UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Trabajo de Fin de Grado

Grado en Ingeniería Informática (GEI)
Especialización en Computación

ENTREGA FINAL

Predicción de la felicidad de los empleados en el lugar de trabajo

Autora: Paula Zhao
Directores: Gerardo Astone
José María Sánchez
Ponente: Javier Béjar
Tutor de GEP: Ferran Marfil Sánchez
Turno Presentación: Octubre de 2020

ÍNDICE

Resumen

El proyecto Kokoro de Slashmobility consiste en una plataforma dirigida a RRHH para poder predecir las acciones de los empleados, la cual una de las acciones es la fuga de talento. Para ello, Kokoro necesita el input de "Felicidad del empleado", es decir, la autorrealización del empleado en la empresa. Este TFG analiza la predicción de esta felicidad en base a unos cuestionarios para determinar la felicidad, distribuidas por un chatbot. Debido a que durante el desarrollo del TFG aún no ha sido desarrollado tal chatbot, se han generado unos datos teóricos y realistas. Se analiza la predicción con diferentes métodos de Machine Learning: Decision Tree, Random Forest, Linear Regression y MLP (Red neuronal). Se ha llegado a la conclusión de que con estos datos teóricos, el mejor método es Random Forest, con una precisión alrededor del 94%, bastante alta precisión, aunque puede ser debido a que los datos son teóricos.

Resum

El projecte de Kokoro de Slashmobility consisteix en una plataforma dirigida a RRHH per poder predir les accions dels empleats, la qual una de les accions és la fuga de talent. Per això, Kokoro necessita el input de “Felicitat del emplea”, és a dir, la autorealització de l'empleat en la empresa. Aquest TFG analitza la predicció d’aquesta felicitat en base a uns qüestionaris per determinar la felicitat, distribuïdes per un chatbot. Degut a que durant el desenvolupament del TFG encara no ha estat desenvolupat aquest chatbot, s’han generat unes dades teòriques y realistes . S’analitza la predicció amb diferents mètodes de Machine Learning: Decision Tree, Random Forest, Linear Regression i MLP (Xarxa Neuronal). S’ha arribat a la conclusió de que amb aquestes dades teòriques, el millor mètode és Random Forest amb una precisió al voltant del 94%, una precisió bastant alta, encara que pot ser degut que les dades són teòriques.

Abstract

Slashmobility's Kokoro project consists of a platform aimed to HR to be able to predict the actions of employees, one of which is the flight of talent. For this, Kokoro needs the input of "Employee Happiness", that is the employee's self-realization in the company. This TFG analyzes the prediction of this happiness based on questionnaires to determine the employee happiness, distributed by a chatbot. Due to the fact that such chatbot has not yet been developed (during the development of the TFG), theoretical and realistic data have been generated. The prediction is analyzed with different Machine Learning methods: Decision Tree, Random Forest, Linear Regression and MLP (Neural Network). It has been concluded that with these theoretical data, the best method is Random Forest, with a precision around 94%, a high precision, although it may be because the data is theoretical.

1. Introducción

1.1 Contextualización

1.1.1. Marco del proyecto

Este proyecto de **Trabajo de Fin de Grado** en Ingeniería Informática (en la especialidad en **Computación**) de la **Facultad de Informática de Barcelona**, se ha desarrollado en la modalidad B con un convenio de cooperación educativa con **Slashmobility**[1]. La cual es dirigida por Gerardo Astone y José María Sánchez, actualmente empleados de Slashmobility. El TFG está supervisado por el doctor Javier Bejar Alonso del Departamento de *Computer Science*.

Slashmobility es una empresa de soluciones *mobile* nacida en 2010 en Barcelona Activa, con sedes en Barcelona, Madrid, Londres y Colombia. Fue fundada por Emilio Avilés Avila.

Su objetivo principal es acelerar la digitalización tecnológica de la sociedad mediante servicios Mobile 360º, es decir, mediante formación, diseño, desarrollo software y *IT Recruitment*. Actualmente, es una empresa referente en el mercado mediante el desarrollo software, formación y captación de talento digital [1].

Slashmobility estableció la primera incubadora de apps en España: *SlashLabs*.

También es ganadora de varios premios:
- 2011 : Emprendedor XXI de la Caixa
- 2011 : Maratón de Finapps Party
- 2012 : AppCircus con su incubadora Dressapp
- 2012 : Hackathon Extreme Android
- 2018 : eAwards como Mejor agencia de creación de apps
- 2018 : Emilio Avilés nominado como Best Digital Leader

Participante de:
- Desde 2011 : Mobile World Congress
- 2012 : expositores en eShowMadrid
- 2015 : ponentes en Eurecat Mobile Forum

El objetivo de este TFG es realizar una parte de un nuevo proyecto puesto en marcha a finales de 2019 por parte de Slashmobility llamado **Kokoro** [2]. Este proyecto tiene como meta desarrollar una plataforma de hiperconexión que analiza grandes masas de datos desestructurados haciendo uso de Inteligencia Artificial con la motivación de poder generar servicios a empresas que deseen aumentar el *Employee Experience* del talento Digital, así como la toma de decisiones en el ámbito de RRHH [3].

Kokoro surgió a raíz de uno de los grandes retos a los que se enfrentan las empresas hoy en día a causa de la globalización y la competitividad en el mundo laboral: **la gestión del**

talento, es decir, la gestión de los empleados, implicando la capacidad de atraer a empleados y la capacidad de retención de estos.

En la actualidad, disponemos de grandes cantidades de información, tanto interna como externa, que pueden ayudar a la toma de decisiones a nivel de RRHH para diferentes situaciones. Y esto será posible con el uso de algoritmos de *Big Data* para estructurar esta información masiva, filtrarlos con algoritmos de *Feature Extraction* y *Feature Selection* para obtener información útil y finalmente poder predecir el comportamiento de los empleados basado en el análisis de los datos con el uso de *Machine Learning*.

Gracias a esta plataforma se podrán resolver diferentes situaciones que pueden surgir en la empresa: como por ejemplo predecir a tiempo una posible fuga de talento; la sobrecarga de trabajo en un empleado causando su descontento o aumento de estrés y por tanto un descenso en su rendimiento; mejoras en el rendimiento como por ejemplo recomendar la priorización de tareas dependiendo del empleado y sus características, la dificultad, el momento del día, etc. Todo ello para mejorar tanto la vida laboral de los empleados como la competitividad de la empresa.

1.1.2. Términos y conceptos

Para la clara comprensión de este trabajo de fin de grado se define a continuación términos y conceptos que se usarán constantemente en este documento y por tanto, se darán por supuesto su entendimiento:

- **RRHH**: Siglas para “Recursos Humanos”. El departamento de una empresa que gestiona las personas de la empresa u organización para conseguir una ventaja competitiva. De manera que maximiza el rendimiento de los empleados para servir a los objetivos estratégicos de la empresa.
- **Talento**: En el ámbito de RRHH, se refiere a los empleados.
- **Employee Experience**: Encapsula lo que el empleado se encuentra, observa o siente durante toda su vida laboral en la empresa, incluyendo desde la entrevista, su interacción con los otros empleados, con sus superiores, con la tecnología y su propio rendimiento en el trabajo [4].
- **Notebook**: Entorno computacional interactivo, en el que se puede combinar ejecución de código, texto enriquecido (markdown), matemáticas, gráficos y medios enriquecidos [24].
- **Máquina**: Se refiere al programa o al ordenador.
- **Inteligencia Artificial**: Es un subcampo de las ciencias de la computación cuyo objetivo es que las máquinas imitan las funciones cognitivas que los humanos consideramos “Inteligentes” (como lo son razonar, aprender, resolver problemas, etc) [5,6]. Lo abreviamos como IA.
- **Machine Learning**: Una rama de la Inteligencia Artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan por sí solas a base de entrenarlas con datos. Una vez entrenada con datos, se obtiene un modelo que debe ser capaz de generalizar comportamiento e inferencias para otro conjunto de datos [7,8]. Existen diferentes métodos de predicción, los cuales dependen de

diferentes parámetros. Estos parámetros moldean el método afectando a la predicción. Para ello, se han de buscar los mejores valores para estos parámetros y así obtener el mejor modelo para ese método con esos datos. Lo abreviamos como ML.

- **Modelo**: Llamamos modelo a una máquina que usa un método predictivo de Machine Learning con valores concretos para cada uno de sus parámetros. Por ejemplo, un modelo A refiriéndose a una máquina de Red Neuronal con 1 capa de neuronas y un modelo B refiriéndose a otra máquina de Red Neuronal con 3 capas de neuronas.
- **Feature Extraction**: De un conjunto de datos, obtener un subconjunto de estos datos simplificados, no redundantes y útiles [9].
- **Feature Selection**: De un conjunto de datos, obtener un subconjunto de estos datos, eliminando datos redundantes o innecesarios [9].
- **Estado anímico**: Estado emocional generalizado y persistente que influye en la manera de percibir el mundo. [10]
- **Big Data**: Conjuntos de datos masivos y complejos de tal manera que hacen falta aplicaciones informáticas no tradicionales de procesamiento de datos para tratarlos adecuadamente. [11]
- **Cross Validation**: Es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba, en que consiste en repetir y calcular la predicción obtenida sobre diferentes particiones, teniendo como datos de entreno el resto de particiones. Lo abreviamos como CV.
- **Aprendizaje supervisado**: Es una técnica para deducir una función a partir de datos de entrenamiento. Los datos de entrenamiento consisten de pares de objetos: par de los datos de entrada y el resultado deseado. La salida de la función debe ser un valor numérico o una etiqueta de la clase (en nuestro caso será una etiqueta de clase, ya que estamos clasificando).El objetivo de esta técnica es crear una función capaz de predecir el valor correspondiente de cualquier objeto de entrada válida después de haber visto una serie de ejemplos correctos, los datos de entrenamiento.

1.1.3. Problema a resolver

Como ya se ha introducido en el apartado de 1.1.1 Marco del proyecto, el proyecto de Kokoro tiene como objetivo generar servicios a empresas que deseen aumentar el *Employee Experience* del talento Digital, así como la toma de decisiones en el ámbito de RRHH.

Actualmente, en el mundo laboral existen diferentes retos a los que cualquier empresa se enfrenta. El proyecto Kokoro identifica y quiere ofrecer solución a dos de estos retos:

1. Con la incorporación de nuevas generaciones al mundo laboral (generación Y, “millennials” y la generación Z, “centennials”), las exigencias del mundo laboral están cambiando. Es decir, exigen una “*Employee Experience*” memorable donde sientan que la empresa los cuida en cada una de sus etapas como empleado y les permite **ser felices** en su lugar de trabajo.

2. Una de las grandes preocupaciones en el ámbito de RRHH, la necesidad de **anticiparse al comportamiento del talento** (como por ejemplo la fuga de talento o anticiparse al posible estrés de un empleado) para poder tomar decisiones a tiempo para solucionar las situaciones que surjan, como ya hemos comentado en la en apartado 1.1.1 Marco del proyecto.

Debido a que el alcance de Kokoro excede al alcance que debería tener un TFG (el predecir el comportamiento de un empleado, que va desde la predicción del aumento de su estrés, satisfacción en el lugar de trabajo, recomendar planificaciones para aumentar el rendimiento según el empleado, hasta predicción de un posible deseo de cambio de trabajo), se ha decidido focalizar el problema a resolver del TFG a una parte de una de las situaciones explicadas anteriormente: la necesidad de anticiparse al comportamiento del empleado, específicamente, a **predecir la felicidad del empleado en el lugar de trabajo**.

1.1.4. Actores implicados

Los actores implicados o *stakeholders*, son todas esas personas o entidades afectadas por o interesadas en el desarrollo del proyecto. En este caso, existen actores implicados directamente en este TFG :

Kokoro

Este TFG es una parte del desarrollo de Kokoro. Por tanto, este trabajo afectará de manera directa porque los resultado obtenidos serán usados por Kokoro para cumplir sus propios objetivos.

Slashmobility

Es la empresa que lanza el proyecto de Kokoro, por lo cual está directamente relacionado con Kokoro y por ende, este TFG. Además, Slashmobility se beneficiará económicamente del producto

Equipo de desarrollo de Slashmobility

El equipo de desarrollo de Slashmobility está interesado en el desarrollo de este TFG porque es parte del proyecto de Kokoro. Además de que me pueden ayudar en el progreso de este TFG.

Además de mi persona, por el desarrollo del mismo TFG y mi interés personal de crecer en el mundo de *Machine Learning* a través de este trabajo.

Los actores implicados indirectamente de este TFG, son aquellos que están relacionados con el proyecto Kokoro:

Empresas que hagan uso de Kokoro

Kokoro es una plataforma pensada para la gestión del talento digital, por lo tanto, está pensado para que lo utilice **cualquier empresa**, ya que toda empresa requiere de gestión de empleados. Específicamente, está **dirigida al departamento de recursos humanos**, o también llamado *talent office*.

No solo Slashmobility se beneficiará de Kokoro, sino toda empresa que use Kokoro, ya que Kokoro proporcionará una mejora del ambiente laboral y la productividad de la empresa, ya que si un empleado está satisfecho con su trabajo y su entorno, tendrá un mejor rendimiento[14].

Empleados y al departamento de recursos humanos

Ya que facilitará la mejora del *Employee Experience* y la toma de decisiones dependiendo de la situación de un empleado y características del empleado, **afectará** de manera directa **a los empleados** y al **departamento de recursos humanos**.

1.2. Justificación

En el marco de este proyecto, es decir, en la empresa de Slashmobility, Kokoro es el primer proyecto con un alto nivel de complejidad por su integración y dependencias con *Big Data* e IA. Por tanto, para Slashmobility es un proyecto ambicioso e innovador con el cual enriquecerá la empresa en estos campos. Y ha sido aprobado por la CDTI (Centro para el Desarrollo Tecnológico Industrial) [12].

Sobre la predicción de la felicidad en el lugar de trabajo han habido algunos trabajos similares, aunque no con el foco de predecir la felicidad del empleado en el lugar de trabajo. Como por ejemplo "***Towards estimating computer users' mood from interaction behaviour with keyboard and mouse***" de Khan, I.A., Brinkman, W. & Hierons, R [13], que estudia el estado anímico según algo fuera del entorno laboral.
O trabajos más centrados en obtener la satisfacción y rendimiento en el trabajo según su inteligencia emocional, como comenta el siguiente estudio: "***Relation of employee and manager emotional intelligence to job satisfaction and performance***" de Thomas Sy, Susanna Tram y Linda A.O'Hara [14].

Aparte de predecir la felicidad del empleado en el lugar de trabajo, este estudio quiere determinar el mejor método de *Machine Learning* para ello. Por tanto, no hace falta reinventar la rueda, así que, se usarán métodos ya existentes en el mundo de *Machine Learning*, se analizarán los resultados y se determinará cuál es el mejor método. En el caso de no obtener resultados satisfactorios, se plantea la modificación de métodos existentes.

1.3. Alcance

El proyecto Kokoro tiene 4 grandes objetivos tecnológicos:

- El **desarrollo de un sistema predictivo**. Desarrollar un sistema que sea capaz de predecir el **comportamiento** de los usuarios.

- **Adaptar y unificar datos no estructurados**. Desarrollar un algoritmo basado en feature extraction y feature selection, capaz de estructurar los datos obtenidos al modelo deseado independientemente de donde provengan. Con esto se refiere a que los datos probablemente tengan una estructura diferente si vienen por ejemplo de Twitter o de Facebook.
- **Ofrecer sugerencias adaptadas a las situaciones**, usando Machine Learning.
- **Recepción y procesamiento de los datos a gran escala**.

Ya que Kokoro es un proyecto muy grande. Se ha decidido centrar el trabajo de fin de grado en uno de los objetivos: El desarrollo de un sistema predictivo.
Aún así, generar el modelo y máquina final que prediga el comportamiento preciso del empleado es excesivo para este TFG, por tanto, este TFG se dedicará a preparar una de las máquinas de entrada para la máquina final: **Predicción de la felicidad** del empleado respecto a la empresa.

1.3.1. Objetivos

El principal objetivo del TFG es desarrollar un sistema que sea capaz de **predecir la felicidad en el trabajo del empleado de manera precisa**, basándonos en el entorno laboral de éste. Para explorar el entorno laboral del empleado, se usarán unas preguntas en formato Escala Likert [30], una escala psicométrica comúnmente utilizada en cuestionarios y en encuestas para la investigación que al responder las preguntas se puede especificar el nivel de acuerdo o desacuerdo con una declaración. En este proyecto, la declaración sería sobre la felicidad del empleado en el lugar de trabajo.

El procedimiento que normalmente se realiza para poder clasificar la felicidad del empleado en base a las preguntas que se detallan en el apartado 10.1 Modelo de Datos, lo que se debe hacer es pasar el formulario periódicamente al usuario, del cual se guardan las respuestas para realizar un estudio estadístico. Además se han de realizar reuniones de seguimiento con el empleado, obteniendo otros resultados cualitativos que también se han de analizar. Y una vez realizado un estudio estadístico y las reuniones de seguimiento, se puede categorizar la felicidad del empleado respecto a la empresa.

Por tanto, el principal objetivo de este TFG, es poder predecir la felicidad del empleado sin tener que realizar todo este largo y complicado proceso desarrollando un sistema predictivo con Machine Learning, de manera que con el input de una encuesta la máquina pueda predecir la felicidad del empleado en la próxima encuesta.

La salida que generará este sistema predictivo será una de las entradas de la máquina predictiva final, que predecirá el comportamiento del empleado más detallado para poder tomar decisiones a tiempo, como puede ser la predicción de que el empleado se querrá irse de la empresa en X tiempo, por Y motivos.

1.3.2. Subobjetivos

Para poder predecir de manera precisa, tenemos que entrenar el modelo con datos suficientes y válidos. Por tanto, es imprescindible realizar un **preprocesamiento** de los

datos. Este preprocesamiento consiste en limpiar y estructurar los datos de entrada de una manera que facilite al modelo encontrar patrones. Por ejemplo, en una base de datos de personas, podría haber toda la información personal de cada persona, pero si nos interesa predecir el ingreso de una persona según el barrio en el que trabaja, solo nos interesaría los campos relacionados al salario, ingresos, trabajo, lugar de trabajo, o parecidos, los cuales analizaríamos estadísticamente si tienen cierta relación o no para decidir su descarte o no. Quedándonos solamente con los campos que son relevantes para alcanzar nuestro objetivo

Además, al ser parte de un proyecto más grande (Kokoro), a parte de ser capaz de producir resultados válidos, tiene que ser eficiente y rápido para no ralentizar el proceso de predicción del comportamiento del empleado, ya que uno de sus inputs será nuestra predicción de la felicidad del empleado. Por lo tanto, se tiene que estudiar cuál es el **mejor método** para definir la máquina que predice la felicidad del empleado entre algoritmos y métodos existentes en el mundo de Machine Learning o si es mejor desarrollar alguna variante de estos ya existentes. De esta manera, en el proyecto de Kokoro se usará directamente el mejor método, obteniendo así mejor rapidez, mejor eficiencia y mejores datos de entrada para la máquina final que predecirá el comportamiento del empleado.

1.3.3. Requisitos funcionales y no funcionales

Este proyecto está bastante focalizado en la generación de un sistema predictivo, sin ser un aplicativo, sino parte de código que se usará para otra parte del proyecto Kokoro. Por consecuencia, la cantidad de requerimientos funcionales y no funcionales no es alta.

Los requerimientos funcionales que ha de cumplir este proyecto son los siguientes:

- **Predecir** la felicidad del empleado, ya que como se ha explicado en el apartado 1.3.1 Objetivos es el principal objetivo de este trabajo de fin de grado.
-

A continuación, se listan los requerimientos no funcionales que ha de cumplir este trabajo:

- La predicción ha de ser **correcta y precisa**. De esta manera, nos aseguramos de tener resultados válidos que se puedan usar en otra parte del proyecto Kokoro sin poner en riesgo la integridad y el correcto funcionamiento del proyecto.

- Al ser parte de un proyecto de grandes dimensiones, la predicción obtenida no solo ha de ser válida y precisa, sino que también debería ser **eficiente y rápida**. Debido a que este trabajo de fin de curso es una parte de Kokoro donde inevitablemente creará dependencias, porque la predicción obtenida será una de las entradas que le daremos a la máquina que predecirá el comportamiento del empleado. Por tanto, es esencial que nuestra parte no tarde mucho para que no cree un cuello de botella en el proyecto. Lo ideal sería que se ejecutará y se obtengan los resultados en menos de 30 segundos a 1 minuto.
- Usar el **mejor método** de *Machine Learning*. De tal manera que podremos asegurar el requerimiento anterior, no solo mejorando la eficiencia y rapidez, sino también obteniendo la mejor optimización. Ya que cada método tiene sus propios parámetros y cada parámetro puede optimizar hasta cierto punto los resultados de nuestro modelo.

1.3.4. Posibles obstáculos o riesgos

Debido a que el proyecto Kokoro se está poniendo en marcha recientemente, finales del 2019, un posible riesgo es **no tener suficientes datos** para poder entrenar el modelo. Pero desde el principio de este TFG, se está poniendo en marcha la planificación y obtención de datos. Por tanto, se espera que en un futuro cercano (semanas o pocos meses) se obtengan los datos suficientes para poder generar algunos resultados. Aún así, se ha planificado **defender este TFG en el siguiente cuatrimestre** (Q2 2019-2020) ya que esperamos tener muchos más datos unos meses antes de la defensa. Por tanto, al tener más datos, probablemente obtendremos mejores resultados, porque si entrenamos una máquina con muchos más datos, el modelo que estamos construyendo para la predicción puede ser menos parcial. Por ejemplo, si queremos predecir de una foto de un animal si es gato o perro, pero entrenamos con muchas fotos de gatos y pocas de perro, la predicción será muy parcial (hacia los gatos). En cambio si entrenamos con fotos en abundancia tanto como perros y gatos, la predicción tendrá mejor perspectiva y por tanto mejorarán la precisión de las predicciones.

En el caso de que aún con abundancia de datos obtengamos un **modelo no preciso**, se tendrá que **recoger de nuevo otra gran cantidad de datos** a través del departamento de recursos humanos de Slashmobility y encuestas online. Desde la primera reunión realizada con los directores, se ha iniciado el procedimiento de plantear la encuesta con los puntos cruciales para calcular la felicidad a través del departamento de recursos humanos y lanzado las encuestas a diferentes empresas directamente y también online a través de los contactos que tiene Slashmobility y yo misma. Además, se tiene en cuenta en la planificación el caso de tener que volver a realizar la recolección, entrenamiento de modelos y análisis sin poner en riesgo la fecha final del proyecto.

Y por último, otro obstáculo a tener en cuenta es que debido a que Slashmobility tiene **muchos clientes y proyectos** con los que se trabaja a la vez, los *developers* solemos tener más de un proyecto en marcha a la vez, por lo que tampoco se podrá garantizar estar al 100% dedicados al proyecto Kokoro por la dinámica de Slashmobility. Y esto puede causar un **retraso en la planificación**. Para prevenir esta situación, mantendré el contacto con el departamento de planificación para que den prioridad al TFG y **evitar posibles retrasos** en la planificación de este proyecto, con una comunicación clara entre mi persona y mis directores y el departamento de planificación, para tener siempre en claro el progreso de mi TFG. Es decir, si en un momento dado, veo que me han planificado pocas horas de TFG cuando voy retrasada, me comunicaré inmediatamente con ellos para obtener un cambio en la planificación y aumentar mis horas en TFG. O en caso contrario, en el que tengo el TFG bastante avanzado en cuanto a mi planificación de TFG, entonces les comunicaré que en esa semana no hace falta priorizar tanto mi TFG, ya que Slashmobility sigue siendo una empresa que está sirviendo a diferentes clientes y éstos esperan su producto y mi TFG no debe ser la causa del retraso de la entrega de sus productos.

2. Metodología y rigor

Para desarrollar este proyecto podemos definir 3 etapas:

1. Obtención de datos y su preprocesamiento (limpieza y estructuración) con algoritmos de *Feature Extraction* y/o *Feature Selection*.
2. Entrenar distintos modelos con los datos con distintos métodos de *Machine Learning*.
3. Análisis de los resultados de las máquinas generadas para seleccionar la mejor máquina en resultados, eficiencia y rapidez.

La metodología que se usará en este TFG será la definida en el proyecto de Kokoro: metodología Agile SCRUM.

La metodología Agile SCRUM implementa el método científico empírico. Consiste en iteraciones cortas (1 a 2 semanas), también conocidas como *sprints,* con el fin de obtener una versión funcional del producto en cada iteración. A cada iteración se le añade valores al producto hasta conseguir el producto final.

Al inicio de un proyecto con esta metodología, se crea el *Product Backlog*, un listado de todas las funcionalidades que se espera que un usuario pueda realizar.

A continuación, empieza a usarse la metodología que acabamos de explicar. En cada iteración sigue el siguiente proceso:

1. **Planificación del *sprint***: Al inicio de cada iteración, se escogen ítems del *Product Backlog* que se espera que se realicen durante ese sprint.
2. **Desarrollo del *sprint***: Gran parte del sprint, se le dedica al desarrollo e implementación de los ítems seleccionados durante la Planificación del *sprint*. En teoría, se ha de realizar una reunión diaria de no más de 10 minutos en la cual se discuten los puntos que ha de realizarse durante ese día.
3. ***Sprint review & retrospective***: Al final de cada *sprint*, se revisan los puntos realizados y se comparan con las planificadas para obtener una perspectiva del *sprint* concluido para abordar mejor el siguiente.
4. **Entrega del producto**: Una vez finalizado el *sprint*, si ya se ha obtenido el producto final, se entrega el producto. En caso negativo, se vuelve al primer paso.

Para abordar las iteraciones, existen tres figuras claves, que en este TFG las 3 figuras las represento yo misma:

- ***SCRUM Master***: Es la figura que lidera los equipos de desarrollo. Su objetivo es que los equipos de trabajo alcancen sus objetivos hasta llegar a la entrega del producto. Es el encargado de la elaboración del *Product Backlog*, *Sprint Backlog* y la planificación del sprint.
- ***Product Owner***: Responsable de maximizar el valor del producto obtenido por el equipo de desarrollo. Normalmente, es el intermediario entre la empresa y el cliente para asegurar que el producto sea el deseado y cumpla las expectativas del cliente.

- **Equipo de desarrollo**: Conjunto de personas responsables de la implementación de los ítems del *Sprint Backlog*.

Para el seguimiento de esta metodología, se usará la herramienta **JIRA Atlassian** que es la usada por el equipo de desarrollo de Slashmobility.

JIRA es una herramienta de desarrollo de software pensada para la metodología Agile donde se puede planificar y supervisar, entre otras funcionalidades que no son relevantes para este trabajo, los *sprints* de un proyecto.

JIRA se basa en el tablero de Kanban [16], una herramienta para mapear y visualizar el flujo de trabajo del *sprint*, gracias al uso de "columnas" y "tarjetas". Las columnas se asocian a estados de la tarea (Generalmente: "*TODO*", "*IN PROGRESS*", "*DONE*", aunque puede variar según el flujo de trabajo de la empresa). Y las tarjetas se asocian a las tareas del sprint que se asignan a un desarrollador del equipo de trabajo.

En cuanto a la validación del modelo, se usarán métodos de validación del mundo del Machine Learning, concretamente, el ***Cross Validation*** [7], que como bien se ha definido en el apartado 1.1.2 Términos y conceptos, se trata de una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba, en que consiste en repetir y calcular la predicción obtenida sobre diferentes particiones, teniendo como datos de entrenamiento el resto de particiones. Es decir, se parten los datos de entrenamiento en K particiones, de estas particiones, se entrenan los datos con K-1 particiones, dejando la partición restante para evaluar la predicción. Pero en vez de hacerlo solo con la última partición, se evalúa cada partición teniendo el resto de K-1 como datos de entrenamiento. Y una vez obtenidas todas las evaluaciones de cada parte, se calcula la media. La evaluación es sobre la precisión de la predicción, por tanto cuanto más alta mejor.

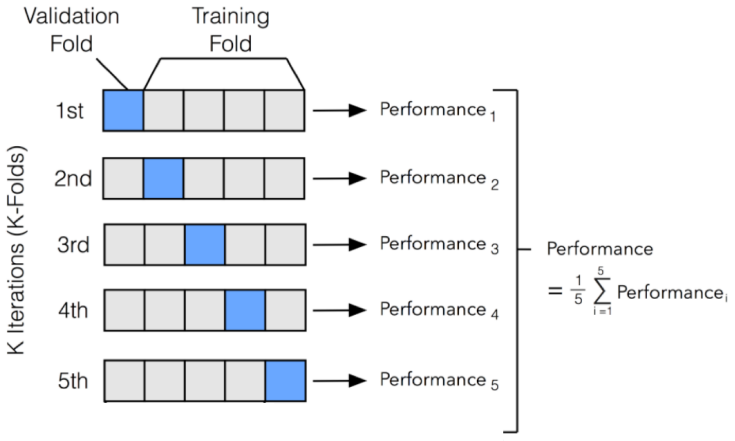


Fig 0 Visualización de lo que consiste Cross Validation: Fuente: [33]

2.1 Seguimiento de la metodología durante el proyecto

Durante la realización de este Trabajo de Fin de Grado ha surgido la pandemia del Covid-19 durante principios de marzo, que ha sido impredecible. Y debido a la cuarentena que se estableció durante el estado de alarma a mediados de marzo, hubo unas semanas de adaptación a la nueva situación que ha sido inevitable el retraso de muchas reuniones, tanto como internas de Slashmobility como las reuniones previstas para mi proyecto.

Aunque esto ha afectado ligeramente a la planificación, que se explican los detalles en el apartado 3.7 Seguimiento de la planificación durante el proyecto, la pandemia no ha afectado a la metodología propuesta para el proyecto y se ha llevado a cabo sin problemas.

3. Planificación temporal

Este trabajo de fin de grado se empezó en febrero de 2020. Su finalización se prevé a finales de agosto de 2020. La duración total prevista del proyecto es de 405 horas.

Teniendo en cuenta que cada día se trabajará en el proyecto 5 horas diarias, la duración esperada total en días son aproximadamente 130 días. Se ha decido trabajar 5 horas diarias, debido a que el convenio dura 735h y por la duración que prevemos del trabajo (de septiembre 2019 a agosto 2020) se ha habido de distribuir las horas de esta manera. La fecha prevista para la defensa del TFG es en octubre de 2020 (véase el apartado Posibles obstáculos y riesgos para su justificación).

A continuación se detallan las tareas épicas (una tarea que agrupa tareas con el mismo objetivo) y el detalle de las tareas que contiene cada una de las épicas. Véase las tareas en detalle en el apartado 4.1 Estimación temporal.

3.1 Gestión del proyecto

Incluye las **tareas correspondientes al curso de GEP**, vinculado al trabajo de fin de grado, y GEP. Se han estimado unas 20h para cada tarea. Cada tarea corresponde a cada entrega de GEP, con un total de 5. La primera tarea que se ha estimado 5h, que requería sólo la lectura del material. Además, incluye las **reuniones para el progreso del TFG**, las cuales se estiman unas 30h.

Para realizar estas tareas, es necesario un ordenador con conexión a Internet. Asimismo, es necesario Google Drive para escribir, almacenar los entregables, Gantter de Google Drive , el Racó y Atenea.

Para las siguientes tareas, es necesario un ordenador para poder desarrollar lo correspondiente y documentar siempre sobre el trabajo realizado.

3.2 Modelo de Datos

Debido a que este proyecto es nuevo, no existen datos reales. Por tanto, se ha de definir **qué modelo de datos** se va a dar uso, consultando con el departamento de RRHH para que se pueda estimar la felicidad del empleado. Una vez definido el modelo de datos, se ha de **generar los datos** de manera que genere unos datos “crudos” que simularán los resultados de las encuestas que más adelante se moldearán al modelo de datos definido anteriormente.

3.3 Algoritmo de procesamiento, limpieza y estructuración

Se ha de **implementar un algoritmo de limpieza de datos**, para eliminar datos no válidos, innecesarios y redundantes, posiblemente con el uso de *Feature Selection*, y **un algoritmo de estructuración de datos**, con la ayuda de *Feature Extraction*, adaptando los datos de entrada al modelo definido. Consiguiendo así uno de los subobjetivos (véase el apartado 1.3.2 Subobjetivos)

3.4 Máquina de predicción de ánimo

Primero hay que **definir los métodos de *Machine Learning*** que se usarán para el estudio del mejor método para la predicción de la felicidad del empleado. A continuación, **integrar el algoritmo de recolección de datos** del apartado anterior. **Entrenar y buscar** los mejores parámetros para cada modelos de los distintos métodos definidos. Y finalmente, **validar los modelos** obtenidos usando *Cross Validation* (véase el apartado de 1.1.2 Términos y conceptos). En este punto habremos logrado el objetivo principal de este TFG (véase el apartado 1.3.1 Objetivos).

3.5 Análisis del mejor modelo

Una vez obtenidos los mejores modelos de cada método, compararlos y **decidir cuál es el mejor**, usando benchmarks de tiempo y precisión en la predicción. Logrando el subobjetivo restante (véase el apartado 1.3.1 Subobjetivos).

3.6 Hito final

Finalmente, hay que acabar la **documentación** de este proyecto y preparar la **presentación** que se usará en la defensa del TFG.

3.7 Seguimiento de la planificación durante el proyecto

Como ya hemos comentado en el apartado 2.1 Seguimiento de la metodología durante el proyecto, durante la realización de este Trabajo de Fin de Grado ha surgido la pandemia del Covid-19 durante principios de marzo, que ha sido impredecible. Y debido a la cuarentena que se estableció durante el estado de alarma a mediados de marzo, hubo unas semanas de adaptación a la nueva situación que ha sido inevitable el retraso de muchas reuniones, tanto como internas de Slashmobility como las reuniones previstas para mi proyecto.

Como en la planificación hemos añadido mucho tiempo de margen por los motivos comentados en el apartado 6. Gestión de Riesgos, estos retrasos realmente no han afectado muy gravemente al proyecto. Y aunque todas las fechas de la planificación se han desplazado debido a estos retrasos, se ha podido llevar a cabo todo sin problema y acabar a tiempo sin necesitar esos meses de margen.

También comentar que al principio se tenía planeado que Slashmobility facilitaría los datos, cosa que conllevaba el riesgo de que la recopilación no fuera suficiente o que la calidad tampoco lo fuera y por ello se habría que volver a rehacer todo. Pero en unas reuniones después de comenzar el proyecto, los de RRHH vieron que los datos que pensaban facilitarme no cumplían con el formato Likert y me avisaron que en vez de esperar a que Slash facilitara los datos, tendría que crearlos yo. Esto implicó realizar varias reuniones más para definir, detallar y comprobar que los datos generados fueran realistas.

Este cambio implica unas reuniones de más, pero con los datos teóricos es imposible que surja el riesgo de que se tuviera que recopilar de nuevo, riesgo explicado en el apartado 6.

Gestión de riesgos y ejecutar todo de nuevo. Por tanto, el “alargamiento” del proyecto quedaría compensado por el “alargamiento” que ha provocado los retrasos debido a la pandemia.

Además, la tarea de “Limpieza de los datos” ya no es necesaria, ya que estamos generando los datos teóricos ya con el modelo de datos ideal. Y la estimación calculada para esta tarea se ha transferido a la nueva tarea de “Generar los datos teóricos”. Por lo que el tiempo total se mantiene y por tanto el presupuesto también.

4. Estimaciones y Gantt

4.1 Estimación temporal

A continuación, vemos una tabla de las tareas de este proyecto, el tiempo estimado en horas de cada tarea, y las dependencias que requiere cada tarea.

Id	Descripción	Tiempo (h)	Dependencias
T1	**Gestión del proyecto**	**115**	
T2	- Herramientas TIC de soporte	5	-
T3	- Contextualización y alcance	20	-
T4	- Planificación temporal	20	T3
T5	- Presupuesto y sostenibilidad	20	T4
T6	- Evaluación final	20	T5
T7	- Reuniones TFG	30	-
T8	**Modelo de Datos**	**30**	
T9	- Definir modelo de datos	5	-
T10	- Generar datos	25	-
T11	**Procesamiento, limpieza y estructuración de datos**	**80**	
T12	- Implementar algoritmo de limpieza	40	-
T13	- Implementar algoritmo de estructuración	40	T9
T14	**Desarrollar máquina de predicción de ánimo**	**65**	
T15	- Definir qué métodos estudiar	5	-
T16	- Preparar los datos de entrenamiento y testeo	5	T13
T17	- Entrenar y buscar los mejores modelos	40	T16
T18	- Validar modelos con Cross Validation	15	T17
T19	**Análisis del mejor modelo**	**20**	
T20	**Hito final**	**75**	
T21	- Documentación del proyecto	60	-
T22	- Preparar defensa del proyecto	15	T21
	Total	385	

Fig 1 Tabla de las tareas con su duración aproximada en horas y sus dependencias. Fuente: Creación propia

4.2 Dependencias entre tareas

4.2.1 Gestión del proyecto

Las dependencias mostradas en esta épica es debido a los límites de entrega de los documentos para la asignatura de GEP. La dependencia crucial la tiene "**Evaluación fina**l" porque es la agregación de éstos.

4.2.2 Dependencias generales

El resto de dependencias tienen una forma similar, ya que para cada implementación, se ha de diseñar o definir primero.

4.3 Concurrencias de tareas

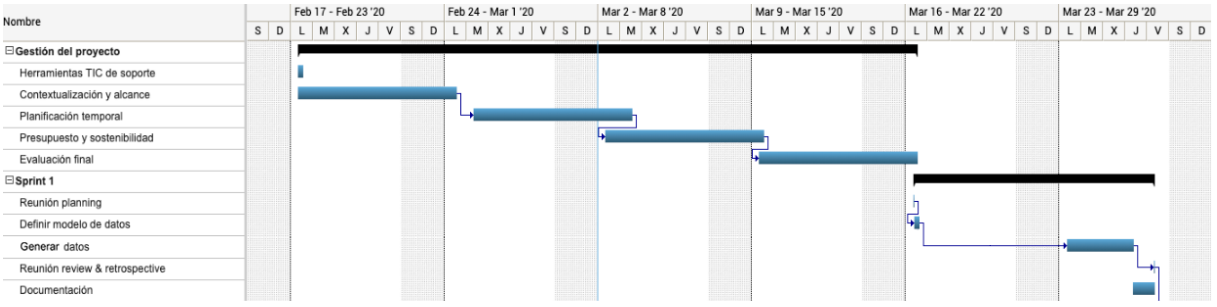
Primero de todo, la **documentación** del trabajo es algo que se puede ir haciendo una vez se acaba una tarea y se empieza la siguiente. Por tanto, es una concurrencia que sucederá varias veces.

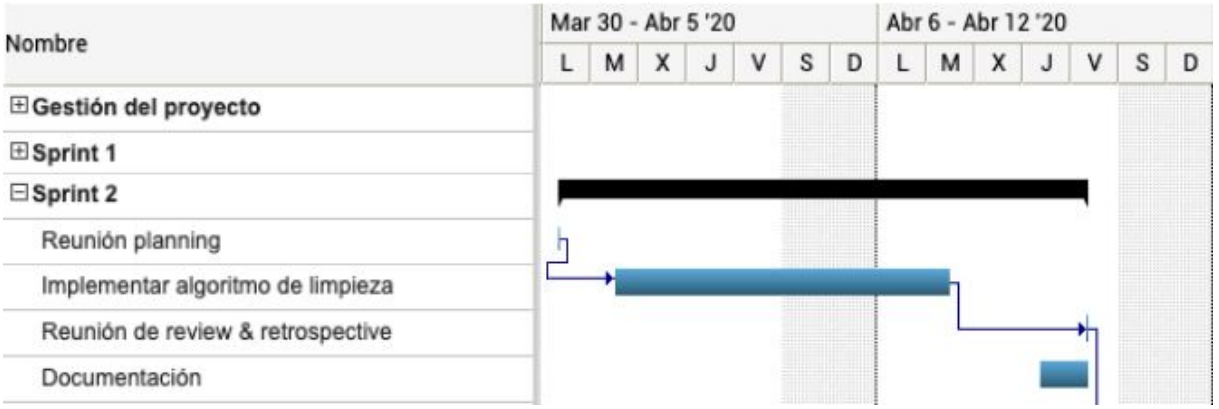
Las otras tareas concurrentes son las de **definición** de conceptos y el **diseño** de algoritmos. El orden para estas tareas las dictará el flujo siguiente:

- Modelo de datos
- Procesamiento de los datos obtenidos
- Máquina de predicción de ánimo
- Análisis de los modelos

De esta manera, tenemos un flujo de trabajo ordenado y organizado en el proyecto.

4.4 Diagrama Gantt





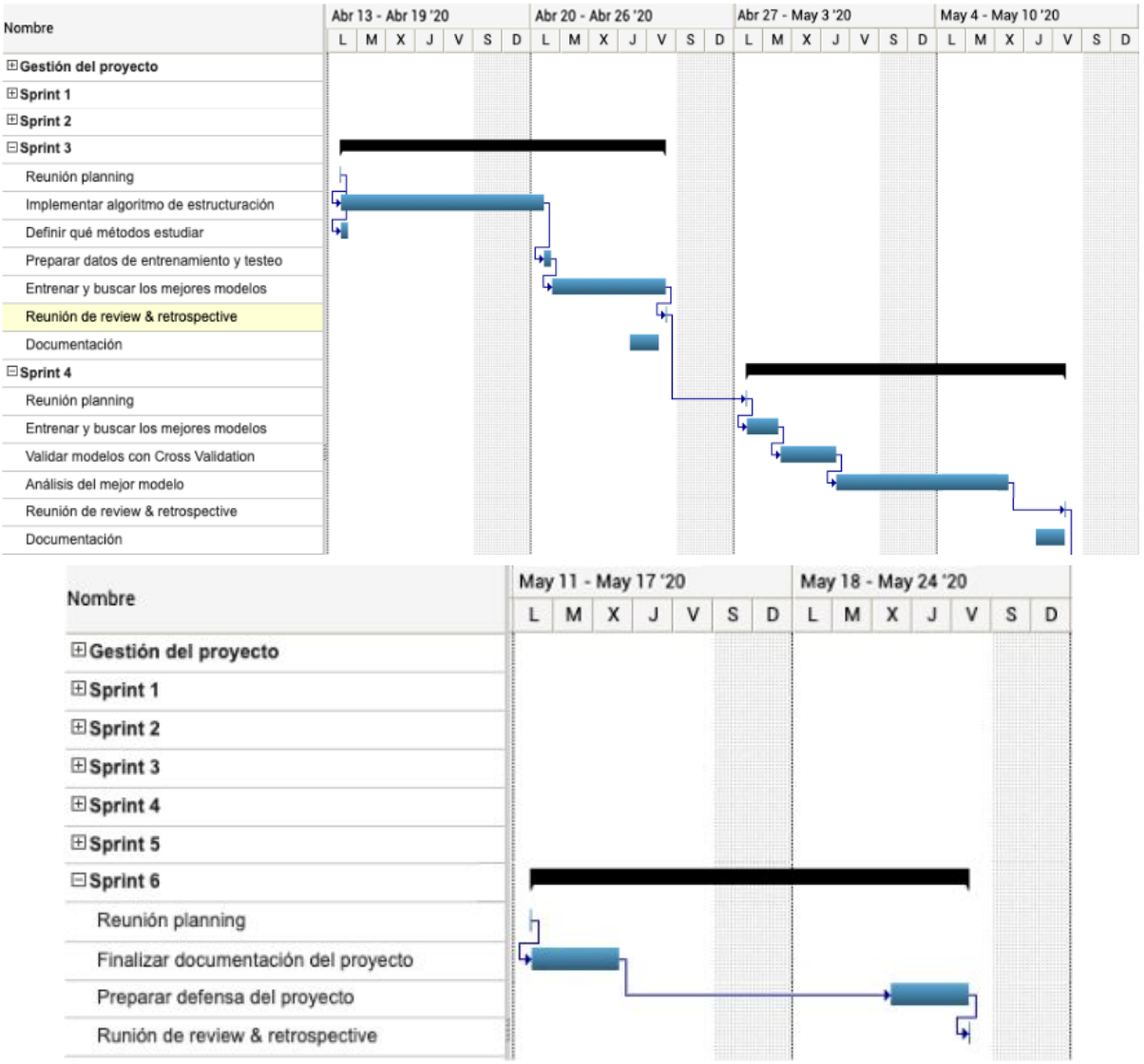


Fig 2 Diagrama de Gantt del proyecto. Fuente: Creación propia

5. Recursos

A continuación se listan todos los recursos usados para este proyecto.

5.1 Recursos humanos

La **autora y responsable** de este TFG. Los **directores** asociados a este trabajo, Gerardo Astone y José María Sánchez, como fuente de información y guías en el proyecto de Kokoro. El doctor **Javier Bejar** del departamento de *Computer Science* como guía en el aspecto técnico del proyecto. Los **profesores de GEP** como guías en la realización de GEP. Y finalmente, por parte de **Slashmobility**, el equipo de desarrolladores asignados al proyecto Kokoro y el departamento de planificación.

5.2 Recursos hardware

- **Ordenador Portátil Asus GL552VW**: 20GB de RAM, procesador Intel Core i7-6700HQ 2.6GHz. Ordenador de trabajo en casa, mayormente para la redacción de las entregas de GEP y la documentación del TFG.
- **Ordenador iMac**: 8BG de RAM, procesador Intel Core i5 2.7GHz. Ordenador de Slashmobility en el cual se realizarán todas las ejecuciones necesarias.

5.3 Recursos Software

- **Documentos de Google Drive**: Para la documentación del TFG y los entregables de GEP.
- **Gantter for Google Drive**: Aplicación de Google Drive que genera diagrama de Gantt. Para la realización del diagrama de gantt presentado en el apartado 4.4 Diagrama de Gantt
- **Visual Studio Code**: Editor de textos refinado y optimizado para el desarrollo de web modernas y aplicaciones *cloud.* Editor que se usará en todo el proyecto.
- **Whatsapp** + **Hangout** + **gmail** : Comunicación entre los recursos humanos que participan en este proyecto.
- **Google Calendar**: Gestión de las reuniones de este proyecto.
- **MACOS Catalina versión 10.15.3 (19D76)**: Sistema operativo del ordenador mac.
- **Windows 10**: Sistema operativo del portátil Asus.
- **Atenea**: Intranet en la cual se encuentra todo lo relacionado con GEP (material y entregas).
- **Raco**: Intranet donde se entrega la documentación final de GEP.
- **JIRA Atlassian**: Herramienta de desarrollo de software pensada para la metodología Agile donde se puede planificar y supervisar, entre otras funcionalidades.

5.4 Recursos Indirectos

Aparte de los recursos que acabamos de listar, también se han de tener en cuenta los recursos indirectos: las **oficinas** de Slashmobility de Barcelona, los **servicios** que proporciona (**internet**, **luz** y **corriente**) y el **mobiliario** en el cual trabajamos.

6. Gestión del riesgo

A continuación se presentan la soluciones respecto a la planificación para cada uno de de los riesgos planteados en el apartado 1.3.4 Posibles obstáculos y riesgos:

6.1 No tener suficientes datos

Este obstáculo se evita, como ya se ha explicado anteriormente, **alargando** el proyecto hasta el siguiente cuatrimestre de tal manera que da margen para que el departamento de RRHH obtenga datos con los que trabajar para cuando empiece el Sprint 3.
Por tanto no es necesario crear nuevas tareas, ni recursos extras.

6.2 Modelo impreciso

Ya que los resultados de un modelo de *Machine Learning* dependen completamente de los datos de entrenamiento, si entrenamos con datos que son parciales, los resultados también lo serán. Por consiguiente, los resultados pueden verse en riesgo a ser irrelevantes, por lo que se tendrá que **volver a realizar la planificación** desde T16-T19 pero usando más datos pero esta vez revisando su imparcialidad, que serán pedidas al departamento de recursos humanos y/o otras empresas. Las horas estimadas no superan las 100 horas (aproximadamente unas 80 horas, ya que el código en sí ya estará hecho, pero se tendrá que hacer pequeñas modificaciones). Cosa que implica **alargar el proyecto** menos de 100h. Por tanto, el plan será de 465h. Por eso la planificación termina unos meses antes del fin del convenio, por si se da este caso. No hacen falta recursos extras.

6.3 Retraso en la planificación

Aparte de alargar el proyecto como se explica en el apartado anterior 6.1, este riesgo se puede evitar manteniendo una **comunicación clara y transparente** entre mi proyecto y el departamento de planificación de Slashmobility. Aún así, la planificación realizada sin riesgos tiene un margen entre la finalización de todo el proyecto y la defensa del proyecto suficiente como para amortizar estos posibles retrasos. Tampoco requiere de tareas ni recursos extras.

7. Presupuesto

En este apartado se detallarán los costes del proyecto según los recursos listados anteriormente (véase el apartado 5. Recursos).

Toda la información que se muestra a continuación ha sido obtenida a través de la empresa Slashmobility.

7.1 Costes de recursos humanos

El coste de los recursos humanos se pueden detallar según los roles implicados en el proyecto y la cantidad de horas dedicadas.

Los roles implicados son: yo, como **gestora del proyecto**, **desarrolladora IA junior**, **desarrolladora *junior*** y **analista IA junior**, el doctor Javier Béjar y los profesores de GEP como **gestores del proyecto** y el equipo de desarrollo de Slashmobility como **ingenieros especialistas de BI**.

El coste por hora de la tabla siguiente son salarios brutos e incluye el coste relacionado a la gestión de la seguridad social y coste de prevenciones.

Rol	Coste por hora (€/h)
Gestor de proyectos	28,00
Ingeniero Especialista BI	46,00
Desarrollador IA junior	9,00
Desarrollador junior	9,00
Analista IA	36,00

Fig 3. Roles implicados y su coste por hora. Fuente: Creación propia

A continuación, calculamos las horas dedicadas de cada rol según las tareas especificadas en el apartado 3. Planificación temporal y visibles en la sección 4.4 Diagrama de Gantt. Como que la mayoría de tareas las desarrolla un rol, el detalle del coste está a nivel de las tareas épicas. En las siguiente tabla distribuimos las horas planificadas en cada tarea épica a los roles correspondientes según la naturaleza de la tarea:

Tarea	Tiempo (h)	Tiempo por rol (h)				
		Gestor	Ing. Esp. BI	Des. IA junior	Des. junior	Analista IA
Gestión del proyecto	**115**	115	-	-	-	-
Modelo de Datos	**30**	-	5	25	-	-
Algoritmo de procesamiento, limpieza y estructuración	**80**	-	-	80	-	-
Máquina de predicción de ánimo	**65**	-	-	65	-	-
Análisis del mejor modelo	**20**	-	-	-	-	20

Hito final	**75**	-	-	-	75	-
Total	**385**	**115**	**5**	**170**	**75**	**20**

Fig 4. Roles implicados y sus horas dedicadas. Fuente: Creación propia

Por tanto, el coste total en el presupuesto se calcula multiplicando las horas totales de cada rol por su salario por hora:

Rol	Horas	Coste (€)
Gestor de proyectos	**115**	3220,00
Ingeniero Especialista BI	5	230,00
Desarrollador IA junior	170	1530,00
Desarrollador junior	75	675,00
Analista IA	20	720,00
Total	**385**	**6375,00**

Fig 5. Roles implicados y su coste total. Fuente: Creación propia

Hay que tener en cuenta que los riesgos comentados en el apartado 1.3.4 Posibles Obstáculos y Riesgos pueden afectar a la cantidad de horas totales (véase el apartado de 6. Gestión del riesgo). Por lo tanto, en los recursos humanos puede haber una **desviación económica** que afecta a la cantidad de horas del desarrollador IA junior y del analista IA.

7.2 Costes de hardware

A continuación se listan los precios, vida útil y la amortización para la duración del proyecto (es decir, 540h, suponemos 1800h de trabajo por año, por tanto, la amortización se calcula de la siguiente manera: coste * 540 / (años vida útil * 1800)) de los recursos hardware:

Hardware	Coste (€)	Vida útil (años)	Amortización (€)
Ordenador Portátil Asus	1200,00	6	60,00
iMac	1694,00	6	84,70
Teclado Mac	72,60	4	5,45
Ratón Mac	72,60	4	5,45
Total	**3039,20**	**-**	**155,59**

Fig 6. Costes Hardware. Fuente: Creación propia

En el caso de los recursos hardware, no debería haber ningún imprevisto grave. El mayor imprevisto sería que los dos ordenadores dejasen de funcionar (si uno dejase de funcionar, se podría trabajar en el otro) que es prácticamente imposible que ocurra. Por tanto, **no existen desviaciones económicas en recursos hardware**.

7.3 Costes de software

El coste de todo el software listado en el apartado de 5.3 Recursos Software son gratis.

A excepción de los sistemas operativos, que están incluidos en el coste de los ordenadores correspondientes, y **JIRA Atlassian** que tiene un coste de 144€/año por usuario. Por tanto la amortización será de 144*540 / 1800 que equivale a **43.20€**.

No puede haber desviaciones ya que todos los productos son software con versiones estables.

7.4 Costes indirectos

A continuación se muestran los costes indirectos del proyecto, que son básicamente el espacio de trabajo, servicios (luz, internet, etc) y el mobiliario. Según Slashmobility, el coste total es de 6200€ anuales, por tanto, la amortización es de 6200 * 540 / 1800, es decir **1860€**, según la empresa. El coste es ligeramente alto debido a que actualmente la empresa está alquilando un espacio en el coworking de Nest City Lab Bcn [23].

No hay desviaciones económicas en este caso ya que las tarifas han sido pagadas por un tiempo más largo que la duración de este proyecto.

7.5 Control de gestión

Como hemos explicado al final de cada apartado, los recursos humanos es el único recurso con opción a una desviación económica significativa. Como se ha explicado en la Gestión de Riesgo en la entrega anterior, la planificación puede alargarse aproximadamente 100 h (una sobreestimación), que requerirá otras 60h dedicadas del desarrollador de IA y otras 20 del analista de IA. Por tanto, el coste humano aumentará como indica la tabla siguiente:

Rol	Horas	Coste (€)
Gestor de proyectos	**115**	3220,00
Ingeniero Especialista BI	5	230,00
Desarrollador IA junior	230	2070,00
Desarrollador junior	75	675,00
Analista IA	40	1440,00
Total	**465**	**7635,00**
Diferencia	**-**	**1260,00**

Fig 7. Costes humanos con el riesgo de alargar el proyecto. Fuente: Creación propia

En el caso de que se dediquen más horas de las estimadas, el presupuesto de los imprevistos tiene que ser calculado por las horas reales dedicadas multiplicado por el salario de la figura 3.

En cuanto a los costes hardware son costes exactos y no pueden variar. Para que haya una afectación grave en el coste, debería ocurrir accidentes graves a ambos ordenadores, como virus que incapaciten el trabajo en el ordenador o daños físicos. Pero las probabilidades son prácticamente nulas.

En cuanto a los costes softwares e indirectos son costes que no variarán ya que son gratuitos o ya han sido pagados por un periodo más largo que la duración de este TFG.

A continuación, se muestra la tabla del presupuesto del proyecto teniendo en cuenta la contingencia y los imprevistos que pueden surgir que afectan al presupuesto (véase el apartado 6. Gestión de riesgos):

Actividad	Coste (€)	Riesgo	Observaciones
Contingencia	853,459	-	Margen de seguridad, calculado como un porcentaje del valor total calculado en un 10% en base del presupuesto.
Imprevistos	-	-	Los costes de las alternativas propuestas para cada plan alternativo de cada riesgo definido en el apartado 6. Gestión del riesgo
Alargamiento Proyecto (Coste: 540€)	108,00	20%	La alternativa será volver a realizar las tareas T16-T19.El coste se ve reflejado en las horas dedicadas de los roles implicados en volver a hacer esas tareas.
Total	**961,459**	**-**	**-**

Fig 8. Coste de contingencia e imprevistos. Fuente: Creación propia

7.6 Presupuesto final

Finalmente, sumamos todos los costes para obtener el presupuesto del proyecto:

Recurso	Coste (€)
Recursos humanos	6375,00
Recursos hardware	155,590
Recursos software	43,20
Recursos indirectos	1860,00
Contingencia e imprevistos	961,459
Total	9395,25
Total (IVA +21%)	**11368,25**

Fig 9. Presupuesto final. Fuente: Creación propia

8. Informe de sostenibilidad

En esta sección se hace una autoreflexión después de contestar la encuesta del conocimiento sobre la sostenibilidad [21].

8.1 Autoevaluación de la competencia de sostenibilidad

Después de haber leído el Módulo 2.6 - El informe de sostenibilidad [22], haber respondido la encuesta anterior y haber realizado la gestión del proyecto (económicamente y de planificación), me he dado cuenta que tenía un conocimiento muy básico sobre el impacto de un proyecto informático.

Económicamente, no tenía noción de todos los recursos implicados en un proyecto (indirectos y humanos), ni la planificación exhaustiva con los riesgos y sus planes alternativos, como se ha realizado en este proyecto.

Ambientalmente, solo tenía en cuenta la huella ecológica una vez lanzado un producto, cuando en realidad existe un consumo energético y/o generación de residuos durante la realización del proyecto.

Socialmente, sólo tenía en cuenta a los agentes directos y algún indirecto. Cuando también puede generar impacto a las personas implicadas en el proyecto durante su realización.

En general, tengo un conocimiento básico de la sostenibilidad en el que se basa en mi propio sentido de mejorar la sociedad, ética y moral, pero no la he aplicado en un proyecto.

8.2 Dimensión económica

➔ **¿Coste que has estimado para la realización del proyecto?**

La estimación del coste del proyecto se detalla en el apartado anterior 7. Presupuesto, en el que se han tenido en cuenta los recursos humanos, hardware, software e indirectos. Con la contingencia y los posibles riesgos. La información sobre los costes han sido obtenidos a través de la propia empresa Slashmobility.

➔ **¿Cómo se resuelven actualmente los aspectos de costes del problema que quieres abordar (estado del arte)? ¿En que mejorará económicamente tu solución respecto a los existentes?**

No existe una herramienta que predice la felicidad del empleado en base a información sobre su trabajo (rendimiento, horas asignadas, entorno laboral, interacción, etc).
Sin embargo, actualmente, la única manera de lidiar con el problema es con el largo proceso explicado en la introducción de este proyecto: pasando encuestas periódicamente, seguidamente de reuniones de seguimiento con el empleado y manteniendo unos análisis de los resultados de las encuestas y las reuniones.

8.3 Dimensión ambiental

➔ **¿Has estimado el impacto ambiental que tendrá la realización del proyecto?**

Actividad	Consumo por hora
Persona en rutina habitual	~ 0'1 kWh
Persona corriendo	~ 1 kWh
Nevera A+	~0'5 kWh
Calefacción eléctrica de baño	~ 1 KWh
Consola PS4	~ 0'125 kWh
Aire acondicionado A+	~ 1 kWh
Televisor LED 32''	~0'03 kWh

Fig 10 Consumos energéticos cotidianos. Fuente: [34]

Este proyecto consume electricidad aunque no genera residuos. El recurso que consume más electricidad es el ordenador, que más o menos consume lo mismo que un televisor (véase la figura 10). Por tanto, la estimación del impacto ambiental sería el número de horas de programación multiplicado por 0.05 kWh.

➔ **¿Te has planteado minimizar el impacto, por ejemplo, reutilizando recursos?**

El único recurso que consume energía es el ordenador, ya que el resto son recursos software usados a través del ordenador. Por tanto, la única manera de minimizar el impacto es apagándolo cuando no se esté usando.

➔ **¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En que mejorará ambientalmente tu solución respecto a los existentes?**

La misma respuesta que en el apartado 8.2, última pregunta.

8.4 Dimensión social

➔ **¿Qué crees que te aportará a nivel personal la realización de este proyecto?**

A manejar un proyecto ambicioso de computación y madurar de alguna manera en el mundo de Machine Learning, tema que me ha estado interesando últimamente, pero no estoy segura de si quiero dedicarme profesionalmente a ello. Con este proyecto tendré una mejor perspectiva de ello y tal vez sea capaz de determinar si quiero continuar por ese camino o no. Además de aprender a gestionar, organizar y planificar un proyecto.

➔ **¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? En que mejorará socialmente (calidad de vida) tu solución respecto a los existentes?**

La misma respuesta que en el apartado 8.2. Aunque en este punto, en el marco del proyecto Kokoro, mejorará la calidad del entorno laboral, tanto como para el empleado como para el departamento de RRHH, que ya no tendrán que dedicar tantas horas a la observación del empleado.

➔ **Existe una necesidad real del proyecto?**

Sí, ya que es parte del proyecto Kokoro y hay módulos que dependen de los resultados obtenidos en este TFG, específicamente la máquina de predicción del comportamiento del empleado el cual recibirá los resultados como uno de los datos de entrada.

9. Leyes y normativas relevantes

En este proyecto podemos distinguir dos partes donde se podrían aplicar leyes o normativas:

1. Los datos usados. Debido a que los datos son generados por mí, y no he usado ningún dato real para la generación, en este caso no aplica ninguna ley. Esto no quita el hecho de que en el proyecto de Kokoro, probablemente si sea sujeta por al menos la Ley de Protección de Datos del Usuario.
2. El código del proyecto. Slashmobility y yo hemos acordado de que sería confidencial, por lo que respecto al código del proyecto, se seguirá el acuerdo de confidencialidad.

A parte de estas dos secciones, se podría considerar la sección de discusión y análisis de los resultados obtenidos. Pero como el contenido es creación u opinión mía, no aplica ninguna ley ni normativa.

10. Desarrollo y aplicación del TFG

A continuación se discutirán el modelo de datos en detalle y el desarrollo del proyecto.

Antes de entrar en detalle, he de recordar que debido a motivos que explicamos con más profundidad en el apartado 3.7 Seguimiento de la planificación durante el proyecto, lo que en un principio sería obtener los datos facilitados por Slashmobility, se ha convertido en generar unos datos teóricos realistas ya que Slashmobility no ha podido facilitar los datos reales. Por tanto, todo este proyecto está basado en estos datos teóricos que generamos específicamente para este TFG siguiendo unas restricciones, en vez de datos reales. En el siguiente apartado, se discute las decisiones tomadas para generar los datos teóricos procurando obtener unos datos realistas.

En cuanto al código del proyecto, por confidencialidad no se mostrará ningún código, pero se explicarán a lo mejor posible lo desarrollado.

El proyecto está desarrollado con el lenguaje de programación python, usando el entorno interactivo de notebooks, específicamente se han creado dos notebooks, que los hemos nombrado “Dataset.ipynb” y “Models Scheme.ipynb”. En el notebook de “Dataset” encontraremos el código usado para generar los datos teóricos y en el notebook de “Models Schemes” encontraremos el código usado para los diferentes métodos ML, buscar los mejores hiper parámetros para cada uno de ellos, entrenarlos y finalmente testearlos.

10.1 Modelo de Datos

El objetivo de este proyecto es predecir la felicidad/autorrealización del empleado en base de las respuestas del empleado de un cuestionario, que se ha de responder periódicamente. Por tanto, el modelo de datos tiene que contener las respuestas o medias de las respuestas.

A partir de ahora, cada vez que use la palabra “felicidad” me refiero a la “felicidad o autorrealización del empleado en dicha empresa”.

Estas preguntas serán distribuidas a cada usuario de la plataforma de Kokoro por un chatbot cada un intervalo de tiempo aún a definir o cuando se quiera entrenar de nuevo la máquina y por tanto, se distribuiría la encuesta con una pregunta final, que la definimos al final de este apartado.

Los datos de Kokoro se basarán en 4 grandes bloques, en el que cada bloque engloba diferentes temas:

- **Compensación**: Salud laboral y bienestar, el salario, clima laboral, beneficios sociales, estabilidad social y la oficina y el equipamiento.
- **Comunicación**: Acompañamiento del empleado (Onboarding a la empresa, evaluación y seguimiento), feedback, evaluación general de la empresa, comunicación interna, red social interna.

- **Cultura**: Satisfacción, compromiso, recomendación de la empresa, felicidad en la empresa.
- **Crecimiento**: Objetivos y plan de carrera, retos profesionales, liderazgo organizacional, formación.

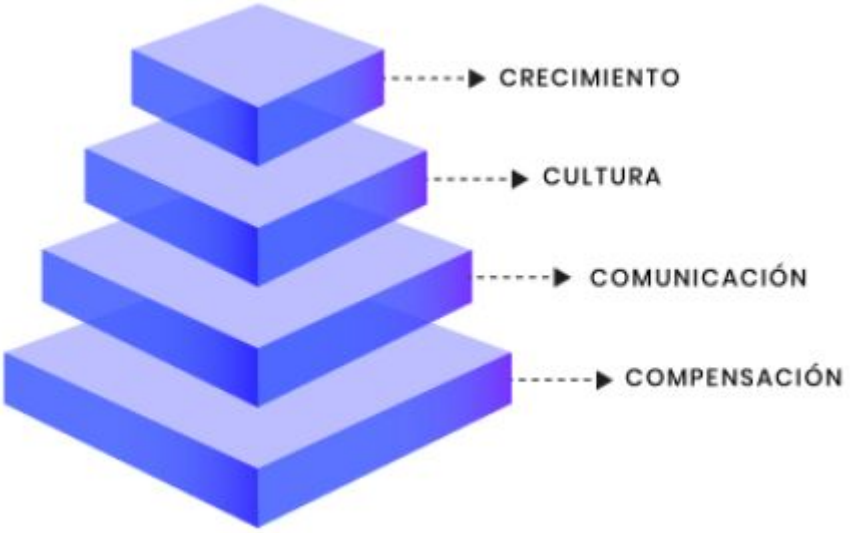


Fig 11. Pirámide de Maslow. Fuente [3]

Estos 4 bloques forman la pirámide Maslow, una jerarquía de necesidades humanas y defiende que conforme se satisfacen las necesidades más básicas (parte inferior de la pirámide), los seres humanos desarrollan necesidades y deseos más elevados (parte superior de la pirámide) [25].

Cada bloque contiene unas preguntas, en las que las respuestas corresponden con un valor:

Respuesta	Valor
Siempre o casi siempre es verdadero.	5
A menudo es verdadero.	4
A veces es verdadero / A veces es falso.	3
A menudo es falso.	2
Siempre o casi siempre es falso.	1

Fig 12. Respuestas Escala Likert y sus valores. Fuente: Creación propia

A continuación definimos las preguntas de cada bloque que está directamente o indirectamente relacionada con la felicidad de un empleado en la empresa, de manera que cuanto más positiva la respuesta, más feliz está en la empresa y viceversa:

COMPENSACIÓN

- Este es un lugar psicológica y emocionalmente saludable para trabajar.

- Resulta sencillo equilibrar la vida profesional y personal.
- Me dan los recursos y equipos para hacer mi trabajo.
- Consideras que recibes un pago justo por tu trabajo.
- Sientes que recibes una parte justa de las ganancias de esta organización.
- Los actuales beneficios sociales de la organización te parecen adecuados.

COMUNICACIÓN

- La comunicación inter e intradepartamental es oportuna y suficiente.
- La dirección nos mantiene informados/as sobre temas y cambios importantes.
- La dirección tiene una visión clara de hacia dónde va la organización y de cómo llegar.
- Mi responsable incentiva, considera y responde sinceramente a mis idea y sugerencias.
- Puede hacer cualquier pregunta razonable a mi responsable de equipo y recibir una respuesta directa.
- Mis superiores reconocen que pueden cometerse errores involuntarios al hacer el trabajo.

CULTURA

- Consideras que la organización contribuye a la sociedad.
- Consideras que tu trabajo supone un valor añadido dentro del equipo.
- Cuando ves que tu equipo logra objetivos, te sientes orgulloso/a.
- Me siento integrado en el equipo.
- Este es un lugar donde lo pasamos bien trabajando.
- Me siento identificado con los valores de la compañía.
- Los responsables implican a las personas en las decisiones que afectan a sus actividades o a su ambiente de trabajo.

CRECIMIENTO

- Aquí todos tenemos la oportunidad de recibir un reconocimiento especial.
- Aquí animamos a las personas a que intenten hacer las cosas de forma distinta o mejor independientemente del resultado.
- Considero que en esta compañía se aprovechan al máximo mis habilidades, conocimientos y experiencia.
- El nivel de exigencia es adecuado.
- Conoces tus responsabilidades individuales.
- Tu responsable te propone periódicamente retos para que crezcas profesionalmente.
- La empresa te ofrece propuestas formativas para fomentar tu crecimiento profesional.

Y finalmente una última pregunta que no se ha de preguntar siempre, sino cada vez que se quiera entrenar la máquina:

- Marca la opción con la te sientes más identificado:

Esta pregunta final es importante para hacer el entreno de manera correcta. Las respuestas de esta pregunta difiere un poco a las demás de la siguiente manera:

Respuesta	Valor
Me siento feliz/autorrealizado en esta empresa.	1
Me siento indiferente en esta empresa.	0
Me siento infeliz/no autorrealizado en esta empresa.	-1

Fig 13 Respuestas de la pregunta final. Fuente: Creación propia

Un total de 26 + 1 preguntas, que como se puede observar, están formuladas de manera que si estás de acuerdo con todo, se podría considerar que te sientes realizado con la empresa, y viceversa.

10.1.1 Modelo de datos teóricos

Estas preguntas se han de pasar periódicamente a los empleados. Por tanto, crearemos datos de tal manera que un empleado, tenga diversas "tandas" de respuestas de la encuesta, a la cual supondremos que son rellenadas 1 vez a la semana. Generamos hasta 16 "tandas" por persona, equivalente a 4 meses de formularios. Remarco el hasta, ya que los empleados podrían haber trabajado menos de 4 meses y por tanto haber contestado menos de 16 formularios. También comentar que el intervalo entre encuestas es teórico y en el marco de Kokoro aún se ha de definir.

Ya que las preguntas están hechas en formato Likert, la media de todas las respuestas es una medida de la felicidad del empleado en la empresa. Es decir, 1 siendo que el empleado no se siente feliz/autorrealizado en la empresa y 5 siendo está completamente feliz y autorrealizado en la empresa.

Entonces, para simular 1 encuesta de 1 empleado, podemos generar para cada pregunta respuestas usando una distribución normal en la cual se centre en un número generado aleatoriamente del 1 al 5 y con cierta variación, que será mínima ya que por la naturaleza de las preguntas, si estas bastante feliz en la empresa, no habrían muchas respuestas muy diferentes (por ejemplo en una pregunta responder con un 1 y en otra responder con un 5).

Pero eso no es todo, con el tiempo un empleado puede aumentar o disminuir su felicidad, debido a diferentes motivos: hacerse amigo de sus compañeros de trabajo, el responsable halaga/culpa al empleado, subida/bajada del salario, etc. Por tanto, cada tanda, aunque no sea muy diferente de la anterior, tendrá cierta variación. Para simular esto, al generar la siguiente tanda, se cogerá la respuesta de la anterior tanda y se le aplicará una pequeña variación. La variación puede ser positiva o negativa para todas las respuestas, ya que de esta manera podremos simular empleados que durante una semana se sienten muy bien en

la empresa pero en la siguiente no tanto, y en la siguiente vuelve como estaba en la primera semana o hasta mejor.

Por lo tanto, cada persona tendrá los siguientes datos:

$$persona = [[Respuestas1], [Respuestas2], \ldots, [RespuestasN]]$$

En el que *RespuestasX* son todas las respuestas de la encuesta *X*, que tendrá 26 valores del 1 al 5. A estos datos, los llamaremos **datos crudos**.

Más adelante, en el preprocesamiento de los datos, convertiremos estos datos crudos de manera que obtengamos lo siguiente:

$$persona = [Media1, Media2, \ldots, Media16, happiness]$$

En el que *MediaX* es la media que tuvo en la encuesta *X* y *happiness* es la respuesta a la pregunta final con valor entre -1 y 1.

De manera que se pueda ver una perspectiva de la felicidad del empleado en el tiempo transcurrido desde la primera encuesta a la última. Con esto, el objetivo principal sería entrenar la máquina con este modelo de datos e intentar predecir la felicidad del empleado actual según las respuestas obtenidas durante las últimas encuestas.

Por tanto, estaríamos prediciendo la felicidad general que tiene el empleado en esta empresa hasta el momento.

10.1.2 Generación de datos teóricos

Los datos que necesitamos generar son las diferentes respuestas en las encuestas que pasará el chatbot.

Por tanto, necesitamos generar un conjunto de 26 números del 1 al 5 para cada encuesta, pero sin que sea totalmente aleatorio, ya que como se ha explicado anteriormente, las preguntas están formuladas de manera que plasman cierta tendencia.

Como no podemos mostrar el código usado directamente, explicaremos y referenciaremos los métodos de las librerías usadas.

Primero, hemos generado una función que genera 26 números enteros con una distribución normal que toma como parámetros una media y una variación. Esta función la llamaremos ***generate_answers(mean, sd)***, en el que usamos el siguiente método:

- ***truncnorm.rvs***: Devuelve variables aleatorias de una distribución normal continua truncada.

Con ***generate_answers(mean, sd)*** podemos generar 26 enteros del 1 al 5 con una distribución normal con centro “mean”. De esta manera simulamos la tendencia del empleado (si en general es feliz sus respuestas “bailarán” a números más altos).

A continuación, hemos generado otra función que popula una persona con diferentes encuestas. Como una persona puede haber estado menos tiempo que las 16 encuestas totales comentadas al principio de este apartado, una persona tendrá N encuestas , donde 1 <= N <= 16. Esta función la llamaremos ***generate_person_answers(N)***.

Para la primera encuesta, determinamos de manera aleatoria usando ***np.random.randint*** la media de la encuesta para que sea el centro de la distribución normal con una desviación aleatoria entre 0.85 y 1.00.

Se ha decidido usar el 0.85 - 1.00 después de experimentar con diferentes intervalos de desviación y comparando los resultados que simularán las respuestas de las encuestas. RRHH confirmó que los resultados usando una desviación del 0.85-1.00 eran las más realistas.

Y con la media aleatoria obtenida y la desviación aleatoria, llamamos ***generate_answers(mean, sd)***, obteniendo la primera encuesta del empleado con respuestas aleatorias de una distribución normal continua truncada centrada en la media aleatoria y desviación que acabamos de obtener.

Para las siguientes encuestas, se escoge aleatoriamente si en la nueva encuesta el empleado tiene una actitud mejor, igual o peor que en la encuesta anterior (mejor, igual o peores resultados). Llamaremos a esta nueva "actitud" la **pendiente de su felicidad**, si en la siguiente encuesta se siente más feliz que en la anterior, su pendiente es positiva, si se siente igual que la anterior, no tendrá pendiente, y si se siente menos feliz, la pendiente será negativa. Según esta pendiente, cogemos la respuesta que tuvo en esa misma respuesta en la anterior encuesta y se le añade una desviación positiva, negativa o sin desviación, dependiendo de la pendiente escogida.

De esta manera, las respuestas entre encuestas serán similares, pero pueden tender a ser más altas o bajas con el tiempo, o tener altibajos, que según RRHH todos estos casos son posibles.

Finalmente, para generar un número alto de ejemplos de empleados, hemos creado una última función llamada ***generate_dataset(X)*** donde la *X* es el número de empleados a generar. En esta función simplemente calculamos para cada empleado una N aleatoria del 1 al 16, que corresponden a las encuestas que ha respondido.

Para darle un toque más realista, en una empresa suelen haber más empleados con más tiempo en la empresa que empleados nuevos, por lo que hemos limitado a que el 70% de los empleados tengan respondidas 16 encuestas y el 30% restante un número aleatorio del 1 al 15.

10.2 Preprocesamiento de datos

Ya que los datos son creados por nosotros, no hay mucho que preprocesar. En este caso, solo queremos obtener las medias de cada encuesta para cada persona.

Para ello aplicaremos Feature Extraction de tal manera que para cada persona obtendremos lo siguiente:

$$persona = [Media1, Media2, \ldots, Media16, happiness]$$

en la cual *MediaX* es la media que obtuvo la persona en la encuesta *X* y *happiness* es el valor que se intenta predecir.

Ya que el dataset y su contenido es algo que estoy creando yo, cada decisión tomada para generar estos datos han sido consultados antes con el personal de RRHH de manera que estos datos fueran lo más realistas posibles.

Para la categorización de un empleado según sus medias en las encuestas tomadas, se ha de tener en cuenta varios puntos:

- La felicidad en la empresa de un empleado, se ve afectada desde el primer momento en el que hace contacto, es decir, todas las encuestas son relevantes hasta que pasa cierto umbral de tiempo.
- Las respuestas en las encuestas, pueden variar según el momento en el que se esté dando, por tanto, puede ser que hayan semanas en el que el empleado se vea peor o mejor, pero si se le pregunta su felicidad, probablemente los eventos recientes tengan mucho más peso que eventos anteriores. Por lo tanto, probablemente, los resultados de las últimas encuestas se vean reflejadas esas opiniones.
- Aun si los eventos más recientes, es decir, las encuestas más recientes tendrán un peso mucho más grande, los resultados anteriores, podrían tener cierta relevancia, aunque menor.
- Para este documento, supondremos que hay cierto límite temporal en el que las encuestas ya no son relevantes. En este caso, he supuesto que después de 16 encuestas, hechas con un intervalo de 1 semana entre ellas, es decir, un total de 4 meses, las encuestas que hizo el empleado ya no son relevantes. Este límite, una vez con datos reales, se puede ir ajustando cambiando el intervalo de tiempo entre las encuestas.

Machine Learning es un método que se usa para predecir comportamientos. Y todo comportamiento, en el trasfondo tiene ciertos patrones.

En este caso, como estoy generando los datos yo misma, tengo que crear/definir cierto patrón que cumpla los puntos anteriores y relacione las medias generadas. Si fuese todo aleatorio, no habría nada que predecir, por eso hemos generado las medias de cada persona como se ha explicado en el apartado anterior (10.1.2 Generación de datos teóricos).

Ya que las encuestas están hechas con el método Likert y las preguntas están hechas de manera que cuanto más alta sea la media más feliz estás en la empresa, puedo decir, que queremos buscar un indicador con el mismo rango, ergo, un indicador que vaya del 1 siendo la peor puntuación a 5 la mejor puntuación. Y además, tiene que estar basado en las

medias, que ya están en este intervalo. Por tanto, el mejor y simple algoritmo es usar una ponderación con pesos.

Esta ponderación tiene que ponderar las medias más antiguas con poco peso y las más recientes con mucho más peso. Se podría pensar hasta en una ponderación con pesos algo exponenciales, pero sin llegar a altas cifras en pocos pasos (x^2, en 5 pasos ya estamos hablando de 25, en 8 ya saltamos a 256 y en 16 se dispara a 65536). Así que, para aplanar la curva exponencial, he decidido usar la siguiente fórmula de pesos:

$$\frac{x^2}{N}$$

Siendo *N* el número de encuestas dadas por el empleado.

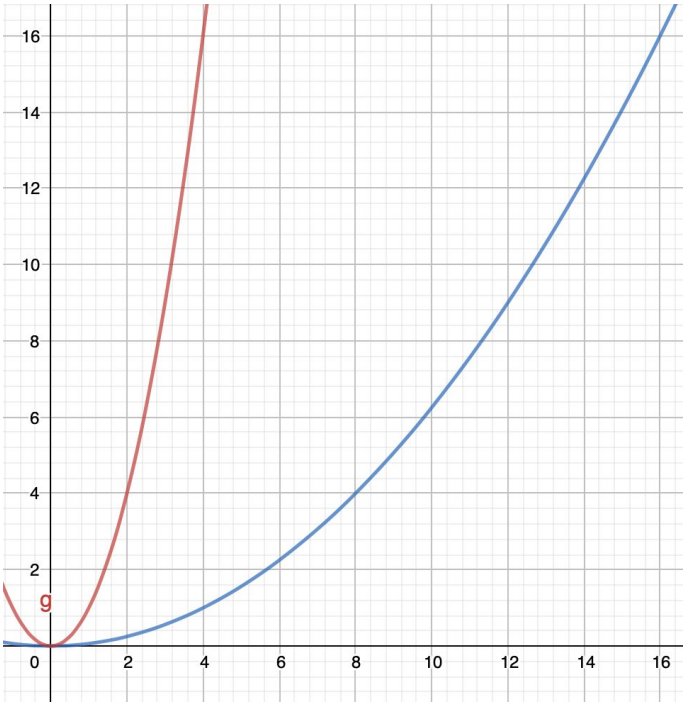


Fig 14 Gráficos de los pesos usando x^2 (rojo) y $\frac{x^2}{16}$ (azul).Fuente: Creación propia con el uso de Geogebra

Una vez definidos los pesos para cada encuesta según el total de encuestas que ha dado el empleado, he definido el indicador de la siguiente manera:

$$score = sum\left(\frac{Y_i \times Media_i}{sum(Y_i)}\right)$$

Siendo Y_i el peso según la fórmula de pesos, $Media_i$ la media de la encuesta i.

Según este indicador, clasificamos de la siguiente manera:

score < 1.75	Infeliz
1.75 <= score <= 3.5	Indiferente
3.5 <= score <= 5	Feliz

Fig 15 Clasificación según el indicador calculado. Fuente: Creación propia

Esta clasificación la asignaremos a la variable *happiness* de la persona.

Este dato, cuando se apliquen datos reales, se obtendrá de la pregunta final en vez de calcularlo con el indicador que acabamos de definir. A través de las encuestas, cada un intervalo de tiempo o cada vez que se quiera volver a entrenar el modelo se debería añadir en la encuesta la pregunta final. Sobreescribiendo el último valor, si es que había.

Por tanto, como en este trabajo no tenemos datos reales, el preprocessing será solamente Feature Extraction.

Los datos que recibimos para cada persona son todas sus respuestas de cada encuesta, es decir, N arrays con 16 números que son las respuestas de las 16 preguntas. Siendo N el número de encuestas que ha hecho.

Por tanto, tenemos que hacer Feature Extraction de cada array a la media de la array. De esta manera, en vez de tener todas las respuestas de cada encuesta, tendremos simplemente las medias de sus encuestas. Que como mucho serán 16 y como mínimo serán 1.

Después calcular el indicador explicado anteriormente para poder clasificar la felicidad del empleado y añadirlo en los datos.

Al ser datos generados, estos datos están siguiendo un patrón a la perfección, cosa que como todos sabemos, en la realidad es imposible. Por tanto, para dar realismo tenemos que generar ruido en los datos. Así que será suficiente añadiendo ruido en el indicador, para hacerlo simplemente añadimos un valor de una distribución normal al indicador.

De esta manera, en la variable target "happiness" no seguirá a la perfección el patrón que he impuesto.

10.3 Modelos de Predicciones

Para este proyecto experimentamos con los siguiente métodos de ML:

- Decision Tree
- Random Forest
- Linear Regression
- Multilayer Perceptron (MLP)

Más adelante explicamos en detalle el concepto básico del método, seguido de los resultados obtenidos.

Todos estos métodos que hemos usado pertenecen a la librería de python skLearn, una de las librerías más potentes de ML en python. Para más detalle podéis consularlo en su página oficial https://scikit-learn.org/stable/.

Para agilizar un poco el código, hemos creado una función llamada ***find_hyperparameters_and_cv(classifier, parameters)*** donde *classifier* es el método clasificador y *parameters* un diccionario con los diferentes parámetros y opciones para este parámetro. Lo que hace esta función es aplicar ***GridSearchCV*** , una función que aparte de buscar los mejores parámetros, realiza el cross validation. El inconveniente de usar esta función es que puede llegar a tardar mucho si se le pasa muchas opciones de parámetros, ya que testea cada combinación de parámetros.

Por lo que en cada método, analizaremos primero con la mayoría de parámetros no triviales e intentaremos descartar aquellos que en diferentes ejecuciones parece mostrar un "mejor parámetro" diferente, ya que si ***GridSearchCV*** siempre encuentra los mejores parámetros.

Cabe comentar, que dependiendo de la generación del dataset (ya que las respuestas de la primera encuesta es aleatoria), la cantidad de cada clase variará. Y seguramente, una de las clases tenga menos ejemplos que otra.

Esto puede resultar en un underfit en el modelo, lo cual lleva a predicciones peores en cuanto a casos similares a la clase con pocos ejemplos.

Para solucionar este problema, podemos usar el método **SMOTE**, **Synthetic Minority Over-sampling Technique**, de la librería imblearn.

SMOTE es una técnica para clasificaciones en el que la población de datos no es equilibrada. Esta técnica añade nuevos datos que pertenecen a la clase minoritaria basándose en ejemplos de la clase minoritaria.

También comentar que para todos los modelos, se usa el mismo conjunto de training/test y que los mejores parámetros obtenidos es dependiente al conjunto de training/test. Por lo que es importante que cada vez que se entrene nuevamente, revisar de nuevo los mejores parámetros del método que discutamos que es el mejor.

A continuación, se detallan para cada método qué conjunto de parámetros y sus opciones hemos aplicado, el resultado de los mejores parámetros.

10.3.1 Decision Tree

Decision Tree [26] es un algoritmo de Aprendizaje Supervisado de ML basado en un Decision Tree. Un Decision Tree consiste en tres componentes:

- **Nodos**: Decisión a tomar. Por ejemplo, "ha respondido la pregunta número 4 con un valor superior a 3?"
- **Aristas**: La respuesta a la pregunta del nodo. Si o No. Las aristas conectan el nodo de la pregunta al siguiente nodo pregunta.

- **Nodos Hojas**: Últimos nodos del árbol. Contienen la clasificación final, en este caso: “Infeliz”, “Indiferente” o “Feliz”.

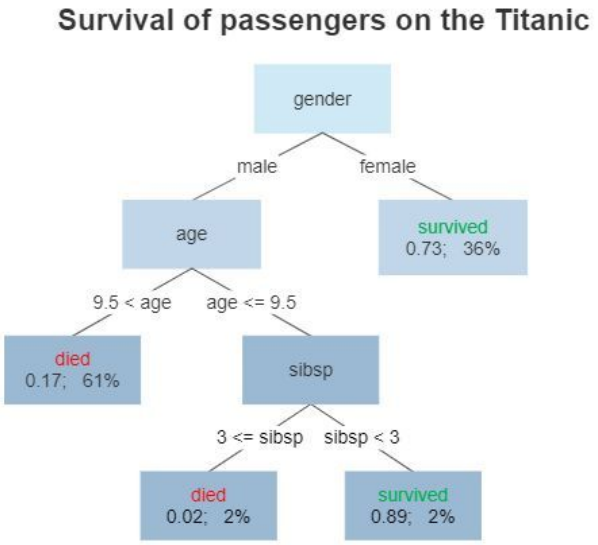


Fig 16. Ejemplo de una Decision Tree sobre la probabilidad de supervivencia de una persona en el Titanic. Fuente:[27]

Por lo que predecir usando Decision Tree es recorrer este árbol desde el nodo inicial hasta llegar a un nodo hoja.

Los hiperparámetros que se le puede pasar los podemos encontrar todos listados en la documentación de Decision Tree Classifier de sklearn.

Los parámetros con los que vamos a experimentar y nos parecen relevantes son los siguientes:

Parámetro	Descripción	Valores a combinar
criterion	Función para medir la calidad de la arista tomada. - ***Gini***: Mide con qué frecuencia un elemento aleatóriamente escogida del conjunto podría estar mal clasificada. - ***Entropy***: La idea es similar a *gini*, pero en vez de usar probabilidades, usa logaritmos.	gini, entropy
splitter	Estrategia usada para escoger qué arista tomar en	best, random

	cada nodo.	
max_depth	Profundidad máxima del árbol.	None, 4, 8, 16, 32, 64, 128

Fig 17 Tabla de los hiperparámetros de Decision Tree, su descripción y los valores con los que experimentamos. Fuente: Creación propia

Los mejores parámetros para Decision Tree obtenidos a través del ***SearchGridCV*** son:

Parámetro	Mejor valor
criterion	**gini**
splitter	**best**
max_depth	**16**

Fig 18 Mejores hiperparámetros Decision Tree encontrados con *SearchGridCV*. Fuente: Creación propia

El **tiempo de entreno** con estos mejores parámetros es de **1.7s.**

Aunque hay que comentar que el *max_depth* en diferentes ejecuciones da diferentes valores.

En cuanto a las predicciones, se han ejecutado varias ejecuciones y obtenemos lo siguiente:

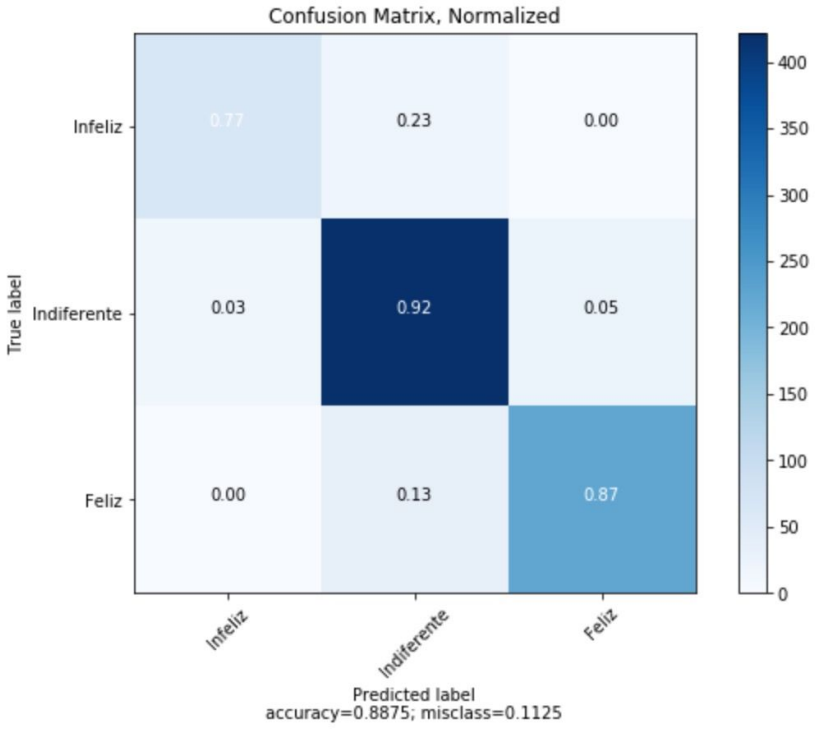


Fig 19 Confusion matrix del Decision Tree. Fuente: Creación propia

En la figura 19, mostramos un gráfico que se llama **confusion matrix** es una tabla o gráfico que permite visualizar la efectividad de un algoritmo, en este caso estamos comprobando la precisión en la predicción de este método. En el eje vertical, tenemos 3 clases (“Infeliz”, “Indiferente”, “Feliz”) verdaderos, y en el eje horizontal, las predicciones hechas por el modelo. Por ejemplo, si tenemos una persona que su resultado es “Infeliz”, pero el modelo predice que es “Indiferente”, esta persona se añadirá a la casilla (“Indiferente”, “Infeliz”). Aparte de la precisión de cada clase, también se obtiene la precisión general de la predicción, en la parte inferior. Este gráfico de ahora en adelante lo usaremos para visualizar los resultados.

Además, hemos normalizado los valores para el mejor entendimiento. Lo ideal sería tener un alto porcentaje en la diagonal (diagonal de izquierda superior a derecha inferior, la diagonal identidad).

Diferentes ejecuciones de las predicciones con los mismos datos suelen dar valores similares a los valores mostrados en la figura 19. Donde podemos ver que la precisión ronda el 89% y específicamente la precisión en las clases “Infeliz”, “Indiferente” y “Feliz” son del **77%, 92%, 87%** , respectivamente.

Así que la precisión para Decision Tree es de **88.75%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.02s**.

Decision Tree + SMOTE

Ejecutamos lo mismo pero aplicando **SMOTE** por si pueden haber diferencias a los resultados que sean significativos:

Los mejores parámetros para **Decision Tree + SMOTE** obtenidos:

Parámetro	Mejor valor
criterion	**gini**
splitter	**best**
max_depth	**32**

Fig 20 Mejores hiperparámetros Decision Tree + SMOTE. Fuente: Creación propia

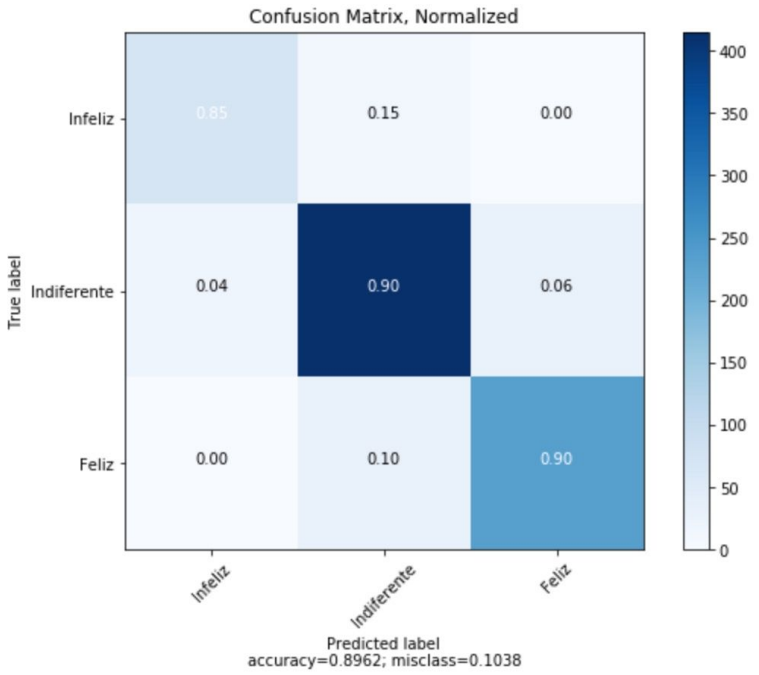


Fig 21 Confusion matrix Decision Tree + SMOTE. Fuente: Creación propia.

Excepto *max_depth* que anteriormente ya hemos comentado que parece ser que *max_depth* en diferentes ejecuciones *SearchGridCV* da distintos valores, el resto parece ser que sigue igual. El tiempo de entreno también se mantiene igual.

Por tanto el **tiempo de entreno** con estos mejores parámetros es de **1.7s.**

En cuanto a las predicciones, observamos que en diferentes ejecuciones los valores tampoco suelen ser muy distintos a los valores mostrados en la figura 21. Vemos en la figura 21 que la precisión ronda el 90%, que es **mejor** que sin SMOTE y específicamente la precisión en las clases "Infeliz", "Indiferente" y "Feliz" son del **85%, 90%, 90%**, respectivamente. Por lo que, ha habido un claro nivelado en la precisión en las clases gracias a la "replicación" de ejemplos en las clases menos abundantes. Aunque podemos notar que ha habido una cierta disminución en la precisión de otras clases ("Indiferente"). Pero aplicando SMOTE tenemos una mejor precisión en las tres clases.

Así que la precisión para Decision Tree + SMOTE es de **89.62%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.07s**.

10.3.2 Random Forest

Random Forest también es un algoritmo Supervisado de ML. Random Forest [28] crea y combina múltiples Decision Trees, creando así un "bosque". En vez de confiar en que un árbol prediga correctamente, este método predice en base a muchos árboles, escogiendo la clase más predecida entre los árboles, como se puede observar en la figura 22. La característica de este método es que la decisión de qué arista tomar es aleatorio.

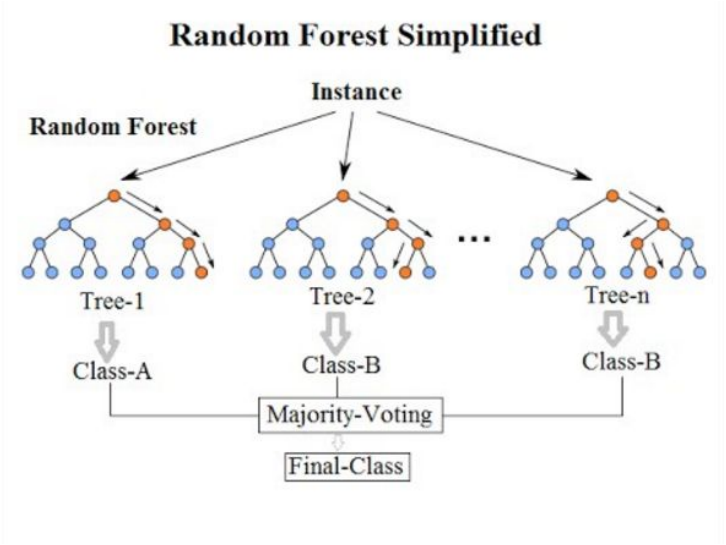


Fig 22 Visualización de cómo funciona un Random Forest. Fuente: [29]

Los hiperparámetros que se le puede pasar los podemos encontrar todos listados en la documentación de Random Forest Classifier de sklearn Que en general se parecen a los de Decision Tree, ya que un Random Forest es un conjunto de Decision Trees.

Los parámetros con los que vamos a experimentar y nos parecen relevantes son los siguientes:

Parámetro	Descripción	Valores a combinar
criterion	Función para medir la calidad del split. - ***Gini***: Mide con qué frecuencia un elemento aleatóriamente escogida del conjunto podría estar mal clasificada. - ***Entropy***: La idea es similar a *gini*, pero en vez de usar probabilidades, usa logaritmos.	gini, entropy
n_estimators	Número de árboles en el bosque.	50, 100, 200
max_depth	Depth máximo del árbol.	None, 4, 8, 16, 32, 64, 128

Fig 23 Hiperparámetros de Random Forest, su descripción y los valores con los que experimentamos. Fuente: Creación propia

Los mejores parámetros para Random Forest obtenidos:

Parámetro	Mejor valor
criterion	**gini**
n_estimators	**100**
max_depth	**16**

Fig 24 Mejores hiperparámetros de Random Forest. Fuente: Creación propia

Las ejecuciones con estos parámetros tardan alrededor de 80s.

Por tanto el **tiempo de entreno** con estos mejores parámetros es de **80s**.

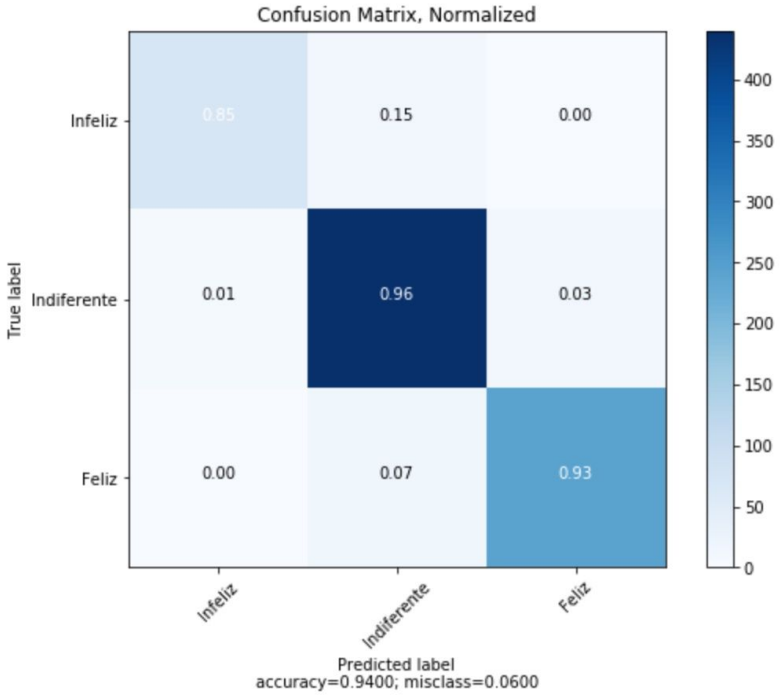


Fig 25 Confusion matrix de Random Forest. Fuente: Creación propia

Y observamos que la precisión ronda el 94% y específicamente la precisión en las clases “Infeliz”, “Indiferente” y “Feliz” son del **85%, 96%, 93%**, respectivamente.

Así que la precisión para Random Forest es de **94.00%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.41s**.

Random Forest + SMOTE

Ejecutamos lo mismo pero aplicando SMOTE y los mejores parámetros obtenidos:

Parámetro	Mejor valor
criterion	**gini**

n_estimators	**50**
max_depth	**None**

Fig 26 Mejores hiperparámetros Random Forest + SMOTE. Fuente: Creación propia

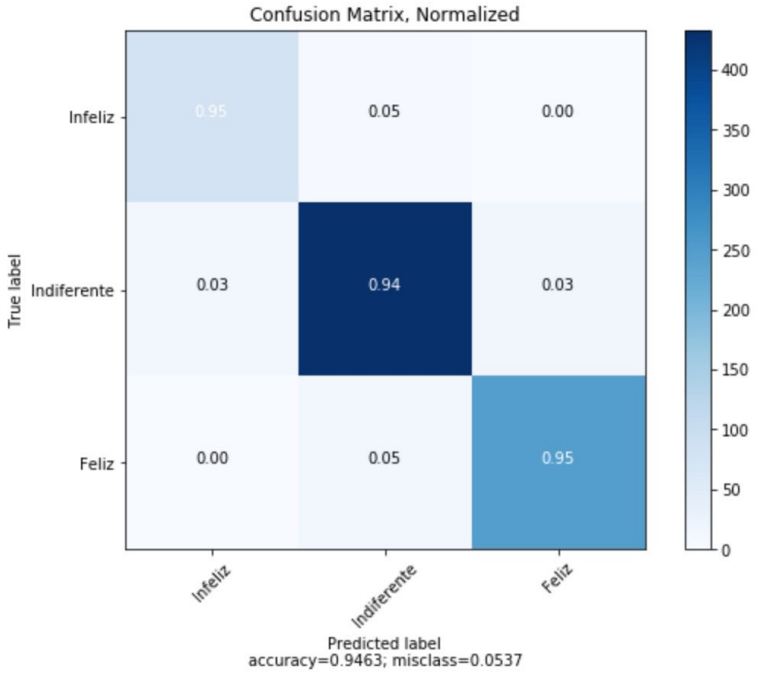


Fig 27 Confusion matrix Random Forest + SMOTE. Fuente: Creación propia

El tiempo de entreno se mantiene igual que el Random Forest normal. Aunque el *n_estimators* y *max_depth* han cambiado respecto al Random Forest sin SMOTE. Pero esto no es debido al hecho de aplicar SMOTE, ya que realizando diversas ejecuciones de Random Forest + SMOTE, estos dos hiperparámetros van cambiando de valores. Pero siguen siendo relevantes en la combinatoria, ya que si se quita de la combinatoria, el parámetro *criterion* no parece tener un valor dominante mostrando muchas veces tanto *'gini'* como *'entropy'*. Por lo que hemos decidido mantener los 2 parámetros en la combinatoria.

El **tiempo de entreno** con estos mejores parámetros es de **80s.**

En cuanto a las predicciones, observamos que en las diferentes ejecuciones los valores rondan por los valores mostrados en la figura 27. Donde podemos ver que la precisión ronda el 95% y específicamente la precisión en las clases "Infeliz", "Indiferente" y "Feliz" son del **95%, 94%, 95%**, respectivamente. Por lo que, ha habido una clara mejora, tanto como en Infeliz como Feliz al coste de un 2% de Indiferente.

Así que la precisión para Random Forest + SMOTE es de **94.63%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.43s**.

10.3.3 Linear Regression

Linear Regression es un algoritmo de aprendizaje supervisado de ML. Este algoritmo, como implica su nombre, usa una regresión lineal para descubrir la relación lineal entre el input, en nuestro caso las medias de las encuestas, y el output, la variable *happiness*.

Los hiperparámetros que se le puede pasar los podemos encontrar todos listados en la documentación de Linear Regression Classifier de sklearn.

Los parámetros con los que vamos a experimentar y nos parecen relevantes son los siguientes:

Parámetro	Descripción	Valores a combinar
C	Inversa de la fuerza de regulación. Cuanto más pequeño el valor, más fuerte se regula.	0.01, 0.1, 1, 2, 5, 10
penalty	El método de penalización. Si es 'none', no se aplica ninguna regularización.	l1, l2, elasticnet, none
dual	Si usa formulación dual o primal.	True, False
fit_intercept	Especifica si una constante (conocida como 'bias' o 'intercept') se añade en la función de la decisión.	True, False
solver	Algoritmo a usar para la optimización del problema. - **'liblinear'** es ideal para conjuntos de datos pequeños, mientras **'sag'** o **'saga'** son más rápidos en conjuntos grandes. - Solamente **'newton-cg'**, **'sag'**, **'saga'** y **'lbfgs'** son capaces de lidiar con problemas de múltiples clases. Por lo que esperamos que salga una de estas 4.	newton-cg, lbfgs, liblinear, sag, saga

max_iter	Número máximo de iteraciones para converger.	100, 250, 500
warm_start	Si **'True'**, usa la solución de la iteración anterior como inicialización .	True, False

Fig 28 Hiperparámetros de Linear Regression, su descripción y los valores con los que experimentamos. Fuente:Creación propia

Los mejores parámetros para Linear Regression obtenidos son:

Parámetro	Mejor valor
C	**0.1**
dual	**False**
fit_intercept	**True**
max_iter	**100**
penalty	**l2**
solver	**newton-cg**
warm_start	**False**

Fig 29 Mejores hiperparámetros para Linear Regression. Fuente: Creación propia

Como esperábamos, el *solver* es uno de los cuatro que habíamos comentado que debería salir.

El **tiempo de entreno** con estos mejores parámetros es de **730s.**

En cuanto a las predicciones, obtenemos lo siguiente:

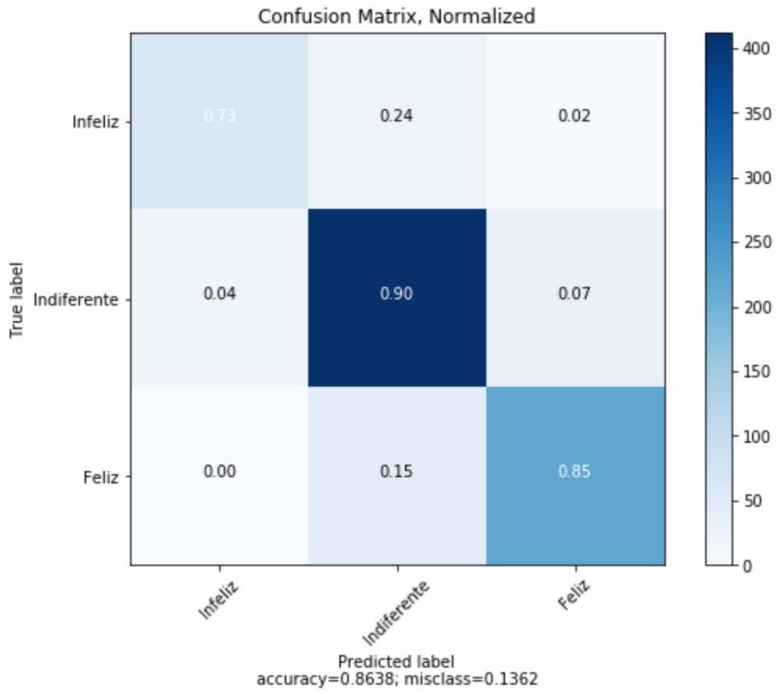


Fig 30 Confusion matrix de Linear Regression. Fuente: Creación propia

Podemos ver que la precisión ronda el 86% y específicamente la precisión en las clases “Infeliz”, “Indiferente” y “Feliz” son del **73%, 90%, 85%**, respectivamente. De momento, es el método con la peor precisión de los métodos vistos.

Así que la precisión para Linear Regression es de **86.38%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.2s**.

Linear Regression + SMOTE

Ejecutamos lo mismo pero aplicando SMOTE y los mejores parámetros son:

Parámetro	Mejor valor
C	**0.1**
dual	**False**
fit_intercept	**True**
max_iter	**100**
penalty	**l2**
solver	**newton-cg**
warm_start	**False**

Fig 31 Mejores hiperparámetros para Linear Regression + SMOTE. Fuente: Creación propia.

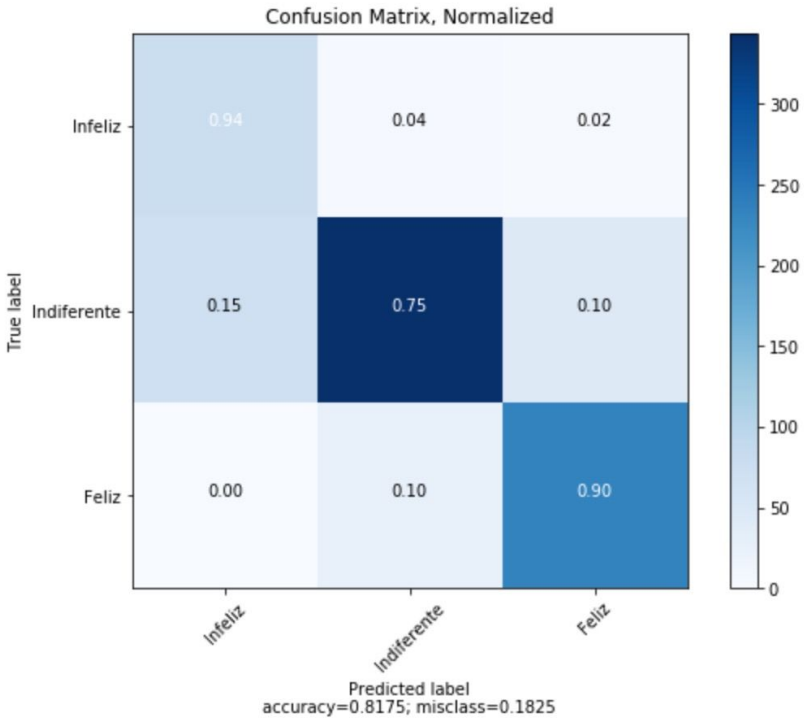


Fig 32 Confusion matrix de Linear Regression + SMOTE. Fuente: Creación propia

El tiempo de entreno se mantiene igual que el Linear Regression normal. No hay diferencias en los valores de los mejores parámetros respecto a Linear Regression sin SMOTE.

Por tanto el **tiempo de entreno** con estos mejores parámetros es de **730s.**

En cuanto a las predicciones, podemos ver que la precisión ronda el 81%, mucho peor que el normal! Aunque la precisión en las clases “Infeliz”, “Indiferente” y “Feliz” son del **84%, 75%, 90%**, respectivamente. Podemos suponer que el descenso de la precisión en la clase Indiferente, ha causado un empeoramiento significativo a la precisión general, poniendo al método Linear Regression + SMOTE con la peor precisión, siguiéndola de cerca Linear Regression con su 86%.

Así que la precisión para Linear Regression + SMOTE es de **81.75%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.23s**.

10.3.4 MLP

MLP es una clase de red neuronal, que consiste en 3 capas mínimas de neuronas: la capa del input, *hidden layers* que pueden ser más de 1, y la capa del output. Cada neurona usa una función de activación [31]. Esta función se encarga de calcular a través de una suma ponderada de la entrada, añadiendo un *bias* y decide si la neurona debería activarse o no [32].

Los hiper parámetros que se le puede pasar los podemos encontrar todos listados en la documentación de MLP Classifier de sklearn.

Los parámetros con los que vamos a experimentar y nos parecen relevantes son los siguientes:

Parámetro	Descripción	Valores a combinar
activation	Función de activación.	identity, logistic, tanh, relu
solver	Algoritmo para la optimización de los pesos.	lbfgs, sgd, adam
hidden_layer_sizes	El formato de *hidden layers*, representado en tuplas en el cada número son las neuronas de esa capa, y cada número representa una capa.	(16, 3), (16, 16), (16, 16, 16), (16, 16, 16, 16)
alpha	Penalización L2 (término de regularización)	0.0001, 0.001, 0.01, 0.1, 1
max_iter	Número máximo de iteraciones.	100, 200, 400

Fig 33 Hiperparámetros de MLP, su descripción y los valores con los que experimentamos. Fuente: Creación propia

Los mejores parámetros para MLP obtenidos:

Parámetro	Mejor valor
activation	**relu**
alpha	**0.01**
hidden_layer_sizes	**(16,16,16)**
max_iter	**200**
solver	**lbfgs**

Fig 34 Mejores hiperparámetros para MLP. Fuente: Creación propia

El **tiempo de entreno** con estos mejores parámetros es de **3651s.**

En cuanto a las predicciones, se han ejecutado varias ejecuciones y obtenemos lo siguiente:

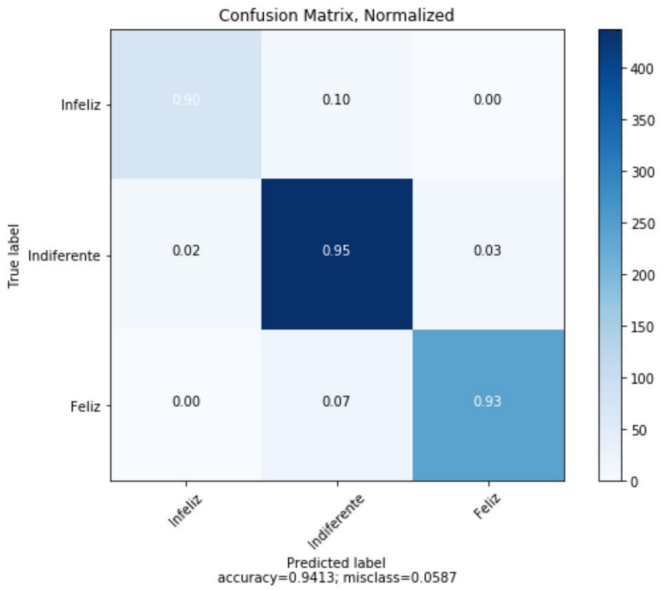


Fig 35 Confusion matrix de MLP. Fuente: Creación propia

Y observamos que en las ejecuciones rondan por los valores mostrados en la figura X. Donde podemos ver que la precisión ronda el 94% y específicamente la precisión en las clases “Infeliz”, “Indiferente” y “Feliz” son del **90%, 95%, 93%,** respectivamente.

Así que la precisión para MLP es de **94.13%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **0.87s**.

MLP + SMOTE

Ejecutamos lo mismo pero aplicando SMOTE y los mejores parámetros son:

Parámetro	Mejor valor
activation	**relu**
alpha	**0.01**
hidden_layer_sizes	**(16,16,16)**
max_iter	**400**
solver	**lbfgs**

Fig 36 Mejores hiperparámetros para MLP + SMOTE. Fuente: Creación propia

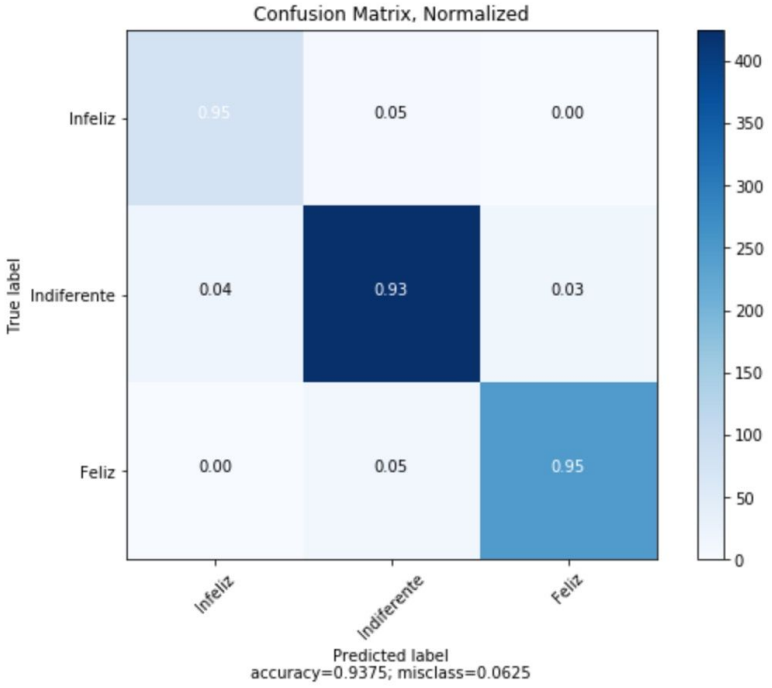


Fig 37 Confusion matrix para MLP + SMOTE. Fuente: Creación propia

La única diferencia en los parámetros es el valor de *max_iter*. El resto parece mantenerse siempre en esos valores, que son los mismos que en MLP sin SMOTE.

Por tanto el **tiempo de entreno** con estos mejores parámetros es de **3648s.**

En cuanto a las predicciones, diferentes ejecuciones rondan por los valores mostrados en la figura 37. Donde podemos ver que la precisión ronda el 94% y específicamente la precisión en las clases “Infeliz”, “Indiferente” y “Feliz” son del **95%, 93%, 95%**, respectivamente. Que en son precisiones muy altas en todas las clases por el coste de 2% en la clase Indiferente.

Así que la precisión para MLP + SMOTE es de **93.75%**. En cuanto al **tiempo de predicción**, en la mayoría de ejecuciones ha tardado **2.11s**.

10.4 Análisis de los modelos

Finalmente, tenemos resultados para cada modelo. A continuación, recopilamos en este apartado todos los resultados para tener una panorámica, compararemos los modelos y finalmente discutiremos cuál método es el mejor, al menos para estos datos teóricos.

10.4.1 Resultados de todos los modelos

Los datos generados para estos resultados son los siguientes:

	Infeliz	**Indiferente**	**Feliz**

Ejemplos	406	2280	1314

Fig 38 La distribución de los datos teóricos en sus clases. Fuente: Creación propia

Los números de la figura 38 son de los datos de la última ejecución, que corresponden a los datos usados para obtener y mostrar durante todo este proyecto.

A continuación, recopilado en la figura 39 el método, su precisión, el % de precisión para cada clase en la predicción, el tiempo de entrenamiento y el tiempo de predicción. En negrita se marcan los mejores valores en cada columna:

Método	**Acc**	**% clase**	**Training Time (s)**	**Predict time (s)**
Decision Tree	88.75	77, 92, 87	**1.7**	**0.02**
Decision Tree SMOTE	89.62	85, 90, 90	**1.7**	0.07
Random Forest	94.00	85, 96, 93	80	0.41
Random Forest SMOTE	**94.63**	**95, 94, 95**	80	0.43
Linear Regression	86.36	73, 90 ,85	730	**0.2**
Linear Regression SMOTE	81.75	84, 75, 90	730	0.23
MLP	94.13	90, 95, 93	3651	1.87
MLP SMOTE	93.75	95, 93, 95	3648	2.11

Fig 39 Recopilación de toda la información obtenida. Fuente: Creación propia

En general, en diferentes ejecuciones, se ha observado que la precisión varía un poco a las de la tabla, pero sin cambios drásticos, cosa que tiene sentido ya que el Cross Validation de todos los métodos han dado una precisión similar a las precisiones de las predicciones con los datos de testeo.

La precisión de un modelo sin aplicar SMOTE respecto al mismo modelo pero aplicando SMOTE suele mantenerse en diferentes ejecuciones, por lo que la mejora/empeoramiento relativo es algo constante. Por ejemplo, Linear Regression sin SMOTE generalmente tiene mejor precisión que aplicando SMOTE.

10.4.2 Comparando precisión

Después de varias ejecuciones, no hemos conseguido que ningún modelo supere el 95%, pero Random Forest y MLP no están muy lejos de ello.

Hablando de precisión, el peor suele ser Linear Regression, sobretodo aplicando SMOTE. Sale bastante peor que Linear Regression sin aplicar SMOTE. Hablamos del orden de 78% a 85%.

Mientras que los demás modelos suelen tener una precisión mayor de 85%, con porcentajes en torno al 90%.

Los mejores en precisión suelen ser Random Forest y MLP. Casi rozando el 95%, con valores alrededor del 92-93%. Sin mucha diferencia entre los dos modelos.

Por tanto, en cuanto a precisión, la mejor opción **es Random Forest o MLP**. Obviamente con los parámetros encontrados durante el proyecto.

10.4.3 Comparando tiempos

Distinguimos tiempo de entreno y predicción porque a lo largo del tiempo, los datos reales irán creciendo en cantidad con nuevos datos, por tanto, estos nuevos datos pueden aportar una mejora en la precisión de los modelos. Ya que cuantos más datos tenga la máquina para entrenar, más probabilidades de que las predicciones sean mejores. Pero para ello, se ha de volver a entrenar con los nuevos datos.

Aunque el entreno no habría que hacerlo siempre, ya que sería demasiado tener que entrenar cada vez que se añade una información nueva a los datos, podemos permitir que el tiempo de entreno sea “algo mayor”.

Pero en cuanto al tiempo de predicción no es el caso. Se quiere realizar una predicción cada vez que haya un nuevo empleado o se actualice un empleado con una nueva encuesta. Por tanto, es importante que la predicción sea bastante rápida.

Mirando la tabla, se puede ver que el modelo más rápido en ambos tiempos es Decision Tree, y que el más lento en ambos tiempos es MLP.

Pero si nos fijamos en Random Forest y Linear Regression pasa algo interesante:

Random Forest entrena y predice en 80s y 0.41s respectivamente mientras que Linear Regression, aunque tarda mucho más (730s, casi 10 veces más) en entrenar, es bastante más rápido prediciendo (0.2s, el doble de rápido).

Si intentamos decidir cuál método es mejor según el tiempo de predicción, vemos que Decision Tree, Random Forest y Linear Regression tardan menos de 1s mientras que MLP

es el único que tarda más, llegando a tardar hasta 2s. Los tiempos de los tres primeros métodos mencionados son bastante similares por lo que el tiempo de predicción no llega a ser un factor decisivo.

Por lo que nos queda, comparar por tiempo de entrenamiento. El tiempo de entreno vemos que va aumentando vertiginosamente en cada método (en el orden en que los hemos introducido). Llegando a tardar hasta una hora aproximadamente en MLP. Pero no podemos tomar una decisión basándonos en el tiempo de entrenamiento solamente ya que, como se ha explicado anteriormente, lo ideal sería entrenar únicamente cuando sea necesario: cuando se obtiene una cantidad considerable de nuevos datos.

Por tanto, pensando en cumplir con el requisito no funcional de ser rápidos (1.3.3 Requisitos funcionales y no funcionales), al menos en la predicción, deberíamos considerar los 3 primeros métodos.

En conclusión, **en cuanto a tiempos**, sería lógico pensar en los 3 primeros métodos, como los mejores: **Decision Tree, Random Forest, Linear Regression**.

Comentar finalmente que la aplicación de SMOTE, casi no afecta en el tiempo, tarda ligeramente más, pero es negligible respecto a diferentes métodos. Por tanto, lo importante no es fijarse en la diferencia de tiempos entre los modelos con y sin SMOTE, sino solamente entre métodos distintos.

11. Conclusiones

En esta sección se discutirán cuál ha resultado ser el mejor modelo, cómo se debería aplicar/adaptar este TFG al proyecto Kokoro y finalmente se describirán brevemente el seguimiento que ha habido de la planificación detallada en el apartado 4.4 Diagrama de Gantt.

11.1 Mejor modelo

El mejor modelo tiene que tener buena precisión y que sea rápido. Así que es necesario que tenga no sólo buena precisión, sino un tiempo de predicción lo menor posible. Por tanto, no podemos decidir cuál es mejor mirando solo por el tiempo o por la precisión, sino por ambos.

Como la precisión es algo crucial para este proyecto, podemos descartar de primeras Linear Regression. Hasta podríamos descartar Decision Tree ya que MLP y Random Forest tienen mucha mejor precisión.

Por tanto, hay que decidir entre MLP y Random Forest.

Ya que en precisión no hay realmente diferencia clara entre MLP y Random Forest, podemos decidir solo mirando sus tiempos:

Claramente se observa que los tiempos con Random Forest son mucho mejor que MLP.

Así que ahora hay que decidir entre Random Forest sin SMOTE o con SMOTE.

Para esta decisión tenemos que tener en cuenta todo, no solo la precisión general y los tiempos, sino la precisión de cada clase:

Método	Precisión	% clase	Training time (s)	Predict time (s)
Random Forest	94.00	85, 96, 93	80	0.41
Random Forest SMOTE	94.63	95, 94, 95	80	0.43

Fig 40 Comparación de los resultados de Random Forest aplicando y no aplicando SMOTE. Fuente: Creación propia

Podemos observar, que la precisión general usando SMOTE, aumenta ligeramente.

Pero no solo aumentaría la precisión general, sino que la precisión de la clase minoritaria aumentaría drásticamente, de 85% a 95%.

Ya que nuestra intención es prioritariamente detectar lo mejor posible cada clase, vamos a concluir que el mejor método para este proyecto es **Random Forest aplicando SMOTE**.

Quiero aclarar que en el caso de que en los datos reales el número de los ejemplos de las clases está bastante equilibrado, no haría falta aplicar SMOTE y por tanto, ahorraría tiempo en entreno y predicción.

11.2 Aplicación del TFG al proyecto Kokoro

Obviamente, este análisis es factible con los datos generados en específicamente para este proyecto. Una vez se recopilen una cantidad alta de datos, ya que cuantos más datos mejor entrenará la máquina, se ha de hacer las siguientes adaptaciones.

11.2.1 Contexto

Para este apartado hay que aclarar primero cómo se conseguirán los datos y cómo y dónde se guardarán.

Slashmobility es una empresa en la que trabajan mucho con base de datos. Por lo que esa parte está controlada.

El cómo guardarlo ya lo hemos especificado en el apartado de 10.1 Modelo de Datos.

El método de obtención de los datos, según la memoria del proyecto de Kokoro, se desarrollará un chatbot en el que cada cierto tiempo mandará el formulario detallado en el apartado 10.1 Modelo de Datos, de manera que una vez rellenado el formulario, se añadirá al empleado su formulario en la base de datos. Recordar que cada vez que se quiera entrenar de nuevo con los nuevos datos, se ha de añadir la pregunta final sobre su felicidad en la empresa.

11.2.2 Substitución del dataset

Para la sustitución del dataset, recoger de la base de datos todos los empleados con los datos en forma del modelo de datos definido en 10.1 Modelo de Datos (datos crudos).

Una vez se obtengan los datos crudos, se puede aplicar casi el mismo código de preprocesamiento, pero con unos pequeños cambios:

- Ya no es necesario calcular un indicador para clasificar la persona en una clase (Feliz, Indiferente, Infeliz). Por lo que el código relacionado con *score* se puede eliminar
- En vez de clasificar según el indicador, simplemente asignar la clase según el dato correspondiente a la pregunta sobre la felicidad en la empresa.
- Como en el dataset actual, cada persona solo son arrays de sus formularios, en los datos reales seguido de los formularios debería estar la respuesta de la pregunta sobre su felicidad. Es decir, en los datos crudos de una persona debería contener N arrays que representan los cuestionarios con sus respuestas y la respuesta sobre su felicidad. Por este punto, se ha de modificar dentro del bucle que procesa una

persona y añadir que si no es una array, es la respuesta a la felicidad y asignarla a la variable target ('happiness').

11.2.3 Ejecución

Una vez sustituido el dataset, no debería haber ningún problema en ejecutar todo el código de los diferentes modelos.

Habría que volver a analizar según los resultados con los datos reales. Para ello, se puede seguir la misma lógica que hemos aplicado en este trabajo, comparando en precisión y tiempo, para determinar qué método es mejor. Una vez determinada, usar ese método con los parámetros encontrados, para predecir la felicidad de los empleados y usar este output como input para la máquina que predice el comportamiento del empleado.

Hay que tener en cuenta, que una vez haya un aumento no trivial en el número de datos, para tener mejor precisión, habría que entrenar de nuevo los datos. No hace falta volver a ejecutar todo el script de los diferentes métodos, aunque al principio, sería recomendable volver a ejecutar todo el script para asegurar que el mejor método encontrado sigue siéndolo.

En notebook de “Models Scheme”, he separado el entrenamiento y predicción en bloques diferentes ya que el tiempo de entreno es mucho más alto que el de predicción en todos los métodos. Esto es debido a que una vez entrenada la máquina, si se le añaden pocos datos nuevos, el cambio no será significativo en la predicción, por tanto, no sería sabio entrenar cada vez que se añade un formulario nuevo a un usuario. Sino entrenar la máquina cada vez que haya un aumento de datos significativo, y por lo tanto ahorrar en tiempo.

11.2.4 Opinión personal

Decidí hacer este proyecto básicamente por dos grandes motivos.

El primero y el principal motivo, la idea de si es posible llegar a predecir la felicidad o autorrealización de un empleado en una empresa en base de cuestionarios, y con qué precisión podría llegarse a predecir.

El proyecto no ha ido de la manera que deseaba, debido a no disponer de datos reales. El hecho de tener que generarlos yo misma puede restar fiabilidad a la predicción. De todas formas, he hecho todo lo posible para que los datos teóricos sean lo más realistas posibles, consultando a expertos de RRHH. Sin restarle importancia a la dificultad de generar estos datos correctamente, me gustaría tener la oportunidad de realizar este trabajo con datos reales.

La segunda motivación fue que cuando me propusieron el proyecto, me recordó al trabajo que se tenía que hacer para la asignatura de “Aprenentatge Automàtic (APA)” de la FiB. En esta asignatura cometí muchos errores básicos que me restaron nota de la asignatura. Pensé que este proyecto era una segunda oportunidad para redimirme de mis errores cometidos. En este aspecto, estoy bastante satisfecha con el resultado.

Dejando de lado el inconveniente de no tener datos reales, el resto del desarrollo del proyecto, y sin contar la pandemia, mi TFG ha sido bastante fluido, en cuanto a reuniones, código y feedback.

En conclusión, aunque no haya podido trabajar con datos reales, la generación de los datos teóricos, preprocesar estos y analizar los resultados de ML ha sido bastante satisfactorio ya que he mejorado y corregido los conocimientos de ML obtenidos durante la carrera.

12. Referencias

[1] Empresa - Slashmobility | Soluciones mobile. [20/02/2020] - https://slashmobility.com/empresa/
[2] Kokoro - The heartbeat of your business. [20/02/2020] - https://www.kokorostudio.es/
[3] Memoria de Kokoro [20/02/2020]
[4] Joline Nicotina. *What is employee experience?* - People-Doc. [22/02/2020]. https://www.people-doc.com/blog/what-is-employee-experience-7-hr-pros-give-their-definition
[5] Russell, S.J; Norvig, P, Prentice Hall. ISBN: 9781292153964. *Artificial intelligence: a modern approach* [21/02/2020] - http://cataleg.upc.edu/record=b1371770~S1*cat
[6] Poole, David. *Computacional Intelligence: A Logical Approach*. [22/02/2020] - https://www.cs.ubc.ca/~poole/ci/ch1.pdf
[7] Alpaydin, E, The MIT Press, 2014. ISBN: 9780262028189. *Introduction to machine learning.* [22/02/2020] - http://cataleg.upc.edu/record=b1468626~S1*cat
[8] Russell, Stuart; Norvig, Peter. 2009. *Inteligencia Artificial: Un Enfoque moderno* (3a edición) [21/02/2020]
[9] *What is the difference between feature extraction and feature selection?* - Quantdare [21/02/2020] - https://quantdare.com/what-is-the-difference-between-feature-extraction-and-feature-selection/
[10] *Estado de ánimo* - Euroresidentes [24/02/2020] - https://www.euroresidentes.com/diccionario-psicologia/estado-de-animo.html
[11] *Macrodatos e inteligancia artificial, alternativas a big data* - Fundeu [24/02/2020] - https://www.fundeu.es/recomendacion/macrodatosalternativa-abig-data-1582/
[12] Centro para el Desarrollo Tecnológico Industrial. [21/02/2020] - https://www.cdti.es/
[13] Khan, I.A., Brinkman, W. & Hierons, R. *Towards estimating computer users' mood from interaction behaviour with keyboard and mouse*. Front. Comput. Sci. 7, 943–954 (2013). [22/02/2020] https://doi.org/10.1007/s11704-013-2331-z
[14] Thomas Sy, Susanna Tram, Linda A.O'Hara. *Relation of employee and manager emotional intelligence to job satisfaction and performance*. [22/02/2020] https://doi.org/10.1016/j.jvb.2005.10.003
[16] *¿Qué es un tablero Kanban? Definición y detalles* - Kanbanize [26/02/2020] - https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-tablero-kanban/
[21] *Encuesta sobre la sostenibilidad* - Proporcionada vía Atenea GEP [6/03/2020] - goo.gl/kWLMLE
[22] *Mòdul 2.6 - El informe de sostenibilidad 2018.pdf* - Material de Atenea [6/03/2020] - https://atenea.upc.edu/mod/folder/view.php?id=2164343
[23] Nest City Lab [14/03/2020] - https://www.apocapocbcn.com/en
[24] The Jupyter Notebook - IPython [30/08/2020] - https://ipython.org/notebook.html
[25] Baena Graciá, Verónica. 2011. *Fundamentos de marketing: entorno, consumidor, estrategia e investigación comercial.* [30/08/2020]
[26] *Decision Tree Classifiers Explained* - Programmerbackpack [01/08/2020] - https://programmerbackpack.com/decision-tree-explained/
[27] Imagen Decision Tree - Decision Tree Learning - Wikipedia [01/09/2020] - https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:Decision_Tree.jpg

[28] *Random Forests Definition* - DeepAI [02/09/2020] - https://deepai.org/machine-learning-glossary-and-terms/random-forest
[29] Imagen Random forest diagram complete - Wikipedia [02/09/2020] - https://en.wikipedia.org/wiki/Random_forest#/media/File:Random_forest_diagram_complete.png
[30] *What is a Likert Scale and How Do You Pronounce Likert?* - core.ecu.edu [02/09/2020] - http://core.ecu.edu/psyc/wuenschk/StatHelp/Likert.htm
[31] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. 2009 *The Elements of Satistical Learning: Data Mining, Inference and Prediction*. [02/09/2020]
[32] *Understanding Activation Functions in Neural Networks* - Medium [02/09/2020] - https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0
[33] Imagen model_selection - ethen8181.github.io [05/09/2020] - http://ethen8181.github.io/machine-learning/model_selection/model_selection.html
[34] Mòdul 2.6 - *El informe de sostenibilidad 2018.pdf* [6/03/2020], Material de Atenea - https://atenea.upc.edu/mod/folder/view.php?id=2164343