionando bien a un empleado es cuando este está feliz con el trabajo que hace.

Muchos autores comparan la relación de empleado con su organización como un matrimonio, lo cual me hizo recordar que el primer modelo de decisión que diseñé fue una cadena de Markov en la asignatura de *procesos estocásticos* de la carrera de matemáticas aplicadas. De esto ya hace tiempo. Con mis compañeros de equipo de trabajo buscamos un modelo sencillo, cotidiano y donde la recolección de la información para alimentar el modelo fuera relativamente fácil y rápido; casi de inmediato se nos ocurrió hacer una cadena de Markov donde cada nodo representara un estado civil: soltero, novio, unión libre, casado, divorciado, viudo, finado. Recolectar los datos para este modelo fue una experiencia muy divertida; hicimos una encuesta buscando que nuestra muestra fuera lo más representativa de la población del barrio donde vivíamos. En ese entonces no teníamos los medios que existen ahora: no había teléfonos inteligentes, el acceso a Internet estaba limitado para cierto número de profesores, usábamos discos para compartir la información y las encuestas las hicimos a pie de calle o en centros comerciales con papel y lápiz; después las concentramos en una hoja de cálculo e hicimos nuestro propio programa para simular los cambios de un estado a otro en nuestro pequeño modelo. Llegamos a la conclusión que la mejor decisión sobre cambiar de estado civil que puede hacer un individuo es no cambiar: si no tienes pareja, no la busques; si ya la tienes, no la abandones; y si quieres pasar del noviazgo al matrimonio, mejor olvídalo. ¿Cómo llegamos a esa conclusión? Porque observamos que todo cambio de estado implica un costo en tiempo y dinero, pero también aporta recompensas. Pero también llegamos a la conclusión de que ésa es una decisión racional, que puede ser tomada como si fuera un proceso para obtener la máxima ganancia o la mínima perdida en tiempo y dinero. Afortunadamente la mayoría de las personas

no toman esa clase de decisiones pensando en el tiempo o el dinero, sino en los sentimientos, que son las recompensan que ofrece cambiar de estado en esa cadena de Markov. El gran fallo de ese proyecto fue solo quedarnos con el modelo de Markov y no llegar a construir un modelo de toma de decisiones. ¿Pero cómo medir la recompensa en términos de sentimientos?

El anterior ejemplo me hizo pensar que tal vez una forma de abordar el problema de la gestión de personal era modelizándolo mediante una cadena de Markov, porque además de que se busca modelar algo parecido a una relación matrimonial, los sistemas de información que gestionan a los empleados son muy sensibles a los tramos de tiempo en que las diferentes variables toman su valor. Esta evolución del empleado dentro de la organización es la que puede modelarse como los *estados* por los que pasa un empleado. El resto es saber que *decisiones* se deben tomar, pero desde la posición de la organización[1] buscando la mejor gestión de los empleados, que para mí es que las personas sean felices y se sientan realizadas con el trabajo que hacen día a día. Éstos son, a mi juicio, los criterios principales que se deberían tener en cuenta en esa toma de decisiones.

1.2. Objetivos

Como se trata de un proceso de toma de decisiones a partir de la información guardada en una base de datos, el modelo adecuado sería un diagrama de influencia, pero como ya también se ha mencionado, la información de personas va cambiando en el tiempo. Los estados en que un empleado pasa mientras esté dentro de la organización también van a ir cambiando, con lo que el proceso se puede modelar como un modelo de Markov y al combinarlo con el diagrama de influencia tendremos el modelo adecuado. Los diagramas de influencia son una generalización de las redes bayesianas en que, además de resolver problemas de inferencia probabilística, también es la herramienta con la que se pueden modelar problemas de toma de decisiones.

Una observación que es importante mencionar, estas soluciones o herramientas suelen encontrarse fuera del propio sistemas de gestión de datos de los empleados, pero utilizan la información generada por la gestión. En resumen, el objetivo es generar diagramas de influencia con cadenas de Markov (MID-Markov Influence Diagrams) a partir de los datos de de la gestión de los recursos humanos para la toma de decisiones en el proceso de separación de un empleado de su puesto de trabajo (despido), pero como paso previo se va a explicar la arquitectura del modelo de base de datos para esta gestión, las tablas historificadas, cómo deben estar normalizadas y qué problemas en cuanto a la calidad y consistencia de la información se deben tomar en cuenta al tratar con el modelo de históricos para el modelado de datos. El modelo de histórico implica no solo saber qué valor tiene una variable sino también cómo ha

[1] Yo no dudo que otro campo de aplicación de la inteligencia artificial sería ayudar al empleado a tomar decisiones sobre su propio puesto de trabajo al igual que hay sistemas que sirven para tomar decisiones sobre inversiones, viajes u otras áreas en las interacciones humanas.

variado ese valor en periodo de tiempo que se ha almacenado en la base de datos. Esta es otra de las aportaciones que ofrece este trabajo.

1.3. Estado del arte

En la gestión de recursos humanos, las redes bayesiana se han aplicado en procesos que pueden conllevar un riesgo para los empleados al efectuar su trabajo (Arnott and Dodson, 2013; Pereira and Alves Lima, 2015; Jan et al., 2018). También se han aplicado a la toma de decisiones, la administración del conocimiento (Hafeez and Abdelmeguid, 2003) y la selección del personal (Ferreira and Borenstein, 2012).

La empresa tecnológica que más ha invertido en desarrollar herramientas para recursos humanos con inteligencia artificial es IBM, pero al igual que en otras empresas, la mayor aplicación está dentro del proceso de reclutamiento de empleados(Nicastro, 2020). Aunque la herramienta de IBM ya puede hacer una proyección del tiempo que el candidato permanecería en la empresa en caso de ser reclutado(Guenole and Feinzig, 2018; García Vega, 2020).

También hay modelos basados en redes bayesianas que sirven para predecir el tiempo que el empleado va a permanecer en una organización y la satisfacción media de la empresa con el empleado sobre la duración del empleo, algo muy parecido al objetivo de este trabajo. Solo que la aportación de este trabajo es, además del modelo de red bayesiana, la forma de extraer la información a partir de los datos históricos en recursos humanos(Ashcroft, 2012).

1.4. Estructura de la memoria

La memoria de esta proyecto se estructura en los siguientes capítulos:

1. Introducción general y objetivos.
2. Conocimientos previos.
3. Modelos de Markov.
4. Análisis de resultados.
5. Discusión y conclusiones.

En el capítulo 1, como ya se ha visto, se expone la motivación personal que me llevó a realizar este trabajo, los objetivos que se pretenden cubrir y el estado del arte que había en el momento de escribir este trabajo.

El capítulo 2 es donde se explica el conocimiento previo (*background*) o la parte donde el conocimiento de RRHH se explica para aplicarlo en este modelo. Es en esta parte donde se expone la problemática del *abandono* (por decisión del empleado) o *separación* (despido)

de un empleado de una organización desde el punto de vista de la propia organización pero a través de los RRHH. Para ello se hace uso de la información generada a partir de los sistemas de información en la gestión de los empleados, se definen las dos variables que reflejan el estado en el que un empleado se encuentra dentro de la empresa, y se muestra cómo con ellas podemos construir un modelo de Markov para modelar los estados por los que pasa el empleado. Finalmente, se explica qué otras variables recuperadas de este sistema de información se pueden usar para construir un diagrama de influencia.

En el modelo, estas variables son los nodos de la red que se va a construir. Para ello se procesan los datos —tanto para obtener los vínculos entre las variables como las relaciones de probabilidad entre ellas— con un programa hecho en Python; de este modo se generan diferentes modelos de diagramas de influencia. Todo esto se explica en el capítulo 3 y el código del programa se encuentra en el anexo A.

En el capítulo 4 se hace una validación cuantitativa y un análisis de casos para los modelos que se han generado, con el fin de seleccionar el mejor de ellos.

Y finalmente, el capítulo 5 se ofrece una discusión de los resultados obtenidos, algunas consideraciones éticas sobre el posible uso de estos modelos, las conclusiones obtenidas y la propuesta de futuros trabajos.

Capítulo 2

Conocimientos previos

"Nada en este mundo debe ser temido... solo entendido. Ahora es el momento de comprender más, para que podamos temer menos."

María Salomea Skłodowska-Curie

2.1. Enmarcamiento: gestión de recursos humanos y el problema del abandono del puesto de trabajo

El mayor activo que puede tener cualquier organización es su capital humano, es el conjunto de empleados de una organización los que hacen funcionar el sistema de producción y organiza todos los componentes que integran este sistema. Un empleado es una persona que durante un periodo de tiempo ejerce ciertas funciones para alcanzar un objetivo, y a cambio de ello recibe una retribución monetaria denominada sueldo. El entorno en donde desarrolla estas funciones y que se encarga de evaluar el cumplimiento de estos objetivos y la retribución es la organización. Una organización puede ser una empresa, un centro educativo, un ente de gobierno entre otros[1]; con lo que apreciamos una diferencia: el empleado es una persona física real mientras la organización es un ente artificial creado por el acuerdo de otras personas u organizaciones.

Dentro de las organizaciones el mayor compromiso interno es satisfacer la retribución monetaria de sus empleados. El cálculo de cuánto y cuándo va a entregar esa retribución a los empleados se denomina *cálculo de nómina*. La nómina es la relación de los empleados a retribuir. Esta retribución suele hacerse en relación a las funciones que el empleado lleva a cabo dentro de la organización en un periodo determinado. Estas definiciones, que pueden parecer obvias, son la forma en que se va a explicar sobre qué entorno se aplica un ERP. Y al ente que gestiona la esta relación entre organización y empleados la vamos a definir como *recursos*

[1]Una ONG tiene la denominación de organización, pero ella gestiona la actividad de voluntarios, y por ello no hay una obligación de recompensar con un salario a estas personas.

humanos de aquí en adelante. Los recursos humanos (RRHH) son entes que se mueven en un mundo real. Pero si lo queremos ver como un modelo basado en agentes, los componentes de ese modelo son:

- Entorno: la organización.
- Agentes: los empleados.
- Interacciones: el tiempo en que el empleado está ligado a la organización y las funciones que desempeña.
- Resultados: pueden obtenerse diferentes resultados de este modelo: cálculo de la nómina, gestión del desempeño del empleado, medición de la satisfacción del empleado dentro de la organización.

La forma en que el empleado está ligado a la organización es mediante un *puesto*. El puesto es el que determina las funciones que va a desempeñar el empleado y a veces es el propio puesto el que define que sueldo debe recibir el empleado. En este modelo tenemos variables que son exclusivas del empleado, como el nombre propio, fecha nacimiento, clave de documento de identificación, etc.; otras son exclusivas del puesto, como las funciones a desempeñar, el lugar jerárquico dentro de la organización, el lugar de trabajo, etc. La relación entre el empleado y el puesto de trabajo es una convención entre la organización y una persona física y está determinada por un periodo de tiempo. Este periodo de tiempo a su vez tiene sus propias variables; las más importantes son: qué puesto tiene un empleado, qué contrato tiene, cuánto tiempo ha dedicado el empleado a cumplir sus objetivos, qué salario ha recibido y qué incidencias se han producido dentro de la organización. En la figura 2.1 se esquematiza como sería este modelo basado en agentes:

Figura 2.1: Modelo basado en agentes de los recursos humanos (RRHH) en una organización.

La obtención de los resultados en este modelo son los procesos propios de los RRHH: selección de personal, gestión del tiempo de trabajo, definición de los puestos de trabajo,

cálculo de la nómina, evaluación del desempeño... entre muchos otros. Como se ha mencionado previamente, el mayor activo que tiene una organización son sus empleados. La forma correcta de gestionarlos y tomar decisiones sobre cómo gestionar este recurso es indispensable, pero muchas veces esas decisiones se toman de una forma empírica y sin un sustento analítico que ayuden a tomar esa decisión. Y de entre todos los procesos que involucran una toma de decisiones de la organización sobre los empleados, este trabajo se va a centrar en el proceso de abandono o separación de un empleado de su puesto de trabajo usando la información disponible.

2.2. Definición de un planificación de recursos empresariales (ERP)

El uso del ordenador personal en diferentes áreas de las empresas y la necesidad de tener centralizada la información de todos los procesos de negocio en los años 1990 del siglo pasado fue detonante del surgimiento de los ERP (Enterprise Resource Planning), o la planificación de recursos empresariales. Estos sistemas se caracterizan por integrar los diferentes procesos de negocio de la empresa —ya no solo la parte de gestión de materiales— en un solo sistema de información, compartiendo la información e interrelacionando la información de las diferentes áreas de la empresa. Otra característica de estos ERPs es el uso de una sola interfaz de conectividad. A este modelo se le conoce como ERP "tradicional". Al inicio de este siglo y como consecuencia de que diversas empresas tenían sus propios ERPs y que entre varias empresas tenían algún tipo de relación (como la de proveedor y comprador), y que entre ellas existía la necesidad de compartir y consultar cierta parte de su información, surgió el concepto de SCM (Supply Chain Management) o gestión de la cadena de suministros.

Un ERP es, en resumen, un sistema de información automatizado de una empresa que en una sola base de datos integra la información generada por las diferentes áreas de la empresa,al cual se accede y se gestiona desde la misma interfaz de conectividad.

La propuesta de nuestra investigación es el uso de la información contenida en un ERP para que, con modelos predictivos de inteligencia artificial, se facilite la toma de decisiones en el área de RRHH como un punto de negocio, y llevar el modelo al área de negocios como parte del activo de la organización. Esto es un aprendizaje de negocio sobre el empleado y las decisiones que afectan a la organización. El ERP aplicado a los RRHH tiene como componentes principales los empleados[2], los puestos y los periodos de trabajo en una organización.

Es importante mencionar que para la correcta explotación de la base de datos del ERP, ésta

[2]Existen ERPs que van más allá de la gestión de empleados, pues el componente principal es la persona, y algunas de esas personas son empleados, otras son candidatos, familiares de los empleados, etc. En este trabajo nos limitaremos a analizar como componente principal al empleado, aunque también es interesante expandir los modelos de análisis a un tipo de agente que es la persona, de modo que una de sus variables sea qué tipo de relación tiene con la organización: empleado, candidato, familiar, proveedor de servicio, etc.

tiene que estar correctamente normalizada (Moya, 2016) y que cada variable que intervienen en los procesos de recursos humanos es un dato histórico. El conjunto de variables que definen los datos de un empleado son una serie de registros históricos que entrelazan datos de tramos de diferentes tablas para cada empleado.

2.3. Características de la organización que se va a usar de ejemplo, alternativas y estrategias

En nuestro trabajo vamos a desarrollar un modelo a partir de los datos reales de una empresa que se dedica a la evaluación de riesgo financiero de negocios. Es una empresa mediana, de entre 400 y 500 empleados, con actividades muy similares entre sus empleados y con características muy particulares que la diferencian del resto de las organizaciones. Para este trabajo, la empresa solo ha proporcionado los datos del ERP de la gestión de recursos humanos, con datos desnaturalizados de los años 2010 al 2017. No nos ha permitido consultar datos de otras bases de datos, como los datos de clientes, transacciones, facturación, etc.

Esta empresa tiene una plantilla de empleados que se ha mantenido durante más de 20 años. Casi no hay rotación de personal, pero la plantilla empieza a envejecer y es necesaria una renovación generacional. Aunque el sector del mercado de esta empresa es muy especifico y prácticamente no ha variado mucho, le preocupa que tanto los empleados que ya llevan tiempo como los que se acaban de incorporar decidan abandonar a la empresa en un futuro próximo. La empresa maneja las siguientes opciones para cada uno de sus empleados:

- Retener al empleado dentro de la empresa.
- Permitir que el empleado abandone a la empresa.
- Separar (despedir) al empleado de su puesto de trabajo.

Si ocurre cualquiera de las dos últimas situaciones, finaliza el puesto de trabajo.

Por política de la propia empresa, las únicas estrategias para retener o separar al empleado son:

- Cambio de salario bruto anual.
- Cambio en el número de cursos formativos que la empresa le da al empleado.

2.4. ¿Separación o abandono?

En el argot de los sistemas orientados a los RRHH, el periodo de tiempo en que un empleado mantiene una relación con la organización está limitado por las fechas de alta y baja. En este trabajo nos interesa saber cómo se llega al momento de la baja. Los motivos de baja se pueden

dividir en dos grandes grupos: la organización decide separar al empleado del puesto (despido) o el empleado decide abandonar a la organización (renuncia). Los motivos particulares del primer grupo pueden ser: finalización de contrato, despido procedente por diferentes causas o aplicación de un ERE, entre otros. Para el segundo grupo las causas pueden ser diversas e incluso una combinación de ellas: motivos familiares, insatisfacción con el puesto de trabajo, inconformidad con el salario, cuestiones de salud, estancamiento, motivación por el cambio y muchas otras más. Desafortunadamente para los ERPs, este segundo grupo solo se almacena como "baja voluntaria" en la base de datos. También se debe tener cuidado con aquellas bajas que son por separación, pues algunas de ellas también son ocasionadas por el propio empleado. Por ejemplo, aquella persona que después de un tiempo no se siente satisfecha con su trabajo y sabe que la organización está obligada a pagarle una indemnización por su separación del puesto. Si ese empleado tiene mucha capacidad y la organización no ha sabido encaminarlo, no solo pierde por la indemnización que hay que pagarle: pierde talento y hasta oportunidades de generar otras actividades que podrían ser provechosas para la organización con la aportación e innovación que podría proporcionar ese empleado, lo cual supone un costo incuantificable.

Es frecuente encontrar ejemplos de empleados que han salido de una forma dolosa de una organización y posteriormente crean su propia organización, que se puede convertir en una competencia muy poderosa, o bien se van a la competencia que ya existe, para fortalecerla. Por eso se ha decidido incluir estos motivos de baja como el proceso de abandono de la organización, porque al fin y al cabo, se trata de evitar que los empleados abandonen el puesto de trabajo. A este fenómeno se le conoce también como "rotación de personal", pero no se debe confundir con el proceso de rotación dentro de la misma organización, que incluso es saludable y en ciertos entornos inevitable. Los clásicos ejemplos son los equipos deportivos profesionales, que por motivos de edad no pueden ni deben mantener a la misma plantilla para afrontar sus compromisos deportivos.

2.5. Eventos inciertos: los motivos que ocasionan el abandono

El análisis de los motivos del abandono de puesto de trabajo se suele realizar a partir de encuestas con la plantilla laboral. Muchas veces este proceso es empírico, sesgado y desactualizado. Sin embargo, algunas causas siguen siendo universales y se pueden medir usando los datos del ERP de RRHH (Branham, 2020; Varios, 2017):

- Incumplimiento de expectativas.
- Desajuste entre la persona y el puesto.
- Falta de seguimiento al empleado.

- Falta de un plan de carrera.
- Falta de reconocimiento.
- Sobrecarga de trabajo.
- Desequilibrio entre el trabajo y la vida personal.
- Falta de confianza en el liderazgo.
- Difícil relación con sus superiores.
- Favoritismo y la falta de meritocracia.
- Falta de compensaciones justas.
- Mal clima laboral.
- Falta de aprendizaje continuo.

Cada empleado valora de diferentes formas cada uno de los anteriores motivos. En una encuesta realizada en los años 1970 en una empresa americana —que contaba con una plantilla de obreros no cualificados, obreros cualificados, administrativos y empleados con grandes cualificaciones— se observó que cuanto menos cualificado era el empleado más influían los motivos externos a la organización, como la familia, la edad, el lugar de trabajo, etc., mientras que para los empleados más cualificados eran más importantes los planes de carrera y los objetivos de la empresa (Bonache and Ángel, 2006). El vídeo de las "generaciones y el trabajo" publicado por PerfilHumano en YouTube[3] nos habla de cómo diferentes generaciones valoran de diferente manera el trabajo y a las organizaciones. Las generaciones que nacieron después de la Segunda Guerra Mundial y hasta los años 1970 valoran más las organizaciones estables y jerárquicas donde el trabajo y la vida personal están muy bien diferenciados. En cambio la generación X, los nacidos después de los años 1970 hasta los 1990, valora más los trabajos donde se pueda crecer rápidamente; nos le importa tener cargas laborales excesivas e incluso convertirse en adictos al trabajo. En cambio, la generación actual, los famosos *millennials* que han nacido después de 1990, buscan más el trabajo flexible, informal y creativo, entre otras muchas cosas[4]. Cada generación tiene diferentes valores, lo cual es un punto que actualmente está cobrando mucha importancia en la gestión de los RRHH. Cuando se implementaron los primeros ERPs en las empresas, primaba la *dirección por objetivos*, que gestionaban a las personas como objetos; los sentimientos eran un factor ajeno al trabajo. Con los cambios generacionales y los

[3] https://www.youtube.com/watch?v=Tk2O2jgmGDM

[4] El autor de este trabajo se puede considerar un nativo de la generación X. Sin embargo, se siente más identificado con la generación de los millennials. De hecho, valoró más permanecer en el trabajo que le diera flexibilidad de horario y tele-trabajo que un buen salario con el fin de poder escribir este trabajo. Por eso, a veces es mejor hablar de individuos que de generaciones.

avances tecnológicos se desarrolla un modelo de *dirección por valores*, donde influye mucho la individualidad de la persona, su cultura, su generación y sus propios intereses. En la figura 2.2 se muestran las diferencias que hay entre las distintas generaciones y qué es lo que más valoran en el trabajo.

Figura 2.2: Las 4 generaciones y el trabajo. Primero tenemos la generación del ***baby boom.*** Son personasnacidas después de la Segunda Guerra Mundial, procedentes de familias estables y seguras, con tendencia a cuestionar la autoridad. No se guían principalmente por reglas, disciplinas o límites. Su huella tecnológica es el uso de la radio y los periódicos impresos. Su principal valor en el trabajo es la estabilidad. La **generación X** está formada por personas nacidas en los años 1970 y 1980, que han vivido el paso tecnológico del siglo XX al siglo XXI, con la mayor transformación cultural y laboral. Son inseguras frente al cambio. Su huella tecnológica consiste en el periódico impreso, la radio y, especialmente, la televisión. Su principal valor en el trabajo es el ascenso jerárquico y el reconocimiento. Los ***millennials***, nacidos a partir de los años 1990, son hijos de padres muy protectores. No están acostumbrados a asumir responsabilidades. Están integrados tecnológica y multiculturalmente; son dinámicos y están bien formados. Son los nativos de la era digital. Sus principal valores en el trabajo son la flexibilidad, la informalidad y la creatividad. Dentro de poco irrumpirá en el mercado laboral la nueva **generación Z**. Son jóvenes nacidos en este siglo y muchos de ellos aún están en la escuela. Su principal característica es la gran exposición a las redes sociales y a la tecnología, y su dependencia de ellas, lo que hace que el principal valor que van a exigir en un trabajo sea la trascendencia, es decir, un trabajo que tenga un impacto para el resto de la sociedad..

Con este nuevo panorama, se tiene que considerar una gestión del talento a partir de una dirección por valores, que tenga en cuenta las generaciones del trabajo. Sin embargo, la opinión del autor argentino Fredy Kofman, resumida en la serie de libros *Metamanagement*, la satisfacción del empleado con su puesto de trabajo se da si se cumplen las siguientes características (Kofman, 2001):

1. La organización le marca al empleado los objetivos claros a cumplir.

2. Lo organización le brinda los medios necesarios para cumplir con esos objetivos.

3. La organización valora el cumplimento de esos objetivos por parte del empleado.

Pero aparte de los motivos que son propios de la organización, debemos de tener en cuenta que el empleado tiene características que influyen en la forma en que evoluciona la relación con la organización. Algunas de las más significativas son:

- Edad.
- Sexo.
- Nivel de estudios.
- Dirección.
- Estado civil.
- Idiomas.
- Capacidades.
- Nacionalidad.

Un ejemplo muy común es de aquel empleado que tiene muy buenas cualificaciones, los objetivos muy bien definidos por la organización y todos los medios para cumplirlos, pero que por situaciones familiares su rendimiento es bajo. Como vemos, tanto la organización como el empleado valoran cómo es la relación que mantienen entre ellos. Para la organización es fácil valorar la relación: basta con medir el desempeño o cumplimiento de los objetivos. Pero por parte de los empleados, el problema radica en cómo valora cada uno de ellos a la organización y si esto puede influir en un proceso de abandono del puesto de trabajo.

2.6. Employee Value Proposition y los estados por los que pasa el empleado

El EVP (Employee Value Proposition) se puede traducir al español como la "propuesta de valor para el empleado". Hay diferentes definiciones para el EVP. Coloquialmente se dice que el EVP son aquellas características de nuestro trabajo de las cuales nos gusta presumir con nuestros conocidos. Según la definición de Minchington (2006) el EVP es "el conjunto de asociaciones y ofertas proporcionadas por una organización a cambio de las habilidades, capacidades y experiencias que un empleado aporta a la organización". El tema de los RRHH, como muchas áreas que implican el estudio y análisis del comportamiento de las personas, es todavía muy empírico. No hace mucho tiempo se pensaba que una gran empresa era garantía de un alto EVP por estabilidad y seguridad. Hoy en día pertenecer a una gran empresa no es garantía de ninguna de las dos. Por otro lado, se confundía al EVP con el salario y el paquete de prestaciones, dejando de lado el clima laboral, el crecimiento profesional o los intereses académicos de los empleados.

En los últimos años se ha empezado a hablar de una gestión de RRHH más responsable y enfocada a las personas como el mayor capital que tienen las organizaciones. Se busca hacer una *dirección por valores*, donde se entiende como valor "el lugar que ocupa una cosa en relación

con la ideales de bondad o maldad regidos por preceptos morales ampliamente aceptados por la sociedad" (Cetti and Simón, 2009). La relación de los empleados con la organización, con otros empleados y con el entorno externo es un sistema complejo donde el EVP de cada individuo es completamente personal e independiente del resto. Los motivos que se expusieron en la sección 2.5 son una parte de ese conjunto de características que forman parte del EVP. Tampoco hay una fórmula matemática exacta para determinar ese EVP, pues va a depender de la organización, de los datos que se almacena en el ERP de RRHH y de si en la organización se realizan evaluaciones de desempeño, Human Side[5], test psicométrico o de 360[6].

Al igual que los empleados, cada organización es única y no todas cuentan con la información exhaustiva y completa para determinar cómo valora el empleado a la organización. Hay que tener en cuenta que la información del ERP se tiene que adecuar a las características que forman el EVP; esta transformación es la que se verá en el siguiente capítulo, teniendo en cuenta que no todas las variables son parte del EVP ni que todos los empleados valoran de la misma forma esas características.

Con el EVP y la evaluación del desempeño que hace la propia organización, tenemos el cuadrante de la figura 2.3.

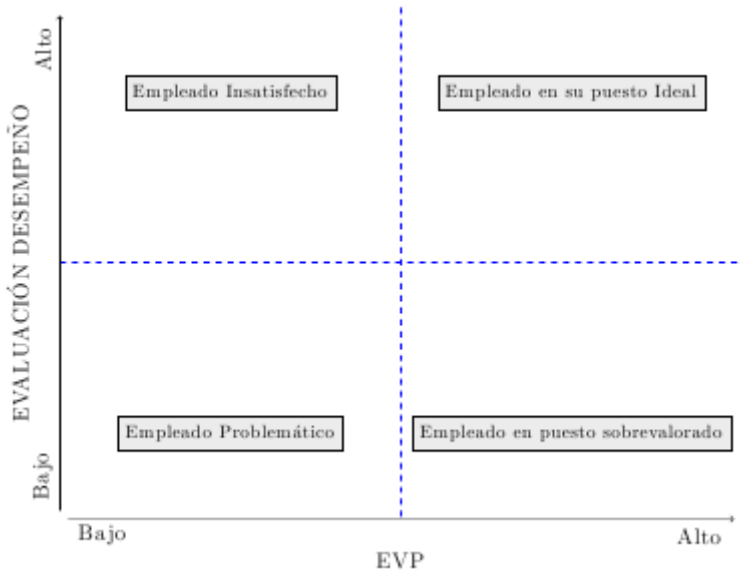

Figura 2.3: Cuadrante que de acuerdo a las valoraciones que tienen la organización y el empleado recíprocamente.

En la figura 2.3 se observa un cuadrante semejante a los que se suelen usar en la gestión de empresas; es semejante al famoso cuadrante de Gartner, pero en este caso colocamos al empleado en algunos de los cuadrantes de acuerdo a cada una de las siguientes situaciones:

- DES+/EVP- (Desempeño alto y EVP bajo): El empleado tiene las capacidades adecuadas

[5]Mide el estilo, los valores y el estilo de pensamiento de la persona.

[6]Mide el ambiente laboral a partir de la opinión que tiene cada empleado con respecto a su jefe, compañeros o colaboradores.

o sobradas para cumplir con sus objetivos, pero no se siente satisfecho con el puesto; puede ser por una remuneración baja, un trabajo que no le presenta ningún reto, un clima laboral tóxico o la falta de confianza en el liderazgo, entre otras causas. Cuando el empleado se encuentra en este cuadrante es sensible de ser tentado a abandonar el puesto de trabajo.

- DES-/EVP+ (Desempeño bajo y EVP alto): El empleado se siente satisfecho con la organización, puede ser que se encuentre cómodo con el salario, el clima laboral, la ubicación de la sede del trabajo o incluso que tenga una cordial relación con su responsable. Sin embargo, sus capacidades no son las adecuadas para el puesto y no cumple los objetivos que se le han asignado. Si la baja no viene por parte de la organización ni se toman medidas correctivas con este empleado, se crea un clima laboral tóxico, desmotiva a empleados cualificados y puede propiciar una baja productividad en el resto de los empleados, o que empleados que sí tienen las cualidades para el puesto se sientan tentados a abandonar la organización.

- DES-/EVP- (Desempeño bajo y EVP bajo): El empleado no solo no cumple con los objetivos que se le han asignado, sino que además no se siente, por diversas causas, satisfecho con el puesto de trabajo. Esta situación se da cuando se ha hecho un mal proceso de selección o recolocación- A veces es más saludable la separación del empleado (despido) que tomar medidas correctivas para que este empleado permanezca en la organización.

- DES+/EVP+ (Desempeño alto y EVP alto): Es la situación a la que tanto el empleado como la organización aspiran llegar. El riesgo en este estado es no saber cuándo es necesario hacer cambios o ajustes para evitar que el empleado pasar a cualquiera de los otros 3. Que un empleado se encuentre en este estado, desde el punto de vista del modelo, no implica que ocasione un abandono o separación por variables externas a la organización. Por ejemplo, la pandemia de este año obligo a muchas empresas a despedir empleados con muy buena valoración ¿pero como sabrían las empresas que vendría una pandemia a nivel mundial?

Cada empleado, cuando permanece por un periodo dentro de la organización, tiene una evolución en que cada uno de los elementos que conforman el EVP, los objetivos, funciones o incluso la posición en el trabajo sufren cambios, algunos para mejor y otros para peor. Incluso aquellos elementos que no sufren cambios también pueden ser beneficiosos o perjudiciales. El típico ejemplo es cuando después de muchos años el sueldo es el mismo pero una política familiarmente amigable incrementa el EVP. Por eso, una vez que el empleado ingresa en la organización, se ubica en cualquiera de estos cuatro estados dentro del cuadrante, y cambia de estado dentro de la organización o permanece en el mismo estado, a menos que ocurra el momento de abandono separación. En resumen, tenemos 6 situaciones por las que pasa el

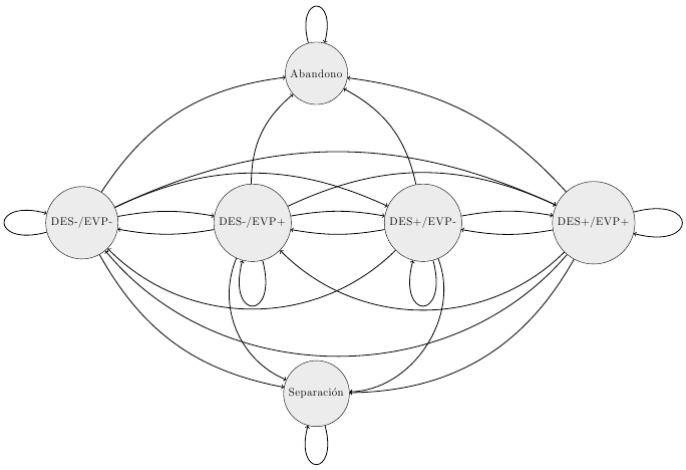

Figura 2.4: Diagrama de transiciones del modelo de Markov. Hay seis estados por los que pasa un empleado, en función de su desempeño (DES+ o DES-) y su EVP.

empleado en el modelo que se propone, con lo que podemos construir un modelo de cadena de Markov de 6 estados, como se muestra en la figura 2.4.

El objetivo de toda gestión de RRHH es que los empleados se encuentren en el cuadrante superior derecho, con desempeño alto y EVP alto, pero eso se da en muy pocos casos. Se dice que solo el 30 % de los empleados a nivel mundial se encuentran conformes con su trabajo (Muñoz, 2019; Schwartz, 2013), y ello repercute en la producción y satisfacción de los usuarios o clientes de este trabajo. Lo importante es detectar en qué estado se encuentra un empleado y qué decisiones se deben tomar para que el estado sea lo más cercana al cuadrante DES+/EVP+, aunque también se debe tener en cuenta que a veces es mejor permitir el abandono o generar la separación.

2.7. Formas de medir el desempeño y la Employee Value Proposition

2.7.1. Evaluación de desempeño

Existen varias formas de evaluar al empleado: se puede evaluar por el supervisor o responsable directo, los compañeros de trabajo, los clientes, una auto-evaluación, una evaluación 360 que sirve para medir el clima laboral y el liderazgo, etc. Hay además de sistemas informáticos que miden el tiempo que el empleado está en su puesto de trabajo. Pero la forma clásica de "evaluación de desempeño" es aquella que hace el supervisor o responsable directo, en la cual califica estos dos aspectos:

- Los objetivos directos del empleado.
- Las capacidades y actitudes que debe cubrir el empleado en el puesto.

Familia de puesto	Objetivos	Competencias
administrativo / técnico	10 %	90 %
coordinador / comercial	20 %	80 %
responsable / supervisor / director	30 %	70 %

Cuadro 2.1: Ejemplo de la forma en que pondera una empresa aseguradora el desempeño de sus empleados a partir de los objetivos y competencias.

Depende del puesto y la cultura empresarial de cada organización ponderar más un aspecto que el otro. Lo lógico es ponderar más las capacidades que el cumplimiento de objetivos cuando el puesto es poco cualificado y con pocos o ningún empleado a cargo; en cambio, la evaluación de un directivo se enfoca más en los objetivos, pues una de sus competencias es tener la capacidad de delegar el cumplimiento de los objetivos a los empleados a su cargo, pensando en las capacidades que tiene cada uno de ellos. Pero eso va a depender de los criterios de cada organización, porque incluso puede existir una ponderación específica para cada competencia u objetivo.

En el ejemplo que se va a desarrollar en este TFM, la evaluación se hace por la familia de puestos. Las ponderaciones que suele manejar esta empresa en sus evaluaciones de desempeño son las que se muestran en el cuadro 2.1. En este ejemplo se puede observar que la empresa pondera más las capacidades en los puestos de técnicos y comerciales, mientras que los cuadros de mando se ponderan más el cumplimiento de los objetivos, que suele ser la práctica generalizada en las diferentes organizaciones.

2.7.2. Medición del Employee Value Proposition

Como ya se ha mencionado, la percepción del valor que tiene el EVP depende de cada uno de los empleados. Generalmente la forma en que las organizaciones obtienen este valor es pasando cuestionarios directos a sus empleados. Uno de los más famosos es el que se usa para obtener la certificación de "Great Place to Work" (Ramírez Salazar, 2008), pero también se puede hacer de forma más casera, haciendo cuestionarios con respuestas abiertas o mediante una lista cerrada (ránking) de los aspectos que valoran más los empleados. Como en la organización de donde hemos tomado los datos para nuestros estudio no se ha hecho este tipo de encuestas o cuestionarios, mediremos el EVP usando el ránking de aspectos más valorados por los empleados, según la encuesta APD-HayGroup sobre los factores motivadores para los directivos y profesionales españoles (Jerico, 2011). Dado que esta empresa tiene una plantilla relativamente homogénea y solamente nos proporciona los datos que tienen en su propio ERP, las únicas variables que incluiremos en nuestro EVP son las que se muestran en el cuadro 2.2.

Algunas variables de esta encuesta no se utilizaron porque el objetivo es medir el EVP desde dentro de la organización, mientras que variables como "empresa de renombre", "reconocimien-

Ránking	Característica	Valoración de los empleados	Ponderación
1	aprendizaje continuo	8'8	27 %
2	desarrollo de carrera	8'3	25 %
3	equilibrio vida personal-profesional	8'2	25 %
4	retribución	7'6	23 %

Cuadro 2.2: Factores que más valoran los empleados según la encuesta APD-HayGroup , usando solo las características que se encuentran en una empresa de evaluación de riesgo de entre 400 y 500 empleados.

to social" o "seguridad en el puesto de trabajo" entre otros son útiles para comparar el EVP entre diferentes empresas. Pero como se ha dicho anteriormente, el EVP es un indicador que se construye de forma personalizada para cada organización y cada empleado le da diferente valor. En nuestro caso, no disponemos de las variables que explícitamente nos den valor a estos indicadores, sino que se tienen que construir a partir de la información existente:

- Aprendizaje continuo: asignamos un valor de 10 si ha tomado al menos dos cursos de formación en el año, 5 si solo ha tomado un curso y 0 cuando no se ha tomado ningún curso.

- Desarrollo de carrera: asignamos 10 si ha cambiado de puesto de trabajo en los últimos 2 años, 5 si ha cambiado en los últimos 5 años y 0 si no ha cambiado o han pasado más de 5 años desde el último cambio[7].

- Equilibrio vida personal-profesional: si tiene más de 30 años de edad entonces se evaluará si el empleado tiene hijos a su cargo, con lo cual se le darán 10 puntos; si tiene menos de 30 años de edad se valorará con 10 puntos si el empleado está continuando con sus estudios universitarios; en otro caso se le dará un valor de 0 puntos [8].

- Retribución: se asigna un 10 si ha tenido un aumento salarial de al menos del 25 % en su SBA (Salario Bruto Anual) en el último año; si el aumento ha sido mayor o igual al 10 % y menos del 25 % se asignará un 5, y 0 cuando este aumento menor del 10 %[9].

Vamos a mostrar dos ejemplos para explicar cómo se calculó el EVP con las propuestas que se presentan en este trabajo:

[7]Este una propuesta simplificada y generalista. Lo ideal sería contar con un plan de carrera personalizado y medir cuánto se acerca el empleado a ese plan en el trascurso de su permanencia en la organización. Desafortunadamente la base de datos que hemos usado para este trabajo no cuenta con información de planes de carrera.

[8]Nuevamente se trata de una propuesta simplificada y generalista para nuestro estudio. La mejor forma de evaluar el equilibrio entre la vida personal y profesional es hablar directamente con el empleado y que él mismo sea el que cuantifique cuánto equilibrio existe entre su trabajo y su vida personal.

[9]En teoría, el aumento salarial debe ser igual o superior al Índice de Precios de Consumo (IPC) para tener un salario justo.

Ejemplo 1: Andrés es un empleado de 26 años que ha pasado de ser becario a programador junior en el último año. Su SBA paso de ser de 7.200 € a 18.000 €. La empresa ha organizado un curso de "gestión del estrés en el trabajo" para todo el personal, pero el curso que él quería, de lenguaje de programación en Python, no fue aprobado y tendrá que esperar hasta el siguiente año para saber si lo aprueban o no. A él le faltan por concluir dos materias en la universidad, pero el traslado de su casa al trabajo es de al menos una hora diaria y esta tan cansado con el nuevo ritmo de trabajo que ha dejado abandonada la actividad académica para otro momento en que ya tenga más estabilidad laboral.

En este ejemplo, las 4 variables que se están usando para calcular el EVP de Andrés tendrán los siguientes valores:

- Aprendizaje continuo: 5 puntos.
- Desarrollo de carrera: 10 puntos.
- Equilibrio de la vida personal-laboral: 0 puntos.
- Retribución: 10 puntos.

Con la anterior información, el EVP tendrá el siguiente valor:

$$EVP = 0'27 \cdot Aprendizaje + 0'25 \cdot Carrera + 0'25 \cdot Equilibrio + 0'23 \cdot Retribución$$

$$EVP = 0'27 \cdot 5 + 0'25 \cdot 10 + 0'25 \cdot 0 + 0'23 \cdot 10 = 6'15$$

En este caso podemos decir que el EVP de Andrés es bueno. Pero si en los siguientes 2 ó 3 años no recibe el curso que quiere, o no logra hacer que su vida personal encaje con su vida laboral, o que su sueldo aumente o se mantiene en el mismo puesto, se corre el riesgo que el EVP pase de bueno a malo y que el empleado se vea tentado a abandonar a la organización.

Ejemplo 2: Beatriz es una madre soltera con una hija de 10 años. Empezó a trabajar en la empresa hace 16 años como administrativa en el departamento de contabilidad. No ha tenido ascensos ni subida salarial en los últimos 5 años. Sin embargo, ella se mantiene en este trabajo porque le he es difícil cambiar a sus 35 años de edad; además, la oficina le queda relativamente cerca de su casa y su jefe le da cierta libertad de horario para que pueda llevar y recoger a su hija a la escuela. Ella fue compañera de Andrés en el mismo curso de formación y no aspira a tener un nuevo curso.

En este ejemplo, aplicando las mismas 4 variables para Beatriz, se tendrán los siguientes valores:

- Aprendizaje continuo: 5 puntos.
- Desarrollo de carrera: 0 puntos.
- Equilibrio de la vida personal-laboral: 10 puntos.
- Retribución: 5 puntos.

Con la anterior información, el EVP tendrá el siguiente valor:

$$EVP = 0'27 \cdot Aprendizaje + 0'25 \cdot Carrera + 0'25 \cdot Equilibrio + 0'23 \cdot Retribución$$

$$EVP = 0'27 \cdot 5 + 0'25 \cdot 0 + 0'25 \cdot 10 + 0'23 \cdot 5 = 5'00$$

El EVP de Beatriz es bajo, y si además este año tiene una mala evaluación de desempeño, corre el riesgo de ser separada de su puesto. Pero si desde RRHH organizan un curso de Excel avanzado que le va a servir para que ella optimice su trabajo y le dé posibilidades de tener más argumentos para pedir un aumento de salario o incluso un ascenso laboral, su EVP aumentaría; en este ejemplo, una simpleacción de RRHH bastaría para fidelizar a esta empleada en su organización.

2.8. El costo de la rotación de personal

Se suele pensar que el costo de que un empleado abandone o sea separado de su puesto de trabajo es sólo el finiquito o la indemnización por despido, dependiendo cuál sea el caso. En el caso del finiquito no es propiamente un costo, pues es el dinero que el empleado ha devengado en el periodo de tiempo que ha permanecido en su puesto y no se le ha remunerado[10]. Para el caso del despido, además de la parte del finiquito que le corresponde al empleado, se debe pagar una indemnización por parte de la organización, la legislación en España, al momento de escribir este trabajo indica que (de Empleo y Seguridad Social, 2015):

- Despido improcedente real: correspondiente al despido en función de la antigua y la nueva normativa siempre que la empresa no esté en crisis: 45 días por año trabajado hasta el 12 de febrero de 2012 y 33 días por año trabajado a partir del 12 de febrero.
- Despido procedente en compañías en crisis: aplicable en empresas con caída de ingresos durante nueve meses consecutivos. En este caso, la indemnización asciende a 20 días por año trabajado, con un máximo de 12 mensualidades.

[10] En España los periodos de tiempo para hacer el cálculo total de remuneraciones devengadas es de un año y generalmente lo que se paga es la parte proporcional de las pagas extras y el número de días de vacaciones no disfrutadas.

- Despido improcedente con nuevo contrato (tras la reforma): en este caso, que será aplicado por defecto a partir del 12 de febrero de 2012, los trabajadores despedidos cobrarán 33 días por cada año trabajado con un máximo de 24 mensualidades.

- Despido improcedente con contrato antiguo (anterior a la reforma): en este caso, aplicable a los contratos anteriores a la reforma laboral. Si el despido fuese también antes de la modificación, la indemnización asciende a 45 días por año trabajado con un máximo de 42 mensualidades.

¿Es este el único costo? La propia experiencia nos dice que no es así, pues cuando un empleado se va, por la razón que sea, el resto del equipo absorbe su trabajo; a veces se realiza un proceso de reclutamiento; cuando ya se tenga la plaza cubierta, habrá un periodo de acogida con los respectivos costos de formación y otro periodo de tiempo en que el nuevo empleado no crea valor para la organización. Esto sin contar que hay empleados estratégicos o especializados por los cuales la organización deja de percibir cierto ingreso. Ese costo real es muy difícil de calcular. Algunos estudios estiman que puede ser 25 % del Salario Bruto Anual del empleado, pero otros lo elevan hasta el 400 %. Todo va a depender de cada organización, pero básicamente son 4 variables las que se utilizan para ese cálculo (Valencia, 2017):

1. Costo de la indemnización por despido.

2. Costos de contratación.

3. Costos de la acogida.

4. Ingresos perdidos.

En el ejemplo que estamos usando, se van a calcular los costos de la siguiente manera:

1. Costo de la indemnización por despido: 33 días por año trabajado con un máximo de 24 mensualidades.

2. Costos de contratación: un número fijo de 3.000 €, que es lo que cobra una empresa especializada en el reclutamiento de personal.

3. Costos de la acogida: para la empresa que estamos tratando, depende del tiempo que se tarde en reclutar el nuevo empleado, que suele ser de 3 meses[11], más el tiempo que el nuevo empleado necesita para formarse dentro de la organización para cumplir sus objetivos, y otros 3 meses que suele ser el periodo de prueba que dan las organizaciones para saber si es conveniente contratar al empleado. En total, son 6 meses del SBA.

[11] En estos 3 meses en que el puesto se encuentra descubierto, el resto del equipo absorbe el trabajo. Una forma de estimar el costo para la organización es usar el mismo salario del empleado que ha dejado descubierta la plaza.

4. Ingresos perdidos: como el ejemplo que estamos usando es una empresa que puede absorber la carga de trabajo de un empleado que se va con el resto del equipo, el costo estimado va a ser nulo.

Con los mismos ejemplos que hemos tratado, vamos a hacer el cálculo de los respectivos costos de abandono y separación:

Ejemplo 1: A Andrés le ha llegado una oferta de trabajo que le permite hacer tele-trabajo y le han prometido que le darán una serie de cursos de diferentes lenguajes de programación. Después del primer año de trabajo, él decide abandonar la organización.

El costo de encontrar otro candidato como Andrés es igual a:

1. Costo de la indemnización por despido: 0 €, a Andrés no se le paga Indemnización.
2. Costos de contratación: 3.000 € del proceso de reclutamiento.
3. Costos de la acogida: 9.000 € que son los 6 meses entre que se cubre el puesto y el nuevo empleado se prepara.

El costo total estimado de esta abandono va a ser de:

$$Costo = Indm + Contr + Acog = 0 + 3{,}000 + 9{,}000 = 12{,}000$$

Ejemplo 2: al final Beatriz ha entrado en un estado de inconformidad con su trabajo y piensa que lo mejor que puede hacer es provocar el despido para cobrar su indemnización y volverse autónoma para trabajar desde casa. Así poder dedicar más tiempo a su hija.

El costo para cubrir el puesto de Beatriz es igual a:

1. Costo de la indemnización por despido: El SBA del puesto de administrativa que tiene Beatriz es de 25.000 €. Cada día de salario para Beatriz es de 68,50 € aproximadamente. Como lleva trabajando 15 años, su indemnización es de 33.904 € aproximadamente.
2. Costos de contratación: 3.000 €, por el proceso de reclutamiento.
3. Costos de la acogida: 12.500 €, por los 6 meses que transcurren desde que se cubre el puesto hasta que el nuevo empleado se prepara.

El costo total estimado de esta separación va a ser de:

$$Costo = Indm + Contr + Acog = 33{,}904 + 3{,}000 + 9{,}000 = 12{,}000 = 49{,}404$$

Como se puede ver, a la organización le sale más costoso separar a Beatriz de su puesto que dejar que Andrés abandone el suyo. Mientras en el primer caso el costo es menos que el

salario de todo un año de Andrés, en el segundo caso, a la organización le cuesta casi dos veces el salario de todo el año de Beatriz. En cualquiera de los dos casos, lo ideal es que ninguno de los empleados abandone su puesto de trabajo.

2.9. Resumen de los componentes del modelo

Con los elementos que se han mencionado, tenemos todos los ingredientes para modelar y analizar este problema. Primero tenemos las variables propias de cada empleado y puesto de trabajo:

- Sexo.
- Edad: joven (menor de 30 años), adulto (mayor de 30 años y menor de 60) y mayor (más de 60) años.
- Familia: sin hijos, con hijos (1 o 2 hijos o personas a cargo), familia numerosa (más de 2 hijos o personas a cargo).
- Familia de puesto: administrativo, coordinador y responsable.
- Antigüedad en la empresa: poca (menos de 3 años), mediana (más de 3 y menos de 10) y mucha (más de 10 años).
- SBA: bajo (menos de 20.000 €), medio (más de 20.000 € y menos de 30.000 €), bueno (más de 30,000 € y menos de 45,000 €) y excelente (más de 45.000 €).
- Nivel educativo: sin formación universitaria y con ella.
- Número de cursos de formación tomados al año: insuficiente (1 curso o ninguno), suficiente (2 cursos), demasiados (más de 2).

Por otro lado, en nuestro modelo de Markov cada periodo de tiempo será de un año, en el cual se hará el cálculo de:

- Desempeño del empleado: bajo (una calificación menor a 90) y alta (una calificación mayor o igual a 90).
- EVP: bajo (una valoración menor o igual a 5 puntos) y alta (una valoración mayor de 5 puntos).
- El estado en que se está el empleado: DES+/EVP- (desempeño alto y EVP bajo), DES-/EVP+ (desempeño bajo y EVP alto), DES-/EVP- (desempeño bajo y EVP bajo), DES+/EVP+ (desempeño alto y EVP alto), baja (tanto se por separación y abandono.

Para cada empleado, se calculará lo que cuesta en cada periodo de tiempo:

- Separación.
- Abandono.

También se va a estimar lo que va a costar cada una de las decisiones que se pueden tomar con los empleados:

- Aumento del 20 % del SBA.
- Costo de formación.
- Tiempo de prueba: 25 % del SBA.

Capítulo 3

Modelo de Markov

"All models are wrong but some are useful."

George E.P. Box

3.1. Consideraciones previas

La herramienta para crear y evaluar los modelos de redes bayesianas de este trabajo es OpenMarkov,[1], un programa de código abierto que permite crear redes de forma gráfica . Los modelos que se generan en esta herramienta se guardan en un formato propio, ProbModelXML[2]. En nuestro estudio, los modelos se crearán de forma dinámica a partir de de la base de datos mediante un programa que hemos desarrollado en Python, luego los guardaremos en archivos y finalmente utilizaremos OpenMarkov como API para evaluar el desempeño de los diferentes modelos.

En esta sección se a exponer las consideraciones previas que se tomaron y que se deben considerar antes de generar los modelos.

3.1.1. Conjunto de datos y partición

En el capítulo anterior se especificó como deberían de ser los componentes del modelo. Pero hace falta definir cómo construir el conjunto de datos a partir de las tablas que conforman el ERP de RRHH de la empresa que estamos estudiando. Este conjunto de datos de construye como una tabla nueva *ad hoc* para nuestros propósitos, siguiendo las siguientes pautas:

- Para cada empleado y cada año en que ha permanecido dentro de la empresa se generará un registro para cada año[3] .

[1] www.openmarkov.org

[2] www.openmarkov.org

[3] En nuestro modelo los periodos de tiempo son anuales, pero dada la dinámica de algunas organizaciones este periodo de tiempo se podría acortar a semestres, trimestres o meses.

- Se incluirán las variables propias del empleado y puesto de trabajo que desempeñó durante ese año:

    1. Sexo.
    2. Edad, que estará categorizada de la siguiente manera: joven (menor de 30 años), adulto (mayor de 30 años y menor de 60) y mayor (más de 60) años.
    3. Número de hijos que tenía en ese año.
    4. Familia, que estará categorizada de la siguiente manera: sin hijos, con hijos (1 o 2 hijos o personas a cargo); familia numerosa (más de 2 hijos o personas a cargo).
    5. Grupo del puesto, que estará categorizada de la siguiente manera: administrativo, coordinador y responsable.
    6. Años de antigüedad.
    7. Antigüedad, que estará categorizada de la siguiente manera: poca (menos de 3 años), mediana (más de 3 y menos de 10) y mucha (más de 10).
    8. Importe del SBA.
    9. SBA, categorizado de la siguiente manera: bajo (menos de 20.000 €), medio (más de 20.000 € y menos de 30.000 €), bueno (más de 30.000 € y menos de 45.000 €) y excelente (más de 45.000 €).
    10. Porcentaje de aumento con respecto al año anterior.
    11. Descripción del porcentaje de aumento, categorizado de la siguiente manera: aumento y sin aumento.
    12. Nivel educativo, que estará categorizada de la siguiente manera: sin estudios universitarios, con estudios universitarios y con posgrado.
    13. Número de cursos de formación recibidos ese año.
    14. Formación, que estará categorizado de la siguiente manera: insuficiente (1 curso o ninguno), suficiente (2 cursos), demasiado (más de 2).

- El siguiente conjunto de variables que se deben incluir en la base de datos tienen que ver con las variables de desempeño y EVP:

    1. La calificación numérica del desempeño del empleado en ese año.
    2. El valor del desempeño: bajo o alto.
    3. El valor numérico del EVP que dio el empleado ese año[4] .

[4]En nuestro ejemplo, el valor total del EVP viene de la suma ponderada del valor que dio el empleado sobre aprendizaje, carrera, equilibrio y retribución dentro de la empresa. Estas variables podrían ser incluidas dentro del conjunto de datos para ampliar el número de variables si se quisiera aplicar otro algoritmo de aprendizaje. Pero para el modelo de este trabajo solamente son necesarios saber si el desempeño fue alto o bajo, y si el EVP fue alto o bajo.

4. El EVP: alto o bajo.

- Con las variables de desempeño y EVP se obtiene en qué estado del modelo de Markov se encontraba el empleado ese año. Entonces también se agregan las siguientes variables al conjunto de datos:

  1. El estado del modelo de Markov en que se encontraba el empleado ese año.
  2. El estado del modelo de Markov en que se encontraba el empleado el año anterior.
  3. Una variable booleana para indicar si ese año el empleado fue separado de su puesto.
  4. Una variable booleana para indicar si ese año el empleado abandonó su puesto.

- Y un conjunto final de variables que van servir para cuantificar las posibles políticas sobre el empleado y el puesto:

  1. La estimación del costo que implica subir el SBA al empleado.
  2. La estimación del costo que implica dar más formación al empleado.
  3. La estimación del costo que implica poner a prueba a un nuevo empleado para cubrir una plaza que ha quedado libre.
  4. Una variable booleana que va registrar si ese año el puesto fue cerrado después del abandono o separación del empleado si lo hubo.

Con los datos proporcionados en el ejemplo, se genera una tabla de 9.287 registros. Un problema es que en los primeros años recuperados para cada empleado se encontraron valores ausentes (*missing values*)[5]. Además, pensando cómo se va a evaluar el modelo, se optó por la validación cruzada de K iteraciones, con $K = 10$. Por eso, también se agregó una variable adicional donde se almacenará el número de partición al que pertenece el registro, que toma valores del 0 al 9, aleatoriamente.

3.1.2. Conjunto de parámetros

Hay varias formas de configurar la red probabilista con la que se puede modelar este conjunto de datos. En nuestro modelo tenemos variables que son fijas y variables que pueden cambiar de un periodo de tiempo al otro, lo que no sabemos todavía cómo se pueden conectar dichas variables (que serán los nodos de la red). Una forma interesante de encontrar esa red ideal es utilizar varias combinaciones de parámetros, para generar diferentes redes, que luego evaluaremos para obtener el mejor modelo para nuestros datos. Esta combinación de parámetros se puede almacenar a la vez en una tabla auxiliar con la que se podrán ir generando los diferentes modelos usando un lenguaje de programación. Los parámetros que se han considerado son:

[5]Una muestra de 1.000 registros de este DataSet se puede consultar en https://github.com/smontesv/tfm/blob/master/TFM_DataSet.csv.

- El número de nodos a enlazar: aquí vamos a variar entre un número máximo de conexiones que puede tener un nodo. Vamos a probar con valores de 4 hasta 8 conexiones, con lo cual podremos crear 5 tipos diferentes de redes[6].

- Conexiones internas parael estado 1: Como se había mencionado, sabemos que la red va a tener por un lado variables fijas y por otro las variables que pueden cambiar de un periodo a otro. A cada una de estas variables que cambian en un periodo de tiempo podemos duplicarlas como el valor que tenía el año anterior y el valor que tiene en el año a considerar. Las variables del año anterior o estado 0 solo las podremos conectar con las variables fijas, en cambio las del año actual o estado 1 las podemos conectar con las variables fijas, las del estado 0 y también con las del estado 1. Aquí la variación es crear redes cuyos nodos del estado 1 tengan conexión con las variables del estado 1 si o no; con este parámetro podemos tener 2 diferentes tipos de redes.

- Otra variable interesante es si queremos incluir políticas de decisión dentro de la red o no, con lo cual tenemos 2 dos tipos de redes más.

- Finalmente podemos diseñar cada red como un diagrama de influencia o un diagrama de influencia de Markov, lo cual da lugar a otros 2 tipos de redes más.

Combinando todas estas posibilidades tenemos $5 \times 2 \times 2 \times 2 = 40$ diferentes tipos de redes, pero como hemos particionado 10 veces la base de datos se generarán 400 redes. Hay 2 parámetros que serán fijos en todos los modelos: el corte de la ratio de ganancia[7], que se ha fijado en 0'75, y la corrección de Laplace, fijada en 0'75; más adelante se explicará el uso de estos dos parámetros, que se van a manejar como una constante. Aprovechando la estructura de esta tabla de parámetros, se van a agregar los campos de número de registros procesados, número de verdaderos positivos , número de verdaderos negativos, número de falsos positivos, número de falsos negativos[8] y la efectividad de la combinación. Estos campos nos servirán para almacenar la efectividad de cada combinación y hacer la comparación entre los diferentes modelos.

3.2. Construcción del modelo

Como se ha comentado, OpenMarkov genera y almacena las redes en formato ProbModelXML: Un archivo XML (eXtensible Markup Language) es, en resumen, un fichero de texto plano que sirve como un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados (Consortium, 2020; Wikipedia, 2020b). En nuestro caso son marcas o

[6] Se intentó generar modelos con 9 y 10 nodos a enlazar, pero eso implicaba generar una combinación tan grande de probabilidades que muchas veces el ordenador no lograba procesar los datos y almacenarlos.

[7] En la subsección 3.2.2 se esplica qué es y cómo se utiliza.

[8] En la subsección 4.1.2 se explica qué son y cómo se utilizan.

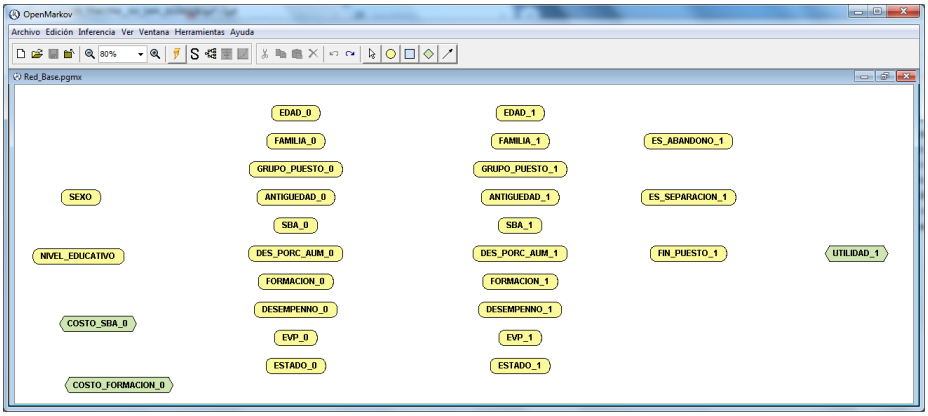

Figura 3.1: Modelo base de las redes bayesianas que se generan en este TFM.

tags que relacionan las variables (nodos de la red), sus relaciones (enlaces) y las probabilidades de esas relaciones entre esas variables.

El conjunto de variables que van a aparecer en todos los modelos: sexo, edad, familia, grupo del puesto, antigüedad, SBA, descuento/aumento de SBA, nivel educativo, formación, desempeño, EVP y estado. De estas, las que van a ser fijas son sexo y nivel educativo, pues estas variables se mantienen constantes en el tiempo. Las otras van a tener dos nodos por cada variable: la variable del año anterior y la variable del año actual. En el caso de que la red sea un modelo de Markov, cada variable tendrá un índice que indique la rodaja temporal a la que pertenece, mientras que en el caso de diagramas de influencia esta diferenciación se hará con un sufijo en el nombre: "_0" para el año anterior y "_1" para el año actual, como puede verse en la figura 3.1.

Las clases que queremos evaluar son: es_abandono, es_separación y es_fin_de_puesto. Distinguiremos si es un modelo de Markov o un diagrama de influencia; en este caso el tiempo se aplica en el año actual, lógicamente.

Finalmente agregamos los costos, que son el de aumentar el SBA (en el periodo anterior) y el de dar más formación (en el periodo anterior), así como la utilidad final. Todo ello constituirá el modelo base, que no tiene enlaces ni probabilidades; es lo que se muestra en la figura 3.1.

En los siguientes apartados se explicarán cómo se añaden los enlaces entre las variables, las clases y los costos, y cómo se calculan las probabilidades.

3.2.1. Construcción del grafo

Las decisiones que se puedan tomar durante el proceso aumentar el SBA y dar más formación al empleado. Esto siempre se hace en el periodo del año anterior, de modo que en el año actual se observan las consecuencias esas políticas. Entonces, por defecto, las variables SBA y formación del periodo anterior se dejan como nodos de decisión en el caso de que sea una modelo que incluya decisiones; el resto van a ser nodos de probabilidades. Estos nodos y las variables fijas van a estar enlazados como antecesores delos nodos SBA_periodo_anterior y

formación_periodo_anterior. A su vez, estos dos nodos estarán enlazados con sus respectivos costos, recordando que todos son del periodo anterior.

El siguiente nivel para enlazar incluye todos los nodos de probabilidad del año actual. El criterio para seleccionar qué nodos se enlazan (teniendo en cuenta la posibilidad de trazar enlaces entre los nodos del periodo actual) utilizaremos los conceptos de "información" y "entropía" (Cetti, 2009; Hernández Orallo and Ferri Ramírez, 2004). Una medida para la información, dada la probabilidad de una variable aleatoria dado el número de bits que se necesita para almacenar ese información, es:

$$I(x) = -\log_2(P(X = x))$$

Aplicando esta fórmula a las probabilidades de los ejemplos que hemos usado, obtendríamos el mismo número de bits que se han mencionado. En cuanto a la "entropía", es una palabra de origen griego (ἐντροπία) que se puede traducir como "vuelta" o "transformación". El término fue acuñado por el físico Rudolf Clausius en el siglo XIX para referirse a la medida del desorden que puede verse en las moléculas de un gas; posteriormente el físico Ludwing Boltzmann fue el que diseño una forma matemática de representar esta medida, desde el punto de vista probabilístico. Pero en el ámbito de la teoría de la información, que es la parte que nos interesa, se puede definir como la medida de incertidumbre de una fuente de información, como lo propuso el matemático Claude Shannon (Cetti, 2009). Entonces, la entropía en una variable aleatoria X se puede definir como "el valor medio ponderado de la información de los diversos estados de la variable":

$$H(X) = \sum_{x \in X} P(x) \cdot \log_2(P(x))$$

Tanto la clase como cada una de las variables aleatorias tienen un nivel de entropía, y nos interesaría identificar aquellas variables que, con respecto a la clase (en este caso la variable del estado actual), nos den mayor cantidad de información o, dicho de otra forma, que reduzcan la entropía. Esto lo podemos obtener haciendo un ránking de las diferencias de los niveles de entropía entra la clase, y la clase con respecto a esa variable:

$$IG(Y, X) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

donde la entropía condicional de Y dado X se obtiene de la siguiente forma:

$$H(Y|X) = -\sum_{x \in X} P(x) \cdot \sum_{y \in Y} P(y|x) \cdot \log_2(P(y|x))$$

También podríamos evaluar la ratio de ganancia con respecto a la variable usando las mismas medidas de entropía de la clase y las variables aleatorias de la siguiente forma:

$$GainR(Y,X) = \frac{H(Y) - H(Y|X)}{H(X)}$$

Vamos a usar un ejemplo con datos del propio modelo para explicar cómo se seleccionaron los enlaces de los nodos del estado actual: tenemos el nodo de EVP en el periodo actual y queremos ver cuál de los nodos del estado_periodo_anterior tiene mayor ratio de ganancia. Vamos a simplificar el ejemplo usando solo los nodos de sexo, formación y el propio nodo de EVP_periodo_anterior. Los valores que tenemos son los que se muestran en el cuadro 3.1.

		EVP año actual		
		Alto	Bajo	Total
Sexo	H	326	305	631
	M	284	233	517
	Total	610	538	1148

(a)

		EVP año actual		
		Alto	Bajo	Total
Formación año anterior	Suficiente	362	179	541
	Insuficiente	248	359	607
	Total	610	538	1148

(b)

		EVP año actual		
		Alto	Bajo	Total
EVP año anterior	Alto	314	19	333
	Bajo	296	519	815
	Total	610	538	1148

(c)

Cuadro 3.1: Tabla de frecuencias de sexo con EVP_periodo_actual (a), de formación_periodo_anterior con EVP_periodo_actual (b) y de EVP_periodo_anterior con EVP_periodo_actual (c).

La entropía para la variable sexo sería:

$$\begin{aligned} H(Sexo) &= P(Sexo = H) \cdot \log_2(P(Sexo = H)) + P(Sexo = M) \cdot \log_2(P(Sexo = M)) \\ &= 0'5497 \cdot (-0'8634) + 0'4504 \cdot (-1'1509) \\ &= -0{,}9929 \end{aligned}$$

La entropía para la variable EVP_periodo_actual sería:

$$\begin{aligned} H(EVP_1) &= P(EVP_1 = Alto) \cdot \log_2(P(EVP_1 = Alto)) + P(EVP_1 = Bajo) \cdot \log_2(P(EVP_1 = Bajo)) \\ &= 0'5314 \cdot (-0'9122) + 0'4686 \cdot (-1'0934) \\ &= -0{,}9972 \end{aligned}$$

La entropía para la variable sexo dado el EVP_periodo_actual sería:

$$\begin{gathered}H(Sexo|EVP_1) = -P(EVP_1 = Alto) \cdot (P(Sexo = H|EVP_1 = Alto) \\ \cdot \log_2(P(Sexo = H|EVP_1 = Alto)) \\ +P(Sexo = M|EVP_1 = Alto) \cdot \log_2(P(Sexo = M|EVP_1 = Alto))) \\ -P(EVP_1 = Bajo) \cdot (P(Sexo = H|EVP_1 = Bajo) \\ \cdot \log_2(P(Sexo = H|EVP_1 = Bajo)) \\ +P(Sexo = M|EVP_1 = Bajo) \cdot \log_2(P(Sexo = M|EVP_1 = Bajo))) \\ = -0'5314 \cdot (0'5344 \cdot (-0'9039) + 0'4656 \cdot (-1'1029)) \\ -0{,}4686 \cdot (0'5669 \cdot (-0'8188) + 0'4331 \cdot (-1'2073)) \\ = -(-0'5295) - (-0'4626) = 0'9921\end{gathered}$$

de modo que la ratio de ganancia de información para sexo y EVP_periodo_actual es:

$$\begin{gathered}GainR(Sexo, EVP_1) = \frac{H(Sexo) - H(Sexo|EVP_1)}{H(EVP_1)} \\ = \frac{-0{,}9929 - 0'9921}{-0{,}9972} = 1'9906\end{gathered}$$

Aplicando los mismos cálculos para la formación_periodo_anterior y EVP_periodo_anterior, obtenemos los siguientes valores:

$$\begin{gathered}GainR(Formacion, EVP_1) = 1'9511 \\ GainR(EVP_0, EVP_1) = 1'5073\end{gathered}$$

Podemos ordenar los ratios de mayor a menor, también podemos obtener la proporción de los 3 ratios y su proporción acumulada, como los del cuadro 3.2 .

	Ratio	Proporción	Proporción Acumulada
$GainR(Sexo\|EVP_1)$	1'9906	36'53 %	36'53 %
$GainR(Formacion, EVP_1)$	1'9511	35'80 %	72'33 %
$GainR(EVP_0, EVP_1)$	1'5073	27'66 %	100 %
Total	5'4491	100 %	

Cuadro 3.2: Comparación de los ratios de sexo, formación del año anterior, EVP del año anterior y año actual.

Entre los parámetros que se han definido para cada uno de los modelos tenemos el número máximo de nodos y la ratio de ganancia; para este ejemplo podemos definir una ratio de ganancia del 75 % y un máximo número de nodos de 2. Por el criterio de ratio de ganancia se deberían usar los 3 nodos del periodo anterior, enlazados al nodo de EVP del año actual, pero como se ha puesto un límite máximo de 2 nodos, solo estarán enlazados los nodos de sexo y formación_periodo_anterior con el EVP del año actual. La red de estas 4 variables quedaría como se muestra en la figura 3.2.

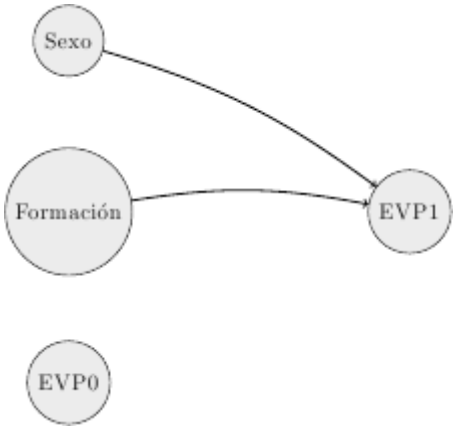

Figura 3.2: Conexión de los nodos del ejemplo de usar las variables de sexo, formación_periodo_anterior, EVP_periodo_anterior y EVP_periodo_actual aplicando el criterio de ratio de ganancia de información.

3.2.2. Obtención de la probabilidad de los nodos

Retomando el ejemplo simplificado en el punto anterior, las probabilidades de los nodos sexo, formación_periodo_anterior y EVP_periodo_anterior las podemos obtener directamente de los datos que nos proporciona el cuadro 3.2, como se muestra en el cuadro 3.3.

		Total	Probabilidad
Sexo	H	631	$P(Sexo = H) = \frac{631}{1148} = 0'5597$
	M	517	$P(Sexo = M) = \frac{517}{1148} = 0'4503$
	Total	1148	

(a)

		Total	Probabilidad
Formación año anterior	Suficiente	541	$P(Formación = Suficiente) = \frac{541}{1148} = 0'4713$
	Insuficiente	607	$P(Formación = Insuficiente) = \frac{607}{1148} = 0'5287$
	Total	1148	

(b)

		Total	Probabilidad
EVP año anterior	Alto	333	$P(EVP_0 = Alto) = \frac{333}{1148} = 0'2900$
	Bajo	815	$P(EVP_0 = Alto) = \frac{815}{1148} = 0'7100$
	Total	1148	

(c)

Cuadro 3.3: Probabilidades de (a) sexo, (b) formació_periodo_anterior y (c) EVP_periodo_anterior.

En el caso de la EVP del año actual se debe hacer el conteo de todas las combinaciones que hay entre las variables de sexo y formación del año anterior y a partir de ahí estimar la probabilidad para cada uno de los valores que puede tomar la EVP del año actual, tal como se muestra en el cuadro 3.4.

		EVP		
Sexo	Formación	Alto	Bajo	Total
H	Insuficiente	129	188	317
H	Suficiente	197	117	314
M	Insuficiente	119	170	289
M	Suficiente	165	62	227

(a)

		$P(EVP_1 \mid Sexo, Formacion)$	
Sexo	Formación	Alto	Bajo
H	Insuficiente	$\frac{129}{317} = 0'4069$	$\frac{188}{317} = 0'5931$
H	Suficiente	$\frac{197}{314} = 0'6274$	$\frac{117}{314} = 0'3726$
M	Insuficiente	$\frac{119}{289} = 0'4118$	$\frac{170}{289} = 0'5882$
M	Suficiente	$\frac{165}{227} = 0'7269$	$\frac{62}{227} = 0'2731$

(b)

Cuadro 3.4: Tablas de (a) frecuencias y (b) probabilidades condicionadas de EVP_periodo_actual dados el sexo y la formación_periodo_anterior.

Puede ocurrir que alguna de las combinaciones tenga una frecuencia de cero, o lo que es peor, que toda una combinación de datos no exista y tengamos una probabilidad de 0 en toda una combinación. Para resolver este problema se aplica la corrección de Laplace (Huang, 2017), que en todos los modelos que se han creado ha sido de $0'75$. Esta corrección o suavizado se define así: si para d observaciones $X = (x_1, x_2, ..., x_d)$, donde tenemos sus respectivas probabilidad $P = (P(X_1 = x_1), P(X_2 = x_2), ..., P(X_d = x_d)$; una versión que corrige los valores nulos:

$$\hat{P}_i = \frac{x_i + \alpha}{N + d \cdot \alpha} \; (i = 1, \ldots, d)$$

donde α es el valor de la corrección de Laplace. Aplicando esta corrección a nuestro ejemplo se obtendrán los resultados del cuadro 3.5.

		$\hat{P}(EVP_1 \mid Sexo, Formacion)$	
Sexo	Formación	Alto	Bajo
H	Insuficiente	$\frac{129+0'75}{317+2\cdot 0'75} = 0'4074$	$\frac{188+0'75}{317+2\cdot 0'75} = 0'5926$
H	Suficiente	$\frac{197+0'75}{314+2\cdot 0'75} = 0'6268$	$\frac{117+0'75}{314+2\cdot 0'75} = 0'3732$
M	Insuficiente	$\frac{119+0'75}{289+2\cdot 0'75} = 0'4122$	$\frac{170+0'75}{289+2\cdot 0'75} = 0'5878$
M	Suficiente	$\frac{165+0'75}{227+2\cdot 0'75} = 0'7253$	$\frac{62+0'75}{227+2\cdot 0'75} = 0'2746$

Cuadro 3.5: Probabilidades EVP del año actual dado el sexo y la formación del año anterior aplicando la corrección de Laplace.

Una de las cuestiones que se debe tomar en cuenta es el tiempo que se tarda en calcular cada una de las probabilidades, que crece de manera exponencial con el número de nodos involucrados, como se muestra en la figura 3.3.

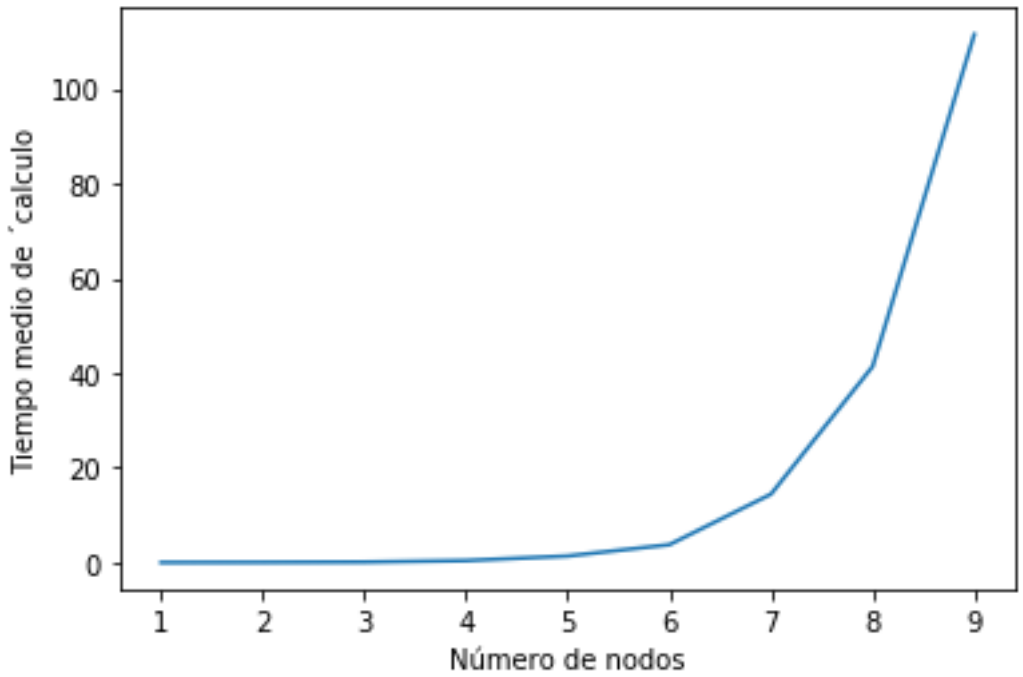

Figura 3.3: Tiempo medio (en minutos) de la estimación de las tablas de probabilidad en función del número de nodos en los modelos generados.

El trabajo más costoso (en tiempo) es el que se explica en este capítulo, ya que al final lo que se obtiene para cada combinación de parámetros es un fichero XML. Si retomamos el ejemplo simplificado de 4 nodos que se ha usado para explicar la obtención de los enlaces y las probabilidades, el programa (en Python) genera el fichero de texto que se puede descargar del siguiente enlace: https://github.com/smontesv/tfm/blob/master/Ejemplo_Simplificado.pgmx. En el siguiente capítulo se explicará la forma en que se evalúa y selecciona el modelo definitivo para aplicar la toma de decisiones.

3.2.3. Almacenamiento y reutilización de las probabilidades

La creación de redes probabilistas implica calcular las probabilidades que relacionan los nodos. En los modelos que se van a generar, las variables son discretas, por lo que todas las probabilidades condicionadas se pueden representar en forma de tablas. A priori, podemos pensar que si lo que va a variar son las conexiones entre los nodos, sin importar si es un modelo de Markov o un diagrama de influencia, algunas de esas probabilidades se pueden reutilizar en diferentes modelos. De hecho, parte de la solución de este problema es crear una tabla en la base de datos donde se van a ir almacenando las diferentes probabilidades. Esta tabla contendrá:

- Un campo que nos indique para qué partición se están calculando las probabilidades.
- Un campo de tipo texto que nos indique de qué nodos se están calculando las probabilidades.

- Un campo de tipo texto para guardar el resultado numérico de esa tabla que componen las diferentes probabilidades obtenidas.
- Un campo para almacenar el nombre del modelo que originó el cálculo de esa probabilidad.
- Unos campos para registrar las horas de inicio y fin del cálculo de esa probabilidad.
- Un campo para almacenar el tiempo en que se realizó el cálculo.[9]

Como dato interesante, también se crea una tabla adicional para registrar qué modelos utilizan las probabilidades de la anterior tabla.

[9]Estos últimos 3 campos se usarán para saber el tiempo que lleva implementar los modelos que queremos aplicar. Además de estos campos, se agregó un campo de tipo entero para indicar el número de la partición de esa probabilidad. El motivo de particionar la información en varios registros es que se utilizó la base de datos MySQL con una versión de libre distribución que tiene ciertas limitaciones para un tamaño de texto demasiado grande.

Capítulo 4

Análisis de resultados

"El verdadero genio reside en la capacidad para la evaluación de la información incierta, peligrosa y en conflicto."

Winston Churchill

En el capítulo anterior se ha explicado cómo generar los modelos a partir de la información de un conjunto de datos, ciertos parámetros y un programa hecho en Python. Al final se generaron 40 diferentes modelos con 10 particiones, con lo cual tenemos 400 redes para evaluar. En este capítulo se explica la forma de evaluar el desempeño y la selección de los modelos para la toma de decisiones.

4.1. Evaluación cuantitativa

4.1.1. Prueba y obtención de los valores para evaluar los modelos

Como hemos indicado en la sección 3.1, la evaluación se va a hacer mediante la herramienta OpenMarkov. Se puede descargar la aplicación e ir evaluando manualmente cada una de las redes con los datos reales, pero al tener tantos archivos y registros, lo conveniente es utilizar la API de esta aplicación. La API está desarrollada en Java, por lo que puede escribir un pequeño algoritmo para probar cada uno de los modelos en ese lenguaje que se conecte directamente a la base de datos. Cada modelo se probara usando los registros que no se usaron en su creación. Es posible, por tanto, guardar los datos reales y las predicciones hechas por OpenMarkov para las variables de interés. El Algoritmo 4.1.1 contiene los pasos necesario para evaluar los modelos con los registros y guardar los resultados en la misma base de datos, recordando que:

- Los parámetros de cada modelo son el número máximo de campos a vincular, si se usan nodos del periodo actual, si se incluyen decisiones, el valor de corte de la ratio de ganancia, el valor de la corrección de Laplace, el tipo de diagrama y el número de la partición.

- Los datos del empleado para crear evidencias con la API de OpenMarkov son sexo, edad, familia, familia de puesto, antigüedad en la organización, salario bruto anual, descuento o aumento de salario, nivel educativo, formación, desempeño, EVP y estado; todos estos valores corresponden al año anterior al que va vamos a evaluar.
- Las variables de interés que se van a evaluar, también se sacarán de la base de datos de cada empleado para el año que se va a evaluar. Dichas variables son el estado, si hubo separación del puesto, si hubo abandono y si se finalizó el puesto de trabajo.

Algoritmo 1 Algoritmo de prueba de modelos y obtención de valor de variables de ínteres

```
1: Ruta ← Ruta donde se encuentran los ficheros XML
2: Conectarse a la base datos
3: Recuperar los registros de los parámetros que generan cada modelo y posicionarse en el
   primer registro.
4: repeat
5:     if Tipo Diagrama = diagrama de influencia then
6:         Con los parámetros ubicar el archivo en Ruta, que es el modelo a evaluar.
7:         Cargar el archivo a la API de OpenMarkov.
8:         Recuperar los registros de cada empleado donde:
9:             Filtrar por el número de partición asociada al modelo.
10:            Los datos de evidencia del año anterior del empleado.
11:            las variables de interés del año actual.
12:        repeat
13:            Con la API de OpenMarkov crear una nueva evidencia.
14:            Crear una nueva propagación con la API de OpenMarkov.
15:            Para cada variable de interés:
16:                Clasificarlas usando las probabilidades generadas por la propagación.
17:            Guardar en la base datos:
18:                Los valores reales del propio empleado.
19:                Los valores obtenidos por API de las variables de interés.
20:            Moverse al siguiente registro de empleados.
21:        until No es último registro de empleados
22:    end if
23:    Moverse al siguiente registro de parámetros.
24: until No es último registro de parámetros
```

4.1.2. Comparación de los valores de prueba con los valores reales

El nivel de exactitud (ACC) de cada **modelo** se define así (Raschka, 2018):

$$ACC = 1 - ERR$$

donde el error de predicción, ERR, se calcula como el valor esperado de la pérdida 0-1 sobre n ejemplos en un conjunto de datos S, y_i es la i-ésima etiqueta de clase verdadera y $\hat{y}i$ la i-ésima etiqueta de clase predicha:

	Real	
Estimado	TP = 55	FP = 13
	FN = 20	TN = 25

Cuadro 4.1: Matriz de confusión para el modelo que tiene 4 nodos como máximo y no incluye enlaces entre los nodos del periodo actual .

$$ERR_s = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

y la pérdida 0-1 se define como:

$$L(\hat{y}_i, y_i) = \begin{cases} 0 & \text{si} \hat{y}_i, = y_i \\ 1 & \text{si} \hat{y}_i, \neq y_i \end{cases}$$

Otra opción para evaluar el desempeño de cada uno de los modelos es usar el área bajo la curva a partir de la matriz de confusión (Hernández Orallo and Ferri Ramírez, 2004). En este análisis se recupera la tasa de verdaderos positivos ($TPR = TP/(TP + FN)$) y la tasa de falsos positivos ($FPR = FP/(FP + TN)$). Por ejemplo, en el cuadro 4.1 tenemos los resultados reales y estimados de un modelo que clasifica con datos con 1 ó 0, un verdadero positivo (TP) es aquel donde tanto el valor real como el estimado son 1; un falso positivo (FP) es cuando el valor real es 0 y el estimado 1; un falso negativo (FN) cuando el valor real es 1 y el estimado 0 y un verdadero negativo (TN) cuando ambos casos son iguales a 0.

Por tanto, las ratios obtenidas son:

$$TPR = \frac{TP}{TP+FN} = \frac{55}{55+20} = 0'733$$
$$FPR = \frac{FP}{FP+TN} = \frac{13}{13+25} = 0'342$$

Con los anteriores datos se puede dibujar una curva que va desde el punto (0,0) al (0'345,0'733) y de éste al punto (1,1). El punto (0,0) corresponde a un modelo que clasificaría a todos los casos como negativos y el (1,1) como a todos positivos. Cualquier punto situado en la diagonal que va de (0,0) a (1,1) corresponde a unmodelo que clasificara los casos de forma aleatoria. La situación perfecta se da cuando el modelo no tiene ningún falso positivo y todos los positivos se han clasificado correctamente; esto sería cubrir un área del 100 %. Se puede clasificar el desempeño de un modelo a partir de los siguientes intervalos (Hernández Orallo and Ferri Ramírez, 2004; Wikipedia, 2020a):

- [0'5] : modelo que clasifica al azar.
- [0'5,0'6) : test malo.
- [0'6, 0'75) : test mediocre.
- [0'75. 0'9) : test bueno.

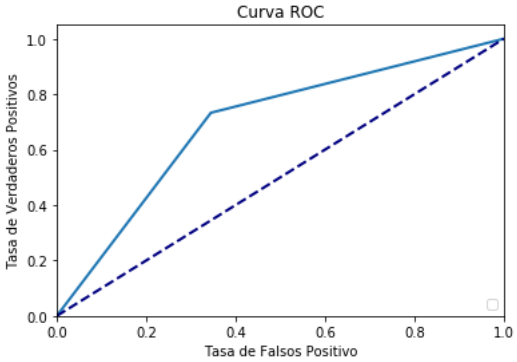

Figura 4.1: Curva ROC para la matriz de confusión del cuadro 6.1.

- [0'9, 0'97) : test muy bueno.
- [0'97,1] : test excelente.

En el ejemplo de la matriz de confusión del cuadro4.1, el área bajo la curva (véase la figura 4.1) es de aproximadamente 0'69, por lo que se puede considerar que es un modelo mediocre.

Y una tercera opción es usar un análisis binario, como $F1\ Score$, aplicando las siguientes formulas (Wikipedia, 2020c), con el número de verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN):

$$Precisión = \frac{TP}{TP + FP}$$

$$Exhaustividad = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2}{Precisión^{-1} + Exhaustividad^{-1}}$$
$$F_1 = 2 \cdot \frac{Precisión \cdot Exhaustividad}{Precisión + Exhaustividad}$$
$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Tener almacenados los valores predichos por el modelo y los valores reales de cada ejemplo nos permite obtener una estadística para medir la certeza y la fiabilidad de cada uno de los modelos. En la creación de los modelos incluíamos el parámetro *partición* como parte de la diferenciación de cada modelo, pero para el análisis este parámetro ya no se necesita. En lugar de comparar 400 modelos solo vamos a comparar 40; pero como solo nos interesa saber cuán acertados son los modelos,basta evaluar los diagramas de influencia, de modo queal final solo tenemos que evaluar 20 modelos.

Para cada uno de estos 20 modelos hemos obtenido un valor predicho para las variables de interés. Tres de ellas son de tipo booleano, pero al existir una gran probabilidad de que no haya abandono, separación, ni finalización del puesto, todos los modelos predicen "no" en todos los casos. Por tanto, seleccionar el modelo en función de estas variables no va a proporcionar

ninguna diferencia, pues todos tienen una exactitud del 0'963, pero un área bajo la curva de 0'5. En cambio, si nos fijamos en los estados, y muy particularmente en ese estado donde el desempeño es alto pero el EVP es bajo (DES+/EVP-), la cuestión es distinta.

Escoger este estado es interesante, pues es aquí donde se encuentran gran parte del conjunto de empleados que le dan valor a la organización, pero también donde los empleados que la valoran mal. Para un empleado que se encuentra en el estado en que su desempeño es bajo y su EVP también (DES-/EVP-), lo más saludable para amabas partes es separarlo de su puesto. Pero cuando se trata de un caso donde el empleado está en DES+/EVP-, tomar la decisión de separarlo de su puesto puede implicar que la organización pierda valor, que el propio empleado abandone la empresa, que contagie su desánimo al resto de compañeros, creando un ambiente tóxico, o que pase a tener un mal desempeño. Saber detectar a tiempo qué empleados podrían pasar a este estado es anticiparse a que se tenga que tomar la decisión de separar al empleado de su puesto o que sea el empleado el que tome la decisión de abandonar.

Además del estado DES+/EVP-, el sistema cuenta con otros 5 estados. Por tanto, para determinar el grado de exactitud y el área bajo la curva de cada uno de los modelos se hará con la siguiente casuística:

- Verdadero positivo: cuando el estado real y el estado predicho sean DES+/EVP-.
- Falso positivo: cuando el estado real sea diferente de DES+/EVP- y el estado predicho sea igual a DES+/EVP-.
- Verdadero negativo: cuando el estado real y el estado predicho sean diferentes de DES+/EVP-.
- Falso negativo: cuando el estado real sea igual a DES+/EVP- y el estado predicho sea diferente de DES+/EVP-.

Con este criterio podemos calcular el nivel de exactitud, el F1 score y el área bajo la curva de cada modelo, como se puede visualizar en el cuadro 4.2.

Del cuadro 4.2 se pueden sacar las siguientes conclusiones:

- Incluir el parámetro decisión no afecta ni al área bajo la curva ni a la exactitud.
- En los modelos que incluyen enlaces entre los nodos del año actual, el número de nodos que da mejores resultados en cuanto exactitud es el de 5 nodos.
- En los modelos que incluyen enlaces entre los nodos del año actual, el modelo que da mejores resultados es el de 8 nodos en cuanto F1 score, pues tiene un valor ligeramente mejor que los modelos de 5 nodos. Dado que es más manejable usar 5 en lugar de 8 nodos y el F1 score es muy parecido en ambos casos, optamos por usar el modelo de 5 nodos.

Núm.	Nodos	I.E.1	I.D.	TP	TN	FP	FN	TPR	FPR	AUC	ACC	F1 score
0	4	1	1	560	251	132	204	0'733	0'345	0'694	0'708	0.7692
1	5	1	1	638	194	189	126	0'835	0'494	0'671	**0'725**	*0.8020*
2	6	1	1	663	141	242	101	0'868	0'632	0'618	0'701	0.7945
3	7	1	1	687	106	277	77	0'899	0'723	0'588	0'691	0.7951
4	8	1	1	738	46	337	26	0'966	0'880	0'543	0'684	**0.8026**
5	4	1	0	560	251	132	204	0'733	0'345	0'694	0'707	0.7692
6	5	1	0	638	194	189	126	0'835	0'494	0'671	**0'725**	*0.8020*
7	6	1	0	663	141	242	101	0'868	0'632	0'618	0'701	0.7945
8	7	1	0	687	106	277	77	0'899	0'723	0'588	0'691	0.7951
9	8	1	0	738	46	337	26	0'966	0'890	0'543	0'684	**0.8026**
10	4	0	1	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
11	5	0	1	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
12	6	0	1	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
13	7	0	1	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
14	8	0	1	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
15	4	0	0	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
16	5	0	0	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
17	6	0	0	484	331	52	280	0.634	0.136	**0'749**	0'711	0.7446
18	7	0	0	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446
19	8	0	0	484	331	52	280	0'634	0'136	**0'749**	0'711	0.7446

Donde: Nodos = número máximos de nodos a enlazar en el modelo, es decir, 1 significa que incluye nodos del periodo actual;
I.D. = incluye decisión; TP = número de verdaderos positivos; TN = número de verdaderos negativos;
FP = número de falsos positivos; FN = número de falsos negativos; TPR = tasa de verdaderos positivos
FPR = tasa de falsos positivos; AUC = área bajo la curva; ACC = exactitud

Cuadro 4.2: Evaluación de los modelos a partir de la variable Estado.

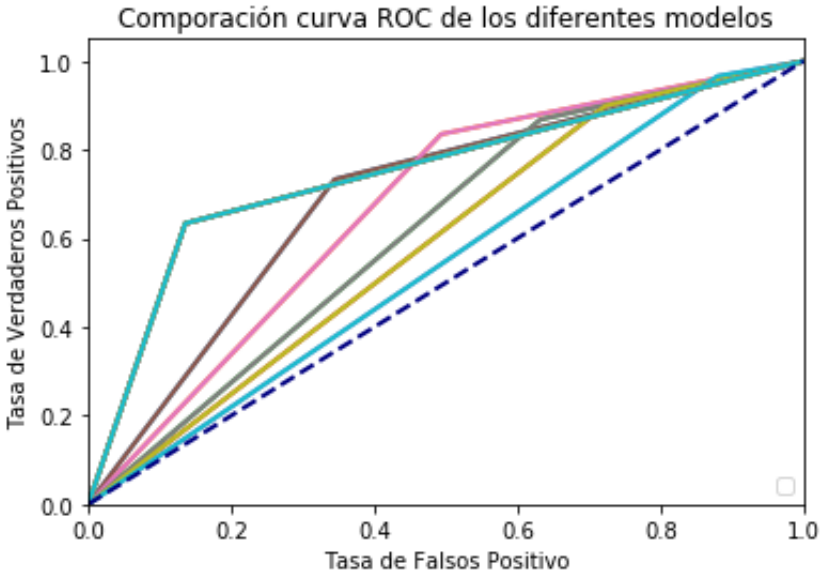

Figura 4.2: Curvas ROC para los modelos evaluados.

- En el caso de no incluir enlaces entre los nodos del año actual, el número de nodos no afecta al resultado. Por ello, para evitar tener que hacer muchos cálculos, lo más adecuado es usar el modelo de 4 nodos.

- Si se toma como criterio de evaluación el área bajo la curva, los modelos que no incluyen enlaces entre los nodos del año actual, son los que dan mejores resultados. Sin embargo, todos los modelos están dentro del rango de un test mediocre, pues ninguno pasa de un valor del 0'75; solo los mejores resultados están a una centésima de poder considerarse buenos. Se hizo un análisis de todos los registros para cada uno de los modelos en cuanto a la forma en que los clasificó, y en todos los casos los registros siempre son clasificados igual en todos los modelos, por lo que aumentar el número de nodos no aporta valor a los resultados.

Con estos datos vamos a escoger dos tipos modelos para comparar los resultados:

Tipo A: modelos que incluyen enlaces entre los nodos del año actual, con 5 nodos.

Tipo B: modelos que no los incluyen, con 4 nodos.

4.1.3. Comparación de modelos mediante la prueba de McNemar

Ahora vamos a comparar mediante pruebas de hipótesis estadísticas el rendimiento de estos dos modelos de aprendizaje automático a partir de los resultados de las predicciones hechas por cada modelo, con el fin de aceptar o rechazar la hipótesis de que el conjunto de resultados de un modelo es diferente del de otro modelo (Brownlee, 2018a; Raschka, 2018).

En primero test que aplicamos es la prueba de McNemar (Brownlee, 2018b; Dietterich, 1998), introducida por Quinn McNemar en 1947, que es recomendad por Thomas Dietterich

		Modelo 2	
		Correctos	Incorrectos
Modelo 1	Correctos	a	b
	Incorrectos	c	d

Cuadro 4.3: Estructura tabla contingencia.

en aquellos casos en los que es costoso o poco práctico entrenar múltiples copias de modelos clasificadores. El procedimiento es el siguiente: a partir de los resultados obtenidos con ambos clasificadores se crea una "tabla de contingencia" que es muy parecida a una matriz de confusión, pues en el caso de los modelos de aprendizaje automático los resultados obtenidos por ambos modelos se comparan con los valores reales y se suman en una tabla, como se muestra en el cuadro 4.3.

El estadístico que se calcula esn χ^2 (chi-cuadrado) con un grado de libertad de la siguiente forma:

$$\chi^2 = \frac{(b-c)^2}{(b+c)}$$

Podemos ver que solo se usan dos elementos de la tabla de contingencia, específicamente que los elementos b y c no se usan en el cálculo de la estadística de prueba. Como tal, podemos ver que este estadístico compara las diferentes predicciones correctas o incorrectas de los dos modelos, no la precisión ni las tasas de error, lo cual sería una limitación. La hipótesis nula, H_0, es que los dos clasificadores tienen la misma tasa de error. Si se rechaza la hipótesis nula, sugiere que hay evidencia que sugiere que los casos tienen diferentes proporción de error

Para cada nivel de significación, α, el valor p calculado por la prueba se puede interpretar de la siguiente manera:

- $p > \alpha$: no se rechaza H_0, es decir, no se observa una diferencia significativa entre los dos modelos predictivos.

- $p \leq \alpha$: se rechaza H_0, es decir, hay diferencia significativa entre los dos modelos predictivos.

La prueba de McNemar es aplicable con cualquier tamaño de muestra, pero si tiene un tamaño bajo, entonces no es apropiado aproximar la distribución nula del estadístico de prueba por una distribución χ^2, sino que para hacer la prueba se construye la tabla de contingencia (cuadro 4.4) recuperando de la base de datos los resultados de evaluar cada prueba; en Python se puede usar la prueba **mcnemar** del paquete **mlxtend**. En nuestro ejemplo, el valor que se obtiene de la tabla de contingencia es $\chi^2 = 130$, con lo que nos da un valor de $p = 0'1722$. Si lo comparamos con $\alpha = 0'05$, se tiene que aceptar la hipótesis nula, es decir, que existe una proporción similar de errores entre ambos modelos.

		Modelo B	
		Correctos	Incorrectos
Modelo A	Correctos	661	154
	Incorrectos	130	202

Cuadro 4.4: Tabla contingencia de los modelos A y B.

4.1.4. Coste de un falso positivo vs. coste de un falso negativo

En los casos en que el modelopredice que la separación o abandono va a ocurrir y esta no ocurre, es decir, se da un falso positivo, no existe un costo como tal. Y aunque se tratase de hacer previsión monetaria y ese recurso no se utiliza es una buena noticia para la organización. El caso contrario —es decir, cuando el modelo predice que un empleado va a permanecer y al final ocurre la separación o el abandono— tiene un gran costo, y el menor mal de ellos es no haber hecho la previsión económica legal de esta separación. Cuando un empleado llega a un punto de separación o abandono es que previamente hubo un desgaste en la relación entre la organización y el empleado, se pierde conocimiento, se pierde tiempo, otros empleados también pueden ser afectados, entre otras muchas cosas.

Por eso es importante prever esta situación y poner la atención en esos empleados que se ubican en el estado DES+/EVP-, pues esos son los empleados que más costo van a tener en el caso de que ocurra la separación o el abandono. Sobre este estado es donde se deben tomar decisiones para que el empleado aumente el valor de su EVP y que repercuta en beneficio de la organización y no solo sobre el empleado.

4.2. Análisis de casos

Una vez que se han creado los ficheros XML para los modelos A y B, se procede a utilizar la aplicación OpenMarkov, cargando los ficheros como se muestra en las figuras 4.3 y 4.4. En ellas se puede ver qque ya se ha hecho la inferencia y se han calculado las respectivas probabilidades y los costos asociados. Las variables de interés son las de "¿es abandono?", "¿es separación?", "¿fin de puesto?" y "estado", además de una estimación de la utilidad esperada, que en este caso refleja cuál sería el costo esperado si un empleado causa baja en la organización. En el cuadro 4.5 se muestra el valor de la probabilidad de las variables de interés cuando toman el valor de 1, es decir, cuando el evento ocurre. En el caso de la variable que indica el estado, se han dejado todos los valores que puede tomar esta variable.

En el cuadro 4.5 podemos ver que en ambos modelos las variables "¿es abandono?", "¿es separación?" y "¿fin de puesto?" alcanzan una probabilidad que resultaría muy preocupante para la organización. Por ejemplo, el modelo A nos dice que la probabilidad de que un empleado de la organización decida abandonar a la empresa el próximo año es del 42'62 %, lo cual sería un desastre para la empresa. En la anterior sección vimos que ninguno de los modelos generados

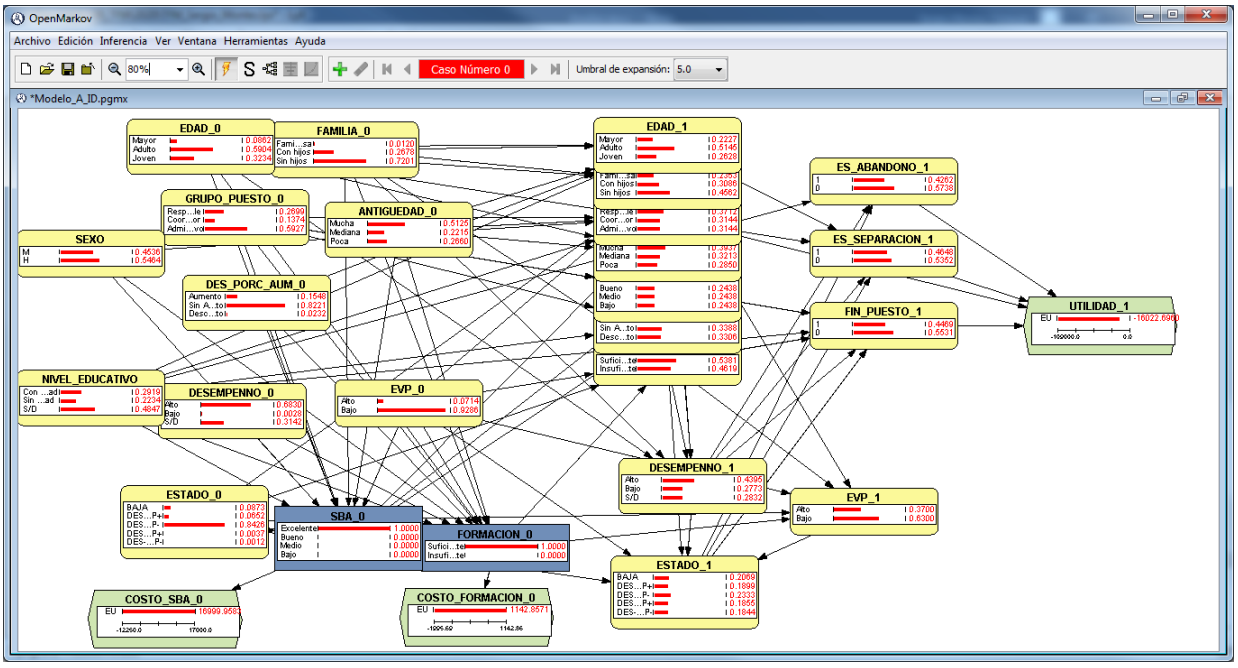

Figura 4.3: Modelo A en modo inferencia en OpenMarkov.

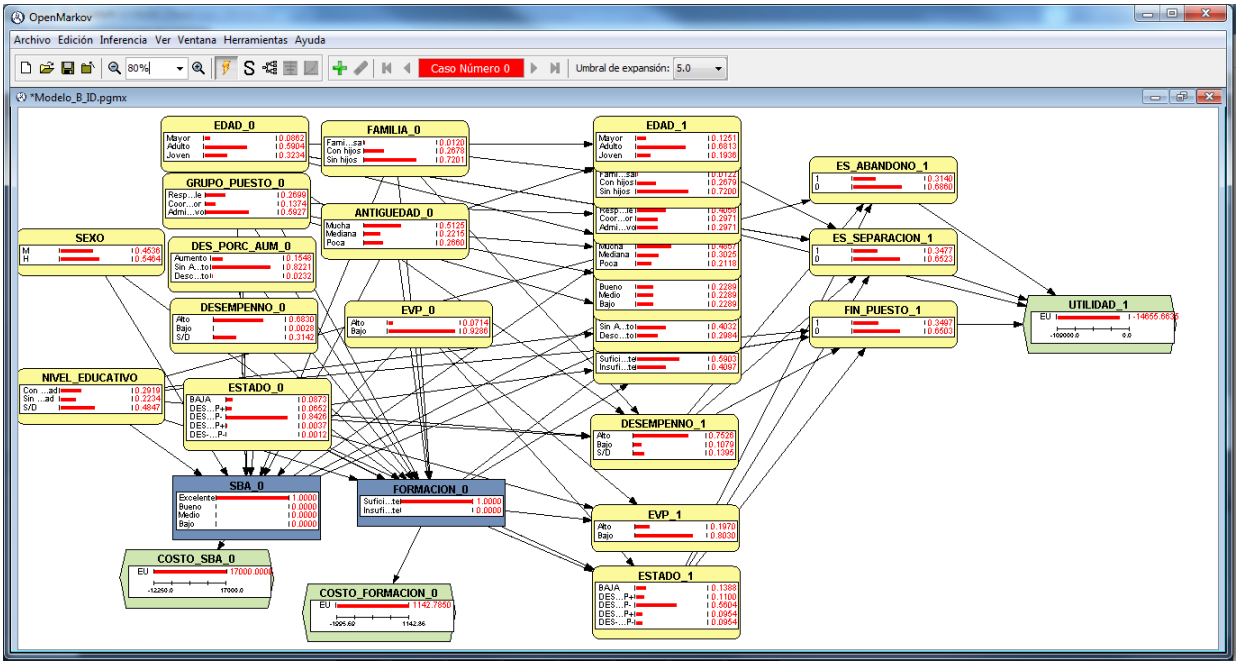

Figura 4.4: Modelo B en modo inferencia en OpenMarkov.

Modelo	A	B
¿Es abandono?	42'62 %	31'40 %
¿Es separación?	46'48 %	34'77 %
¿Fin de puesto?	44'69 %	34'97 %
Baja	20'69 %	13'88 %
DES+/EVP+	18'99 %	11'00 %
DES+/EVP-	23'33 %	56'04 %
DES-/EVP+	18'55 %	9'54 %
DES-/EVP-	18'44 %	9'54 %
Utilidad esperada	-16.022€	-14.655€

Cuadro 4.5: Comparación de las probabalidades y la utilidad para los modelos A y B.

Año	Ratio Baja
2014	1'7021 %
2015	4'1494 %
2016	8'9184 %
2017	6'5056 %

Cuadro 4.6: Ratio de bajas en la empresa ejemplo de los años del 2014 al 2017.

logró tener buenos resultados para estas variables, mientras queen el caso de la variable "estado" tuvimos resultados buenos.

Para la variable "estado" la probabilidad de baja en el modelo A es de 20'69 % y en el B del 13'88 %. La ratio de bajas de los últimos 4 años (de los cuales se ha obtenido información) se encuentra en el cuadro 4.6. Aquí se aprecia que el modelo B esta más cercano a ese valor. Sin embargo, lo que se aprecia en ambos modelos es que las condiciones que desencadenaron labaja en los anteriores años son las que se están presentando en el año actual y pueden desembocar en una rotación del más del 10 % el próximo año. Además los dos modelos predicen que el estado más probable para el empleado el próximo año es DES+/EVP-, justo lo que necesita la empresa detectar, pues es el talento que puede abandonar la empresa en los próximos años. La probabilidad de regreso el modelo A para este estado es más cercana a lo esperado que la del modelo B, pues el modelo A nos dice que puede ser del 23'33 % para el próximo año, lo cuales demasiado alto. En cambio,en el modelo B llega al 56'04 %, lo cual habla de una empresa tóxica, donde los empleados solo van a ganar un sueldo sin sentirse comprometidos ni con la empresa ni con el resto de sus compañeros.

Si estas mismas probabilidades las vemos como un gráfico de barras, también llama la atención que las probabilidades de tener un desempeño bajo, sin importar el EVP que tenga el empleado, son muy parecidas a la de tener un desempeño alto con un EVP alto. Es otro dato que tiene que tomar en cuenta la empresa si quiere ser competitiva.

Como se aprecia, el modelo A es el que mejor resultados nos da según el conocimiento que se tiene dentro de la empresa.

Ahora vamos a realizar una evaluación a partir de los dos casos ficticios que hemos presen-

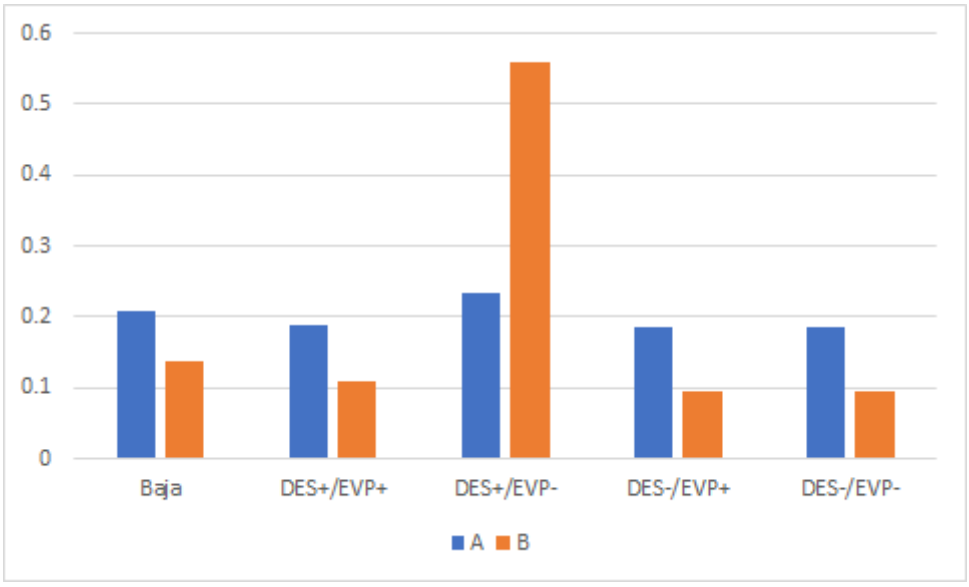

Figura 4.5: Comparación ratios en la variable de interés "estado" entre los modelos A y B.

Modelo	A	B
¿Es abandono?	26'05 %	8'51 %
¿Es separación?	31'88 %	0'91 %
¿Fin de puesto?	28'62 %	11'31 %
Baja	14'68 %	4'92 %
DES+/EVP+	8'83 %	0'39 %
DES+/EVP-	61'33 %	90'49 %
DES-/EVP+	7'78 %	0'51 %
DES-/EVP-	7'38 %	0'17 %
Utilidad esperada	-13.036,21€	-6.557,31€

Cuadro 4.7: Comparación de los modelos A y B para el caso de Andrés.

tado como ejemplos en el capítulo 2.

4.2.1. Caso de Andrés

Recordemos que Andrés es un empleado de 26 años que ha pasado de ser becario a programador junior en el último año; su sueldo bruto anual (SBA) pasó de 7.200 € a 18.000 €, la empresa no le proporcionó el curso que él quería, y el traslado de su casa al trabajo no le permite concluir sus estudios en la universidad. Su situación se puede representar en ambos modelos como se ve en la figura 4.6.

El resumen de las variables de interés de ambos modelos las tenemos en el cuadro 4.7. Vemos que este empleado no tiene una gran probabilidad de abandonar, pero la probabilidad que pase a ser parte del grupo de talento que está insatisfecho con la empresa es muy alto. La empresa solo tiene dos herramientas para cambiar esta situación: formación y SBA. Para este empleado, que ya tiene un SBA medio, como corresponde a su la experiencia, lo mejor que se

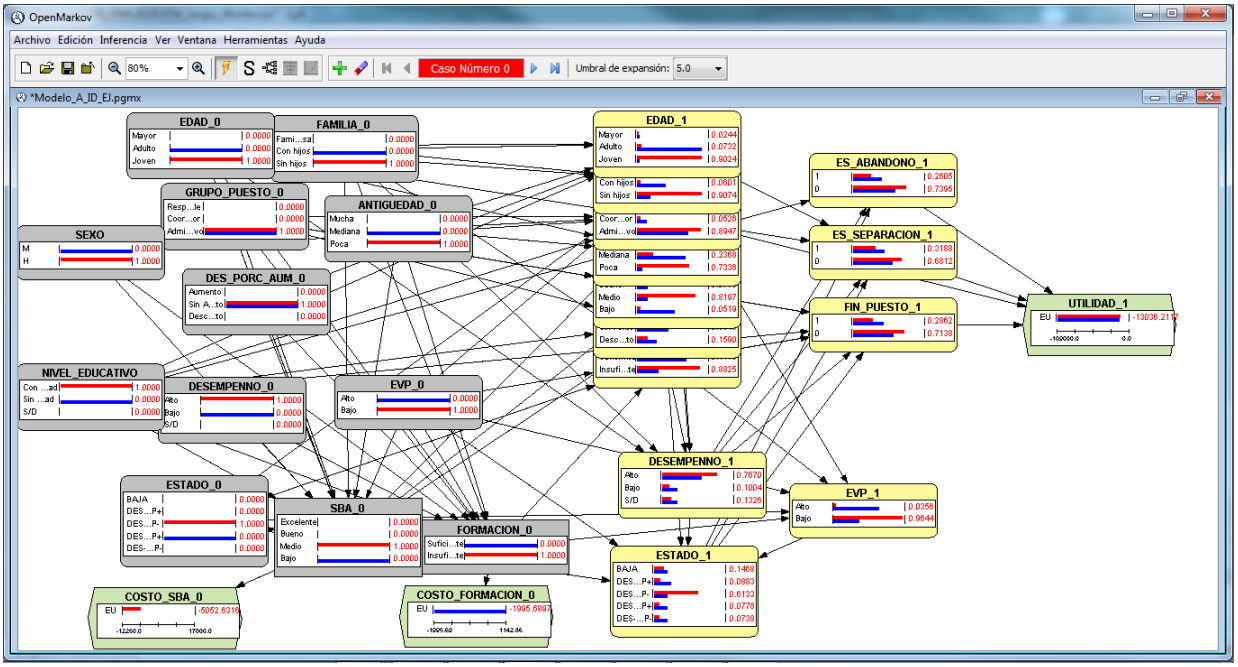

Figura 4.6: Caso de Andrés, en el modelo A.

Modelo	A			B		
Formación	Insuficiente	Suficiente	Diferencia	Insuficiente	Suficiente	Diferencia
¿Es abandono?	26'05 %	28'41 %	-2'36 %	8'51 %	11'03 %	-2'52 %
¿Es separación?	31'88 %	33'60 %	-1'72 %	9'10 %	11'58 %	-2'48 %
¿Fin de Puesto?	28'62 %	31'75 %	-3'13 %	11'31 %	12'15 %	-0'84 %
Baja	14'68 %	14'23 %	0'45 %	4'92 %	8'45 %	-3'53 %
DES+/EVP+	8'83 %	10'11 %	-1'28 %	0'39 %	2'68 %	1'22 %
DES+/EVP-	61'33 %	58'15 %	3'18 %	90'49 %	88'45 %	2'04 %
DES-/EVP+	7'78 %	8'88 %	-0'11 %	0'51 %	0'21 %	0'30 %
DES-/EVP-	7'38 %	8'64 %	-1'26	0'17 %	0'21 %	0'04 %
Utilidad esperada	-13.036,21€	-13.301,52€	265,31€	-6.557,31 %	-7.782,49€	1.225,19€

Cuadro 4.8: Comparación de los modelos A y B, variando la formación de Andrés.

puede hacer es darle la formación que solicita. Este cambio lo podemos observar en el cuadro 4.8.

Aquí vemos que en ambos modelos la probabilidad de que sea baja o que se encuentre en el estado DES+/EVP- disminuyen; concretamente, en el modelo A disminuye unos 3 puntos porcentuales. También aumenta en este modelo la probabilidad de que el empleado pase al estado DES+/EVP+, mientras que en el modelo B baja ligeramente. Sin embargo, ambos modelos indican que darle al empleado la formación que quiere puede repercutir positivamente a la empresa.

En el cuadro 4.9 podemos ver qué pasaría en el caso de afectar al empleado de forma negativa, bajando el SBA de medio a bajo.

Lo que vemos es que las probabilidades no han variado, lo que se explica porque este

Modelo	A			B		
SBA	Medio	Bajo	Diferencia	Medio	Bajo	Diferencia
¿Es abandono?	26'05 %	26'05 %	0'00 %	8'51 %	8'51 %	0'00 %
¿Es separación?	31'88 %	31'88 %	0'00 %	9'10 %	9'10 %	0'00 %
¿Fin de puesto?	28'62 %	28'62 %	0'00 %	11'31 %	11'31 %	0'00 %
Baja	14'68 %	14'68 %	0'00 %	4'92 %	4'92 %	0'00 %
DES+/EVP+	8'83 %	8'83 %	0'00 %	0'39 %	0'39 %	0'00 %
DES+/EVP-	61'33 %	61'33 %	0'00 %	90'49 %	90'49 %	0'00 %
DES-/EVP+	7'78 %	7'78 %	0'00 %	0'51 %	0'51 %	0'00 %
DES-/EVP-	7'38 %	7'38 %	0'00 %	0'17 %	0'17 %	0'00 %
Utilidad esperada	-13.036,21€	-13.479,50€	443,29€	-6.557,31 %	-6.568,17 %	10,87€

Cuadro 4.9: Comparación de los modelos A y B, variando el SBA de Andrés.

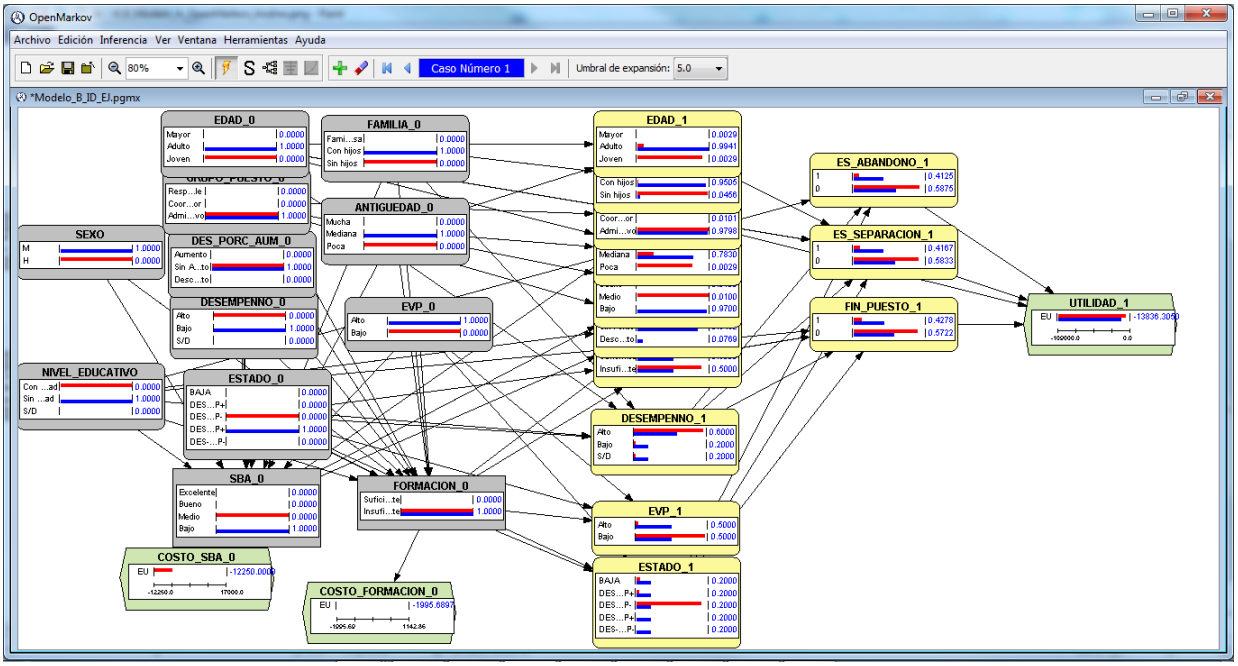

Figura 4.7: Caso de Beatriz en el modelo B.

empleado valora más la experiencia y el conocimiento que el SBA.

4.2.2. Caso de Beatriz

El otro caso que usamos de ejemplo fue el de Beatriz, una madre soltera con una hija de 10 años; trabaja en la empresa desde hace 16 años como administrativa, no ha tenido un ascenso ni subida salarial en los últimos 5 años, pero tiene cierto equilibrio entre vida familiar y trabajo. Haciendo el mismo ejercicio que con Andrés, primero comparamos los resultados de los modelos A y B. En la la figura 4.7 vemos la situación de Beatriz, usando ahora el modelo B, y en el cuadro 4.10 el resumen de los dos modelos.

Observando las probabilidades obtenidas con ambos modelos, vemos que Beatriz es un caso típico de empleados que llevan muchos años en la empresa, se sienten parte de ella y van a ser

Modelo	A	B
¿Es abandono?	58'84 %	35'82 %
¿Es separación?	44'14 %	36'78 %
¿Fin de puesto?	42'79 %	37'87 %
Baja	19'59 %	20'00 %
DES+/EVP+	24'12 %	60'00 %
DES+/EVP-	18'77 %	6'67 %
DES-/EVP+	18'77 %	6'67 %
DES-/EVP-	18'77 %	6'67 %
Utilidad esperada	-14.828,21€	-13.332,90€

Cuadro 4.10: Comparación de los modelos A y B para el caso de Beatriz.

Modelo	A			B		
SBA	Bajo	Medio	Diferencia	Bajo	Medio	Diferencia
¿Es abandono?	58'84 %	41'16	17'68 %	35'82 %	35'82 %	0'00 %
¿Es separación?	44'14 %	44'14 %	0'00 %	36'78 %	36'78 %	0'00 %
¿Fin de Puesto?	42'79 %	42'79 %	0'00 %	37'87 %	37'87 %	0'00 %
Baja	19'59 %	19'59 %	0'00 %	20'00 %	20'00 %	0'00 %
DES+/EVP+	24'12 %	24'12 %	0'00 %	60'00 %	60'00 %	0'00 %
DES+/EVP-	18'77 %	18'77 %	0'00 %	6'67 %	6'67 %	0'00 %
DES-/EVP+	18'77 %	18'77 %	0'00 %	6'67 %	6'67 %	0'00 %
DES-/EVP-	18'77 %	18'77 %	0'00 %	6'67 %	6'67 %	0'00 %
Utilidad esperada	-14.828,21€	-14.267,71€	-560,50€	-13.332,90€	-13.313,27€	-19,63€

Cuadro 4.11: Comparación de los modelos A y B variando el SBA de Beatriz.

productivos y felices mientras se encuentren en ella. Estas probabilidades nos sirven para poder ver que ninguno de los dos modelos predice bien las variables de interés "¿es abandono?", "¿es separación?" y "¿fin de puesto?", sino que en el caso de Beatriz esas variables dan resultados completamente erróneos. En cambio, para la variable "estado" nos dan un resultado muy cercano a las circunstancias de la empleada; la probabilidad de causar baja el siguiente año tiene que ver con situaciones externas que no se pueden controlara desde el modelo[1]. Vamos a intentar experimentar con el SBA de Beatriz, pasándolo de bajo a medio; los resultados se muestran en el cuadro 4.11.

Vemos es que las probabilidades apenas han variado para este caso, lo que se explica porque esta empleada valora más la estabilidad y el equilibrio familiar. Otra explicación es que casi no hay casos en que bajar el suelo a un empleado repercuta en las variables de interés del año siguiente. Veamos que pasa en el sentido negativo, es decir, pasar de darle una formación suficiente a una insuficiente (cf. cuadro 4.12).

En el modelo A se refleja mejor el escenario esperado: baja la probabilidad de que la empleada se encuentre en el estado DES+/EVP+ y aumenta la de los demás estados, pero no

[1]No se pueden controlar o medir porque no tenemos dentro del modelo la información que nos ayude a detectar que variables externas a la empresa pueden influir en que Beatriz cause baja el siguiente año.

Modelo	A			B		
Formación	Suficiente	Insuficiente	Diferencia	Suficiente	Insuficiente	Diferencia
¿Es abandono?	58'84 %	41'42 %	17'42 %	35'82 %	41'25 %	-5'43 %
¿Es separación?	44'14 %	43'99 %	0'15 %	36'78 %	41'67 %	-4'89 %
¿Fin de puesto?	42'79 %	42'93 %	-0'14 %	37'87 %	42'78 %	-4'91 %
Baja	19'59 %	20'14 %	-0'55 %	20'00 %	20'00 %	0'00 %
DES+/EVP+	24'12 %	21'73 %	2'39 %	60'00 %	20'00 %	40'00 %
DES+/EVP-	18'77 %	19'38 %	-0'61 %	6'67 %	20'00 %	-13'33 %
DES-/EVP+	18'77 %	19'38 %	-0'61 %	6'67 %	20'00 %	-13'33 %
DES-/EVP-	18'77 %	19'38 %	-0'61 %	6'67 %	20'00 %	-13'33 %
Utilidad esperada	-14.828,21€	-14.776,42€	-51,79€	-13.332,90€	-13.816,27	483,36€

Cuadro 4.12: Comparación de los modelos A y B variando la formación para el caso de Beatriz.

en una proporción que pueda resultar alarmante para la empresa. Ello podría llevar a la empresa a tomar ciertas políticas para esta clase de empleados, dándoles menos formación para abaratar costos, sabiendo que el impacto en cuanto a bajas va a ser mínimo el siguiente año. Finalmente, vemos que el modelo B lo podríamos rechazar y solo quedarnos con el modelo A. Dadas las probabilidades que tenemos para la variable de interés "estado", al hacer el cambio en valor de la formación, el modelo B distribuyó todas las probabilidades de forma homogénea, lo cual no debería ocurrir si tuviéramos los enlaces entre los nodos bien definidos.

Capítulo 5

Discusión y conclusiones

"*Mittler zwischen Hirn und Hand muss das Herz sein*." ("*Mediador entre el cerebro y la mano ha de ser el corazón*.")

Fritz Lang en su película Metrópolis, Alemania 1927

5.1. Discusión

Una vez que se han generado los modelos y se ha comprobado que el mejor de ellos es aquel que incluye enlaces entre los nodos del año actual, con 5 nodos como máximo en las relaciones entre nodos, vemos que este modelo final sirve para predecir la variable de interés "estado". No es un resultado tan negativo si tenemos en cuenta que uno de los estados de esa variable es "baja", que con el análisis de casos y el propio conocimiento que se tiene sobre la rotación de la empresa. Esta herramienta puede ser ideal para predecir en qué estado se va a encontrar cada empleado al año siguiente.

La probabilidad general de que los empleados sean baja el siguiente año es un reflejo de la situación en que vivía la empresa en el momento en que se tomaron los datos de nuestro estudio: la mayoría de los empleados se encontraban en un estado de DES+/EVP- (alto desempeño y baja motivación). Recordemos que este estado es el precisamente el que más interesa convertir a DES+/EVP+. La empresa solo proporciona dos políticas para cambiar esa situación: formación y SBA. La primera de ellas es la única que puede tener un efecto positivo, aunque tal vez no lo suficiente para que los empleados salgan del estado DES+/EVP-. Saber esto también es útil, pues es una forma de que la empresa añada más políticas para mejorar el EVP.

5.2. Consideraciones éticas

En esta sección vamos a aplicar las consideraciones de las cinco C's (Kosta Loukides and Patil, 2018), pues constituyen una buena guía para aquellas personas que han desempeñado

su vida profesional y académica en el área de las tecnologías de la información, más adecuada que en el de humanidades:

Consent (Consentimiento): se refiere al consentimiento explícito que existe entre el proveedor de la información (el empleado) y el usuario de esta información (la organización). Dada la relación entre ambos, la organización, al recolectar la información dentro de su ERP, es dueña de esta información. A pesar de ello, algunas empresas ya piden autorización a sus empleados para recopilar información personal o sensible. Una ventaja de este modelo es que, al ser una herramienta de uso interno dentro de la organización[1], se puede construir con datos anonimizados.

Clarity (Claridad): se refiere a que el proveedor de la información (el empleado) sepa y entienda cuál va a ser el uso de dicha información. Esto se logra con un ejercicio de pedagogía donde el empleado sepa que el uso de los datos que recaba el área de RRHH no solo sirve para la gestión, sino también para beneficio del propio empleado. Este ejercicio sería ideal implantarlo en el curso de introducción o propedéutico que suelen dar las empresas a los nuevos empleados.

Consistency and Trust (Consistencia y Confianza): "la confianza requiere consistencia durante todo el tiempo". Este punto se refiere a que cuando se ha establecido un vínculo entre proveedor y usuario de la información, este último tiene que demostrar que su comportamiento siempre va a ser el mismo con respecto al uso que se ha dado de la información. Cualquier comportamiento anormal implica la rupturade dicha confianza. Este aspecto va más allá del modelo propuesto en el TFM, pues tiene que ser parte de las políticas de tratamientos de la información de datos de los ERPs de RRHH. Si no se hiciera así, podría tener consecuencias legales.

Control and Transparency (Control y Transparencia): este apartado se refiere al hecho de que el proveedor de la información no solo sepa el uso que se le va a dar, sino que además tenga control sobre ese uso. Este punto está ligado con el anterior, en el sentido que el uso de la información de los datos de los empleados es sensible, y un mal uso de ella conlleva consecuencias legales. El empleado sabe que hay control y transparencia cuando ve que los productos resultantes de procesar esa información llegan al empleado: pago de la nómina, planes de carreras, medición del ambiente laboral, evaluación de desempeño y aplicación del modelo predictivo para aumentar el EVP para cada empleado.

Consequences (Consecuencias): supuestamente la causa directa de que el empleado proporcione cierta información a la organización es porque ya existe una relación entre ellos, y el objetivo de esa información es crear valor para la organización. Como hemos visto, en la gestión de las organizaciones a partir de los valores, una forma de crear más valor es aumentar el EVP de cada uno de los empleados, que es lo que se busca en este modelo. La información

[1]Algunos ERPs ya se encuentran como SaaS (Software as a Service) en la nube. Esto quiere decir que se podrían construir modelos mediante aprendizaje automático a partir de la información de varias empresas, pero en una situación así sería necesario pedir un consentimiento no solo al empleado sino también a la propia empresa dueña de esa información.

debería ser de uso exclusivo para la organización, en los productos o servicios que proporciona el departamento de RRHH a los empleados. Cualquier otro uso que desconozca el empleado o la mayor parte de la organización sería una falta de ética.

5.3. Conclusiones

La primera conclusión es la relativa a los empleados como personas. Para las actuales generaciones laborales debe existir compatibilidad emocional con el trabajo. Esto implica alinear los valores de la organización con valores individuales de cada uno de los empleados, y eso es lo que mide el EVP, pues no solo es la forma en que un candidato puede evaluar su ingreso en la organización, sino que también es un indicador de cómo el empleado puede valorar su permanencia en la organización.

Para poder explicar las conclusiones a las que he llegado realizando este proyecto, voy a utilizar los pasos que propone Roberto Ley Borrás para la toma de decisiones (Ley Borrás, 2009):

1. **¿Qué situación conviene abordar?** La situación de la que ha tratado este TFM es la rotación de personal, una situación que para la empresa tiene un costo en tiempo, dinero e incluso en el propio ambiente laboral.

2. **¿Qué se desea lograr?** El objetivo principal es reducir el número de bajas (tanto por abandono como por separación) de los empleados de la organización, en especial de aquellos que por su conocimiento o desempeño dan valor a la organización.

3. **¿Cómo se puede lograr?** La propuesta de este TFM era utilizar diagramas de influencia, y aunque también se podrian utilizar MIDs, a partir de los datos que procesados y almacenados en el ERP del área de RRHH de una empresa que nos ha permitido acceder a ellos. Analizando estos datos mediante unprograma escrito en Python, hemos construidos los modelos y los hemos almacenado en el formato ProbModelXML, desarrollado en la UNED, y tanto con la API como con la aplicación de OpenMarkov, hemos explotado dichos modelos en la toma de decisiones individuales o grupales, para medir el impacto y evaluar políticas de retención de empleados.

4. **¿Qué factores incontrolables tienen impacto?** Uno de los factores que más pesan para saber si un empleado va a causar baja en una organización es la *propuesta de valor para el empleado* (EVP). Sin embargo, este factor pocas veces se mide. Una de las principales aportaciones de este TFM ha sido construir una fórmula para calcular el EVP de los empleados a partir de la información contenida en el ERP. Esta formula debe ser personalizada para cada organización, pero en el caso de que no se cuente con la opinión directa de cada empleado que mida su propio EVP. Otros factores difíciles de medir

en una organización son aquellos que vienen de fuera; por ejemplo, competencia que ofrece mejores condiciones a los empleados, situaciones familiares o eventos totalmente imprevisibles, como una pandemia que altera totalmente la economía mundial.

5. **¿Cómo se relacionan los elementos de la situación?** La forma de relacionar los datos del EPR con la situación esperada de cada empleado ha sido construir un modelo de Markov con los 5 posibles estados por los que puede transitar cada empleado. Hemos visto que nuestro modelo, construido con los datos disponibles, no es adecuado para predecir ciertas variables de interés, como "¿es abandono?" o "es separación?". En cambio,para la variable "estado", que es la de mayor relevanci, nuestro modelo de Markov ha dado resultados buenos o aceptables, suficientes para describir la situación de los empleados y útil para saber si las políticas de la empresa pueden tener impacto positivo o negativo sobre la permanencia del empleado dentro de la empresa. El modelo de Markov y el diagrama de influencia que hemos contruido permiten tomar rápidamente decisiones sobre esta situación.

6. **¿Cuál es el valor de cada estrategia?** El valor directo de cada estrategia consiste en reducir el costo de que un empleado cause baja. Hemos hecho una estimación de cual sería ese costo en términos monetarios, pero el costo real en términos de tiempo y en el ánimo del resto de los empleados es casi imposible de medir. En este punto podemos señalar que, como una medida de aprovisionamiento a presupuesto, se puede reservar en una partida anual el valor esperado de que un empleado sea baja, por los costos indirectos que eso conlleva. Con los modelos que proponemos, también se busca medir el EVP de cada empleado, como indicador de valor en cada estrategia.

7. **¿Qué es lo esencial y cuél es la mejor estrategia?** En esta investigación hemos visto que la empresa que nos ha proporcionado los datos solo cuenta con dos herramientas para cambiar el estado de cada empleado: formación y salario (SBA). Hemos visto también que estas políticas son insuficientes, no causan el impacto deseado en los empleados y, aunque tal vez el riesgo de baja de un empleado no sea alarmante, el clima laboral de esta empresa no es el deseado por los empleados, como se observa en el bajo valor de su EVP.

8. **¿Cómo llevar a cabo la práctica la mejor estrategia?** Los pasos que podría dar esta empresa si quiesiera utilizar este modelo como una herramienta para la gestión de sus empleados, actualizándolo periódicamente a partir de los datos que se almacenan dentro del ERP de RRHH, son los siguientes:

    a) Definir el periodo de tiempo en que se va actualizar dicho modelo; en nuestro estudio ha sido anual, pero los periodos de tiempo podrían ser más cortos: semestres, meses, etc.

b) Definir que variables son fijas (o casi fijas) en el modelo; por ejemplo, los datos personales de cada empleado.

c) Definir qué variables son temporales. Estos datos se encuentran en cada uno de los registros de los diferentes históricos que tiene el ERP para cada empleado.

d) Definir las políticas. Estas variables son también temporales. Deben estar almacenadas como datos históricos dentro del ERP y tienen un coste asociado. En nuestro estudio hemos considerado dos variables, la formación y el SBA, pero también podrían considerarse servicios de guardería, actividades lúdicas y de integración, tele-trabajo, etc.

e) Construir, si no se cuenta con un histórico que guarde el dato del EVP, con los datos de históricos del ERP que el experto en RRHH sepa que sirven para medirlo.

f) Construir, con el EVP, el resultado del desempeño y los casos en que ha habido baja del empleado, construir una variable adicional, que es el estado en que se encuentra el empleado dentro de la empresa, recordando que los 5 estados posibles son: baja, DES+/EVP+, DES+/EVP-, DES-/EVP+ y DES-/EVP-.

g) Construir, con la ayuda de algoritmo y el código en Python que se proporciona en este TFM, las diferentes combinaciones de modelos. El formato ProbModelXML es, a nuestro juicio, el más adecuado para almacenarlos.

h) Probar y evaluar dichos modelos utilizando la API de OpenMarkov, con el fin de seleccionar el mejor de ellos.

i) Observar diversas situaciones —particulares, por grupos de empleados o incluso en toda la organización— utilizando ese modelo y OpenMarkov. Ello permitirá medir o modificar dichas políticas en la toma de decisiones y crear presupuestos en función de los costes esperados al aplicar dichas políticas. Con los medios físicos limitados y la cantidad de información que hemos manejado en este trabajo no hemos podido realizar simulaciones con los modelos de Markov ni generar árboles de decisión, pero con equipos más potentes y con el tiempo suficiente también se podrían obtener esta clase de productos.

5.4. Trabajos futuros

Los dos elementos que, en mi opinión, quedaron pendientes son: primero la construcción del modelo tienen que ver en la forma que variables externas pueden afectar a la decisión que tome un empleado para abandonar la organización. Y el segundo tiene que ver con la posibilidad de que tales modelos, con una gran cantidad de estados en cada uno de las variables y que generan una gran cantidad de combinaciones en las probabilidades, puedan generar árboles de decisión y simulaciones con modelos de Markov, usando equipos de computación muy potentes, o bien

generando algún algoritmo que reduzca o calcule de una forma más simplificada o explorar otras estrategias.

He iniciado este trabajo inicia hablando de los sentimientos y quiero concluir con una reflexión abierta sobre este mismo punto. Un ERP no está diseñado para gestionar la información relativa a emociones y sentimientos. Pero la generación emergente, la generación Z, nos está mostrando el camino, pues en su interacción en las redes sociales está expresando las emociones en forma de emoticonos; si pudiéramos ir más allá de los datos que nos dan los ERPs convencionales y tenemos en cuenta las emociones y sentimientos de las personas como una red compleja de interacciones entre naciones, culturas, organizaciones y generaciones, pero principalmente como interacciones entre personas, podremos proporcionar, a partir de la inteligencia artificial, herramientas para alcanzar un objetivo que va más allá de este trabajo: que una persona sea feliz con el trabajo que hace.

Anexo A

Código utilizado en la generación de Modelos

A.1. Estructura de los componentes del código

En la figura A.1 se explica como interacciona el programa en Python con la base de datos y la API de OpenMarkov. Además vemos cómo este programa está compuesto por un modulo principal y otros cuatro con tareas especificas. El código en Python se puede consultar en las siguientes secciones de este anexo o en `https://github.com/smontesv/tfm`.

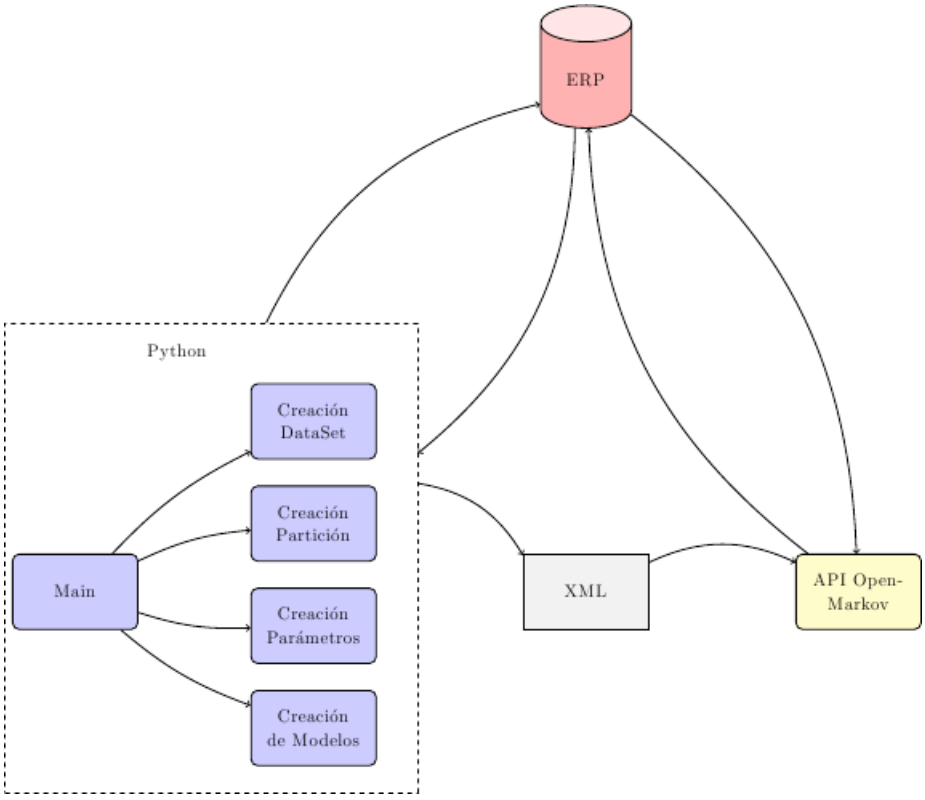

Figura A.1: Estructura de los componentes del código.

A.2. Código del módulo "Main"

```
# -*- coding: utf-8 -*-
"""
UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA
Master: Inteligencia Artificial Avanzada
Materia: Trabajo de Fin de Master
Módulo: Main
@author: Sergio Montes Vázquez Noviembre 2019
"""

#Librerias
import datetime as dt
import time import pymysql.cursors from mysql.connector
import Error
from Creacion_DataSet import Crea_DataSet
from Creacion_Particion import Crea_Particion
from Creacion_Parametros import Crea_Parametros
from Creacion_Modelo import Crea_Modelo

#0.- Subrutinas
#0.0.- Diferencia entre fechas
def Dif_Tiempo(Inicio,Fin):
        vAux1 = Inicio.year + 30 * (Inicio.month - 1) + Inicio.day + Inicio.hour / 24 + Inicio.minute / 60 / 24
        vAux1 = vAux1 + Inicio.second / 60 / 60 / 24
        vAux2 = Fin.year + 30 * (Fin.month - 1) + Fin.day + Fin.hour / 24 + Fin.minute / 60 / 24 + Fin.second / 60 / 60 / 24
        vAux = 24 * (vAux2 - vAux1)
        vHoras = int(vAux)
        vAux = (vAux - vHoras) * 60
        vMinutos = int(vAux)
        vAux = (vAux - vMinutos) * 60
        vSegundos = int(vAux)
        stResultado  = str(vHoras) + ":" + str(vMinutos).zfill(2) + ":"  + str(vSegundos).zfill(2)
        return(stResultado)

#0.1.- Creación de Registro Proceso
def Crea_Proceso(stHost,stUser,stPassword,stDB,stProceso,idPaso,numExperimento,stPaso,
        stExperimento,vInicio):

        myConexion = pymysql.connect(host=stHost,user=stUser,password=stPassword,db=stDB)
        try:
                with myConexion.cursor() as myCursor:
                        stSQL = "INSERT INTO " + stDB + ".TB_REG_PROCESO "
                        stSQL = stSQL + " (PROCESO,ID_PASO,NUM_EXPERIMENTO,DES_PASO,DES_EXPERIMENTO,INICIO)"
                        stSQL = stSQL + " VALUES('" + stProceso + "'," + str(idPaso) + "," + str(numExperimento)
                        stSQL = stSQL + ",'" + stPaso + "','" + stExperimento + "','" + str(vInicio) + "')"
                        myCursor.execute(stSQL)
                        myConexion.commit()

                        stSQL = "SELECT * FROM " + stDB + ".TB_REG_PROCESO "
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()

        finally:
                pass

        myConexion.close()

#0.2.- Actualización de Registro Proceso
def Actualiza_Proceso(stHost,stUser,stPassword,stDB,stProceso, idPaso,numExperimento,vFin,vTiempo):

        myConexion = pymysql.connect(host=stHost, user=stUser, password=stPassword, db=stDB)

        try:
                with myConexion.cursor() as myCursor:
                        stSQL = "UPDATE " + stDB + ".TB_REG_PROCESO "
                        stSQL = stSQL + " SET FIN = '" + str(vFin) + "',"
                        stSQL = stSQL + " TIEMPO = '" + vTiempo + "'"
                        stSQL = stSQL + " WHERE PROCESO = '" + stProceso + "'"
                        stSQL = stSQL + " AND ID_PASO = " + str(idPaso)
                        stSQL = stSQL + " AND NUM_EXPERIMENTO = " + str(numExperimento)

                        myCursor.execute(stSQL)
                        myConexion.commit()
```

```
                        stSQL = "SELECT * FROM " + stDB + ".TB_REG_PROCESO "
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()

        finally:
                pass

        myConexion.close()

#1.- Declaración de Variables Generales
#1.1.- Conexión MySQL
stHost =  'localhost'
stUser = 'root'
stPassword = '######'
stDB = "tfm_smv"

#1.2.- Indicadores de Ejecuciones y Variables Generales
parParticio = 1
banCreaDataset = 1
banCreaParticion = 1
banCreaParametros = 1
banCreaModelo = 1
banCreaModeloA = 1
banCreaModeloB = 1 tamParticion = 10

stRuta = "C:\\Users\\sergiomontes\\Documents\\TFM\\Modelo\\"

stProceso = 'CM_' + time.strftime("%y%m%d") + '_' + time.strftime("%H%M%S")
idPaso = 0
stPaso = ''
idExperimento = 0
stExperimento = ''
Inicio = dt.datetime

print ('0-> Inicia Proeso ' + stProceso)

#3.- Creación del DataSet
if banCreaDataset == 1:
        print ('1-> Entra en Crear DataSet')
        idPaso = 1
        stPaso = 'Crea DataSet'
        idExperimento = 0
        stExperimento = 'S/E'
        Inicio = dt.datetime.today()
        Crea_Proceso(stHost, stUser, stPassword, stDB, stProceso, idPaso, idExperimento,stPaso, stExperimento, Inicio)
        Crea_DataSet(stHost, stUser, stPassword, stDB)
        Fin = dt.datetime.today()
        Tiempo = Dif_Tiempo(Inicio,Fin)
        Actualiza_Proceso(stHost, stUser, stPassword, stDB, stProceso, idPaso, idExperimento, Fin, Tiempo)

#4.- Creación de las Particiones
if banCreaParticion == 1:
        print ('2-> Entra en Crear Particiones')
        idPaso = 2
        stPaso = 'Crea Particiones'
        idExperimento = 0
        stExperimento = 'S/E'
        Inicio = dt.datetime.today()
        Crea_Proceso(stHost, stUser, stPassword, stDB, stProceso, idPaso, idExperimento,stPaso, stExperimento, Inicio)
    Crea_Particion(stHost, stUser, stPassword, stDB, tamParticion)
        Fin = dt.datetime.today()
        Tiempo = Dif_Tiempo(Inicio,Fin)
        Actualiza_Proceso(stHost, stUser, stPassword, stDB, stProceso, idPaso, idExperimento, Fin, Tiempo)

#5.- Creación de Parámetros de los Modelos
if banCreaParametros == 1:
        print ('3-> Entra en Crear Parámetros')
        idPaso = 3
        stPaso = 'Crea Parámetros'
        idExperimento = 0
        stExperimento = 'S/E'
        Inicio = dt.datetime.today()
        Crea_Proceso(stHost, stUser, stPassword, stDB, stProceso, idPaso, idExperimento, stPaso, stExperimento, Inicio)
        Crea_Parametros(stHost, stUser, stPassword, stDB, tamParticion)
```

```
        Fin = dt.datetime.today()
        Tiempo = Dif_Tiempo(Inicio,Fin)
        Actualiza_Proceso(stHost, stUser, stPassword, stDB, stProceso, idPaso, idExperimento, Fin, Tiempo)

#6.- Creacion y Prueba de Modelo
if banCreaModelo == 1:
        myConexion = pymysql.connect(host=stHost,user=stUser,password=stPassword,db=stDB)
        print ('4-> Entra en Crear/Probar Modelo')

        try:
                with myConexion.cursor() as myCursor:
                print ('4.0--> Limpia Experimentos')
                stSQL = "DELETE FROM " + stDB + ".TB_REG_PROCESO WHERE (TIEMPO IS NULL OR TIEMPO ='0:0:0')"
                if (parParticio > -1):
                        stSQL = stSQL + " AND DES_EXPERIMENTO LIKE '%Partición: "
                        stSQL = stSQL + str(parParticio) + "%'"
                myCursor.execute(stSQL)

                stSQL = "UPDATE " + stDB + ".TB_MODELO_PARM SET Seleccionado = FALSE "
                stSQL = stSQL + " WHERE (Parametros = FALSE OR Parametros IS NULL )"
                if (parParticio > -1):
                        stSQL = stSQL + " AND idParticion = " + str(parParticio)
                myCursor.execute(stSQL)

                print ('4.1--> Abre la tabla de Parámetros')
                idExperimento = 0
                stSQL = "SELECT * FROM " + stDB + ".TB_MODELO_PARM WHERE Seleccionado = False AND NumMaxCampos <= 8"
                if (parParticio > -1):
                        stSQL = stSQL + " AND idParticion = " + str(parParticio)
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for i in range(0,len(regAux)):
                        print(i,regAux[i])

                        #6.0.- Se Prepara los datos del experimento
                        varParametro = regAux[i]

                        #6.1.- Se hace una comporbación de que no se ha seleccionado
                        #      este registro en otro proceso
                        stSQL = "SELECT Count(*) FROM " + stDB + ".TB_MODELO_PARM "
                        stSQL = stSQL + " WHERE stVer = '" + varParametro[0] + "'"
                        stSQL = stSQL + " AND NumMaxCampos =" + str(varParametro[1])
                        stSQL = stSQL + " AND banIncluye_Estado_1 =" + str(varParametro[2])
                        stSQL = stSQL + " AND banDecision =" + str(varParametro[3])
                        stSQL = stSQL + " AND vCorte_GainR =" + str(varParametro[4])
                        stSQL = stSQL + " AND vCorrecionLaplace =" + str(varParametro[5])
                        stSQL = stSQL + " AND idTipoDiagrama =" + str(varParametro[6])
                        stSQL = stSQL + " AND idParticion =" + str(varParametro[7])
                        stSQL = stSQL + " AND Seleccionado = FALSE"
                        myCursor.execute(stSQL)
                        regCtl = myCursor.fetchall()

                        if regCtl[0][0] == 1:
                                #6.2.- Se bloque el registro de prámetros
                                stSQL = "UPDATE " + stDB + ".TB_MODELO_PARM "
                                stSQL = stSQL + " SET Seleccionado = TRUE"
                                stSQL = stSQL + " WHERE stVer = '" + varParametro[0] + "'"
                                stSQL = stSQL + " AND NumMaxCampos =" + str(varParametro[1])
                                stSQL = stSQL + " AND banIncluye_Estado_1 =" + str(varParametro[2])
                                stSQL = stSQL + " AND banDecision =" + str(varParametro[3])
                                stSQL = stSQL + " AND vCorte_GainR =" + str(varParametro[4])
                                stSQL = stSQL + " AND vCorrecionLaplace =" + str(varParametro[5])
                                stSQL = stSQL + " AND idTipoDiagrama =" + str(varParametro[6])
                                stSQL = stSQL + " AND idParticion =" + str(varParametro[7])
                                res1 = myCursor.execute(stSQL)
                                myConexion.commit()

                                #6.3.- Se crea el modelo
                                stExperimento = "Ver:" + varParametro[0]
                                stExperimento = stExperimento + " Nodos:" + str(varParametro[1])
                                stExperimento = stExperimento + " Estado 1: " + str(varParametro[2])
                                stExperimento = stExperimento + " Decisión: " + str(varParametro[3])
                                stExperimento = stExperimento + " Corte GainR: " + str(varParametro[4])
                                stExperimento = stExperimento + " Correlación Laplace: " + str(varParametro[5])
                                stExperimento = stExperimento + " Tipo Diagrama: " + str(varParametro[6])
                                stExperimento = stExperimento + " Partición: " + str(varParametro[7])

                                print ('4.2---> Crea Modelo ' + stExperimento)
                                idPaso = 4
```

```
                                stPaso = 'Crea Modelo'
                                Inicio = dt.datetime.today()
                                Crea_Proceso(stHost,stUser,stPassword,stDB,stProceso,idPaso,idExperimento,stPaso,stExperimento,Inicio)
                                Crea_Modelo(stHost, stUser, stPassword, stDB, stRuta, varParametro, stExperimento)
                                Fin = dt.datetime.today()
                                Tiempo = Dif_Tiempo(Inicio,Fin)
                                Actualiza_Proceso(stHost,stUser,stPassword,stDB,stProceso,idPaso,idExperimento,Fin,Tiempo)

                                #6.4.- Se prueba el modelo

                                #6.5.- Se marca el modelo como ejecutado
                                stSQL = "UPDATE " + stDB + ".TB_MODELO_PARM "
                                stSQL = stSQL + " SET Parametros = TRUE"
                                stSQL = stSQL + " WHERE stVer = '" + varParametro[0] + "'"
                                stSQL = stSQL + " AND NumMaxCampos =" + str(varParametro[1])
                                stSQL = stSQL + " AND banIncluye_Estado_1 =" + str(varParametro[2])
                                stSQL = stSQL + " AND banDecision =" + str(varParametro[3])
                                stSQL = stSQL + " AND vCorte_GainR =" + str(varParametro[4])
                                stSQL = stSQL + " AND vCorrecionLaplace =" + str(varParametro[5])
                                stSQL = stSQL + " AND idTipoDiagrama =" + str(varParametro[6])
                                stSQL = stSQL + " AND idParticion =" + str(varParametro[7])
                                myCursor.execute(stSQL)
                                myConexion.commit()

                        idExperimento = idExperimento + 1
        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()
                varResultado = -1

        finally:
                pass

        #99.- Cierra conexiones
        myConexion.close()

if banCreaModeloA == 1:
        print ('5-> Entra en Crear Modelo Final A')

        stVer= '01'
        NumMaxCampos = 5
        banIncluye_Estado_1 = 1
        banDecision = 1
        vCorte_GainR = 0.7500
        vCorrecionLaplace = 0.5000
        idTipoDiagrama = 0
        idParticion = -1

        varParametro = [stVer, NumMaxCampos, banIncluye_Estado_1, banDecision, vCorte_GainR,
                vCorrecionLaplace, idTipoDiagrama, idParticion]

        stExperimento = "Ver:" + varParametro[0] + " Nodos:" + str(varParametro[1])
        stExperimento = stExperimento + " Estado 1: " + str(varParametro[2])
        stExperimento = stExperimento + " Decisión: " + str(varParametro[3])
        stExperimento = stExperimento + " Corte GainR: " + str(varParametro[4])
        stExperimento = stExperimento + " Correlación Laplace: " + str(varParametro[5])
        stExperimento = stExperimento + " Tipo Diagrama: " + str(varParametro[6])
        stExperimento = stExperimento + " Partición: " + str(varParametro[7])
        Crea_Modelo(stHost, stUser, stPassword, stDB, stRuta, varParametro, stExperimento)

if banCreaModeloB == 1:
        print ('6-> Entra en Crear Modelo Final B')

        stVer= '01'
        NumMaxCampos = 4
        banIncluye_Estado_1 = 0
        banDecision = 1
        vCorte_GainR = 0.7500
        vCorrecionLaplace = 0.5000
        idTipoDiagrama = 0
        idParticion = -1

        varParametro = [stVer, NumMaxCampos, banIncluye_Estado_1, banDecision, vCorte_GainR,
                vCorrecionLaplace, idTipoDiagrama, idParticion]

        stExperimento = "Ver:" + varParametro[0] + " Nodos:" + str(varParametro[1])
        stExperimento = stExperimento + " Estado 1: " + str(varParametro[2])
        stExperimento = stExperimento + " Decisión: " + str(varParametro[3])
        stExperimento = stExperimento + " Corte GainR: " + str(varParametro[4])
```

```
        stExperimento = stExperimento + " Correlación Laplace: " + str(varParametro[5])
        stExperimento = stExperimento + " Tipo Diagrama: " + str(varParametro[6])
        stExperimento = stExperimento + " Partición: " + str(varParametro[7])
        Crea_Modelo(stHost, stUser, stPassword, stDB, stRuta, varParametro, stExperimento)

print ('99-> Termina Proeso ' + stProceso)
```

A.3. Código del módulo "creación del DataSet"

```
# -*- coding: utf-8 -*-
"""
UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA
Master: Inteligencia Artificial Avanzada
Materia: Trabajo de Fin de Master
Módulo: Carga y Transformación de los Datos en el DataSet
@author: Sergio Montes Vázquez Noviembre 2019
"""

#0.- Librerias e inicia variables
import pymysql.cursors
from mysql.connector import Error

def Crea_DataSet(stHost, stUser, stPassword, stDB):

        #1.- Conexión MyServer
        varResultado = 1
        myConexion = pymysql.connect(host = stHost,user=stUser,password=stPassword,db=stDB)

        print ('-> 0.- Inicia creación ')
        try:
                with myConexion.cursor() as myCursor:
                        print ('--> 1.- Entra en Cursor MySql')

                        #2.- Creación de la tabla de orden
                        print ('--> 2.- Creación de la tabla de orden')

                        #2.1.- Se borra la tabla si es que existe
                        stTabla = "CAMPO_VALOR_ORDEN"
                        stSQL = "DROP TABLE IF EXISTS " + stDB + "." + stTabla
                        myCursor.execute(stSQL)

                        #2.2.- Se crea la estructura de la tabla
                        stSQL = "CREATE TABLE IF NOT EXISTS " + stDB + "." + stTabla + " (\n"
                        stSQL = stSQL + "  CAMPO   VARCHAR(50),\n"
                        stSQL = stSQL + "  VALOR   VARCHAR(50),\n"
                        stSQL = stSQL + "  ORDEN   INT )"
                        myCursor.execute(stSQL)

                        #2.3.- Preparación de los datos
                        vecOrden = []
                        vecOrden.append(["NIVEL_EDUCATIVO",["S/D","Sin Universidad","Con Universidad"]])
                        vecOrden.append(["EDAD",["S/D","Joven","Adulto","Mayor"]])
                        vecOrden.append(["FAMILIA",["S/D","Sin hijos","Con hijos","Familia numerosa"]])
                        vecOrden.append(["GRUPO_PUESTO",["S/D","Administrativo","Coordinador","Responsable"]])
                        vecOrden.append(["ANTIGUEDAD",["S/D","Poca","Mediana","Mucha"]])
                        vecOrden.append(["SBA",["S/D","Bajo","Medio","Bueno","Excelente"]])
                        vecOrden.append(["DES_PORC_AUM",["S/D","Descuento","Sin Aumento","Aumento"]])
                        vecOrden.append(["FORMACION",["S/D","Insuficiente","Suficiente","Demasiado"]])
                        vecOrden.append(["DESEMPENNO",["S/D","Bajo","Alto"]])
                        vecOrden.append(["EVP",["S/D","Bajo","Alto"]])
                        vecOrden.append(["ESTADO",["S/D","DES-/EVP-","DES-/EVP+","DES+/EVP-","DES+/EVP+","BAJA"]])
                        for iOrden in vecOrden:
                                for i in range(0,len(iOrden[1])):
                                        stSQL = "INSERT INTO " + stDB + "." + stTabla
                                        stSQL = stSQL + "(CAMPO,VALOR,ORDEN) VALUES('" + iOrden[0] + "','"
                                        stSQL = stSQL + iOrden[1][i] + "'," + str(i) + ")"
                                        myCursor.execute(stSQL)

                        myConexion.commit()

                        #3.- Creación de la tabla de DataSet
                        #3.1.- Se borra la tabla si es que existe
                        stTabla = "MID_DATOS"
                        stSQL = "DROP TABLE IF EXISTS " + stDB + "." + stTabla
                        myCursor.execute(stSQL)

                        #3.2.- Se crea la estructura de la tabla
```

```
stSQL = "CREATE TABLE IF NOT EXISTS " + stDB + "." + stTabla + " (\n"
stSQL = stSQL + "  ID_EMPLEADO      VARCHAR(12),\n"
stSQL = stSQL + "  ANNO             INT,\n"

#Variables propias de cada Empleado y el Puesto de Trabajo
stSQL = stSQL + "  SEXO             VARCHAR(1),\n"
stSQL = stSQL + "  ANNO_NAC         INT,\n"
#Joven (menor de 30 años); Adulto (Mayor de 30 años y menor de 60) y Mayor (más de 60) años.
stSQL = stSQL + "  EDAD             VARCHAR(20),\n"
#Número de hijos
stSQL = stSQL + "  NUM_HIJOS        INT,\n"
#Sin hijos; con hijos (1 o 2 hijos o personas a cargo); familia numerosa (más de 2 hijos o personas a cargo).
stSQL = stSQL + "  FAMILIA          VARCHAR(20),\n"
stSQL = stSQL + "  PUESTO               VARCHAR(20),\n"
#Administrativo, Coordinador y Responsable.
stSQL = stSQL + "  GRUPO_PUESTO     VARCHAR(20),\n"
#Años Antiguedad
stSQL = stSQL + "  ANNO_ANTIGUEDAD  INT,\n"
#Poca (menos de 3 años), Mediana (más de 3 y menos de 10) y Mucha (más de 10 Años).
stSQL = stSQL + "  ANTIGUEDAD       VARCHAR(20),\n"
#Importe Salario Bruto Anual
stSQL = stSQL + "  SBA_IMPORTE      NUMERIC(18,2),\n"
#Bajo (Menos de 20.000€), Medio (más de 20.000€ y menos de 30.000€),
#Bueno (más de 30.000€ y menos de 45.000€) y Excelente (más de 45.000€).
stSQL = stSQL + "  SBA              VARCHAR(20),\n"
#Porcentaje de aumento con respecto al año anterior
stSQL = stSQL + "  PORC_AUMENTO     NUMERIC(18,2),\n"
#Descripción Porcentaje de aumento con respecto al año anterior
stSQL = stSQL + "  DES_PORC_AUM     VARCHAR(50),\n"
#Sin Estudios Universitarios, Con Universidad y Con Posgrado.
stSQL = stSQL + "  NIVEL_EDUCATIVO  VARCHAR(20),\n"
#Número de Cursos de Formación
stSQL = stSQL + "  NUM_CURSOS       INT,\n"
#Insuficiente (0 ó 1 curso), Suficiente (2 cursos), Demasiado (más de 2 cursos).
stSQL = stSQL + "  FORMACION        VARCHAR(20),\n"

#Variables que se cálculan durante el modelaje y relaticvas al EVP
stSQL = stSQL + "  DESEMPENNO_NUM   NUMERIC(8,2),\n"
stSQL = stSQL + "  DESEMPENNO       VARCHAR(20),\n"     #Alto, Bajo
stSQL = stSQL + "  EVP_APRENDIZAJE  NUMERIC(8,2),\n"
stSQL = stSQL + "  EVP_CARRERA      NUMERIC(8,2),\n"
stSQL = stSQL + "  EVP_EQUILIBRIO   NUMERIC(8,2),\n"
stSQL = stSQL + "  EVP_RETRIBUCION  NUMERIC(8,2),\n"
stSQL = stSQL + "  EVP_NUM          NUMERIC(8,2),\n"
stSQL = stSQL + "  EVP              VARCHAR(20),\n"     #Alto, Bajo

#Variables Propias del modelo de Markov
stSQL = stSQL + "  ESTADO           VARCHAR(20),\n"
stSQL = stSQL + "  ESTADO_ANTERIOR  VARCHAR(20),\n"
stSQL = stSQL + "  ES_SEPARACION    NUMERIC(1),\n"      #Es separación
stSQL = stSQL + "  ES_ABANDONO      NUMERIC(1),\n"      #Es abandono

#Clase que nos interesa obtener
stSQL = stSQL + "  SEPARACION       NUMERIC(18,4),\n"
stSQL = stSQL + "  ABANDONO         NUMERIC(18,4),\n"

#Estimación de Precios y decisión de fin de puesto.
stSQL = stSQL + "  PRE_AUMENTO      NUMERIC(18,4),\n"
stSQL = stSQL + "  PRE_FORMACION    NUMERIC(18,4),\n"
stSQL = stSQL + "  PRE_PRUEBA       NUMERIC(18,4),\n"
stSQL = stSQL + "  FIN_PUESTO       NUMERIC(1),\n"      #Variable de Fin de Puesto
stSQL = stSQL + "  MUESTRA          NUMERIC(1) )"
myCursor.execute(stSQL)
myConexion.commit()

#4.- Creación del DataSet
print ('--> 3.- Creación de la tabla DataSet')

#4.1.- Se recorre la lista de Empleados y periodos
stSQL = "SELECT A.STD_ID_PERSON, A.STD_ID_GENDER, "
stSQL = stSQL + "ifnull(A.STD_DT_BIRTH,date_add(B.STD_DT_START,interval -30 YEAR)), "
stSQL = stSQL + "B.STD_DT_START, B.STD_DT_END, "
stSQL = stSQL + "ifnull(B.SSP_FEC_ANTIGUEDAD,B.STD_DT_START), "
stSQL = stSQL + ifnull(B.STD_ID_HRP_END_REA,'S/D') \n"
stSQL = stSQL + "FROM " + stDB + ".STD_PERSON A, " + stDB + ".STD_HR_PERIOD B \n"
stSQL = stSQL + "WHERE A.STD_ID_PERSON = B.STD_ID_HR \n"
stSQL = stSQL + "AND NOT( A.STD_ID_PERSON LIKE 'BC%') \n"
stSQL = stSQL + "ORDER BY A.STD_ID_PERSON, B.STD_DT_START "
myCursor.execute(stSQL)
```

```
regEmpleados = myCursor.fetchall()

numRegistros = 0
for i in range(0,len(regEmpleados)):
        idEmpleado = regEmpleados[i]
        stEmpleado = idEmpleado[0]

        #4.2.- Se recuperan y transforman datos
        vSTD_DT_BIRTH = idEmpleado[2]
        vANNO_INI = idEmpleado[3].year
        vANNO_FIN = idEmpleado[4].year
        vFEC_ANTIGUEDAD = idEmpleado[5]
        vDT_START = idEmpleado[3]
        vSTD_ID_HRP_END_REA = idEmpleado[6]
        vANNO = vANNO_INI

        if vANNO_FIN == 4000:
                vANNO_FIN = 2017

        vSTD_ID_GENDER = 'M'
        if idEmpleado[1] == '1':
                vSTD_ID_GENDER = 'H'

        #4.3.- Hace un recorrido para los años de cada empleado
        while vANNO <=  vANNO_FIN:
                #4.4.- Se comprueba que exista el registro en MID_DATOS
                stSQL = "SELECT COUNT(*) "
                stSQL = stSQL + "FROM " + stDB + "." + stTabla
                stSQL = stSQL + " WHERE ID_EMPLEADO = '" +  stEmpleado + "' "
                stSQL = stSQL + " AND ANNO = " + str(vANNO)
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                if regAux[0][0] == 0:
                        stSQL = "INSERT INTO " + stDB + "." + stTabla + "(ID_EMPLEADO,ANNO ) "
                        stSQL = stSQL + "VALUES('" + stEmpleado + "'," + str(vANNO) + ")"
                        myCursor.execute(stSQL)

                #4.5.- Recuperación de Datos Personales  y del Puesto de Trabajo
                #4.5.1.- Edad
                auxEdad = vANNO - vSTD_DT_BIRTH.year
                vEDAD = 'Joven'
                if auxEdad > 30 and auxEdad <= 60 : vEDAD = 'Adulto'
                if auxEdad > 60: vEDAD = 'Mayor'

                #4.5.2 Familia
                vFAMILIA = 'Sin hijos'
                stSQL = "SELECT COUNT(*) "
                stSQL = stSQL + " FROM " + stDB + ".M4CSP_FAM_EMPLEADOS "
                stSQL = stSQL + " WHERE STD_ID_PERSON = '" + stEmpleado + "' "
                stSQL = stSQL + "       AND STD_ID_ACT_DEP_TYP = '2' "
                stSQL = stSQL + " AND YEAR(STD_DT_BIRTH) < " + str(vANNO)
                stSQL = stSQL + "       AND " + str(vANNO) + " - YEAR(STD_DT_BIRTH) < 18 "
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                vNUM_HIJOS = regAux[0][0]
                if vNUM_HIJOS == 1 or vNUM_HIJOS == 2 :
                        vFAMILIA = 'Con hijos'
                if vNUM_HIJOS > 2:
                        vFAMILIA = 'Familia numerosa'

                #4.5.3.- Grupo de Puesto
                vPUESTO = 'S/D'
                vGRUPO_PUESTO = 'S/D'
                vAuxSBA = '0'
                stSQL = "SELECT  COUNT(*) "
                stSQL = stSQL + " FROM " + stDB + ".M4SCO_H_HR_JOB A "
                stSQL = stSQL + " LEFT JOIN " + stDB + ".STD_HT_JOB_DEF B ON "
                stSQL = stSQL + "    (A.ID_ORGANIZATION = B.ID_ORGANIZATION  "
                stSQL = stSQL + " AND B.STD_ID_JOB_CODE = A.SCO_ID_JOB_CODE "
                stSQL = stSQL + "   AND B.STD_DT_START <= A.SCO_DT_END "
                AstSQL = stSQL + " ND B.STD_DT_END >= A.SCO_DT_START) "
                stSQL = stSQL + " LEFT JOIN " + stDB + ".STD_LU_JOB_INT_CLA C "
                stSQL = stSQL + "ON (B.STD_ID_JOB_INT_CLA = C.STD_ID_JOB_INT_CLA) "
                stSQL = stSQL + " LEFT JOIN " + stDB + ".STD_LU_JOB_INT_FAM D "
                stSQL = stSQL + "ON (C.STD_ID_JOB_INT_FAM = D.STD_ID_JOB_INT_FAM) "
                stSQL = stSQL + " WHERE A.SCO_ID_HR = '" + stEmpleado + "' "
                stSQL = stSQL + " AND YEAR(A.SCO_DT_START) <= " + str(vANNO)
                stSQL = stSQL + " AND YEAR(A.SCO_DT_END) >= " + str(vANNO)
                myCursor.execute(stSQL)
```

```
regAux = myCursor.fetchall()
if regAux[0][0] == 1:
        stSQL = "SELECT A.SCO_ID_JOB_CODE, ifnull(C.STD_ID_JOB_INT_FAM,'S/D'), "
        stSQL = stSQL + " ifnull(D.SCO_COMMENT , '-1') "
        stSQL = stSQL + " FROM " + stDB + ".M4SCO_H_HR_JOB A "
        stSQL = stSQL + " LEFT JOIN " + stDB + ".STD_HT_JOB_DEF B ON "
        stSQL = stSQL + "   (A.ID_ORGANIZATION = B.ID_ORGANIZATION "
        stSQL = stSQL + " AND B.STD_ID_JOB_CODE = A.SCO_ID_JOB_CODE "
        stSQL = stSQL + "   AND B.STD_DT_START <= A.SCO_DT_END "
        stSQL = stSQL + " AND B.STD_DT_END >= A.SCO_DT_START) "
        stSQL = stSQL + " LEFT JOIN " + stDB + ".STD_LU_JOB_INT_CLA C "
        stSQL = stSQL + " ON (B.STD_ID_JOB_INT_CLA = C.STD_ID_JOB_INT_CLA) "
        stSQL = stSQL + " LEFT JOIN " + stDB + ".STD_LU_JOB_INT_FAM D "
        stSQL = stSQL + " ON (C.STD_ID_JOB_INT_FAM = D.STD_ID_JOB_INT_FAM) "
        stSQL = stSQL + " WHERE A.SCO_ID_HR = '" + stEmpleado + "' "
        stSQL = stSQL + " AND A.SCO_DT_START = (SELECT MAX(SCO_DT_START) "
        stSQL = stSQL + "       FROM " + stDB + ".M4SCO_H_HR_JOB "
        stSQL = stSQL + "       WHERE SCO_ID_HR = '" + stEmpleado + "' "
        stSQL = stSQL + "       AND YEAR(SCO_DT_START) <= " + str(vANNO)
        stSQL = stSQL + "       AND YEAR(SCO_DT_END) >= " + str(vANNO) + ")"

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        vPUESTO = regAux[0][0]
        vGRUPO_PUESTO = regAux[0][1]
        vAuxSBA = regAux[0][2]

        #Administrativo, Coordinador y Responsable.
        if vGRUPO_PUESTO=='ADMIN'
                or vGRUPO_PUESTO=='SECRET'
                or vGRUPO_PUESTO=='TEC_ADMIN'
                or vGRUPO_PUESTO=='TECNICOS'
                or vGRUPO_PUESTO=='COND'
                or vGRUPO_PUESTO=='ALT'
                or vGRUPO_PUESTO=='ANALISTAS'
                or vGRUPO_PUESTO=='ASES_TECN'
                or vGRUPO_PUESTO=='AUX_ADMIN':
                vGRUPO_PUESTO = 'Administrativo'

        if vGRUPO_PUESTO=='COMERCIAL'
                or vGRUPO_PUESTO=='COORD'
                or vGRUPO_PUESTO=='FJ0'
                or vGRUPO_PUESTO=='FJ1'
                or vGRUPO_PUESTO=='FJ2'
                or vGRUPO_PUESTO=='GER_COMER'
                or vGRUPO_PUESTO=='GES_COMER'
                or vGRUPO_PUESTO=='LETRADOS':
                vGRUPO_PUESTO = 'Coordinador'

        if vGRUPO_PUESTO=='MB'
                or vGRUPO_PUESTO=='RA'
                or vGRUPO_PUESTO=='RD'
                or vGRUPO_PUESTO=='RDS'
                or vGRUPO_PUESTO=='RGC'
                or vGRUPO_PUESTO=='ROC'
                or vGRUPO_PUESTO=='RS'
                or vGRUPO_PUESTO=='RU':
                vGRUPO_PUESTO = 'Responsable'

#4.5.4.- Antigüedad
vANNO_ANTIGUEDAD = vANNO - vFEC_ANTIGUEDAD.year
vANTIGUEDAD = 'Poca'
if vANNO_ANTIGUEDAD > 3 and vANNO_ANTIGUEDAD <= 10:
        vANTIGUEDAD = 'Mediana'
if vANNO_ANTIGUEDAD > 10:
        vANTIGUEDAD = 'Mucha'

#4.5.5.- SBA
if float(vAuxSBA) <=0:
        stSQL = "SELECT COUNT(*) "
        stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
        stSQL = stSQL + " WHERE ID_EMPLEADO =  '" + stEmpleado + "' "
        stSQL = stSQL + " AND ANNO + 1 = " + str(vANNO)

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        if regAux[0][0] == 1:
```

```
                    stSQL = "SELECT GRUPO_PUESTO, SBA_IMPORTE "
                    stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
                    stSQL = stSQL + " WHERE ID_EMPLEADO = '" + stEmpleado + "' "
                    stSQL = stSQL + " AND ANNO + 1 = " + str(vANNO)

                    myCursor.execute(stSQL)
                    regAux = myCursor.fetchall()

                    vGRUPO_PUESTO = regAux[0][0]
                    vAuxSBA = str(regAux[0][1])

if float(vAuxSBA) <=0: vSBA = 'S/D'
if float(vAuxSBA) > 0 and float(vAuxSBA) <= 20000:
        vSBA = 'Bajo'
if float(vAuxSBA) > 20000 and float(vAuxSBA) <= 30000:
        vSBA = 'Medio'
if float(vAuxSBA) > 30000 and float(vAuxSBA) <= 45000:
        vSBA = 'Bueno'
if float(vAuxSBA) > 45000: vSBA = 'Excelente'

#4.5.6.- Nivel Educativo
vNIVEL_EDUCATIVO = 'S/D'
stSQL = "SELECT COUNT(*) "
stSQL = stSQL + " FROM " + stDB + ".STD_HR_ACAD_BACKGR A, " + stDB + ".STD_LU_EDU_TYPE B "
stSQL = stSQL + "       WHERE A.STD_ID_EDU_TYPE = B.STD_ID_EDU_TYPE "
stSQL = stSQL + "       AND A.STD_ID_HR = '" + stEmpleado + "' "

myCursor.execute(stSQL)
regAux = myCursor.fetchall()

if regAux[0][0] == 1:
        stSQL = "SELECT STD_N_EDU_TYPEITA "
        stSQL = stSQL + " FROM " + stDB + ".STD_HR_ACAD_BACKGR A, "
        stSQL = stSQL + stDB + ".STD_LU_EDU_TYPE B "
        stSQL = stSQL + " WHERE A.STD_ID_EDU_TYPE = B.STD_ID_EDU_TYPE "
        stSQL = stSQL + " AND A.STD_ID_HR = '" + stEmpleado + "' "

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        vNIVEL_EDUCATIVO = regAux[0][0]

#4.5.7.- Formación
vFORMACION = 'Insuficiente'
if vANNO < 2016:
        vNUM_CURSOS = 2
else:
        stSQL = "SELECT COUNT(*) "
        stSQL = stSQL + " FROM " + stDB + ".M4SCO_ENROLLMENT "
        stSQL = stSQL + " WHERE STD_ID_PERSON = '" + stEmpleado + "' "
        stSQL = stSQL + " AND YEAR(DT_END) = " + str(vANNO)

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()
        vNUM_CURSOS = regAux[0][0]

if vNUM_CURSOS == 1 or vNUM_CURSOS == 2:
        vFORMACION = 'Suficiente'
if vNUM_CURSOS > 2 :
        vFORMACION = 'Demasiado'

#4.6.- Recuperación de Desempeñio y EVP
#4.6.1.- Desempeño
vDESEMPENNO_NUM = 0
vDESEMPENNO = 'S/D'
stSQL = "SELECT COUNT(*) "
stSQL = stSQL + " FROM " + stDB + ".M4CSP_EVALUATOR "
stSQL = stSQL + " WHERE CSP_ID_HR = '" + stEmpleado + "' "
stSQL = stSQL + " AND YEAR(CSP_DT_START_EVAL) = " + str(vANNO)

myCursor.execute(stSQL)
regAux = myCursor.fetchall()

if regAux[0][0] == 1:
        stSQL = "SELECT ifnull(CSP_EVAL_GLOBAL,'0') "
        stSQL = stSQL + " FROM " + stDB + ".M4CSP_EVALUATOR "
        stSQL = stSQL + " WHERE CSP_ID_HR = '" + stEmpleado + "' "
        stSQL = stSQL + " AND YEAR(CSP_DT_START_EVAL) = " + str(vANNO)
```

```
        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        vDESEMPENNO_NUM = float(regAux[0][0])

if vDESEMPENNO_NUM < 90 and vDESEMPENNO_NUM > 0:
        vDESEMPENNO = 'Bajo'
if vDESEMPENNO_NUM >= 90: vDESEMPENNO = 'Alto'

#4.6.2.- EVP
#4.6.2.1.- Aprendizaje
vEVP_APRENDIZAJE = 0
if vNUM_CURSOS == 1: vEVP_APRENDIZAJE = 5
if vNUM_CURSOS >= 2: vEVP_APRENDIZAJE = 10

#4.6.2.2.- Plan de Carrera
vEVP_CARRERA = 0
stSQL = "SELECT COUNT(*) "
stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
stSQL = stSQL + " WHERE ID_EMPLEADO = '" + stEmpleado + "' "
stSQL = stSQL + " AND ANNO < " + str(vANNO)
stSQL = stSQL + "        AND PUESTO = '" + vPUESTO + "'"

myCursor.execute(stSQL)
regAux = myCursor.fetchall()

if regAux[0][0] <= 2: vEVP_CARRERA = 10
if regAux[0][0] > 2 and regAux[0][0] <= 5:
        vEVP_CARRERA = 5

#4.6.2.3.- Equilibrio vida personal-profesional
vEVP_EQUILIBRIO  = 0
if auxEdad <= 30 and vNIVEL_EDUCATIVO == 'Sin Universidad':
        vEVP_EQUILIBRIO  = 10
if auxEdad > 30 and vNUM_HIJOS != 0:
        vEVP_EQUILIBRIO  = 10

#4.6.2.4.- Retribución
vPORC_AUMENTO = 0
vEVP_RETRIBUCION = 0
vESTADO_ANTERIOR = 'S/D'
stSQL = "SELECT COUNT(*) "
stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
stSQL = stSQL + "        WHERE ID_EMPLEADO = '" + stEmpleado + "' "
stSQL = stSQL + "        AND ANNO + 1 = " + str(vANNO)

myCursor.execute(stSQL)
regAux = myCursor.fetchall()

if regAux[0][0] == 0: vPORC_AUMENTO = 100
if regAux[0][0] == 1:
        stSQL = "SELECT SBA_IMPORTE, ESTADO "
        stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
        stSQL = stSQL + " WHERE ID_EMPLEADO = '" + stEmpleado + "' "
        stSQL = stSQL + " AND ANNO + 1 = " + str(vANNO)

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        if float(regAux[0][0]) > 0:
                vPORC_AUMENTO = 100 * (float(vAuxSBA) - float(regAux[0][0]))
                vPORC_AUMENTO= vPORC_AUMENTO / float(regAux[0][0])
        vESTADO_ANTERIOR = regAux[0][1]

#4.6.2.5.- Descripción de Porcentaje de Aumento
if vPORC_AUMENTO <= 0: vDES_PORC_AUM = 'Descuento'
if vPORC_AUMENTO == 0: vDES_PORC_AUM = 'Sin Aumento'
if vPORC_AUMENTO > 0: vDES_PORC_AUM = 'Aumento'

if vPORC_AUMENTO >=25: vEVP_RETRIBUCION = 10
if vPORC_AUMENTO >=10 and vPORC_AUMENTO < 25:
        vEVP_RETRIBUCION = 10

vEVP_NUM = 0.27 * vEVP_APRENDIZAJE + 0.25 * vEVP_CARRERA
vEVP_NUM = vEVP_NUM + 0.25 * vEVP_EQUILIBRIO + 0.23 * vEVP_RETRIBUCION
vEVP = 'Bajo'
if vEVP_NUM > 5: vEVP = 'Alto'

#4.7.- Estado para el modelo de Markov
```

```
vESTADO = 'S/D'
if vDESEMPENNO == 'Alto' and vEVP == 'Alto':
        vESTADO = 'DES+/EVP+'
if vDESEMPENNO == 'Alto' and vEVP == 'Bajo':
        vESTADO = 'DES+/EVP-'
if vDESEMPENNO == 'Bajo' and vEVP == 'Alto':
        vESTADO = 'DES-/EVP+'
if vDESEMPENNO == 'Bajo' and vEVP == 'Bajo':
        vESTADO = 'DES-/EVP-'
vES_ABANDONO = 0
vES_SEPARACION = 0

#4.8.- Estados terminales del modelo de Markov (si aplica)
if vANNO ==  vANNO_FIN and vSTD_ID_HRP_END_REA != 'S/D':
        stSQL = "SELECT COUNT(*) "
        stSQL = stSQL + " FROM " + stDB + ".STD_HR_PERIOD "
        stSQL = stSQL + " WHERE STD_ID_HR = '" + stEmpleado + "' "
        stSQL = stSQL + " AND STD_DT_START > '" + str(vDT_START) + "' "

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        if regAux[0][0] == 0:
                vESTADO = 'BAJA'
                vES_ABANDONO = 1
                if vSTD_ID_HRP_END_REA == '004'
                        or vSTD_ID_HRP_END_REA == '009'
                        or vSTD_ID_HRP_END_REA == '111'
                        or vSTD_ID_HRP_END_REA == '54':
                        vES_ABANDONO = 0
                        vES_SEPARACION = 1

#4.9.- Cálculo Separación
vSEPARACION = vANNO_ANTIGUEDAD * 22 * float(vAuxSBA)/365
if vSEPARACION > float(vAuxSBA) * 2:
        vSEPARACION = float(vAuxSBA) * 2
vSEPARACION = vSEPARACION + 3000 + float(vAuxSBA) / 2

#4.10.- Cálculo Abandono
vABANDONO = 3000 + float(vAuxSBA) / 2

#4.11.- Cálculo de Precios adicionales
vPRE_AUMENTO = float(vAuxSBA) * 0.20
vPRE_FORMACION = 3000
vPRE_PRUEBA = float(vAuxSBA) * 0.25

#4.12.- Fin de Puesto
vFIN_PUESTO = 0
stSQL = "SELECT COUNT(*) "
stSQL = stSQL + " FROM " + stDB + ".STD_JOB "
stSQL = stSQL + " WHERE YEAR(STD_DT_END) = " + str(vANNO)
stSQL = stSQL + " AND STD_ID_JOB_CODE = '" + vPUESTO + "'"

myCursor.execute(stSQL)
regAux = myCursor.fetchall()

if regAux[0][0]  > 0: vFIN_PUESTO = 1

#4.99.- Se hace Update con los dato ya calculados
stSQL = "UPDATE " + stDB + "." + stTabla + " \n"
stSQL = stSQL + "SET  \n"
stSQL = stSQL + "    SEXO = '" + vSTD_ID_GENDER + "', \n"
stSQL = stSQL + "    ANNO_NAC = " + str(vSTD_DT_BIRTH.year) + ", \n"
stSQL = stSQL + "    EDAD = '" + vEDAD + "', \n"
stSQL = stSQL + "    NUM_HIJOS = " + str(vNUM_HIJOS) + ", \n"
stSQL = stSQL + "    FAMILIA = '" + vFAMILIA + "', \n"
stSQL = stSQL + "    PUESTO = '" + vPUESTO + "', \n"
stSQL = stSQL + "    GRUPO_PUESTO = '" + vGRUPO_PUESTO + "', \n"
stSQL = stSQL + "    ANNO_ANTIGUEDAD = " + str(vANNO_ANTIGUEDAD) + ", \n"
stSQL = stSQL + "    ANTIGUEDAD = '" + vANTIGUEDAD + "', \n"
stSQL = stSQL + "    SBA_IMPORTE = " + vAuxSBA + ", \n"
stSQL = stSQL + "    SBA = '" + vSBA + "', \n"
stSQL = stSQL + "    PORC_AUMENTO = " + str(vPORC_AUMENTO) + ", \n"
stSQL = stSQL + "    DES_PORC_AUM = '" + vDES_PORC_AUM + "', \n"
stSQL = stSQL + "    NIVEL_EDUCATIVO = '" + vNIVEL_EDUCATIVO + "', \n"
stSQL = stSQL + "    NUM_CURSOS = " + str(vNUM_CURSOS) + ", \n"
stSQL = stSQL + "    FORMACION = '" + vFORMACION + "', \n"
stSQL = stSQL + "    DESEMPENNO_NUM = " + str(vDESEMPENNO_NUM) + ", \n"
stSQL = stSQL + "    DESEMPENNO = '" + vDESEMPENNO + "', \n"
```

```
                                        stSQL = stSQL + "    EVP_APRENDIZAJE = " + str(vEVP_APRENDIZAJE) + ", \n"
                                        stSQL = stSQL + "    EVP_CARRERA = " + str(vEVP_CARRERA) + ", \n"
                                        stSQL = stSQL + "    EVP_EQUILIBRIO = " + str(vEVP_EQUILIBRIO) + ", \n"
                                        stSQL = stSQL + "    EVP_RETRIBUCION = " + str(vEVP_RETRIBUCION) + ", \n"
                                        stSQL = stSQL + "    EVP_NUM = " + str(vEVP_NUM) + ", \n"
                                        stSQL = stSQL + "    EVP = '" + vEVP + "', \n"
                                        stSQL = stSQL + "    ESTADO = '" + vESTADO + "', \n"
                                        stSQL = stSQL + "    ESTADO_ANTERIOR = '" + vESTADO_ANTERIOR + "', \n"
                                        stSQL = stSQL + "    ES_SEPARACION = " + str(vES_SEPARACION) + ", \n"
                                        stSQL = stSQL + "    ES_ABANDONO = " + str(vES_ABANDONO) + ", \n"
                                        stSQL = stSQL + "    SEPARACION = " + str(vSEPARACION) + ", \n"
                                        stSQL = stSQL + "    ABANDONO = " + str(vABANDONO) + ", \n"
                                        stSQL = stSQL + "    PRE_AUMENTO = " + str(vPRE_AUMENTO) + ", \n"
                                        stSQL = stSQL + "    PRE_FORMACION = " + str(vPRE_FORMACION) + ", \n"
                                        stSQL = stSQL + "    PRE_PRUEBA = " + str(vPRE_PRUEBA) + ", \n"
                                        stSQL = stSQL + "    FIN_PUESTO = " + str(vFIN_PUESTO) + " \n"
                                        stSQL = stSQL + " WHERE ID_EMPLEADO = '" +  stEmpleado + "' \n"
                                        stSQL = stSQL + " AND ANNO = " + str(vANNO)
                                        myCursor.execute(stSQL)

                                        numRegistros = numRegistros + 1

                                        myConexion.commit()
                                        vANNO = vANNO + 1

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()
                varResultado = -1

        finally:
                pass

        #99.- Cierra conexiones
        myConexion.close()
        print("-> 99.- Fin Migración")
        return varResultado
```

A.4. Código del módulo "creación de la partición"

```
# -*- coding: utf-8 -*-
"""
UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA
Master: Inteligencia Artificial Avanzada
Materia: Trabajo de Fin de Master
Módulo: Partición del DataSet
@author: Sergio Montes Vázquez Noviembre 2019
"""

#0.- Librerias e iniciación de variables
import random
import pymysql.cursors
from mysql.connector import Error

def Crea_Particion(stHost, stUser, stPassword, stDB, numParticiones):
        varResultado = 1
        myConexion = pymysql.connect(host=stHost,user=stUser,password=stPassword,db=stDB)

        print ('-> 0.- Inicia particiones ')

        try:
                with myConexion.cursor() as myCursor:
                        print ('--> 1.- Entra en Cursor MySql')

                        #1.- limpia el campo de la Muestra la tabla de datos
                        print ('--> 2.- Limpia el campo de la Muestra ')
                        stSQL = "UPDATE " + stDB + ".MID_DATOS "
                        stSQL = stSQL + " SET MUESTRA = NULL "
                        stSQL = stSQL + " WHERE MUESTRA IS NOT NULL "
                        myCursor.execute(stSQL)
                        myConexion.commit()

                        idParticion = 0
                        banAvanzar = True

                        #2.- Bucle de particiones hasta que se hayan cubierto
                                todos los registros
```

```
                while banAvanzar == True:
                        stSQL = "SELECT  COUNT(*) "
                        stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
                        stSQL = stSQL + " WHERE MUESTRA IS NULL"
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()
                        numReg = regAux[0][0]
                        if numReg > 0:
                                idRegMarca = random.randrange(numReg)

                                stSQL = " SELECT ID_EMPLEADO, ANNO "
                                stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
                                stSQL = stSQL + " WHERE MUESTRA IS NULL"
                                stSQL = stSQL + " ORDER BY ID_EMPLEADO, ANNO "
                                myCursor.execute(stSQL)
                                regAux = myCursor.fetchall()

                                stEmpleado = ""
                                if type(regAux[idRegMarca][0]) == "str":
                                        stEmpleado = regAux[idRegMarca][0]
                                else:
                                        stEmpleado = str(regAux[idRegMarca][0])

                                stSQL = "UPDATE " + stDB + ".MID_DATOS "
                                stSQL = stSQL + " SET MUESTRA = " + str(idParticion)
                                stSQL = stSQL + " WHERE ID_EMPLEADO  = '" + stEmpleado + "'"
                                stSQL = stSQL + " AND ANNO  = " + str(regAux[idRegMarca][1])
                                myCursor.execute(stSQL)
                                myConexion.commit()
                                idParticion = idParticion + 1

                                if idParticion >= numParticiones:
                                        idParticion = 0
                        else:
                                banAvanzar = False

                stSQL = "SELECT  MUESTRA, count(*) "
                stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
                stSQL = stSQL + " GROUP BY MUESTRA "
                stSQL = stSQL + " ORDER BY MUESTRA "
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for i in range (0,len(regAux)):
                        print(str(i) + '.- ' + str(regAux[i][0]) + ', ' + str(regAux[i][1]) )

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()
                varResultado = -1

        finally:
                pass

        #99.- Cierra conexiones
        myConexion.close()
        print("-> 99.- Fin Particiones")
        return varResultado
```

A.5. Código del módulo “creación de los parámetros”

```
# -*- coding: utf-8 -*-
"""
UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA
Master: Inteligencia Artificial Avanzada
Materia: Trabajo de Fin de Master
Módulo: Creación de Parámetros para los Modelos
@author: Sergio Montes Vázquez
Noviembre 2019
"""

#0.- Librerias e iniciación de variables
import pymysql.cursors
from mysql.connector import Error

def Crea_Parametros(stHost, stUser, stPassword, stDB, numParticiones):
        varResultado = 1
        myConexion = pymysql.connect(host=stHost,user=stUser,password=stPassword,db=stDB)
```

```
        print ('-> 0.- Inicia Creación de Parámetros ')
        try:
                with myConexion.cursor() as myCursor:
                        print ('--> 1.- Entra en Cursor MySql')

                        stVer = "01"
                        vCorte_GainR = 0.75
                        vCorrecionLaplace = 0.5

                        for NumMaxCampos in range(4,11):
                                print ('---> 2.- Número de Campos: ' + str(NumMaxCampos))

                                for banIncluye_Estado_1 in (False,True):
                                        print ('----> 3.- Incluye estado 1: ' + str(banIncluye_Estado_1))

                                        for banDecision  in (False,True):
                                                print ('-----> 4.- Decisión: ' + str(banDecision))

                                                for idTipoDiagrama in (0,1):
                                                        print ('------> 5.- Tipo de Diagrama: ' + str(idTipoDiagrama))

                                                        for idParticion in range(0,numParticiones):
                                                                print ('-------> 6.- Id. Partición: ' + str(idParticion))

                                                                        stSQL = "INSERT INTO " + stDB + ".TB_MODELO_PARM ("
                                                                        stSQL = stSQL + "stVer, NumMaxCampos, banIncluye_Estado_1, "
                                                                        stSQL = stSQL + "banDecision,vCorte_GainR, vCorrecionLaplace,"
                                                                        stSQL = stSQL + " idTipoDiagrama, idParticion, "
                                                                        stSQL = stSQL + "Seleccionado, Ejecutado) "
                                                                        stSQL = stSQL + "VALUES('" + stVer + "',"
                                                                        stSQL = stSQL + str(NumMaxCampos) + ","
                                                                        stSQL = stSQL + str(banIncluye_Estado_1) + ","
                                                                        stSQL = stSQL + str(banDecision) + ", "
                                                                        stSQL = stSQL + str(vCorte_GainR) + ", "
                                                                        stSQL = stSQL + str(vCorrecionLaplace) + ", "
                                                                        stSQL = stSQL + str(idTipoDiagrama) + ", "
                                                                        stSQL = stSQL + str(idParticion) + ", "
                                                                        stSQL = stSQL + "False,False) "

                                                                        myCursor.execute(stSQL)
                                                                        myConexion.commit()

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()
                varResultado = -1

        finally:
                pass

        #99.- Cierra conexiones
        myConexion.close()
        print("-> 99.- Fin Creación de Parámetros")
        return varResultado
```

A.6. Código del módulo “creación de los modelos”

```
# -*- coding: utf-8 -*-
"""
UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA
Master: Inteligencia Artificial Avanzada
Materia: Trabajo de Fin de Master
Módulo: Creación de Modelos
@author: Sergio Montes Vázquez Noviembre 2019
"""

#0.- Librerias e iniciación de variables
import datetime as dt
import pymysql.cursors
from mysql.connector import Error
import math import time

#1.- Subrutinas Auxiliares
#1.1.- Probabilidad de la variable X
def probX(pX):
        matResultado = []
```

```
        for i in range(0,len(pX)):
                banAgregar = True
                for j in range(0,len(matResultado)):
                        if pX[i] == matResultado[j][0]:
                                banAgregar = False
                                matResultado[j][1] = matResultado[j][1] + 1/len(pX)

                if banAgregar:
                        ren = [pX[i],1/len(pX)]
                        matResultado.append(ren)
        return(matResultado)

#1.2.- Probabilidad de la variable Y dado X
def probY_X(pX,pY):
        matResultado = []
        vPX = probX(pX)
        vPY_X = []
        for i in range(0,len(pX)):
                banAgregar = True
                for j in range(0,len(vPY_X)):
                        if pX[i] == vPY_X[j][0] and pY[i] == vPY_X[j][1]:
                                banAgregar = False
                                vPY_X[j][2] = vPY_X[j][2] + 1/len(pX)
                if banAgregar:
                        ren = [pX[i],pY[i],1/len(pX)]
                        vPY_X.append(ren)

        for i in range(0,len(vPY_X)):
                for j in range(0,len(vPX)):
                        if vPY_X[i][0] == vPX[j][0]:
                                ren = [vPY_X[i][0],vPY_X[i][1],vPY_X[i][2]/vPX[j][1]]
                                matResultado.append(ren)

        return(matResultado)

#1.3.- Entropia de la variable X
def EntropiaX(pX):
        varH = 0
        vPX = probX(pX)
        for i in range(0,len(vPX)):
                varH = varH - vPX[i][1]*math.log(vPX[i][1],2)

        return(varH)

#1.4.- Entropia de la variable Y dado X
def EntropiaY_X(pX,pY):
        varH = 0
        vPX = probX(pX)
        vPY_X = probY_X(pX,pY)

        for i in range(0,len(vPX)):
                vSumY_X = 0
                for j in range(0,len(vPY_X)):
                        if vPX[i][0] == vPY_X[j][0]:
                                vSumY_X = vSumY_X - vPY_X[j][2]*math.log(vPY_X[j][2],2)
                varH = varH + vPX[i][1] * vSumY_X

        return(varH)

#1.5.- Diferencia entre fechas
def Dif_Tiempo(Inicio,Fin):
        vAux1 = Inicio.year + 30 * (Inicio.month - 1) + Inicio.day + Inicio.hour / 24 + Inicio.minute / 60 / 24
        vAux1 = vAux1 + Inicio.second / 60 / 60 / 24
        vAux2 = Fin.year + 30 * (Fin.month - 1) + Fin.day + Fin.hour / 24 + Fin.minute / 60 / 24 + Fin.second / 60 / 60 / 24

        vAux = 24 * (vAux2 - vAux1)
        vHoras = int(vAux)
        vAux = (vAux - vHoras) * 60
        vMinutos = int(vAux)
        vAux = (vAux - vMinutos) * 60
        vSegundos = int(vAux)
        stResultado  = str(vHoras) + ":" + str(vMinutos).zfill(2) + ":" + str(vSegundos).zfill(2)
        return(stResultado)

#2.- Subrutinas para gestionar el cálculo de las probsbilidades por Base de Datos
#2.1.- Guarda las probabvilidades
def Guarda_Probabilidad(myConexion, stDB, idParticion, stVariables, stProbabilidad, stExperimento, vInicio, vFin, vTiempo):

        stAux = stProbabilidad
```

```
        idParte = 1
        try:
                with myConexion.cursor() as myCursor:
                        while True:
                                stProb = stAux[0:16000]
                                stAux = stAux[16000:]
                                if len(stProb)>0:
                                        stSQL = "INSERT INTO " + stDB + ".TB_PROBABILIDAD "
                                        stSQL = stSQL + " (idParticion, stVariables, idParte, stProbabilidad,"
                                        stSQL = stSQL + " ORIGEN_EXPERIMENTO, INICIO, FIN, TIEMPO) "
                                        stSQL = stSQL + " VALUES("  + str(idParticion) + ","
                                        stSQL = stSQL + "'" + stVariables + "'," + str(idParte) + ",'" + stProb + "',"
                                        stSQL = stSQL + "'" + stExperimento + "','" + str(vInicio) + "','" + str(vFin) + "',"
                                        stSQL = stSQL + "'" + vTiempo + "')"
                                        idParte = idParte + 1

                                        myCursor.execute(stSQL)
                                        myConexion.commit()
                                else:
                                        break

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()

        finally:
                pass

#2.2.- Busca probabailidad
def Busca_Probabilidad(myConexion, stDB, idParticion, stVariables, stExperimento):
        stProbabilidad = "N/E"

        try:
                with myConexion.cursor() as myCursor:
                        #Registra el uso de la probabilidad
                        stSQL = "SELECT Count(*) FROM " + stDB + ".TB_PROBABILIDAD_USO "
                        stSQL = stSQL + " WHERE idParticion = "  + str(idParticion)
                        stSQL = stSQL + " AND stVariables ='" + stVariables + "'"
                        stSQL = stSQL + " AND DES_EXPERIMENTO ='" + stExperimento + "'"
                        myCursor.execute(stSQL)
                        regCtl = myCursor.fetchall()
                        if regCtl[0][0] == 0:
                                stSQL = "INSERT INTO " + stDB + ".TB_PROBABILIDAD_USO "
                                stSQL = stSQL + " (idParticion, stVariables, DES_EXPERIMENTO) "
                                stSQL = stSQL + " VALUES("  + str(idParticion) + ","
                                stSQL = stSQL + "'" + stVariables + "','" + stExperimento + "')"
                                myCursor.execute(stSQL)

                        #Regresa la Probabilidad
                        stSQL = "SELECT Count(*) FROM " + stDB + ".TB_PROBABILIDAD "
                        stSQL = stSQL + " WHERE idParticion = "  + str(idParticion)
                        stSQL = stSQL + " AND stVariables ='" + stVariables + "'"
                        myCursor.execute(stSQL)
                        regCtl = myCursor.fetchall()

                        if regCtl[0][0] != 0:
                                stSQL = "SELECT stProbabilidad FROM " + stDB + ".TB_PROBABILIDAD "
                                stSQL = stSQL + " WHERE idParticion = "  + str(idParticion)
                                stSQL = stSQL + " AND stVariables ='" + stVariables + "'"
                                stSQL = stSQL + " ORDER BY idParte"
                                myCursor.execute(stSQL)
                                regProb = myCursor.fetchall()
                                stProbabilidad = ""
                                for i in range(0,len(regProb)):
                                        stProbabilidad = stProbabilidad + regProb[i][0]

        except Error as e:
                print("Error reading data from MySQL table", e)
                myConexion.close()
        finally:
                pass

        return(stProbabilidad)

#3.- Subrutina de Creación de Modelos
def Crea_Modelo(stHost, stUser, stPassword, stDB, stRuta, varParametro, stExperimento):

        print('0.- Entro ' + time.strftime("%H:%M:%S"))
```

```
stVer = varParametro[0]
NumMaxCampos = varParametro[1]
banIncluye_Estado_1 = varParametro[2]
banDecision = varParametro[3]
vCorte_GainR = varParametro[4]
vCorrecionLaplace = varParametro[5]
stCorrecionLaplace = str(vCorrecionLaplace)
idTipoDiagrama = varParametro[6]
idParticion = varParametro[7]

stArchivo = "Modelo_v" + stVer
if banIncluye_Estado_1 == True:
        stArchivo = stArchivo + "_ie1"
else:
        stArchivo = stArchivo + "_ne1"
if banDecision == True:
        stArchivo = stArchivo + "_id"
else:
        stArchivo = stArchivo + "_nd"
stCorte_GainR = str(vCorte_GainR)
stArchivo = stArchivo + "_g" + stCorte_GainR.replace(".","") + "_c" + str(NumMaxCampos )
stArchivo = stArchivo + "_cl" + stCorrecionLaplace.replace(".","")
if idTipoDiagrama == 0:
        stTipoDiagrama = "MID"
        stArchivo = stArchivo + "_MID"
if idTipoDiagrama == 1:
        stTipoDiagrama = "InfluenceDiagram"
        stArchivo = stArchivo + "_ID"

if idParticion != -1:
        stArchivo = stArchivo + "_Par_" + str(idParticion)

stArchivo = stArchivo + ".pgmx"

#1.- Conexión e Inicialización Variables
varResultado = 1
myConexion = pymysql.connect(host = stHost, user = stUser, password = stPassword, db = stDB)

vecVariables_Fijas = ['SEXO','NIVEL_EDUCATIVO']
vecVariables = ['SEXO','EDAD','NUM_HIJOS','FAMILIA','PUESTO',
        'GRUPO_PUESTO','ANNO_ANTIGUEDAD','ANTIGUEDAD','SBA_IMPORTE','SBA',
        'PORC_AUMENTO','NIVEL_EDUCATIVO','NUM_CURSOS','FORMACION','DESEMPENNO_NUM',
        'DESEMPENNO','EVP_APRENDIZAJE','EVP_CARRERA','EVP_EQUILIBRIO','EVP_RETRIBUCION',
        'EVP_NUM','EVP','ESTADO']
vecClase = ['EDAD','NUM_HIJOS','FAMILIA','PUESTO', 'GRUPO_PUESTO',
        'ANNO_ANTIGUEDAD','ANTIGUEDAD','SBA_IMPORTE','SBA','PORC_AUMENTO',
        'NUM_CURSOS','FORMACION','DESEMPENNO_NUM', 'DESEMPENNO','EVP_APRENDIZAJE',
        'EVP_CARRERA','EVP_EQUILIBRIO','EVP_RETRIBUCION',
        'EVP_NUM','EVP','ESTADO']
vecVariables = ['SEXO','EDAD','FAMILIA',
        'GRUPO_PUESTO','ANTIGUEDAD','SBA','DES_PORC_AUM',
        'NIVEL_EDUCATIVO','FORMACION',
        'DESEMPENNO','EVP','ESTADO']
vecDecision = ['SBA','FORMACION']
vecClase = ['EDAD','FAMILIA','GRUPO_PUESTO',
        'ANTIGUEDAD','SBA','DES_PORC_AUM',
        'FORMACION', 'DESEMPENNO','EVP','ESTADO']
vecClase_Fija = ['ES_ABANDONO','ES_SEPARACION','FIN_PUESTO']
vecClase_Fija_Conteo = ['B.ID_EMPLEADO','B.ID_EMPLEADO','B.PUESTO']
vecVariables_Utilidad = ['GRUPO_PUESTO','ES_ABANDONO','ES_SEPARACION','FIN_PUESTO']

try:
        with myConexion.cursor() as myCursor:
                print ('--> 1.- Entra en Cursor MySql')

                #1.- Consulta General
                stSQLFiltro = " FROM " + stDB + ".MID_DATOS A, " + stDB + ".MID_DATOS B"
                stSQLFiltro = stSQLFiltro + " WHERE A.ID_EMPLEADO = B.ID_EMPLEADO "
                stSQLFiltro = stSQLFiltro + " AND A.ANNO + 1 = B.ANNO "
                stSQLFiltro = stSQLFiltro + " AND A.ANNO_ANTIGUEDAD IS NOT NULL "
                stSQLFiltro = stSQLFiltro + " AND A.GRUPO_PUESTO <> 'S/D'"
                stSQLFiltro = stSQLFiltro + " AND A.SBA <> 'S/D'"
                stSQLFiltro = stSQLFiltro + " AND A.PUESTO <> 'S/D'"
                stSQLFiltro = stSQLFiltro + " AND A.ESTADO <> 'S/D'"
                stSQLFiltro = stSQLFiltro + " AND B.ESTADO <> 'S/D'"
                stSQLFiltro = stSQLFiltro + " AND A.MUESTRA <> " + str(idParticion) + " "

                #2.- Encabezado Red
                print('1.-- Inicia XML ' + time.strftime("%H:%M:%S"))
```

```
arhcPGMX = open (stRuta  + stArchivo,'w')
arhcPGMX.write('<?xml version="1.0" encoding="UTF-8"?>\n')
arhcPGMX.write('<ProbModelXML formatVersion="0.2.0">\n')
arhcPGMX.write('  <ProbNet type="' + stTipoDiagrama + '">\n')
arhcPGMX.write('    <Comment showWhenOpeningNetwork="false">
        <![CDATA[<<Pulse dos veces para incluir/modificar comentario>>]]></Comment>\n')
arhcPGMX.write('    <DecisionCriteria>\n')
arhcPGMX.write('      <Criterion name="---" unit="---" />\n')
arhcPGMX.write('    </DecisionCriteria>\n')
if idTipoDiagrama == 0:
        arhcPGMX.write('    <TimeUnit unit="YEAR" Value="1.0" />\n')
arhcPGMX.write('    <AdditionalProperties />\n')

#3.- Variables
arhcPGMX.write('    <Variables>\n')
vecNodo = []
#3.1.- Variables Estátitcas
print('2.-- Variables Estátitcas ' + time.strftime("%H:%M:%S"))
posX = 100
posY = 100
for i in range(0,len(vecVariables_Fijas)):
        posY = 200 * (i+1)
        arhcPGMX.write('      <Variable name="' + vecVariables_Fijas[i]
                + '" type="finiteStates" role="chance">\n')
        arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="' + str(posY) + '" />\n')
        arhcPGMX.write('        <States>\n')
        stSQL = "SELECT A." + vecVariables_Fijas[i]
        stSQL = stSQL + stSQLFiltro
        stSQL = stSQL + " GROUP BY A." + vecVariables_Fijas[i]
        stSQL = stSQL + " ORDER BY A." + vecVariables_Fijas[i]
        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()
        vecEstados = []
        for row in regAux:
                vecEstados.append(row[0])

        #Revisa si hay orden en los valores
        NumOrden = 0
        stSQL = "SELECT COUNT(*) FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
        stSQL = stSQL +  vecVariables_Fijas[i] + "'"
        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()
        for resSQL in regAux:
                NumOrden = resSQL[0]
        if NumOrden > 0:
                auxEstados = vecEstados
                vecEstados = []
                stSQL = "SELECT VALOR FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
                stSQL = stSQL +  vecVariables_Fijas[i] + "' ORDER BY ORDEN"
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for resSQL in regAux:
                        for row in auxEstados:
                                if resSQL[0] == row:
                                        vecEstados.append(row)
        for row in vecEstados:
                arhcPGMX.write('          <State name="' + row + '" />\n')
        arhcPGMX.write('        </States>\n')
        arhcPGMX.write('      </Variable>\n')
        vecNodo.append([vecVariables_Fijas[i],vecEstados])

#3.2.- Variables Estado [0]
print('3.-- Variables Estado 0 ' + time.strftime("%H:%M:%S"))
posX = 500
if banDecision == True:
        posX = 300
posY = 100
k = 1
for i in range(0,len(vecVariables)):
        banAgregar = True
        for j in range(0,len(vecVariables_Fijas)):
                if vecVariables[i] == vecVariables_Fijas[j]:
                        banAgregar = False

        stRole="chance"
        if banDecision == True:
                for row in vecDecision:
                        if vecVariables[i] == row:
```

```
                                banAgregar = False
                                stRole="decision"
                if banAgregar == True:
                        posY = 50 * k
                        k = k + 1

                        stNombreNodo = vecVariables[i] + '" timeSlice="0"'
                        if idTipoDiagrama == 1:
                                stNombreNodo = vecVariables[i] + '_0"'

                        arhcPGMX.write('      <Variable name="' + stNombreNodo +
                                ' type="finiteStates" role="' + stRole + '">\n')
                        arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="' + str(posY) + '" />\n')
                        arhcPGMX.write('        <States>\n')
                        stSQL = "SELECT A." + vecVariables[i]
                        stSQL = stSQL + stSQLFiltro
                        stSQL = stSQL + " GROUP BY A." + vecVariables[i]
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()
                        vecEstados = []
                        for row in regAux:
                                vecEstados.append(row[0])

                        stSQL = "SELECT B." + vecVariables[i]
                        stSQL = stSQL + stSQLFiltro
                        stSQL = stSQL + " GROUP BY B." + vecVariables[i]
                        stSQL = stSQL + " ORDER BY B." + vecVariables[i]
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()
                        for row in regAux:
                                banAgregar = True
                                for j in range(0,len(vecEstados)):
                                        if vecEstados[j]==row[0]:
                                                banAgregar = False
                                if banAgregar == True:
                                        vecEstados.append(row[0])

                        #Revisa si hay orden en los valores
                        NumOrden = 0
                        stSQL = "SELECT COUNT(*) FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
                        stSQL = stSQL +  vecVariables[i] + "'"
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()
                        for resSQL in regAux:
                                NumOrden = resSQL[0]
                        if NumOrden > 0:
                                auxEstados = vecEstados
                                vecEstados = []
                                stSQL = "SELECT VALOR FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
                                stSQL = stSQL +  vecVariables[i] + "' ORDER BY ORDEN"
                                myCursor.execute(stSQL)
                                regAux = myCursor.fetchall()
                                for resSQL in regAux:
                                        for row in auxEstados:
                                                if resSQL[0] == row:
                                                        vecEstados.append(row)

                        for row in vecEstados:
                                arhcPGMX.write('          <State name="' + str(row) + '" />\n')

                        arhcPGMX.write('        </States>\n')
                        arhcPGMX.write('      </Variable>\n')
                        vecNodo.append([vecVariables[i],vecEstados])

#3.3.- Variables Estado [0]
print('4.-- Variables Decisión estado 0 ' + time.strftime("%H:%M:%S"))
posX = 500
if banDecision == True:
        for i in range(0,len(vecDecision)):
                banAgregar = True
                for j in range(0,len(vecVariables_Fijas)):
                        if vecDecision[i] == vecVariables_Fijas[j]:
                                banAgregar = False

                if banAgregar == True:
                        posX = posX + 75 * i
                        posY = posY + 75 * i
                        k = k + 1
```

```
                stNombreNodo = vecDecision[i] + '" timeSlice="0"'
                if idTipoDiagrama == 1:
                        stNombreNodo = vecDecision[i] + '_0"'

                arhcPGMX.write('      <Variable name="' + stNombreNodo
                        + ' type="finiteStates" role="decision">\n')
                arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="'
                        + str(posY) + '" />\n')
                arhcPGMX.write('        <States>\n')
                stSQL = "SELECT A." + vecDecision[i]
                stSQL = stSQL + stSQLFiltro
                stSQL = stSQL + "GROUP BY A." + vecDecision[i]
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                vecEstados = []
                for row in regAux:
                        vecEstados.append(row[0])

                stSQL = "SELECT B." + vecDecision[i]
                stSQL = stSQL + stSQLFiltro
                stSQL = stSQL + "GROUP BY B." + vecDecision[i]
                stSQL = stSQL + " ORDER BY B." + vecDecision[i]
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for row in regAux:
                        banAgregar = True
                        for j in range(0,len(vecEstados)):
                                if vecEstados[j]==row[0]:
                                        banAgregar = False
                        if banAgregar == True:
                                vecEstados.append(row[0])

                #Revisa si hay orden en los valores
                NumOrden = 0
                stSQL = "SELECT COUNT(*) FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
                stSQL = stSQL +  vecDecision[i] + "'"
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for resSQL in regAux:
                        NumOrden = resSQL[0]
                if NumOrden > 0:
                        auxEstados = vecEstados
                        vecEstados = []
                        stSQL = "SELECT VALOR FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
                        stSQL = stSQL +  vecDecision[i] + "' ORDER BY ORDEN"
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()
                        for resSQL in regAux:
                                for row in auxEstados:
                                        if resSQL[0] == row:
                                                vecEstados.append(row)

                for row in vecEstados:
                        arhcPGMX.write('          <State name="' + str(row) + '" />\n')

                arhcPGMX.write('        </States>\n')
                arhcPGMX.write('      </Variable>\n')
                vecNodo.append([vecDecision[i],vecEstados])

#3.4.- Variables Estado [1], Clases
print('5.-- Variables Estado 1 ' + time.strftime("%H:%M:%S"))
posX = 900
posY = 100
k = 1
for i in range(0,len(vecClase)):
        posY = 50 * (i+1)
        k = k + 1

        stNombreNodo = vecClase[i] + '" timeSlice="1"'
        if idTipoDiagrama == 1:
                stNombreNodo = vecClase[i] + '_1"'

        arhcPGMX.write('      <Variable name="' + stNombreNodo + ' type="finiteStates" role="chance">\n')
        arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="' + str(posY) + '" />\n')
        arhcPGMX.write('        <States>\n')
        for row in vecNodo:
                if row[0] == vecClase[i]:
                        for col in row[1]:
                                arhcPGMX.write('          <State name="' + str(col) + '" />\n')
```

```
        arhcPGMX.write('        </States>\n')
        arhcPGMX.write('      </Variable>\n')

#3.5.- Variables Clases Fija
print('6.-- Variables Estado Fijo ' + time.strftime("%H:%M:%S"))
posX = 1200
posY = 500
k = 1
for i in range(0,len(vecClase_Fija)):
        posY = 100 * (i+1)
        k = k + 1

        stNombreNodo = vecClase_Fija[i] + '" timeSlice="1"'
        if idTipoDiagrama == 1:
                stNombreNodo = vecClase_Fija[i] + '_1"'

        arhcPGMX.write('      <Variable name="' + stNombreNodo + ' type="finiteStates" role="chance">\n')
        arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="' + str(posY) + '" />\n')
        arhcPGMX.write('        <States>\n')

        stSQL = "SELECT B." + vecClase_Fija[i]
        stSQL = stSQL + stSQLFiltro
        stSQL = stSQL + "GROUP BY B." + vecClase_Fija[i]
        stSQL = stSQL + " ORDER BY B." + vecClase_Fija[i]
        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()
        vecEstados = []
        for row in regAux:
                vecEstados.append(row[0])

        #Revisa si hay orden en los valores
        NumOrden = 0
        stSQL = "SELECT COUNT(*) FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
        stSQL = stSQL +  vecClase_Fija[i] + "'"
        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()
        for resSQL in regAux:
                NumOrden = resSQL[0]
        if NumOrden > 0:
                auxEstados = vecEstados
                vecEstados = []
                stSQL = "SELECT VALOR FROM " + stDB + ".CAMPO_VALOR_ORDEN WHERE CAMPO ='"
                stSQL = stSQL +  vecClase_Fija[i] + "' ORDER BY ORDEN"
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for resSQL in regAux:
                        for row in auxEstados:
                                if resSQL[0] == row:
                                        vecEstados.append(row)

        for row in vecEstados:
                arhcPGMX.write('          <State name="' + str(row) + '" />\n')

        arhcPGMX.write('        </States>\n')
        arhcPGMX.write('      </Variable>\n')
        vecNodo.append([vecClase_Fija[i],vecEstados])

#3.6.- Utilidad / Costos de las Decisiones
print('7.-- Utilidad / Costos de las Decisiones ' + time.strftime("%H:%M:%S"))
posX = 1500
posY = 300

stNombreNodo = 'UTILIDAD" timeSlice="1"'
if idTipoDiagrama == 1:
        stNombreNodo = 'UTILIDAD_1"'

arhcPGMX.write('      <Variable name="' + stNombreNodo + ' type="numeric" role="utility">\n')
arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="' + str(posY) + '" />\n')
if idTipoDiagrama == 1:
        arhcPGMX.write('        <AdditionalProperties>\n')
        arhcPGMX.write('          <Property name="Purpose" value="cost" />\n')
        arhcPGMX.write('        </AdditionalProperties>\n')

arhcPGMX.write('        <Unit />\n')
arhcPGMX.write('        <Precision>0.01</Precision>\n')
arhcPGMX.write('        <Criterion name="---" />\n')
arhcPGMX.write('      </Variable>\n')
```

```
for i in range(0,len(vecDecision)):
        posX = 300 + i * 200
        posY = 650 + i * 50

        stNombreNodo = 'COSTO_' + vecDecision[i] + '" timeSlice="0"'
        if idTipoDiagrama == 1:
                stNombreNodo = 'COSTO_' + vecDecision[i] + '_0"'

        arhcPGMX.write('      <Variable name="' + stNombreNodo + ' type="numeric" role="utility">\n')
        arhcPGMX.write('        <Coordinates x="' + str(posX) + '" y="' + str(posY) + '" />\n')
        arhcPGMX.write('        <Unit />\n')
        arhcPGMX.write('        <Precision>0.01</Precision>\n')
        arhcPGMX.write('        <Criterion name="---" />\n')
        arhcPGMX.write('      </Variable>\n')

arhcPGMX.write('    </Variables>\n')

arhcPGMX.write('    <Links>\n')

print('8.-- Links Variables -> Nodos Decisión ' + time.strftime("%H:%M:%S"))
vecLinksDecision = []
for i in range(0,len(vecDecision)):
        vecNodoLink = []
        for j in range(0,len(vecVariables)):
                banAgregar = True
                for k in range(0,len(vecDecision)):
                        if vecVariables[j] == vecDecision[k]:
                                banAgregar = False
                if banAgregar == True:
                        stTimeSlice = 'timeSlice="0"'
                        for k in range(0,len(vecVariables_Fijas)):
                                if vecVariables[j] == vecVariables_Fijas[k]:
                                        stTimeSlice = ''

                        stNombreNodo = vecVariables[j] + '" ' + stTimeSlice
                        if idTipoDiagrama == 1 and stTimeSlice == 'timeSlice="0"':
                                stNombreNodo = vecVariables[j] + '_0"'

                        arhcPGMX.write('      <Link directed="true">\n')
                        arhcPGMX.write('        <Variable name="' +  stNombreNodo + ' />\n')

                        stNombreNodo = vecDecision[i] + '" timeSlice="0"'
                        if idTipoDiagrama == 1 :
                                stNombreNodo = vecDecision[i] + '_0"'

                        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')
                        arhcPGMX.write('      </Link>\n')
                        vecNodoLink.append(vecVariables[j])
        vecLinksDecision.append([vecDecision[i],vecNodoLink])

print('9.-- Links Clases ' + time.strftime("%H:%M:%S"))
#4.- Reducción de varaiables y links Clases

vecLinks = []
for iClase in range(0,len(vecClase)):
        #4.1.- Recuperación de datos
        i = len(vecClase) - iClase - 1

        stSQL = "SELECT "
        for j in range(0,len(vecVariables)):
                stSQL = stSQL + "A." + vecVariables[j] + ", "
        vecClase_1 = []
        if banIncluye_Estado_1 == True:
                for j in range(0,len(vecClase)):
                        if vecClase[i] != vecClase[j]:
                                stSQL = stSQL + "B." + vecClase[j] + " " + vecClase[j] + "_1, "
                                vecClase_1.append(vecClase[j] + "_1")

        stSQL = stSQL + "B." + vecClase[i] + " CLASE"
        stSQL = stSQL + stSQLFiltro

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        matDatos=[]
        for row in regAux:
                vRec = []
                for k in range(0,len(row)):
                        vRec.append(row[k])
```

```
        matDatos.append(vRec)

#4.2.- Cálculo de la Ganancia de la Información y del Ratio de Ganancia
matGainR = []
sumGainR = 0
for j in range(0,len(matDatos[0])-1):
        auxColX = [row[j] for row in matDatos]
        auxColY = [row[len(matDatos[0])-1] for row in matDatos]
        IG_Y_X = EntropiaX(auxColY)-EntropiaY_X(auxColX,auxColY)
        GainR_Y_X = IG_Y_X/EntropiaX(auxColX)
        if j < len(vecVariables):
                vRen = [vecVariables[j],IG_Y_X,GainR_Y_X,0,0]
        else:
                vRen = [vecClase_1[j-len(vecVariables)],IG_Y_X,GainR_Y_X,0,0]
        matGainR.append(vRen)
        sumGainR = sumGainR + IG_Y_X

#4.3.- Ordenación de los valores por Ratio
for j in range(0,len(matGainR)-1):
        for k in range (j+1,len(matGainR)):
                if matGainR[j][1] < matGainR[k][1]:
                        vRen = matGainR[j]
                        matGainR[j] = matGainR[k]
                        matGainR[k] = vRen

#4.4.- Cálculo de los acumulados y selección de las variables
vAcumulado = 0
vecNodoLink = []
vCorte = float(vCorte_GainR)
maxCampo = NumMaxCampos
for j  in range(0,len(matGainR)):
        vAcumulado = vAcumulado + matGainR[j][1]
        matGainR[j][3] = vAcumulado
        matGainR[j][4] = vAcumulado / sumGainR
        banAgregar = True
        stAuxCampo = matGainR[j][0]
        if stAuxCampo.find("_1") != -1:
                stAuxCampo = stAuxCampo.replace("_1","")
                for k in range(0,len(vecLinks)):
                        if vecLinks[k][0] == stAuxCampo:
                                banAgregar = False

        if banAgregar == True:
                if vCorte > 0 and maxCampo > 0:
                        vecNodoLink.append(matGainR[j][0])
                        maxCampo = maxCampo - 1
                vCorte = vCorte - matGainR[j][1] / sumGainR

#4.5.- Creación de los links
vecNodos = []
for j in range(0,len(vecNodoLink)):

        stTimeSlice = ' timeSlice="0"'
        stNodoLink = vecNodoLink[j]
        for k in range(0,len(vecVariables_Fijas)):
                if vecNodoLink[j] == vecVariables_Fijas[k]:
                        stTimeSlice = ''

        if stNodoLink.find("_1") != -1:
                stTimeSlice = ' timeSlice="1"'
                stNodoLink = stNodoLink[:-2]

        arhcPGMX.write('      <Link directed="true">\n')

        stNombreNodo = stNodoLink + '" ' + stTimeSlice
        if idTipoDiagrama == 1 and stTimeSlice == ' timeSlice="0"':
                stNombreNodo = stNodoLink + '_0"'
        if idTipoDiagrama == 1 and stTimeSlice == ' timeSlice="1"':
                stNombreNodo = stNodoLink + '_1"'

        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')

        stNombreNodo = vecClase[i] + '" timeSlice="1"'
        if idTipoDiagrama == 1:
                stNombreNodo = vecClase[i] + '_1"'

        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')
        arhcPGMX.write('      </Link>\n')
        vecNodos.append(vecNodoLink[j])
```

```
        vecLinks.append([vecClase[i],vecNodos])

#5.- Reducción de varaiables y links Clases Fijas
print('10.-- Links Clases Fijas ' + time.strftime("%H:%M:%S"))
for i in range(0,len(vecClase_Fija)):
        #5.1.- Recuperación de datos

        stSQL = "SELECT "
        for j in range(0,len(vecVariables)):
                stSQL = stSQL + "A." + vecVariables[j] + ", "
        vecClase_1 = []

        for j in range(0,len(vecClase)):
                stSQL = stSQL + "B." + vecClase[j] + " " + vecClase[j] + "_1, "
                vecClase_1.append(vecClase[j] + "_1")

        stSQL = stSQL + "B." + vecClase_Fija[i] + " CLASE"
        stSQL = stSQL + stSQLFiltro

        myCursor.execute(stSQL)
        regAux = myCursor.fetchall()

        matDatos=[]
        for row in regAux:
                vRec = []
                for k in range(0,len(row)):
                        vRec.append(row[k])
                matDatos.append(vRec)

        #5.2.- Cálculo de la Ganancia de la Información y del Ratio de Ganancia
        matGainR = []
        sumGainR = 0
        for j in range(0,len(matDatos[0])-1):
                auxColX = [row[j] for row in matDatos]
                auxColY = [row[len(matDatos[0])-1] for row in matDatos]
                IG_Y_X = EntropiaX(auxColY)-EntropiaY_X(auxColX,auxColY)
                GainR_Y_X = IG_Y_X/EntropiaX(auxColX)
                if j < len(vecVariables):
                        vRen = [vecVariables[j],IG_Y_X,GainR_Y_X,0,0]
                else:
                        vRen = [vecClase_1[j-len(vecVariables)],IG_Y_X,GainR_Y_X,0,0]
                matGainR.append(vRen)
                sumGainR = sumGainR + IG_Y_X

        #5.3.- Ordenación de los valores por Ratio
        for j in range(0,len(matGainR)-1):
                for k in range (j+1,len(matGainR)):
                        if matGainR[j][1] < matGainR[k][1]:
                                vRen = matGainR[j]
                                matGainR[j] = matGainR[k]
                                matGainR[k] = vRen

        #5.4.- Cálculo de los acumulados y selección de las variables
        vAcumulado = 0
        vecNodoLink = []
        vCorte = float(vCorte_GainR)
        maxCampo = NumMaxCampos
        for j  in range(0,len(matGainR)):
                vAcumulado = vAcumulado + matGainR[j][1]
                matGainR[j][3] = vAcumulado
                matGainR[j][4] = vAcumulado / sumGainR
                if vCorte > 0 and maxCampo > 0:
                        vecNodoLink.append(matGainR[j][0])
                        maxCampo = maxCampo - 1
                vCorte = vCorte - matGainR[j][1] / sumGainR

        #5.5.- Creación de los links
        vecNodos = []
        for j in range(0,len(vecNodoLink)):

                stTimeSlice = ' timeSlice="0"'
                stNodoLink = vecNodoLink[j]
                for k in range(0,len(vecVariables_Fijas)):
                        if vecNodoLink[j] == vecVariables_Fijas[k]:
                                stTimeSlice = ''

                if stNodoLink.find("_1") != -1:
                        stTimeSlice = ' timeSlice="1"'
```

```
                        stNodoLink = stNodoLink[:-2]

                arhcPGMX.write('      <Link directed="true">\n')

                stNombreNodo = stNodoLink + '" ' + stTimeSlice
                if idTipoDiagrama == 1:
                        if stTimeSlice == ' timeSlice="0"':
                                stNombreNodo =  stNodoLink + '_0"'
                        if stTimeSlice == ' timeSlice="1"':
                                stNombreNodo =  stNodoLink + '_1"'

                arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')

                stNombreNodo = vecClase_Fija[i] + '" timeSlice="1"'
                if idTipoDiagrama == 1:
                        stNombreNodo = vecClase_Fija[i] + '_1"'

                arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')
                arhcPGMX.write('      </Link>\n')
                vecNodos.append(vecNodoLink[j])

        vecLinks.append([vecClase_Fija[i] , vecNodos])

print('11.-- Links para las Utilidades / Costos ' + time.strftime("%H:%M:%S"))
for row in vecVariables_Utilidad:
        arhcPGMX.write('      <Link directed="true">\n')

        stNombreNodo = row + '" timeSlice="1"'
        if idTipoDiagrama == 1:
                stNombreNodo = row + '_1"'

        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')

        stNombreNodo = 'UTILIDAD" timeSlice="1"'
        if idTipoDiagrama == 1:
                stNombreNodo = 'UTILIDAD_1"'

        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')
        arhcPGMX.write('      </Link>\n')

for i in range(0,len(vecDecision)):
        arhcPGMX.write('      <Link directed="true">\n')

        stNombreNodo = vecDecision[i] + '" timeSlice="0"'
        if idTipoDiagrama == 1:
                stNombreNodo = vecDecision[i] + '_0"'

        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')

        stNombreNodo = 'COSTO_' + vecDecision[i] + '" timeSlice="0"'
        if idTipoDiagrama == 1:
                stNombreNodo = 'COSTO_' + vecDecision[i] + '_0"'

        arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')
        arhcPGMX.write('      </Link>\n')

arhcPGMX.write('    </Links>\n')

#6.-Obtención de las probabilidades
arhcPGMX.write('    <Potentials>\n')
#6.1.- Probabilidades Variables Fijas y Estado 0
print('12.-- Potencias Variables Fijas y Estado 0 ' + time.strftime("%H:%M:%S"))
for row in vecNodo:
        banAvanzar = True

        for j in range(0,len(vecClase_Fija)):
                if row[0] == vecClase_Fija[j]:
                        banAvanzar = False

        if banDecision == True:
                for j in range(0,len(vecDecision)):
                        if row[0] == vecDecision[j]:
                                banAvanzar = False

        if banAvanzar == True:
                arhcPGMX.write('      <Potential type="Table" role="joinProbability">\n')
                arhcPGMX.write('        <Variables>\n')
                stTimeSlice = ' timeSlice="0"'
                stVariables = row[0] + "_0"
```

```
                for k in range(0,len(vecVariables_Fijas)):
                        if row[0] == vecVariables_Fijas[k]:
                                stTimeSlice = ''
                                stVariables = row[0]

                stNombreNodo = row[0] + '" ' + stTimeSlice
                if idTipoDiagrama == 1 and stTimeSlice == ' timeSlice="0"':
                        stNombreNodo = row[0] + '_0"'

                arhcPGMX.write('          <Variable name="' + stNombreNodo + '  />\n')
                arhcPGMX.write('        </Variables>\n')
                vecProb = []
                varTotal = 0
                stProb = Busca_Probabilidad(myConexion, stDB, idParticion, stVariables, stExperimento)
                if stProb == "N/E":
                        Inicio = dt.datetime.today()
                        stProb = ""
                        for valor in row[1]:
                                stSQL = "SELECT COUNT(*) FROM " + stDB + ".MID_DATOS WHERE " + row[0]
                                stSQL = stSQL + "='" + str(valor) + "' AND ANNO=2017 AND MUESTRA <> "
                                stSQL = stSQL + str(idParticion)
                                myCursor.execute(stSQL)
                                regAux = myCursor.fetchall()
                                for resSQL in regAux:
                                        vecProb.append(resSQL[0])
                                        varTotal =  varTotal + resSQL[0]

                               #Correción de Laplace
                                if varTotal + vCorrecionLaplace * len(vecProb) > 0 :
                                        for i in range(0,len(vecProb)):
                                                vecProb[i] = (vecProb[i] + vCorrecionLaplace ) /
                                                        (varTotal + vCorrecionLaplace * len(vecProb))

                                for i in range(0,len(vecProb)):
                                        stProb = stProb + ' ' + str(vecProb[i])
                                Fin = dt.datetime.today()
                                Tiempo = Dif_Tiempo(Inicio,Fin)
                                Guarda_Probabilidad(myConexion, stDB, idParticion, stVariables, stProb,
                                        stExperimento, Inicio, Fin, Tiempo)
                arhcPGMX.write('        <Values>' + stProb +'</Values>\n')
                arhcPGMX.write('      </Potential>\n')

#6.2.- Probabilidades Variables Estado 1
print('13.-- Potencias Estado 1 ' + time.strftime("%H:%M:%S"))
for row in vecLinks:

        stConteo = '*'
        stTimeSlice = ' timeSlice="1"'

        arhcPGMX.write('      <Potential type="Table" role="conditionalProbability">\n')
        arhcPGMX.write('        <Variables>\n')

        stNombreNodo = row[0] + '" ' + stTimeSlice
        if idTipoDiagrama == 1 and stTimeSlice == ' timeSlice="1"':
                stNombreNodo = row[0] + '_1"'

        stVariables = row[0] + '_1'
        arhcPGMX.write('          <Variable name="' + stNombreNodo + ' />\n')
        vClaseVal = []
        for auxNodo in vecNodo:
                if row[0] == auxNodo[0]:
                        vClaseVal = auxNodo[1]
        vecMax = []
        vecCon = []
        vVariableVal = []

        for auxVar in row[1]:
                auxStVar = auxVar + '_0'
                stTimeSlice = ' timeSlice="0"'
                for k in range(0,len(vecVariables_Fijas)):
                        if auxVar == vecVariables_Fijas[k]:
                                stTimeSlice = ''
                                auxStVar = auxVar

                if auxVar.find("_1") != -1:
                        stTimeSlice = ' timeSlice="1"'
                        auxVar = auxVar[:-2]
                        auxStVar = auxVar + '_1'
```

```
        for k in range(0,len(vecClase_Fija)):
                if auxVar == vecClase_Fija[k]:
                        stTimeSlice = ' timeSlice="1"'
                        stConteo = 'DISTINCT ' + vecClase_Fija_Conteo[k]
                        auxStVar = auxVar + '_1'

        stVariables =  stVariables + "," + auxStVar
        stNombreNodo = auxVar + '" ' + stTimeSlice
        if idTipoDiagrama == 1:
                if stTimeSlice == ' timeSlice="0"':
                        stNombreNodo = auxVar + '_0"'
                if stTimeSlice == ' timeSlice="1"':
                        stNombreNodo = auxVar + '_1"'

        arhcPGMX.write('          <Variable name="' + stNombreNodo + ' />\n')
        for auxNodo in vecNodo:
                if auxVar == auxNodo[0]:
                        vVariableVal.append(auxNodo[1])
                        vecMax.append(len(auxNodo[1])-1)
                        vecCon.append(0)

vecProb = []
banAvanzar = True
while True:
        vecCampos = []
        for i in range(0,len(vecCon)):
                vecCampos.append(vVariableVal[i][vecCon[i]])
        vecProb.append(vecCampos)

        auxCol = 0 #(len(vecCon)-1)
        vecCon[auxCol] = vecCon[auxCol] + 1
        for i in range(0,len(vecCon)):
                auxCol = i    #(len(vecCon)-1) - i
                if vecCon[auxCol] > vecMax[auxCol]:
                        vecCon[auxCol] = 0
                        auxCol = auxCol + 1
                        if auxCol < len(vecCon):
                                vecCon[auxCol] = vecCon[auxCol] + 1
                        else:
                                banAvanzar = False
        if banAvanzar == False:
                break

stProb = Busca_Probabilidad(myConexion, stDB, idParticion, stVariables, stExperimento)
if stProb == "N/E":
        Inicio = dt.datetime.today()
        stProb = ""
        for vProb in vecProb:
                stSQL = "SELECT  Count(" + stConteo + ")"
                stSQL = stSQL + stSQLFiltro
                stMismoCampoEstado_0 = ""
                for i in range(0,len(vProb)):
                        stCampo = "A." + row[1][i]
                        if row[1][i].find("_1") != -1:
                                stCampo = "B." + row[1][i][:-2]
                        else:
                                if row[1][i] == row[0]:
                                        stMismoCampoEstado_0  = " AND " + stCampo + "='"
                                        stMismoCampoEstado_0  = stMismoCampoEstado_0 + str(vProb[i]) + "'"
                        stSQL = stSQL + " AND " + stCampo + "='" + str(vProb[i]) + "'"

                vecProbRes = []
                varTotal = 0
                for i in range(0,len(vClaseVal)):
                        myCursor.execute(stSQL + " AND B." + row[0] +"='" + str(vClaseVal[i]) + "'")
                        regAux = myCursor.fetchall()
                        for resSQL in regAux:
                                vecProbRes.append(resSQL[0])
                                varTotal =  varTotal + resSQL[0]

                if varTotal == 0 :

                        stSQL = "SELECT  Count(" + stConteo + ")"
                        stSQL = stSQL + stSQLFiltro
                        myCursor.execute(stSQL)
                        regAux = myCursor.fetchall()
                        for resSQL in regAux:
                                varTotal = resSQL[0]
                        if varTotal > 0:
```

```
                                        for i in range(0,len(vClaseVal)):
                                                myCursor.execute(stSQL + " AND B." + row[0] +"='"
                                                        + str(vClaseVal[i]) + "' " )
                                                for resSQL in regAux:
                                                        vecProbRes[i]=resSQL[0]

                                #Correción de Laplace
                                if varTotal + vCorrecionLaplace * len(vClaseVal) > 0 :
                                        for i in range(0,len(vecProbRes)):
                                                vecProbRes[i] = (vecProbRes[i] + vCorrecionLaplace ) /
                                                        (varTotal + vCorrecionLaplace * len(vClaseVal))

                                for i in range(0,len(vecProbRes)):
                                        stProb = stProb + ' ' + str(vecProbRes[i])

                        Fin = dt.datetime.today()
                        Tiempo = Dif_Tiempo(Inicio,Fin)
                        Guarda_Probabilidad(myConexion, stDB, idParticion, stVariables, stProb,
                                stExperimento, Inicio, Fin, Tiempo)

                arhcPGMX.write('        </Variables>\n')
                arhcPGMX.write('        <Values>' + stProb + '</Values>\n')
                arhcPGMX.write('      </Potential>\n')

        #7.- Cálculo de Utilidad de Estado Final
        print('14.-- Cálculo de las Utilidades ' + time.strftime("%H:%M:%S"))
        if idTipoDiagrama == 0:
                #7.1.-  Markov Influence Diagram
                arhcPGMX.write('       <Potential type="Tree/ADD" role="utility">\n')

                stNombreNodo = 'UTILIDAD" timeSlice="1"'
                if idTipoDiagrama == 1:
                        stNombreNodo = 'UTILIDAD_1"'

                arhcPGMX.write('        <UtilityVariable name="' + stNombreNodo + ' />\n')
                arhcPGMX.write('        <Variables>\n')
                for row in vecVariables_Utilidad:

                        stNombreNodo = row + '" timeSlice="1"'
                        if idTipoDiagrama == 1:
                                stNombreNodo = row + '_1"'

                        arhcPGMX.write('          <Variable name="' + stNombreNodo + ' />\n')
                arhcPGMX.write('        </Variables>\n')

                stNombreNodo = 'FIN_PUESTO" timeSlice="1"'
                if idTipoDiagrama == 1:
                        stNombreNodo = 'FIN_PUESTO_1"'

                arhcPGMX.write('        <TopVariable name="' + stNombreNodo + ' />\n')
                arhcPGMX.write('        <Branches>\n')

                #7.1.1.- Rama cuando no hay fin de plaza
                arhcPGMX.write('          <Branch>\n')
                arhcPGMX.write('            <States>\n')
                arhcPGMX.write('              <State name="0" />\n')
                arhcPGMX.write('            </States>\n')
                arhcPGMX.write('            <Potential type="Tree/ADD">\n')

                stNombreNodo = 'UTILIDAD" timeSlice="1"'
                if idTipoDiagrama == 1:
                        stNombreNodo = 'UTILIDAD_1"'

                arhcPGMX.write('              <UtilityVariable name="' + stNombreNodo + ' />\n')
                arhcPGMX.write('              <Variables>\n')
                for row in vecVariables_Utilidad:

                        stNombreNodo = row + '" timeSlice="1"'
                        if idTipoDiagrama == 1:
                                stNombreNodo = row + '_1"'

                        arhcPGMX.write('                <Variable name="' + stNombreNodo + ' />\n')
                arhcPGMX.write('              </Variables>\n')

                stNombreNodo = 'GRUPO_PUESTO" timeSlice="1"'
                if idTipoDiagrama == 1:
                        stNombreNodo = 'GRUPO_PUESTO_1"'

                arhcPGMX.write('              <TopVariable name="' + stNombreNodo + ' />\n')
```

```
                archhPGMX.write('              <Branches>\n')
                for row in vecNodo:
                        if row[0] == 'GRUPO_PUESTO':
                                for i in range(0,len(row[1])):
                                        stSQL = "SELECT "
                                        stSQL = stSQL + " AVG(SBA_IMPORTE * 0.15) PERMANENCIA, "
                                        stSQL = stSQL + " (-1) * AVG(SBA_IMPORTE * 0.15 + SEPARACION ) SEPARACION, "
                                        stSQL = stSQL + " (-1) * AVG(SBA_IMPORTE * 0.15 + ABANDONO ) ABANDONO "
                                        stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
                                        stSQL = stSQL + " WHERE ANNO >= 2017 "
                                        stSQL = stSQL + " AND MUESTRA <> " + str(idParticion) + " "
                                        stSQL = stSQL + " AND GRUPO_PUESTO = '" + row[1][i]  + "'"
                                        myCursor.execute(stSQL)
                                        regAux = myCursor.fetchall()
                                        stUtilidades = "0 0 0 0"
                                        for resSQL in regAux:
                                                stUtilidades = str(resSQL[0]) + " " + str(resSQL[1])  + " "
                                                stUtilidades = stUtilidades + str(resSQL[2]) + " 0"
                                        archPGMX.write('                <Branch>\n')
                                        archPGMX.write('                  <States>\n')
                                        archPGMX.write('                    <State name="' + row[1][i] + '" />\n')
                                        archPGMX.write('                  </States>\n')
                                        archPGMX.write('                  <Potential type="Table">\n')

                                        stNombreNodo = 'UTILIDAD" timeSlice="1"'
                                        if idTipoDiagrama == 1:
                                                stNombreNodo = 'UTILIDAD_1"'

                                        archPGMX.write('                    <UtilityVariable name="'
                                                + stNombreNodo + ' />\n')
                                        archPGMX.write('                    <Variables>\n')

                                        stNombreNodo = 'ES_SEPARACION" timeSlice="1"'
                                        if idTipoDiagrama == 1:
                                                stNombreNodo = 'ES_SEPARACION_1"'

                                        archPGMX.write('                      <Variable name="' + stNombreNodo + ' />\n')

                                        stNombreNodo = 'ES_ABANDONO" timeSlice="1"'
                                        if idTipoDiagrama == 1:
                                                stNombreNodo = 'ES_ABANDONO_1"'

                                        archPGMX.write('                      <Variable name="' + stNombreNodo + ' />\n')
                                        archPGMX.write('                    </Variables>\n')
                                        archPGMX.write('                    <Values>' + stUtilidades + '</Values>\n')
                                        archPGMX.write('                  </Potential>\n')
                                        archPGMX.write('                </Branch>\n')

                archhPGMX.write('              </Branches>\n')
                archhPGMX.write('            </Potential>\n')
                archhPGMX.write('          </Branch>\n')

                #7.1.1.- Rama cuando hay fin de plaza
                archhPGMX.write('          <Branch>\n')
                archhPGMX.write('            <States>\n')
                archhPGMX.write('              <State name="1" />\n')
                archhPGMX.write('            </States>\n')
                archhPGMX.write('            <Potential type="Table">\n')

                stNombreNodo = 'UTILIDAD" timeSlice="1"'
                if idTipoDiagrama == 1:
                        stNombreNodo = 'UTILIDAD_1"'

                archhPGMX.write('              <UtilityVariable name="' + stNombreNodo + ' />\n')
                archhPGMX.write('              <Values>0.0</Values>\n')
                archhPGMX.write('            </Potential>\n')
                archhPGMX.write('          </Branch>\n')
                archhPGMX.write('        </Branches>\n')
                archhPGMX.write('      </Potential>\n')

        if idTipoDiagrama == 1:
                #7.1.- Diagrama de Influencia
                archhPGMX.write('      <Potential type="Table" role="utility">\n')
                archhPGMX.write('        <UtilityVariable name="UTILIDAD_1" />\n')
                archhPGMX.write('        <Variables>\n')
                vecMax = []
                vecCon = []
                vVariableVal = []
                for row in vecVariables_Utilidad:
```

```
                arhcPGMX.write('          <Variable name="' + row + '_1" />\n')
                for auxNodo in vecNodo:
                        if row == auxNodo[0]:
                                vVariableVal.append(auxNodo[1])
                                vecMax.append(len(auxNodo[1])-1)
                                vecCon.append(0)

        vecUtilidad = []
        banAvanzar = True
        while True:
                vecCampos = []
                for i in range(0,len(vecCon)):
                        vecCampos.append(vVariableVal[i][vecCon[i]])
                vecUtilidad.append(vecCampos)

                auxCol = 0 #(len(vecCon)-1)
                vecCon[auxCol] = vecCon[auxCol] + 1
                for i in range(0,len(vecCon)):
                        auxCol = i    #(len(vecCon)-1) - i
                        if vecCon[auxCol] > vecMax[auxCol]:
                                vecCon[auxCol] = 0
                                auxCol = auxCol + 1
                                if auxCol < len(vecCon):
                                        vecCon[auxCol] = vecCon[auxCol] + 1
                                else:
                                        banAvanzar = False
                if banAvanzar == False:
                        break

        stUtilidad = ""
        for vUtilidad in vecUtilidad:
                stSQL = "SELECT "
                stSQL = stSQL + " IFNULL((-1) * AVG(IFNULL((SBA_IMPORTE * 0.15 "
                stSQL = stSQL + " + SEPARACION) * ES_SEPARACION,0))  "
                stSQL = stSQL + " + (-1) * AVG(IFNULL((SBA_IMPORTE * 0.15 + ABANDONO)"
                stSQL = stSQL + " * ES_ABANDONO,0)),0) "
                stSQL = stSQL + " FROM " + stDB + ".MID_DATOS "
                stSQL = stSQL + " WHERE ANNO >= 2017 "
                stSQL = stSQL + " AND MUESTRA <> " + str(idParticion) + " "
                stMismoCampoEstado_0 = ""
                for i in range(0,len(vUtilidad)):
                        stSQL = stSQL + " AND " + vecVariables_Utilidad[i] + "='" + str(vUtilidad[i]) + "'"
                myCursor.execute(stSQL)
                regAux = myCursor.fetchall()
                for resSQL in regAux:
                        stUtilidad  = stUtilidad  + ' ' + str(resSQL[0])

        arhcPGMX.write('        </Variables>\n')
        arhcPGMX.write('       <Values>' + stUtilidad + '</Values>\n')
        arhcPGMX.write('      </Potential>\n')

#8.- Cálculo de los costos
print('15.-- Costos de Decisión ' + time.strftime("%H:%M:%S"))
for i in range(0,len(vecDecision)):
        arhcPGMX.write('      <Potential type="Table" role="utility">\n')

        stNombreNodo = "COSTO_" + vecDecision[i] + '" timeSlice="0"'
        if idTipoDiagrama == 1:
                stNombreNodo = "COSTO_" + vecDecision[i] + '_0"'

        arhcPGMX.write('        <UtilityVariable name="' + stNombreNodo + ' />\n')
        arhcPGMX.write('        <Variables>\n')
        stNombreNodo = vecDecision[i] + '" timeSlice="0"'
        if idTipoDiagrama == 1:
                stNombreNodo = vecDecision[i] + '_0"'
        arhcPGMX.write('          <Variable name="' + stNombreNodo + ' />\n')
        arhcPGMX.write('        </Variables>\n')

        for auxNodo in vecNodo:
                if vecDecision[i] == auxNodo[0]:
                        vClaseVal = auxNodo[1]
        stCostos = ""
        if i == 0:
                stSQL = "SELECT IFNULL(AVG(B.SBA_IMPORTE - A.SBA_IMPORTE),0) "
        if i == 1:
                stSQL = "SELECT IFNULL(AVG(B.NUM_CURSOS * B.PRE_FORMACION/ 3 "
                stSQL = stSQL + " - A.NUM_CURSOS * A.PRE_FORMACION/3),0) "
        stSQL = stSQL  + " FROM " + stDB + ".MID_DATOS A, " + stDB + ".MID_DATOS B "
        stSQL = stSQL  + " WHERE A.ID_EMPLEADO = B.ID_EMPLEADO "
```

```
            stSQL = stSQL  + " AND A.ANNO + 1 = B.ANNO "
            stSQL = stSQL  + " AND A.MUESTRA <> " + str(idParticion) + " "
            stSQL = stSQL  + " AND A." + vecDecision[i] + " <> 'S/D' "
            stSQL = stSQL  + " AND A." + vecDecision[i] + " <> B." + vecDecision[i] + " "
            stSQL = stSQL  + " AND A." + vecDecision[i] + "='"
            for row in vClaseVal:
                    myCursor.execute(stSQL + row + "'")
                    regAux = myCursor.fetchall()
                    for resSQL in regAux:
                            stCostos = str(resSQL[0]) + ' ' + stCostos
            arhcPGMX.write('        <Values>'+ stCostos + '</Values>\n')

            #Aqui va los datos numéricos
            arhcPGMX.write('      </Potential>\n')

    arhcPGMX.write('    </Potentials>\n')
    arhcPGMX.write('  </ProbNet>\n')

    #9.- Opciones de Infirencia
    arhcPGMX.write('  <InferenceOptions>\n')
    arhcPGMX.write('    <MulticriteriaOptions>\n')
    arhcPGMX.write('      <SelectedAnalysisType>UNICRITERION</SelectedAnalysisType>\n')
    arhcPGMX.write('      <Unicriterion>\n')
    arhcPGMX.write('        <Scales>\n')
    arhcPGMX.write('          <Scale Criterion="---" Value="1.0" />\n')
    arhcPGMX.write('        </Scales>\n')
    arhcPGMX.write('      </Unicriterion>\n')
    arhcPGMX.write('      <CostEffectiveness>\n')
    arhcPGMX.write('        <Scales>\n')
    arhcPGMX.write('          <Scale Criterion="---" Value="1.0" />\n')
    arhcPGMX.write('        </Scales>\n')
    arhcPGMX.write('        <CE_Criteria>\n')
    arhcPGMX.write('          <CE_Criterion Criterion="---" Value="Cost" />\n')
    arhcPGMX.write('        </CE_Criteria>\n')
    arhcPGMX.write('      </CostEffectiveness>\n')
    arhcPGMX.write('    </MulticriteriaOptions>\n')
    if idTipoDiagrama == 0:
            arhcPGMX.write('    <TemporalOptions>\n')
            arhcPGMX.write('      <Slices>20</Slices>\n')
            arhcPGMX.write('      <Transition>BEGINNING</Transition>\n')
            arhcPGMX.write('      <DiscountRates>\n')
            arhcPGMX.write('        <DiscountRate Criterion="---" value="0.0" unit="YEAR" />\n')
            arhcPGMX.write('      </DiscountRates>\n')
            arhcPGMX.write('    </TemporalOptions>\n')
    arhcPGMX.write('  </InferenceOptions>\n')

    #10.- Políticas
    print('16.-- Potencias/Políticas Decisión ' + time.strftime("%H:%M:%S"))
    if banDecision == True:
            arhcPGMX.write('  <Policies>\n')
            for row in vecLinksDecision:
                    stVariables = row[0] + '_0'

                    stTimeSlice = ' timeSlice="0"'
                    arhcPGMX.write('    <Potential type="Table" role="joinProbability">\n')
                    arhcPGMX.write('      <Variables>\n')

                    stNombreNodo = row[0] + '" timeSlice="0"'
                    if idTipoDiagrama == 1:
                            stNombreNodo = row[0] + '_0"'

                    arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')
                    vClaseVal = []
                    for auxNodo in vecNodo:
                            if row[0] == auxNodo[0]:
                                    vClaseVal = auxNodo[1]

                    vecMax = []
                    vecCon = []
                    vVariableVal = []
                    for i in range(0,len(row[1])):

                            auxStVar = row[1][i] + '_0'
                            stTimeSlice = ' timeSlice="0"'
                            for j in range(0,len(vecVariables_Fijas)):
                                    if row[1][i] == vecVariables_Fijas[j]:
                                            stTimeSlice = ''
                                            auxStVar = row[1][i]
```

```
                stNombreNodo = row[1][i] + '" ' + stTimeSlice
                if idTipoDiagrama == 1 and stTimeSlice == ' timeSlice="0"':
                        stNombreNodo = row[1][i] + '_0"'

                stVariables =  stVariables + "," + auxStVar

                arhcPGMX.write('        <Variable name="' + stNombreNodo + ' />\n')

                #Prepara las variable auxiliares para el cálculo de las probabilidades
                for auxNodo in vecNodo:
                        if row[1][i] == auxNodo[0]:
                                vVariableVal.append(auxNodo[1])
                                vecMax.append(len(auxNodo[1])-1)
                                vecCon.append(0)

        arhcPGMX.write('      </Variables>\n')

        #Prepara el vector para las ID. de Probabilidades
        stProb = Busca_Probabilidad(myConexion, stDB, idParticion, stVariables, stExperimento)
        if stProb == "N/E":
                Inicio = dt.datetime.today()
                stProb = ""
                vecProb = []
                banAvanzar = True
                while True:
                        vecCampos = []
                        for i in range(0,len(vecCon)):
                                vecCampos.append(vVariableVal[i][vecCon[i]])
                        vecProb.append(vecCampos)

                        auxCol = 0 #(len(vecCon)-1)
                        vecCon[auxCol] = vecCon[auxCol] + 1
                        for i in range(0,len(vecCon)):
                                auxCol = i    #(len(vecCon)-1) - i
                                if vecCon[auxCol] > vecMax[auxCol]:
                                        vecCon[auxCol] = 0
                                        auxCol = auxCol + 1
                                        if auxCol < len(vecCon):
                                                vecCon[auxCol] = vecCon[auxCol] + 1
                                        else:
                                                banAvanzar = False
                        if banAvanzar == False:
                                break

                for vProb in vecProb:
                        stSQL = "SELECT  Count(*)"
                        stSQL = stSQL + "FROM " + stDB + ".MID_DATOS A "
                        stSQL = stSQL + " WHERE 1 = 1 "
                        stSQL = stSQL + " AND A.MUESTRA <> " + str(idParticion) + " "
                        for i in range(0,len(vProb)):
                                stCampo = "A." + row[1][i]
                                stSQL = stSQL + " AND " + stCampo + "='" + str(vProb[i]) + "'"
                        vecProbRes = []
                        varTotal = 0
                        for i in range(0,len(vClaseVal)):

                                myCursor.execute(stSQL + " AND A." + row[0] +"='" + str(vClaseVal[i]) + "'")
                                regAux = myCursor.fetchall()

                                for resSQL in regAux:
                                        vecProbRes.append(resSQL[0])
                                        varTotal =  varTotal + resSQL[0]
                        #Correción de Laplace
                        if varTotal + vCorrecionLaplace * len(vClaseVal) > 0 :
                                for i in range(0,len(vecProbRes)):
                                        vecProbRes[i] = (vecProbRes[i] + vCorrecionLaplace )
                                                / (varTotal + vCorrecionLaplace * len(vClaseVal))

                        for i in range(0,len(vecProbRes)):
                                stProb = stProb + ' ' + str(vecProbRes[i])

                Fin = dt.datetime.today()
                Tiempo = Dif_Tiempo(Inicio,Fin)
                Guarda_Probabilidad(myConexion, stDB, idParticion, stVariables, stProb,
                        stExperimento, Inicio, Fin, Tiempo)

        arhcPGMX.write('      <Values>' + stProb + '</Values>\n')
        arhcPGMX.write('    </Potential>\n')
```

```
                arhcPGMX.write('  </Policies>\n')

        #99.- Cierra Red
        arhcPGMX.write('</ProbModelXML>\n')
        arhcPGMX.close()

except Error as e:
        print("Error reading data from MySQL table", e)
        myConexion.close()
        varResultado = -1

finally:
        pass

#99.- Cierra conexiones
myConexion.close()
print('99.- Salgo ' + time.strftime("%H:%M:%S"))
return varResultado
```

Bibliografía

Arnott, D. and Dodson, G. (2013). Decision Support Systems Failure Success and Failure in DSS Projects. *Decision Support Systems*.

Ashcroft, M. (2012). Bayesian networks in business analytics. urlannals-csis.org/proceedings/2012/pliks/190.pdf.

Bonache, J. and Ángel, C. (2006). *Dirección de Personas. Evidencias y perpectivas para el siglo XXI*. Prentice-Hall, España.

Branham, L. (2020). 7 razones ocultas por las que los empleados se van. urlhttps://www.leadersummaries.com/resumen/7-razones-ocultas-por-las-que-los-empleados-se-van.

Brownlee, J. (2018a). A gentle introduction to statistical hypothesis testing. urlhttp://machinelearningmastery.com/statistical-hypothesis-tests/.

Brownlee, J. (2018b). How to calculate mcnemar's test to compare two machine learning classifiers. urlhttps://machinelearningmastery.com/mcnemars-test-for-machine-learning/ .

Cetti, J. (2009). Using Information Gain Attribute Evaluation to Classify Sonar Targets. *Telecommunications forum TELFOR 2009*, pages 24–26.

Cetti, L. and Simón, V. (2009). Dirección por Valores. *Visión de Enfermería Actualizada, Junio 2009*, pages 32–35.

Consortium, W. W. W. (2020). Extensible markup language (xml). urlhttps://www.w3.org/XML/.

de Empleo y Seguridad Social, M. (2015). Boe núm. 255, real decreto legislativo 2/2015, de 23 de octubre, por el que se aprueba el texto refundido de la ley del estatuto de los trabajadores. urlhttps://www.boe.es/buscar/pdf/2015/BOE-A-2015-11430-consolidado.pdf.

Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation, Volume 10, Issue 7, October 1, 1998, p.1895-1924*.

Ferreira, L. and Borenstein, D. (2012). A fuzzy-Bayesian model for supplier selection. *Expert Systems with Applications*, 39:7834–7844.

García Vega, M. A. (2020). La era del reinado del jefe digital. urlhttps://retina.elpais.com/retina/2020/07/16/tendencias/1594858796-446067.html.

Guenole, N. and Feinzig, S. (2018). The business case for ai in hr. urlhttps://www.ibm.com/downloads/cas/AGKXJX6M.

Hafeez, K. and Abdelmeguid, H. (2003). Dynamics of human resource and knowledge management. *Journal of the Operational Research Society*, 54:153–164.

Hernández Orallo, J. R. Q. M. J. and Ferri Ramírez, C. (2004). *Introducción a la Minería de Datos.* Pearson Prentice Hall, Spain.

Huang, O. (2017). Applying multinomial naive bayes to nlp problems: A practical explanation. urlhttps://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf.

Jan, M. J., Khalid1, M. S. K., Awan, A. A., and Nisar, S. (2018). Proposing Probabilistic Operational Risk Assessment Model for Textile Industry Using Bayesian Approach.

Jerico, P. (2011). *La Nueva gestión del talento.* PRENTICE-HALL, España.

Kofman, F. (2001). *Metamanagement, tomos I, II y III.* Granica, México.

Kosta Loukides, Michael; Mason, H. and Patil, D. (2018). *Ethics and Data Science.* O'Reilly Media, United States of America.

Ley Borrás, R. (2009). *Análisis de Decisiones Integral: Una guía para quienes desean ayudar a otros (y a sí mismos) a tomar mejores decisiones en situaciones complejas.* Consultoría en Decisiones, México.

Minchington, B. (2006). *Your employer brand : attract, engage, retain.* Mile end, sa : Collective learning, Australia.

Moya, M. (2016). Resumen reglas de codd. urlhttps://www.clubensayos.com/TecnologÃ-a/Resumen-Reglas-de-Codd/3152758.html.

Muñoz, C. (2019). ¿cómo evitar una relación tóxica? urlhttps://youtu.be/H4zZ7ec2OGI.

Nicastro, D. (2020). 7 ways artificial intelligence is reinventing human resources. urlhttps://www.cmswire.com/digital-workplace/7-ways-artificial-intelligence-is-reinventing-human-resources/.

Pereira, J. C. and Alves Lima, G. B. (2015). Probabilistic risk analysis in manufacturing situational operation, application of modelling techniques and causal structure to improve safety performance. *International Journal of Production Management and Engineering (IJPME)*, 3:33–42.

Ramírez Salazar, M. d. P. (2008). Cómo lograr ser una de las mejores empresas para trabajar en colombia? urlhttps://www.redalyc.org/pdf/206/20611455006.pdf.

Raschka, S. (2018). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. *arXiv:1811.12808v1 [cs.LG] 13 Nov 2018*.

Schwartz, T. (2013). What would make you more satisfied and productive at work? urlhttps://hbr.org/2013/11/what-would-make-you-more-satisfied-and-productive-at-work.

Valencia, E. (2017). Guía para calcular el coste del abandono. urlhttp://www.eduardovalencia.com/2017/07/guia-para-calcular-el-coste-del-abandono.html.

Varios (2017). 10 razones por las que los buenos profesionales abandonan una empresa. urlhttps://nexian.es/10-razones-las-los-buenos-profesionales-abandonan-una-empresa/.

Wikipedia (2020a). Curva roc. urlhttps://es.wikipedia.org/wiki/Curva-ROC.

Wikipedia (2020b). Extensible markup language. urlhttps://es.wikipedia.org/wiki/Extensible-Markup-Language/.

Wikipedia (2020c). F1 score. urlhttps://en.wikipedia.org/wiki/F1-score/.