ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

ALGORITMOS AVANZADOS DE MACHINE LEARNING EN LA ESTIMACIÓN DE TIEMPOS DE RESOLUCIÓN DE PROCESOS FINANCIEROS

Autor: Natalia Gila Briñas

Director: Gabriel Pierobon

Madrid

Junio 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

ALGORITMOS AVANZADOS DE MACHINE LEARNING EN LA ESTIMACIÓN DE TIEMPOS DE RESOLUCIÓN DE PROCESOS FINANCIEROS

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/21 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Natalia Gila Briñas Fecha: …21…/ …06…/ …2021…

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Gabriel Pierobon

Fdo.: Gabriel Pierobon Fecha: …14…/ …06…/ …2021…

Vº Bº del Coordinador de Proyectos

Fdo.: Carlos Morrás Ruiz-Falcó Fecha: ……/ ……/ ……

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D._____ Natalia Gila Briñas_____

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: ALGORITMOS AVANZADOS DE MACHINE LEARNING EN LA ESTIMACIÓN DE TIEMPOS DE RESOLUCIÓN DE PROCESOS FINANCIEROS, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

- El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ……14….. de …….JUNIO…….... de …2021…….

ACEPTA

Fdo……Natalia Gila Briñas……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

ALGORITMOS AVANZADOS DE MACHINE LEARNING EN LA ESTIMACIÓN DE TIEMPOS DE RESOLUCIÓN DE PROCESOS FINANCIEROS

Autor: Natalia Gila Briñas

Director: Gabriel Pierobon

Madrid

Junio 2021

ALGORITMOS AVANZADOS DE MACHINE LEARNING EN LA ESTIMACIÓN DE TIEMPOS DE RESOLUCIÓN DE PROCESOS FINANCIEROS

Autor: Natalia Gila Briñas
Director: Gabriel Pierobon
Entidad Colaboradora: KPMG España

RESUMEN DEL PROYECTO

Palabras clave: Machine Learning, Modelos, Big Data

1. Introducción

En esta sección del proyecto me he enfocado en la planificación del mismo, los objetivos y las motivaciones. Asimismo, se ha hecho hincapié en la metodología empleada para el desarrollo del trabajo. Al comienzo de este apartado, podemos encontrar una breve puesta en contexto que nos pondrá en la situación del cliente y se entenderá el porqué de la realización de este proyecto.

2. Herramientas

En esta parte del proyecto, he explicados las diferentes herramientas que he empleado, las cuales han servido de base para el desarrollo del trabajo. Además, toda la información expuesta en este apartado, me ha servido para entender el funcionamiento de cada una de las herramientas y así poder hacer uso de ellas de manera eficiente.

3. Fuente de datos

Este apartado lo considero uno de los más importantes ya que abarca la explicación de todas y cada una de las variables. Durante el proyecto se han ido creando nuevas variables, cuya explicación se recoge en esta sección. Por otro lado, tanto la depuración de los datos (etapa de gran de importancia en un proyecto de Machine Learning), como la variable objetivo han sido definidas en este apartado.

Figura 1. Imagen ilustrativa de la obtención de los datos

4. **Modelado**

   En esta sección del trabajo, comento todos los modelos empleados para el entrenamiento de los datos y posterior predicción de la variable objetivo. Del mismo modo, se detallan las principales ventajas y desventajas que posee cada uno de los modelos.

5. **Evaluación y resultados**

   En la evaluación, se muestran los resultados de todos los modelos que se han explicado en el apartado anterior y se hace una comparativa de la métrica elegida con el objetivo de mostrar en la sección de los resultados, aquellos que han sido considerados como mejor predicción.

   En resultados, únicamente se comentan los valores obtenidos del mejor modelo, dejando a un lado el resto de modelos que no han sido capaces de adaptarse de manera óptima a los datos.

6. **Conclusiones y trabajo futuro**

   En conclusiones se recoge una breve reflexión sobre el proyecto y los resultados obtenidos junto con lo esperado.

   Finalmente, en trabajo futuro se propone una idea detallada sobre una posible continuación y futura implementación de un proyecto que obtenga resultados más existosos.

REAL-TIME PRICE PREDICTION MODEL FOR THE RESIDENTIAL ELECTRICITY MARKET

Author: Natalia Gila Briñas
Supervisor: Gabriel Pierobon
Collaborating Entity: KPMG Spain

ABSTRACT

Keywords: Machine Learning, Models, Big Data

1. Introduction

In this section of the project I have focused on project planning, objectives and motivations. Likewise, emphasis has been placed on the methodology used for the development of the work. At the beginning of this section, we can find a brief contextualization that will put us in the client's situation and we will understand why this project was carried out.

2. Tools

In this part of the project, I have explained the different tools that I have used, which have served as a basis for the development of the work. In addition, all the information presented in this section has helped me to understand the operation of each of the tools and thus be able to use them efficiently.

3. Data source

I consider this section to be one of the most important, since it covers the explanation of each and every one of the variables. During the project, new variables have been created, the explanation of which is included in this section. On the other hand, both the data cleaning (a very important step in a Machine Learning project) and the target variable have been defined in this section.

Figure 1. Illustrative image of data collection.

4. Modelling

In this section of the paper, I discuss all the models used for data training and subsequent prediction of the target variable. Likewise, the main advantages and disadvantages of each of the models are detailed.

5. Evaluation and results

In the evaluation, the results of all the models that have been explained in the previous section are shown and a comparison of the chosen metric is made in order to show in the results section, those that have been considered as the best prediction.

In the results, only the values obtained from the best model are commented, leaving aside the rest of the models that have not been able to adapt optimally to the data.

6. Conclusions and future work

Conclusions include a brief reflection on the project and the results obtained along with the expected results.

Finally, in future work, a detailed idea about a possible continuation and future implementation of a project that obtains more successful results is proposed.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

1.1 CONTEXTO Y DEFINICIÓN

El objetivo del proyecto fue predecir el tiempo en minutos hasta la resolución de una operación vinculada con la gestión de la financiación para compra de automóviles.

Dado que una operación está conformada por una serie de tareas, las cuales tienen dependencias no solo del analista sino también del concesionario (por ejemplo tener que enviar nueva documentación), hemos considerado que la unidad con mejores chances de tener una buena predicción es la tarea.

En términos de experiencia de usuario creemos que enseñar el tiempo estimado de resolución de la tarea más inmediata cumpliría con el objetivo de negocio de esta conceptualización, el de proveer al usuario con una indicación de cómo va avanzando su gestión en todo momento. Por otro lado, intentar predecir el tiempo estimado para la operación completa supondría incrementar los niveles de desvío.

Siendo así, para poder estimar los minutos que toma la tarea de principio a fin, se subdivide la duración total en tres etapas diferentes que conformarán el tiempo total de demora de la misma:

- Tiempo que toma desde que se abre la tarea hasta que se asigna la tarea a un analista (técnicamente referido como lag)
- Tiempo desde que el analista acepta realizar la tarea hasta que la finaliza (técnicamente referido como work)
- Tiempo estimado que se tomar el analista el descanso durante la realización de la tarea (técnicamente referido como break)

Hemos considerado esta descomposición pensando en entrenar tres modelos separados, ya que las variables que influyen en el tiempo de demora de asignación (I) no son

necesariamente las mismas y no influyen en la misma medida que en el tiempo de resolución de la tarea (II) ni mucho menos que del tiempo de descanso (III). En este sentido, buscamos que al entrenar tres modelos por separado eso nos permita encontrar particularidades y patrones de cada uno de ellos, que en la suma final resulten en una predicción más precisa que la que se obtendría entrenando un único modelo (lag+work+break).

Finalmente, el documento incluye tanto una explicación sobre los datos utilizados para la generación de los modelos como una explicación sobre los modelos generados y seleccionados. También se explica cuál ha sido el criterio de validación de los modelos.

1.2 OBJETIVOS

En referencia al objetivo del proyecto, lo que se está intentando lograr, es la satisfacción del usuario. Esto se obtendrá si se consigue una buena predicción del tiempo en desarrollar una tarea, que se adecúe a la realidad, de esta forma, el usuario estará recibiendo una información acertada que le ayudará en los procesos con nuestro cliente.

Otros objetivos a destacar son el gran manejo, control y conocimiento de las herramientas empleadas. Este objetivo junto con el siguiente, hacen referencia a lo personal. Uno de los objetivos perseguidos en este proyecto, es poder adquirir un gran manejo y control del lenguaje de Python así como sus estructuras de datos. Por otro lado, he buscado poder tener conocimiento de las librerías más utilizadas y potentes del ámbito profesional para el cual me estoy formando.

Por último y más importante, lo que principalmente me gustaría lograr, es llevar a la práctica los conocimientos adquiridos en mis estudios del máster realizado, para de este modo, ver una aplicación real y que resulte útil en el día a día del mundo laboral; pues mi corta experiencia laboral, no me ha permitido ver las aplicaciones de mis estudios en más allá de lo académico.

1.2.1 MOTIVACIÓN

Como bien se ha comentado en la introducción, este proyecto busca predecir el tiempo en minutos hasta la resolución de una operación vinculada con la gestión de la financiación para la compra de automóviles, es decir, se busca satisfacer al usuario, ofreciéndole información del tiempo que va a tardar en resolverse su trámite.

Son frecuentes las consultas por parte del usuario sobre la duración de la resolución del trámite (lo que se ha considerado como tarea), es por ello, que nuestro cliente, considera interesante y de gran utilidad, disponer de un sistema que pueda ofrecer al usuario una estimación de este tiempo sin necesidad de consultar, simplemente que lo tenga a su disposición y pueda acceder a ello cuando le sea necesario y pueda resultarle útil.

Por todo lo descrito, me encuentro realizando el proyecto para satisfacer las necesidades de un cliente, lo cual ha sido la motivación principal.

1.3 ENFOQUE Y MÉTODO SEGUIDO

En este proyecto, se ha tratado de dar un enfoque y metodología muy común que se ha seguido hasta el final, esta metodología para poner orden en los proyectos, es conocida como CRISP-DM.

A continuación, se muestra la Figura 2 que muestra los pasos que sigue CRISP-DM y servirá de apoyo para explicar cada etapa de esta metodología seguida.

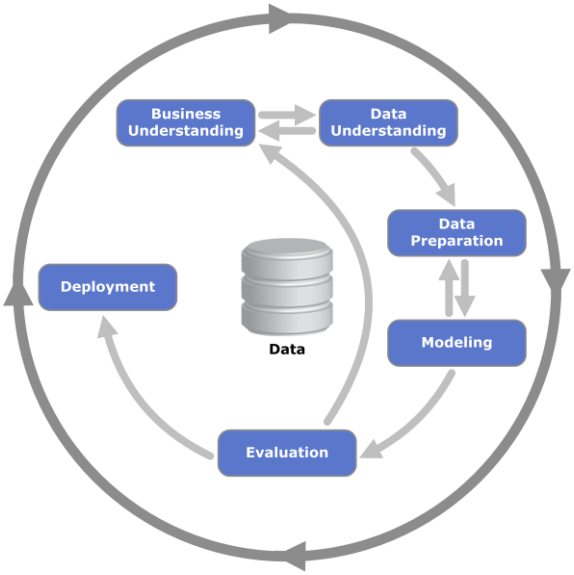


Figura 2. Metodología CRISP-DM

En la Figura 2, se muestra el orden de las fases, pero este orden no es rígido, pudiendo volver a las fases anteriores. Si nos fijamos en las flechas internas, simbolizan el transcurso de las fases a medida que finalizamos cada una.

El círculo externo, nos indica que este tipo de proyectos de análisis de datos, son proyectos de naturaleza cíclica. Esto quiere decir que el proyecto no finaliza con la fase de Deployment. La información descubierta durante el proceso de realización del proyecto y la solución desplegada pueden producir nuevas iteraciones del modelo. Los procesos de análisis subsecuentes se beneficiarán de las experiencias previas (1).

Un proyecto de minería de datos consta de seis fases:

- **Fase I. Business Understanding.**

  Esta fase inicial se enfoca en la comprensión de los objetivos de proyecto. Después se convierte este conocimiento de los datos en la definición de un problema de minería de datos y en un plan preliminar diseñado para alcanzar los objetivos (1).

- **Fase II. Data Understanding.**

La fase de entendimiento de datos comienza con la colección de datos inicial y continúa con las actividades que permiten familiarizarse con los datos, identificar los problemas de calidad, descubrir conocimiento preliminar sobre los datos, y/o descubrir subconjuntos interesantes para formar hipótesis en cuanto a la información oculta (1).

- **Fase III. Data Preparation.**

  La fase de preparación de datos cubre todas las actividades necesarias para construir el conjunto final de datos (los datos que se utilizarán en las herramientas de modelado) a partir de los datos en bruto iniciales. Las tareas incluyen la selección de tablas, registros y atributos, así como la transformación y la limpieza de datos para las herramientas que modelan (1).

- **Fase IV. Modeling.**

  En esta fase, se seleccionan y aplican las técnicas de modelado que sean pertinentes al problema (cuantas más mejor), y se calibran sus parámetros a valores óptimos. Típicamente hay varias técnicas para el mismo tipo de problema de minería de datos. Algunas técnicas tienen requerimientos específicos sobre la forma de los datos (1).

- **Fase V. Evaluation.**

  En esta etapa en el proyecto, se han construido uno o varios modelos que parecen alcanzar calidad suficiente desde la una perspectiva de análisis de datos. Antes de proceder al despliegue final del modelo, es importante evaluarlo a fondo y revisar los pasos ejecutados para crearlo, comparar el modelo obtenido con los objetivos de negocio.. Al final de esta fase, se debería obtener una decisión sobre la aplicación de los resultados del proceso de análisis de datos (1).

- **Fase VI. Deployment.**

Generalmente, la creación del modelo no es el final del proyecto. Incluso si el objetivo del modelo es de aumentar el conocimiento de los datos, el conocimiento obtenido tendrá que presentarse para que el cliente pueda usarlo (1).

1.4 PLANIFICACIÓN

En cuanto a la planificación, se realizó un cronograma al comienzo del proyecto que se muestra a continuación en la Tabla 1:

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
	8-14 feb	15-21 feb	22-28 feb	1-7 mar	8-14 mar	15-21 mar	22-28 mar	29-4 abr	5-11 abr	12-18 abr
Project Planning										
Generación del entorno de trabajo										
Comprobación de datos										
Exploración de datos										
Limpieza de datos										
Variable objetivo y problema a solucionar										
Métrica de evaluación - medida del éxito										
Preprocesamiento de datos										
Generación de variables										
Separación de set en test/train										
Modelo baseline										
Otros modelos sencillos										
Modelos más avanzados										
Estudio de resultados										
Decisión del modelo final										
Documentación/Realización de memoria final										

Tabla 1.Planificación semanal

Como se puede ver, el proyecto lo he desarrollado en 10 semanas. Se pasa a comentar lo que se ha realizado en cada una de las fases indicadas en la Tabla 1.

- Project Planning. El proyecto comenzó tras una reunión en la que se me puso en conocimiento del problema a resolver y del tiempo del que disponía. Tras esta reunión, comenzó una etapa de comprensión del problema y del caso ante el que me enfrentaba. Una vez entendido el caso y proporcionados los datos, se realizó el planning que muestra la Tabla 1. Esta fase de Project Planning tuvo una duración de una semana, ya que tuve que realizar una estimación de lo que podría durar cada fase y contar con que problemas podrían surgirme.

En esta semana, se habló sobre qué es lo que queríamos como objetivo, que tipo de modelos se querían implementar y desde un primer momento se hizo hincapié que se querían ver resultados de algoritmos más avanzados como XGBoost o LightGBM.

- Generación del entorno de trabajo. Esta tarea se realizó en una semana a la vez que la siguiente. Esta sección se explicará en profundidad en el capítulo 2. Se hizo elección de un lenguaje de programación y se realizó una investigación de las librerías con las cuales se podrían realizar los entrenamientos de los algoritmos y modelos elegidos.
- Comprobación de los datos. Se hizo una primera "exploración" que fue más bien una comprobación, ya que lo que nos interesa en esta semana es corroborar que la información enviada era correcta y no íbamos a necesitar más del cliente.
- Exploración de los datos. Pasadas las primeras tres etapas clave para el proyecto, comencé con la exploración. En esta fase tocó entender los datos que se nos proporcionaron, qué variables tengo, qué significan estas variables, qué información nos proporcionan, en definitiva, un estudio descriptivo. Esta fase se realizó a la vez que la limpieza y duró un poco más de lo previsto, ya que son una de las fases más importantes para obtener unos u otros resultados en el entrenamiento de los modelos.
- Limpieza de datos. En esta etapa, se tuvo que tomar una serie de decisiones y se encontraron problemas que se explicarán más adelante. Esta fase es crucial para que el modelado sea lo más acertado posible. Se limpiaron valores nulos, se crearon variables, se eliminaron filas de tablas con información redundante, se juntaron tablas con técnicas de join, se identificaron el formato de las variables para comprobar si sería necesaria una recodificación de estas.
- Variable objetivo y problema a solucionar. El previo análisis del problema, nos ofrecía una visión sobre qué es lo que queríamos predecir, en este caso los tiempos de realización de tareas bancarias, es por ello, que se debía analizar como variable objetivo aquella que coincidía con la predicción del tiempo.
- Métrica de evaluación – medida del éxito. En este punto, se realizó una investigación sobre las diferentes medidas que podían medir el error en base al problema que

teníamos que solucionar y se debía elegir una de estas mediciones para las posteriores etapas de modelado.

- Pre-procesamiento de datos. En esta fase, se incluye una preparación adecuada de los datos para su fase de modelado. En esta etapa debemos generar datos de calidad, los que nos llevarán a resultados de calidad. Si es cierto que este paso podría englobar los anteriores como la limpieza y exploración, pero se ha realizado una separación para dar importancia a la calidad que se quiere conseguir en este proceso de pre-procesado.
- Generación de variables. Como parte del pre-procesado, está la generación de variables que se forman a partir de otras ya existen que se consideran de importancia pero se desconoce la importancia que tendrán para el modelo entrenado. Se ha separado del pre-procesamiento ya que es una tarea que llevará más tiempo que otras, y al hacer la planificación de los tiempos, se ha querido considerar como una tarea independiente del pre-procesado, cosa que no es del todo cierta pero nos ofrece una visión del tiempo de realización de las tareas mucho más acertada.
- Separación del set de train y test. Llegados a este punto hemos acabado la parte de limpieza de datos y se comienza la parte del modelado. Tenemos todos los datos que se van a utilizar y debemos hacer una separación de estos en dos dataset distintos, uno de entrenamiento, que nos servirá como bien dice el nombre, de entrenamiento para nuestro modelo, y otro dataset de validación o test, que será el que utilizará el modelo únicamente para validar resultados y comprobar el sobreentrenamiento o generalización de nuestro modelo.
- Modelo baseline. Un modelo baseline, es un modelo más "sencillo" que aquellos que queremos conseguir entrenar que son más complejos. Se establece un baseline y se obtiene un resultado, si ese resultado se mejora con nuestro modelos más complejos, entonces nuestra práctica es correcta.
- Otros modelos sencillos. No solo he querido entrenar un modelo baseline, sino que he querido tener una batería de ellos para corroborar que con modelos sencillos es imposible conseguir resultados que se logran con modelos más complejos y potentes.

- Modelos más avanzados. Como bien indica el título del proyecto, se quieren comprobar resultados de algoritmos avanzados de machine learning, este ha sido el objetivo del trabajo de fin de máster y del proyecto para la empresa, centrar nuestras miras en este tipo de algoritmos que nos ofrecerán (o no) resultados distintos a los de los modelos baseline.
- Estudio de resultados. A esta fase junto con la siguiente se le dedicó una semana. Una semana donde se estudiaron los valores obtenidos por los modelos y se comparaban y entendían mediante gráficas eso resultaos.
- Decisión del modelo final. Por último, se tuvo que decidir un modelo que evaluase de forma más adecuada nuestros datos y estimase de una manera correcta y con menos error los tiempos de ejecución de tareas financieras. Esta decisión será explicada de manera más extensa junto con el estudio de resultados en el capítulo 5.
- Documentación/Realización de la memoria. La última etapa del proyecto, es la realización de una documentación para aquellas personas que deban realizar cambios en el proyecto para futuros trabajos y en mi caso también la realización de esta memoria. Si bien es cierto que se le ha asignado una semana, este tiempo es orientativo, ya que la duración ha sido desde que se finaliza la decisión del modelo final hasta la entrega del proyecto.

1.5 GENERACIÓN DE TAREAS

Las tareas que se han estudiado están relacionadas con el proceso de revisión de los documentos necesarios para aprobar la financiación de compra de automóviles.

Las tareas se inician cuando se recibe un correo electrónico por parte del concesionario, que debería contener documentación correspondiente a la operación, la cual resulta fundamental para el éxito o fracaso de la financiación.

El correo electrónico se añade a una cola de trabajo, gestionada mediante la herramienta Appian, la cual asigna automáticamente la tarea a un analista cuando se detecta que ha quedado libre.

Posteriormente, cuando algún analista queda libre y le es asignada una tarea, comienza la etapa de revisión de la documentación. En este punto pueden darse dos casuísticas:

1. La documentación es correcta y completa, por lo que se termina la fase de revisión documental, o
2. que el correo electrónico no contenga los documentos necesarios, lo que inicia un flujo de correos entre el analista y el concesionario.

Cuando la documentación no es correcta pueden llegar a generarse entonces diferentes tareas para gestionar una misma operación.

1.5.1 FUNCIONAMIENTO DE APPIAN

Appian ayuda a las organizaciones a crear aplicaciones y workflows rápidamente, con una plataforma de automatización de low-code. Al combinar personas, tecnologías y datos en un único workflow, Appian puede ayudar a las empresas a maximizar sus recursos y mejorar los resultados empresariales (2).

Appian, lo que hace con las tareas de nuestro cliente, es colocarlas en una única cola, esperando a ser asignadas a un analista. Aparentemente, este método puede parecer el menos acertado, se puede pensar que las tareas, se encontrarán más tiempo en espera al ser una cola que si se formasen varias una detrás de cada tarea que se está ejecutando por un gestor (2).

El cliente del banco, acude a un concesionario donde pide los papeles, pero el cliente no es quien manda éstos al banco, sino el concesionario mediante un correo que se pone en la cola de la aplicación Appian del banco. En esta aplicación, se distribuye el trabajo, cuando uno de los gestores del banco se queda libre, le asignan un correo nuevo para revisar. Si ven que tienen algún problema o se quieren ir a tomar un descanso, siguen con el expediente abierto hasta que lo devuelvan o lo cierren.

Capítulo 2. HERRAMIENTAS

2.1 LENGUAJE

El lenguaje empleado para el desarrollo de este proyecto, ha sido Python. Se barajó la idea de emplear otro lenguaje potente de programación, R, pero finalmente se optó por Python debido a lo que se cuenta a continuación.

Es un lenguaje de programación versátil multiplataforma y multiparadigma. Es importante recalcar, que Python cuenta con una licencia de código abierto. Gracias a esto, es un lenguaje utilizado por gran cantidad de gente. Esta razón ha sido clave para la elección del lenguaje para el proyecto, ya que cuanta más gente ha indagado con el lenguaje, más dudas y aclaraciones hay resueltas por la red, a priori puede parecer una razón con poco peso, pero cuando te encuentras haciendo un proyecto como este, te das cuenta de la importancia de que exista una gran comunidad conocedora del lenguaje Python.

Además, Python tiene dos características principales que le destacan por encima de otros lenguajes informáticos (4).

En primer lugar, su corta curva de aprendizaje, puesto que es un lenguaje sencillo y elegante que contempla un conjunto de normas muy intuitivas.

En segundo lugar, es un lenguaje práctico que optimiza el tiempo del programador, ya que con pocas líneas de código se pueden programar algoritmos complejos. Por tanto, como se ha comentado anteriormente, el programador dispone de librerías ya implementadas de las cuales aprovecha funcionalidades ahorrando tiempo de desarrollo (4).

En los últimos 30 años, Python ha evolucionado y ha ganado el apoyo de la comunidad, lo que lo hace especialmente importante en el desarrollo de aplicaciones del lado del servidor. Esto, combinado con su simplicidad, le ha permitido posicionarse en el espacio de los grandes datos, especialmente en el desarrollo de algoritmos de aprendizaje automatizado (4).

Dado que Python tiene una fuerte presencia en la educación, siendo utilizado como lenguaje de modelo en escuelas, centros educativos y universidades. Como resultado, muchas herramientas que han surgido en este campo han sido desarrolladas en este lenguaje y están siendo utilizadas por ingenieros y científicos de datos. Algunos ejemplos pueden son PySpark (big data), o Pandas, NumPy, Matplotlib entre otros o Jupyter (ciencia de datos) (4).

2.2 LIBRERÍAS

En esta sección, se muestra una pequeña explicación de las librerías empleadas en el desarrollo del proyecto. Al comienzo del proyecto, se ha utilizado la información que muestro a continuación, para obtener unas nociones básicas de cada librería y para obtener conocimiento para su posterior uso.

Se han elegido librerías potentes y que son utilizadas para desarrollar proyectos de Machine Learning.

2.2.1 PANDAS

Seguidamente, se tratará sobre Pandas. Pandas es un paquete de código abierto de Python que se utiliza sobre todo para tareas de ciencia de datos/análisis de datos y aprendizaje automático. Está construido sobre otro paquete llamado Numpy, que proporciona soporte para matrices multidimensionales. Como uno de los paquetes más populares para el manejo de datos, Pandas funciona bien con muchos otros módulos de ciencia de datos dentro del ecosistema de Python, y normalmente se incluye en todas las distribuciones de Python, desde las que vienen con su sistema operativo hasta las distribuciones de proveedores comerciales como ActivePython de ActiveState. El nombre de Pandas se deriva del término “Panel Data” y es la librería de los análisis de datos que utiliza los desarrolladores informáticos principalmente en Python (5).

Pandas nace debido a la necesidad de una herramienta de análisis de datos que sea flexible y de alto rendimiento. En los comienzos, Python se utilizaba para la manipulación y

preparación de datos, pero no para el lenguaje automático, y es por ello que aparece la librería Pandas y solventa el problema. Para el procesamiento y análisis de datos, realizamos 5 pasos: carga, preparación, manipulación, modelado y análisis, los cuales se consiguen con Pandas como se muestra en la Figura 3 (5).

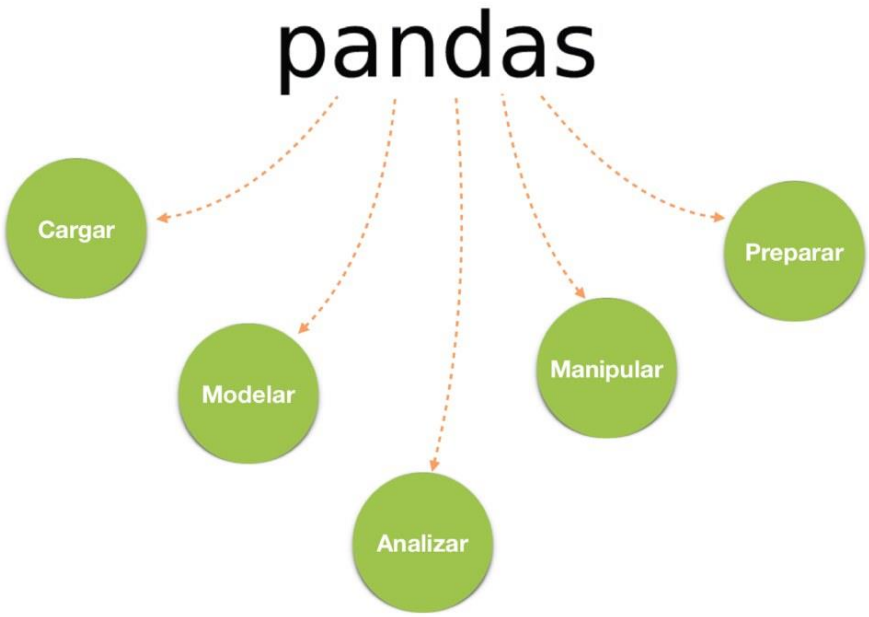


Figura 3. Pasos para el análisis con pandas

Entre sus estructuras más conocidas se encuentra DataFrame. El DataFrame de Pandas es una estructura de datos tabular bidimensional, potencialmente heterogénea, con ejes etiquetados (filas y columnas). Un DataFrame es una estructura de datos bidimensional, es decir, los datos se alinean de forma tabular en filas y columnas. El DataFrame de Pandas consta de tres componentes principales, los datos, las filas y las columnas, estos componentes son vitales para manipular datos y la implementación de modelos predictivos (5).

Los tres tipos de estructura de datos que dispone Panda son: Series, DataFrame y Panel.

En primer lugar, las series son matrices de una dimensión que puede almacenar todo tipo de datos. Además, puede contener datos homogéneos y al mismo tiempo datos de tamaño inmutable y valores de datos mutables. No obstante, lo importante e innovador es con

NumPy que tiene un índice para las columnas, que son establecidas personalmente por los desarrolladores de Pandas y que luego pueden ser manipuladas por el programador (6).

Seguidamente, el dataframe son tablas de bidimensionales definida por columnas, formadas por variables heterogéneas e índices que puede ser personalizados por el programador para poder adaptarlo a su proyecto (6).

Finalmente, el panel es una estructura de tres dimensiones, también conocido como cubos. Es necesario importarlo a través del alias pd, que se utiliza para esa librería. Por ejemplo, en el caso de np, se emplea la librería NumPy (6).

2.2.2 NUMPY

Numpy es una librería de Python enfocada principalmente al análisis de datos y al cálculo numérico, especialmente cuando se trata de un gran volumen de datos (7).

Una característica que diferencia a NumPy del resto de librerías es que tiene una nueva clase de objetos que permiten representar funciones muy eficientes para su manipulación y alguna colecciones de datos en varios dimensiones (7).

NumPy, es un paquete vital para desarrolladores que se dedican al Machine Learning debido a que proporciona una estructura de datos de matriz los cuales tiene ciertos beneficios sobre las listas regulares que facilita Python. Los beneficios más destacados son: es muy rápido en la accesibilidad para leer y escribir artículos, es el más conveniente y eficiente, también es él más compacto en comparación a otros paquetes como Pandas, Matplotlib o Scikit-Learn (8).

Para poder implementar NumPy en Python, es necesario un intérprete de código de bytes no optimizado, también conocido como CPython. Por ello, NumPy sirve para una misma versión de Python facilitar la ejecución de algoritmos matemáticos de manera más rápida que los equivalentes compilados, ya que Numpy aporta matrices multidimensionales, operadores y funciones que operan de manera óptima en mtraice. Para ello, es necesario

reescribir algo del código, sobre todo bucles internos mientras estés usando NumPy con los algoritmos matemáticos (9).

Por un lado, MATLAB contiene diversas cajas de herramientas adicionales, como por ejemplo Simulink, no obstante NumPy, está completamente integrado con Python, como bien ya se ha mencionado, un lenguaje de programación más compacto y moderno. Por otro lado, dentro de Python existen paquetes complementarios como es SciPy, una biblioteca que se asemeja a la funcionalidad de MATLABA y Matplotlib, un paquete de trazado que permite al desarrollador una funcionalidad de trazado parecida a MATLAB. Finalmente, tanto NumPy como MATLAB se basan en los mis cálculos de álgebra lineal eficientes, como son BLAS y LAPACK (9).

Asimismo, cabe destacar los bindings de Python que se utilizan en la biblioteca de visión por computadora, OpenCV el cual emplea matrices de NumPy con la intención de operar y almacenar datos. Debido a que al ser imágenes con múltiples canales son formas muy eficientes de conceder a píxeles específicos de una imagen. Del mismo modo, la matriz NumPy se puede utilizar como estructura de datos universal para: imágenes en OpenCV, también para núcleos de filtrado, puntos de características extraídos y sobre todo para simplificar el enorme flujo de trabajo de depuración y programación (7).

A continuación, se muestra una imagen con una matriz bidimensional ya que tiene filas y columnas, concretamente tiene cuatro filas y tres columnas. Por tanto, se convierte en una matriz de dos dimensiones. No obstante, si solo tuviera una hilera, se estaría hablando de una matriz unidimensional. (7).

2.2.3 DATETIME

Datetime es una librería que se emplea para trabajar con fechas en Python, puesto que incorpora datos de date, time y datetime con el fin de representa funciones y fechas para operar con ellas. Algunas de las funciones más habituales que permiten son (10):

Acceso a diferentes componentes de una fecha, ya sea mes, día, hora, año, minutos, incluso llegando a los segundos y microsegundos.

Crear cadenas con formato de fecha en los tipos times, datetime o date

Comparar diferente fechas para comprobar si es necesario hacer un ajuste en el tipo de formato de fecha o en los tipos date, time o datetime.

2.2.4 WARNINGS

Los mensajes warnings o también conocidos como mensajes de advertencia, aparecen cuando es necesario avisar al usuario de que es posible que alguna condición en un programa no esté justificada y termine el programa. Un ejemplo puede ser cuando se emiten el warning de que un programa está utilizando un módulo obsoleto, en ese caso saltaría un mensaje de advertencia alertando del dicho problema (11).

Por un lado, los desarrolladores de Python suelen transmitir estos warnings meidate la función warn() definida en el módulo. Por otro lado, los programadores de C emplean PyErr_WarnEX () (11).

Además, en la mayoría de los casos los mensajes warning se escriben en svs.stderr. No obstante, existe warning filter, una secuencia de acciones y reglas que controlan todos los mensajes de advertencia. Asismismo, es posible incluir nuevas reglas al filtro llamando filterwanings() y así se reestablece el estado por defecto llamándolo a resetwarnings() (11).

Finalmente, la impresión de los mensajes warning se realizan a través de showwarning(), el cual se puede anular, y su implementación por defecto de dicha función consigue dar formato al mensaje gracias a formatwarning() (11).

2.2.5 MATPLOTLIB

Python cuenta con numerosas librerías para presentar los datos. Es importante comprender como optimizar los resultados de un proyecto mediante la visualización, puesto que dicha visualización facilitará la presentación del análisis, resultados o ideas que desees presentar. Por ello, es fundamental mencionar la librería Matplotlib (12).

Seguidamente, existe un módulo en el paquete de matplotlib, que aparece cuando se importa la librería matplolib pyplot (12).

Por último, existe un módulo que se instala al mismo tiempo que matplotlib que permite hacer cálculos de manera interactiva y solo requiere las funciones de trazado. Este módulo se llama pylba y es recomendable utilizarlo cuando se emplean matrices. Es un módulo que se utilizaba antiguamente (12).

2.2.6 SEABORN

Seaborn es otra librería que existe en Python que se utiliza para generar gráficos de manera fácil y elegante. Esta librería, está principalmente basada en matplotlib y dispone de una interfaz de gran nivel, la cual es muy intuitiva, debido a su gran popularidad. Del mismo modo, se encuentra instalada directamente en la distribución de Anaconda (13).

Igualmente, en algunas ocasiones es necesario las librerías de Matplotlib cuando se utiliza Seaborn, ya que la funcionalidad la ofrece Matplotlib. También, para crear estructuras de datos basadas en dichas librerías será necesario Pandas, Numpy e incluso la librería de warning con el objetivo omitir algunos mensajes de advertencia que genera Seabron (especialmente en futuros cambios de funcionalidad) (14).

Seaborn ofrece diferentes gráficas para la visualización que tienen los datos almacenados de un DataFrame, Por ello, ¿Qué gráfico se recomienda utilizar en cada caso?. Los gráficos más relevantes son: Barplots, histogramas, KDE plots, Box plots, Violin plot, Catplot y por último Scatter plots (15).

En primer lugar, los **barplots** se utilizan para visualizaciones de datos agregados, por ejemplo la media (15).

```
sns.barplot(x='Food', y='Age', hue='Gender', data=df, estimator=np.median)
```

En segundo lugar, los **histogramas** son herramientas básicas que sirven para analizar la distribución de los datos. También, esta gráfica está influenciada por el número bins que selecciona el desarrollador y el ancho de cada valor (15).

```
sns.distplot(df['Age'])
```

KDE plots, su acrónimo viene de Kernel Density Estimation plots. Estas gráficas representan mejor la distribución de los datos que un histograma, debido a que suaviza la forma de lo valores. No obstante, no proporciona información estadística (15).

```
sns.kdeplot(data=df['Age'], shade=True)
```

A continuación, los **box plots** se utilizan para conocer rangos de datos, sobre todo si existen outliers o el rango intercuantil donde se distribuyen los datos. También, sirve para identificar la media fácilmente (15).

```
sns.boxplot(data=df, x='Food', y='Age')
```

En sexto lugar, los **violín plots** proviene de la unión de KDE + box plots, por tanto sirven para comparar diferentes distribuciones de muestras o poblaciones. Un violín representa en su conjunto la media, el rango intercuartil, el intervalo de confianza en el que se reparten los mapas (normalmente 95%) y la distribución (15).

```
sns.violinplot(data=df, x='Gender', y='Age')
```

En séptimo lugar, el **catplot** se utiliza cuando el desarrollador representa variables categóricas, es decir, es una alternativa muy útil para el diagrama de barras (15).

```
sns.catplot(x='Day', y='Age', hue='Gender', kind="swarm", data=df)
```

En último lugar, están los **scatter plots,** muy útiles para representar cómo una variable afecta a otra y las correlaciones (15).

2.2.7 RANDOM

Random es un módulo que utiliza generadores de números pseudoaleatorios para diferentes distribuciones (16).

En primer lugar, hay que definir si estamos en una secuencia o en un número entero. Por un lado, para las distribuciones enteras, hay una selección uniforme en una franja seleccionada.

Además, para los números reales se utilizan funciones basadas en distribuciones uniformes, log-normales, exponenciales negativas, beta, normales y beta, con el objetivo de generar distribuciones angulares y también está accesible en la distribución de von Mises (16).

La mayoría de las funciones del módulo son dependientes de la función básica random(), que proporciona de manera uniforme un número flotante aleatorio en la franja semiabierta [0.0, 1.0).

Además, la implementación subyacente en C es segura y veloz para subprocesos. Mersenne Twister es conocido como uno de los generadores de números aleatorios más utilizados que existen. No obstante, al ser totalmente determinado, no conviene para cualquier propósito y es muy inadecuado sobre todo para fines criptográficos, por tanto es fundamental tener en cuenta la finalidad del proyecto (16).

La clase Random puede utilizar un generador básico diferente para su propio diseño, en caso de que el desarrollador quiera convertir la clase en una subclase. Por ello, esta subclase de Random invalidará los métodos random(), gestate(), setstate() y seed(). De manera opcional, es posible sustituir un nuevo generador por el método getrandbits(), esto consigue a randrange() proporcionar selecciones sobre un rango arbitrariamente amplio (17).

Finalmente, el módulo random produce la clase SystemRandom, que utiliza la función del sistema os.urandom() con el objetivo de proporcionar números aleatorios mediante fuentes obtenidas por el sistema operativo (17).

Personalmente, en este proyecto, he utilizado la siguiente función principal de la librería random:

- random.seed(): Inicializa el generador de números aleatorios.

```
random.seed(1998)
```

2.2.8 MATH

Math es un módulo que da aceso a las funciones matemáticas definidas en el estándar empleado por C (18).

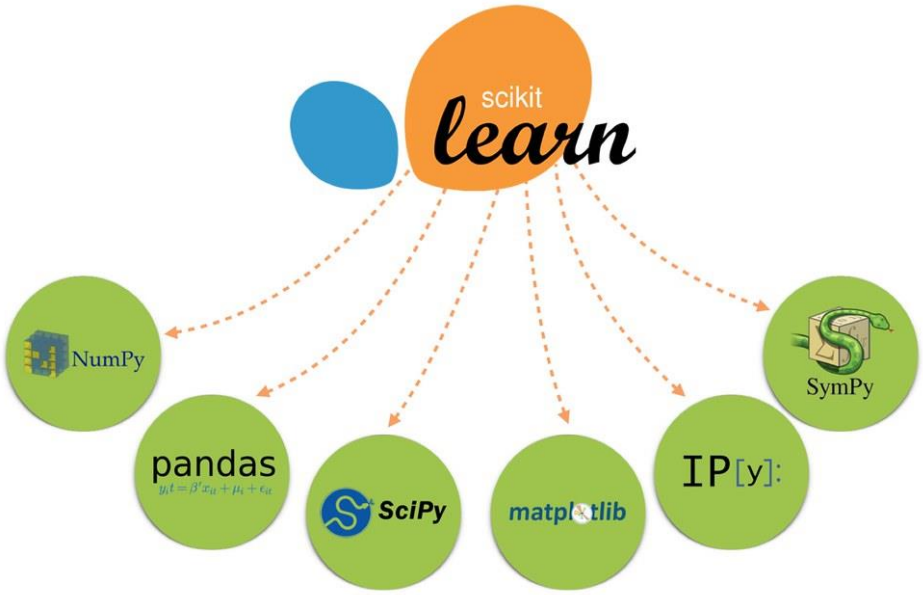
Del mismo modo, cabe destacar que estas funciones matemáticas no pueden contener números complejos, en cuyo caso es recomendable utilizar las funciones con el mismo nombre del módulo cmath. La principal diferencia entre las funciones que permiten utilizar números complejos y las que no, provocan que la gran mayoría de programadores no quieran formarse en el campo de las matemáticas

Para concluir, este módulo permite crear diferentes funciones utilizadas en C, aunque si los valores son flotantes se indican explícitamente para no cometer errores posteriormente (18).

2.2.9 SCIKIT-LEARN

El scikit-learn es conocida como la librería más interesante para Machine Learning en Python, debido a que es un código abierto y se puede reutilizar en varios contextos, favoreciendo así su uso comercial y académico. Asimismo, scikit-learn presenta una amplia selección de algoritmos de aprendizaje tanto supervisado como no supervisado en Python (19).

Scikit-learn está fabricada sobre SciPy, la cual proviene de Scientific Python. Además, esta funcionalidad incluye otras librerías o paquetes que muestra la siguiente imagen aquí abajo:



Los principales componentes de scikit-learn son que existen muchas funciones que se van a explicar más adelante como son: algoritmos de aprendizaje, validación cruzada, algoritmos de aprendizaje no supervisados (19).

En primer lugar, los algoritmos de aprendizaje supervisados se utilizan en la mayoría de los casos en Machine Learning desde los modelos lineales generalizados, como una regresión lineal, o hasta máquinas de vectores de soporte, también conocidos como (SVM). Asimismo, sirven para árboles de decisión, incluso para métodos bayesianos, todos los mencionados en este párrafo forman parte de las herramientas de scikit-learn y sirven especialmente para algoritmos supervisados (19).

Asimismo, los algoritmos de aprendizaje no supervisados cuentan (como los algoritmos supervisados) con una variedad algoritmos disponibles para su aprendizaje. Empezando por el análisis factorial, continuando con agrupación, análisis de componentes principales y finalizando con las redes neuronales no supervisadas, que son muy intuitivas (19).

Finalmente, la comunidad una de las principales ventajas al ser un código abierto. Gracias a ello, scikit-learn atrae a nuevos usuarios de alrededor del mundo, debido a que existen hasta 35 colaboradores quitando los errores y perfeccionándola con actualizaciones (19).

2.2.10 SCIPY

SciPy es una biblioteca libre con la ventaja de estar en código abierto para Python. Esta biblioteca contiene algoritmos matemáticos y una gran variedad de herramientas. SciPy, fue creada por Travis Oliphant en 1999, con el nombre de Multipack, porque en esa época así se llamaban los paquetes de netlib, los cuales reúnen a QUADPACJ, MINPACK y ODEPACK (20).

Para finalizar con SciPy, cabe destacar que es una biblioteca que se fundamenta en el objeto de matriz de NumPy, ya mencionado anteriormente. Por ello, incluye herramientas como Matplotlib, SymOy y Pandas, sin contar el conjunto de bibliotecas de computación científica que contiene NumPy. Este conglomerado de herramientas se dirige al mismo tipo de usuarios, como las aplicaciones GNU, MATLAB, Scilab y Octave. Asimismo, es frecuente hacer referenica a este conjunto de herramientas cuando se está trabajando en bibliotecas como SciPy (20).

2.2.11 XGBOOST

XGBoost es una librería de gradient boosting optimizada y distribuida, que pone en práctica algoritmos de aprendizaje automático bajo el marco del Gradiente Boosting XGBoost, obteniendo un boosting de árbol paralelo, también conocido como GBM, GBDT y consigue resolver problemas de la ciencia de los datos de manera rápida (21).

2.2.12 LIGHTGBM

LightGBM es un gradient boosting framework que utiliza un algoritmo de aprendizaje basado en árboles. El LightGBM proviene del árbol verticalmente mientras que el otro algoritmo crece los árboles horizontalmente, lo que significa que el LightGBM crece el árbol por hojas mientras que el otro algoritmo crece por niveles. Elegirá la hoja con la máxima

pérdida delta para crecer. Cuando crece la misma hoja, el algoritmo Leaf-wise puede reducir más pérdidas que el algoritmo level-wise (22).

Capítulo 3. FUENTE DE DATOS

Los datos utilizados para el desarrollo del proyecto han sido proporcionados por el cliente. Aunque inicialmente se recibieron 12 tablas, se han utilizado 3 de ellas por ser las más relevantes tras realizar un estudio de todas ellas.

La razón por la que no se han empleado las restantes, se debe a que la tabla de métricas se encuentra construida a partir de muchas de las otras tablas, por lo que la información resultaría redundante y no nos aporta información nueva a nuestro modelo.

En algunas ocasiones la tabla no aportaba información valiosa como sucede con los datos relativos a información de los usuarios o información sobre los sla.

3.1 TABLAS EMPLEADAS

Se han utilizado como he comentado, 3 tablas con información no repetida, estas tablas son las siguientes:

- **urdm_process_task_metric**: es la tabla más importante ya que contiene las métricas principales sobre las que se realizan los análisis. Es decir, información sobre cada una de las tareas así como sus características: sla, fechas, trabajador que ha gestionado la tarea, etc.
  Esta tabla la he considerado como la principal, el resto derivan de esta.
- **urdexpediente**: aporta información sobre la operación y la carga de trabajo.
- **urdhistoricosistema**: aporta información sobre la carga de trabajo y la actividad de los usuarios.

Los datos fueron proporcionados en formato csv.

En el proceso de depuración de datos se trataron los valores faltantes, se eliminaron columnas con información repetida y crearon variables nuevas en función de las necesidades

encontradas durante el proceso de entrenamiento de los modelos y al criterio de expertos consultados en KPMG, conocedores del negocio.

Se añadió a la tabla principal (urdm_process_task_metric), información de las otras dos tablas empleadas (urdexpediente y urdhistoricosistema) mediante técnicas de join y merge de tablas, creando una única y concentrando toda la información necesaria.

3.2 DATOS QUE ENRIQUECERÍAN EL ANÁLISIS

Se cree que se puede mejorar la precisión de los modelos en futuras versiones a través de la incorporación de datos de los correos electrónicos (con los que se inicia la operación) ya que el tipo de usuario que envía la solicitud, el tamaño del correo o el número de ficheros adjuntos pueden ser información útil a la hora de predecir tiempos. Se presupone que un correo electrónico de un concesionario que contenga una cierta cantidad de adjuntos tiene mayor probabilidad de resolución rápida que uno que venga sin documentación y por la cual haya que reclamar al concesionario el envío de los documentos.

Para este principio de proyecto, recibimos alrededor de 40 correos que pudimos explorar pero no utilizar en los modelos.

3.3 DEPURACIÓN DE LOS DATOS

El primer paso para estudiar la adecuación de los datos ha sido un estudio descriptivo de los mismos. Los principales aspectos a destacar han sido:

1. la presencia de datos faltantes que no estaban distribuidos de manera aleatoria
2. la repetición de la misma información en diferentes columnas.

Inicialmente había variables que solo tomaban valores cuando otra variable tomaba cierto valor (variables correlacionadas), situación que se presentaba en pocos casos. La presencia de datos faltantes con patrones se ve, por ejemplo, en variables vinculadas a la prioridad de

la tarea. Para solucionar este problema primero se ha hecho un estudio de los patrones de los valores faltantes y se han eliminados aquellos que tuvieran un alto porcentaje de los mismos.

El siguiente paso ha sido identificar el formato de las variables para recodificarlas de la manera correcta así como crear variables útiles para los modelos dividiendo las variables relativas a las fechas en distintos valores: mes, día de la semana, día del mes y hora.

Por último, de los datos obtenidos tras realizar los cambios, ajustes y depuraciones anteriores, se han eliminado las filas del fichero que contenían los últimos valores faltantes, ya que los modelos de machine learning normalmente no emplean variables con datos faltantes.

En la depuración de los datos, se observó un detalle que podría ayudar a entender mejor el proceso de modelado. En la Figura 4, se muestra el tiempo medio de tareas a lo largo del tiempo. Observo, que en las etapas que coinciden con las fiestas navideñas, como es diciembre, el tiempo de resolución de las tareas es más largo de lo habitual.

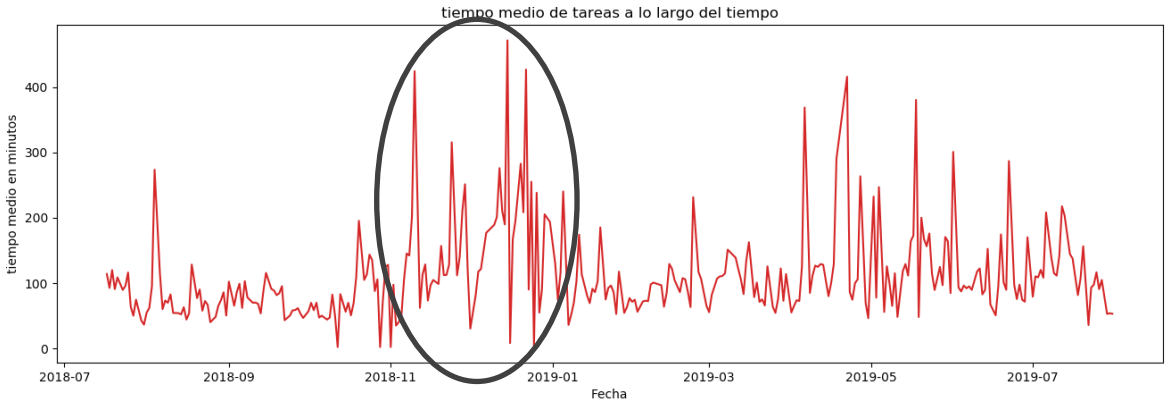


Figura 4. Gráfica de duración media

Asimismo, se realizó un estudio de la distribución de las variables numéricas como muestra la Figura 5 a continuación.

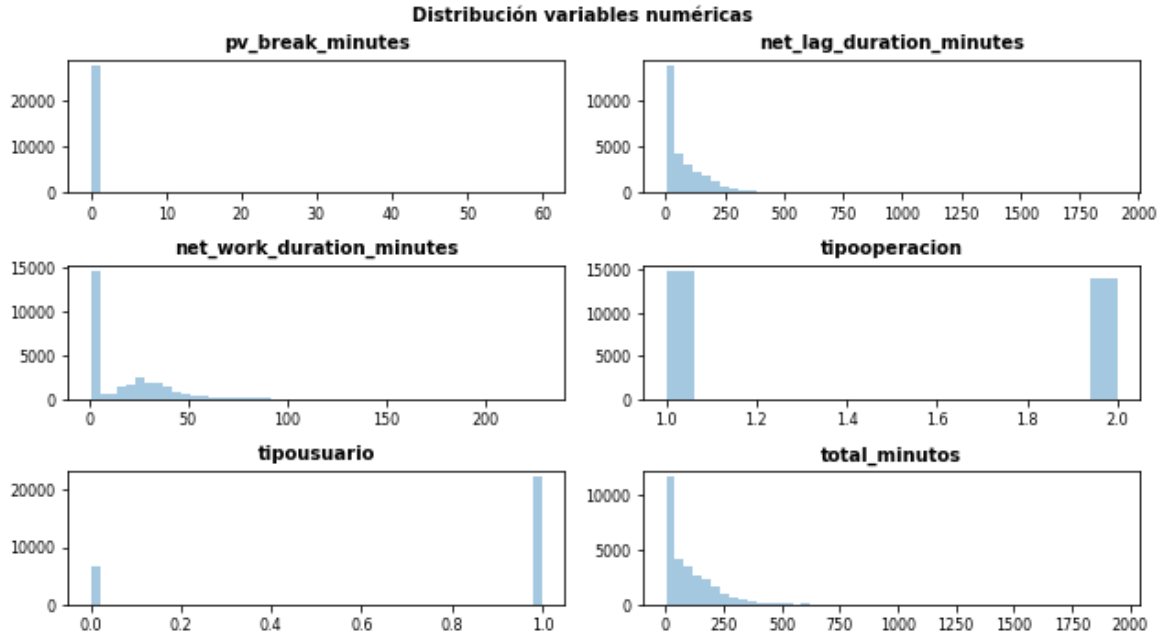


Figura 5. Distribución de las variables numéricas

Únicamente recalcar de esta Figura 5, la distribución de la variable pv_break_minutes, la cual, no tiene los datos balanceados apreciándose simplemente aquellos valores que son 0. Esto puede afectar a nuestro modelo de descansos prediciendo de manera correcta los valores 0 pero cometiendo grandes errores en aquellos que se desvían de esta cantidad.

3.4 POBLACIÓN

Con el propósito de implementar modelos adecuados que permitan extrapolar los resultados obtenidos, se ha pretendido obtener una muestra representativa de la población.

De los 48.000 registros (filas que conformaban el set de datos inicial) que se tenían inicialmente, tras el tratamiento de los datos han quedado unas 3.000 tareas que consideramos que contienen una correcta representación de la población actual. La drástica reducción se debe principalmente en la selección de los registros de interés ya que se ha filtrado por distintas variables (por fecha y pv_tipo_operacion_name) y la presencia de numerosos valores perdidos, pero esencialmente a la detección de mejoras en los procesos que fueron detectadas por criterio de expertos.

Fundamentalmente, se detectó en el set de datos una notable tendencia a la disminución de cantidad de tareas (nuevas herramientas que reducen tiempos y cantidad de casos). Esta situación fue contrastada con expertos de KPMG que han participado de la evolución del proceso de URD quienes han alertado acerca de la puesta en marcha de diversas mejoras en los procesos como la implantación de la herramienta FICRES que marcaron una tendencia a reducir cantidades y tiempos de resolución. Es así como se decidió finalmente elegir un punto de corte sobre los datos iniciales y considerar únicamente aquellos con fecha posterior a 01/04/2019. Esta fecha está indicada en Figura 6 con una línea gris vertical. Esto significó una reducción de la cantidad de datos en un 77.99%. No realizar esta eliminación de datos hubiera supuesto entrenar modelos no representativos de los niveles de productividad actuales.

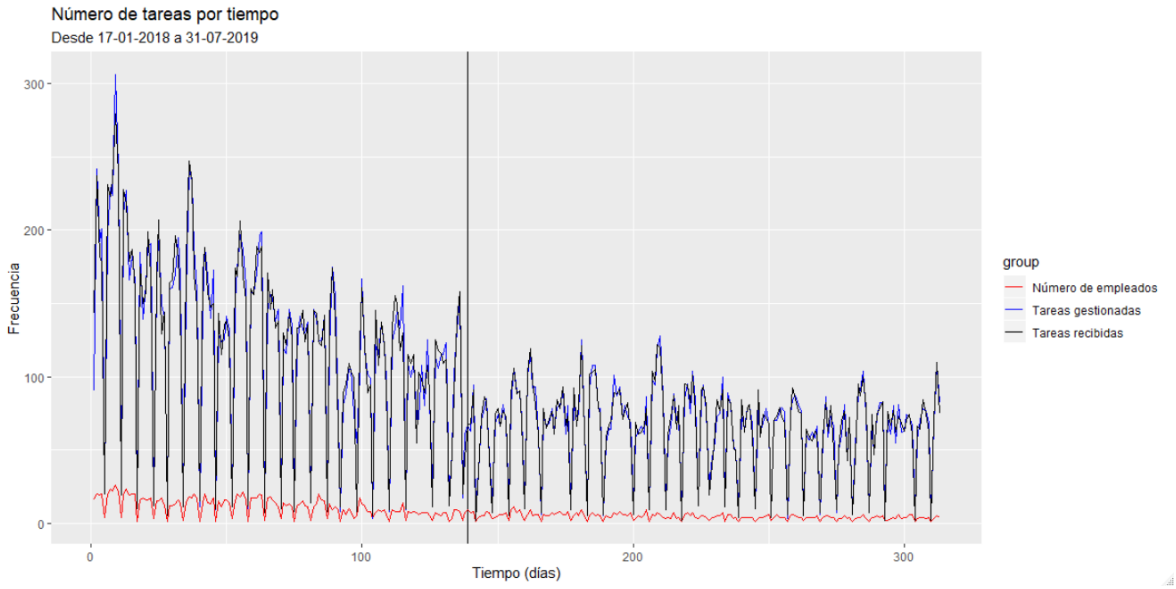


Figura 6. Número de tareas por tiempo

Tras posteriores análisis se observó la necesidad de eliminar diferentes valores extremos (outliers) que desplazaban las estimaciones notablemente. Estos valores extremos dependen de la variable que se esté prediciendo, por lo que variará en función del modelo. La reducción de datos supuso un máximo del 5.00% de eliminación de registros depurados.

Los aproximadamente 3.000 registros finales, se dividen en dos conjuntos de datos: el conjunto de entrenamiento con el 80% de los datos y el 20% restantes como conjunto de test. El conjunto de entrenamiento es el empleado para entrenar los modelos mientras que el conjunto de test es el empleado para estudiar si existe sobreajuste del modelo y su capacidad de generalización. El objetivo, como en todo modelo de machine learning es el de entrenar los modelos sobre un conjunto y hacer la evaluación sobre otro para proveer al modelo de datos sobre los que no se entrenó, y así poder validar su capacidad predictiva.

3.5 VARIABLES

Para construir este modelo las variables empleadas han sido las explicadas en la Tabla 2:

VARIABLE	DESCRIPCION
net_lag_duration_minutes	Tiempo en minutos desde que se abre la tarea hasta que se asigna.
net_work_duration_minutes	Tiempo en minutos desde que el trabajador empieza a trabajar en la tarea hasta que termina.
pv_break_minutes	Tiempo en minutos de descanso que se toma el trabajador durante la realización de la tarea.
task_owner	Trabajador/usuario que gestiona la tarea
pv_tipo_operacion_name	Tipo de cliente que solicita la financiacion. Interesan los Físicos.
user_rol	Rol que tiene el usuario que ha gestionado la tarea.
estado	Estado final en el que se encuentra la tarea.
prioridad	Indica si se ha asignado prioridad a la tarea.
sla	Tipo de sla contratado para la gestión.
activo	Toma valores 0 y 1, aunque el 0 es muy poco frecuente.
igualcerradopor	Usuario que termina la tarea. Puede diferir de task_owner.
dia_start_task	Día de la semana en que se ha comenzado a trabajar en la tarea
mes_start_task	Mes en el que se ha comenzado a trabajar en la tarea.
hora_start_task	Hora a la que se ha empezado a trabajar en la tarea
numdia_start_task	Día del mes en que se ha comenzado a trabajar en la tarea.
dia_start_process	Día de la semana en que se ha recibido la tarea
mes_start_process	Mes en el que se ha recibido la tarea.
hora_start_process	Hora a la que se ha recibido la tarea
numdia_start_process	Día del mes en que se ha recibido la tarea.
cont1	Número de tareas recibidas.

Tabla 2. Descripción de variables

3.5.1 VARIABLE OBJETIVO

El objetivo es predecir el tiempo que toma resolver una tarea desde que se recibe un correo electrónico del concesionario hasta que se aprueba el pago de la operación.

Cuando hablo de tiempo, me refiero al tiempo como variable continua compuesta por las suma de las siguientes tres variables que se muestran a continuación en la tabla 3.

Tiempo completo =
net_lag_duration_minutes
+ net_work_duration_minutes
+ pv_break_minutes

Tabla 3. Desglose de la variable tiempo

La tabla 3 anterior, muestra las componentes de la variable tiempo al completo, que está formada por:

- net_lag_duration_minutes: es el tiempo en minutos desde que se recibe el correo electrónico y se abre la tarea hasta que a un trabajador se le asigna una tarea y comienza a trabajar en ella.
- net_work_duration_minutes: es el tiempo en minutos que el trabajador tarda en resolver la tares, es decir, desde que comienza a trabajar hasta que termina y la tarea queda marcada como finalizada.
- pv_break_minutes: es el tiempo que el trabajador se toma de descanso durante la resolución de la tarea, este tiempo puede ser 0 en algunos casos.

Como podemos observar, el tiempo completo se desglosa en 3 tiempos totalmente distintos entre sí. Al comienzo del proyecto, se desconocía esta información, y debido a los malos resultados, se hizo un estudio con expertos de la empresa para ver lo que podía estar

ocurriendo. Se llegó finalmente a la siguiente conclusión. Los tiempos, no tienen mucha relación entre sí, sin embargo, al tratarlos como una suma los estamos tratando todos por igual. Se decide entonces, evaluar modelos distintos para cada uno de los tiempos, pues cada modelo se centrará en aquellas variables que mejor predigan estos tiempos de manera individual. Por lo tanto y en resumen, cada modelo tomará variables y les dará pesos en función de cada tiempo a predecir, sin tener en cuenta las otras variables tiempo. Finalmente, como evaluación, se sumarán los tiempos predichos por cada modelo y se obtendrá el valor resultado de la predicción final.

3.5.2 VARIABLES DERIVADAS

Dentro del proceso de limpieza de datos, se encuentra también el proceso de selección de variables que finalmente modelarán nuestro resultado. Durante esta selección de las variables se han creado distintas variables nuevas que han surgido a partir de los datos originales y que han sido tratadas para que se adapten mejor al modelo y consigan predecir de una manera más óptima el resultado.

Esta etapa lo que busca es la generación de nuevos cálculos mediante la interacción de variables ya existentes pero que a la vez, permitan agregar al modelo nuevos enfoques que quizás mediante las variables dadas sin tratar, no serían capaces de enfocarse en esa información que hemos sido capaces de extraer en la limpieza y transformación de dichos datos.

Las variables creadas han sido:

- Igualcerradopor: esta variable se crea con el propósito de reducir el número de variables dummys generadas en el proceso, ya que transforma una variable categórica, en una variable dummy.
  Las variables dummys, son variables binarias que indican si una variable categórica dividida en categorías, toma un valor u otro. Véase la explicación en Figura 7 siguiente:

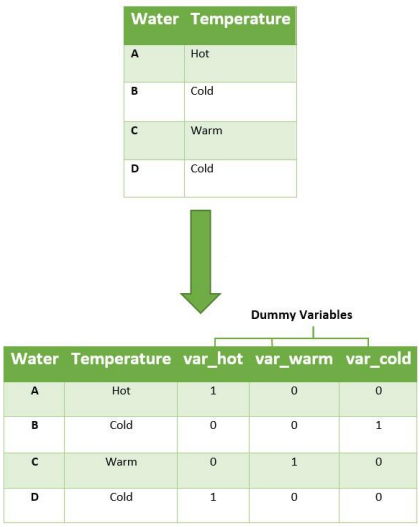


Figura 7. Variables dummies

En la Figura 7, vemos como la variable Temperature, se divide en sus categorías y simplemente contendrá un 1 en aquella variable ficticia creada que corresponda con la categoría adecuada, en otro caso, se obtendrá el valor de 0 en esa posición.

- hora_start_task: esta variable se ha extraído de la fecha de comienzo de la tarea y nos indica la hora a la que ha comenzado dicha tarea. Se ha tenido que realizar una transformación sobre esta variable, debido a que los modelos no son capaces de interpretar correctamente las fechas.
- dia_start_task: esta variable se ha extraído de la fecha de comienzo de la tarea y nos indica el día que ha comenzado dicha tarea. Del mismo modo, se ha realizado una transformación que la anterior.
- numdia_start_task: esta variable se ha extraído de la fecha de comienzo de la tarea y nos indica el número del día que ha comenzado dicha tarea. Del mismo modo, se ha realizado una transformación que la anterior.
- mes_start_task: esta variable se ha extraído de la fecha de comienzo de la tarea y nos indica el número del mes que ha comenzado dicha tarea. Del mismo modo, se ha realizado una transformación que la anterior.
- hora_start_process: esta variable se ha extraído de la fecha de comienzo del proceso y nos indica la hora en la que ha comenzado dicha proceso. Del mismo modo, se ha realizado una transformación que la anterior.

- dia_start_process: esta variable se ha extraído de la fecha de comienzo del proceso y nos indica el día que ha comenzado dicho proceso. Del mismo modo, se ha realizado una transformación que la anterior.
- numdia_start_process: esta variable se ha extraído de la fecha de comienzo del proceso y nos indica el número del día que ha comenzado dicho proceso. Del mismo modo, se ha realizado una transformación que la anterior.
- mes_start_process: esta variable se ha extraído de la fecha de comienzo del proceso y nos indica el mes que ha comenzado dicho proceso. Del mismo modo, se ha realizado una transformación que la anterior.
- cont1: es la carga de trabajo que se ha estimado como el número de tareas que se han recibido al día. Según fuentes de información, para saber cuántos trabajadores deben dedicarse a revisar la documentación de la financiación se debe estimar previamente la carga de trabajo.

3.6 ESTIMADOR DE ÉXITO

El Error Cuadrático Medio (MAE en inglés), es uno de los criterios de evaluación más usados para problemas de regresión. Se usa sobre todo cuando usamos aprendizaje automático supervisado. Vemos en la siguiente fórmula, como se calcula:

$$error\ cuadrático = (real - estimado)^2$$

El valor estimado, es aquel que predice nuestro modelo, siendo el real el valor ocultado al modelo durante su entrenamiento.

También se puede estudiar con la siguiente fórmula que nos ofrece la misma información:

$$MAE = \frac{\sum_{i=1}^{N} |x_i - \hat{x}_i|}{N}$$

La métrica elegida para medir el desempeño de los modelos, ya que es una medida absoluta de la desviación entre la predicción y el valor real. Se evaluaron otras métricas como el MAPE, pero considero que para este tipo de problemas, no son medidas adecuadas. Por

ejemplo, si una tarea se predice que se resolverá en 1 minuto, pero su valor real son 2 minutos, supondría un desvío de 100%, pero a ojos del usuario, la diferencia es prácticamente inexistente. Entonces se propone evaluar a los modelos con la diferencia absoluta media o también conocido como MAE.

Este estimador del error, estimará nuestro éxito con un valor cercano a 0, ya que indicará que nuestro modelo predijo todos los tiempos correctamente. Este valor tomaría 0 en un escenario ideal, como no es el caso, se deberá evaluar en base a resultados de otros modelos empleados, si los modelos finales entrenados, estiman de manera satisfactoria la variable resultado.

El valor considerado como éxito, es algo subjetivo, ya que no se sabe cuándo una estimación es buena y a partir de que minuto deja de ser buena y pasa a ser una estimación errónea desviada de la realidad. El criterio para determinar si el desvío en minutos entre la predicción y el valor real es bueno, deberá ser *negocio* quien determine qué niveles de desvío son aceptables, mientras tanto, se tratará de reducir este valor lo más posible.

Capítulo 4. MODELADO

En este capítulo, se explicará el funcionamiento de los modelos implementados en la realización del proyecto.

4.1 REGRESIÓN LINEAL

Un modelo de regresión lineal se basa en relaciones, de este modo, se intenta explicar una variable dependiente, que en nuestro caso será la variable salida (que dependerá de en qué proceso de la tarea nos encontremos), con un conjunto de variables independientes unas de otras, que para nuestro caso, serán el resto de variables menos la salida (24).

Hay distintos tipos de regresión lineal, por un lado tenemos la regresión simple, que trata de explicar la salida con el comportamiento de una única variable. La regresión lineal tiene una expresión que es la siguiente:

$$Y = \alpha + \beta X + \varepsilon,$$

La α es la ordenada en el origen, es decir, es el valor que toma Y cuando la X vale 0. La β es la pendiente de la recta y lo que nos va indicando es como cambia la Y a medida que cambia la X. La ε es una variable que incluye un conjunto de factores, que influye de manera mínima en la variable respuesta, esta variable la llamamos error. La variable Y es nuestra variable salida, la cual, será explicada con la variable independiente X (24).

En la Figura 8, se muestra de forma gráfica la ecuación de la recta.

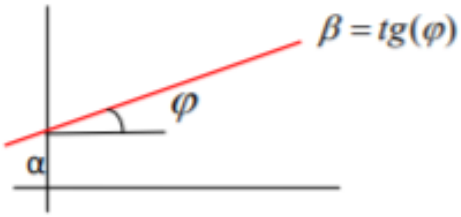


Figura 8. Gráfica de ecuación de una recta

Como se puede ver, el modelo de regresión lineal simple es un modelo tanto de comprensión como ejecución sencilla, pero en este caso, nos encontramos frente a un modelo de regresión lineal múltiple (24).

La regresión lineal múltiple trata de ajustar modelos lineales o linealizables entre una variable dependiente y más de una variables independientes o predictores. En este tipo de modelos es importante testar correlación entre las variables. Es una extensión de la regresión lineal simple (24).

Los modelos de regresión múltiple suelen emplearse para predecir el valor de la variable dependiente o para evaluar la influencia que tienen los predictores sobre ella. La regresión lineal tiene una expresión que es la siguiente:

$$Y_i = (\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_n X_{ni}) + \varepsilon_i,$$

De donde, β_0 es la ordenada en el origen, es decir, es el valor de la variable dependiente Y cuando todos los predictores son cero. β_n También conocido como coeficientes parciales de regresión, es el efecto medio que tiene el incremento de la variable predictor X_i sobre la variable salida Y. ε_i al igual que en la regresión lineal simple, es el residuo o error, es el resultado de la diferencia del valor real y el predicho por el modelo (25).

Cabe destacar, que el valor de cada coeficiente parcial de regresión, no está asociado con la importancia de la variable predictor correspondiente, sino con la unidad en la que ésta se

mide. Por lo tanto, si queremos determinar la importancia que tiene sobre el modelo, el impacto que causa sobre éste, se realiza un proceso de estandarización (sustraer la media y dividir entre la desviación estándar) y posteriormente se utilizarán los coeficientes parciales estandarizados para el ajuste del modelo (25).

Es importante tener en cuenta la colinialidad o multicolinialidad entre variables, pues las variables predictores, deben ser todas independientes entre sí. Hay colinialidad cuando una variables predictor está linealmente relacionada con una o muchas variables predictores. Esto se debe cumplir ya que en caso de existir colinialidad, no podremos distinguir de forma precisa el efecto que tiene cada variable de manera individual sobre la variable respuesta Y.

En el caso de que exista colinialidad entre variables, tendríamos dos alternativas, por un lado podría eliminar una de le las variables correlacionadas. Esto no tiene mucho impacto en la capacidad predictiva del modelo. Por otro lado, podríamos hacer una combinación de las variables correlacionadas de tal manera que quedara una sola con la información de ambas, pero esta segunda opción podría hacer que perdamos su interpretación (25).

El código utilizado para la creación y entrenamiento de éste modelo ha sido el siguiente:

```
modelo = LinearRegression()
modelo.fit(X = X_train, y = y_train)
maeLR_Lag = evaluar('LR',y_test_Lag, modelo,X_test,y_test)
```

Se crea una variable modelo donde se guarda el objeto creado de Regresión Linear. Esa variable creada, es nuestro modelo que es entrenado en la segunda línea del código anterior con la función fit(), a la cual, se le pasa el set de entrenamiento.

En la variable maeLR_Lag, se guarda el resultado de “Evaluar()”, función que se ha creado y su código es el siguiente:

```
def evaluar(nombre,dataset,model,X_test,y_test):
    predicciones = model.predict(X = X_test)
    predicciones = [0 if i < 0 else i for i in predicciones] # sustitución por 0
    mae = mean_absolute_error(
            y_true  = y_test,
            y_pred  = predicciones
```

```
        )
    print(f"El error (mae) de test es: {mae}")
    dataset[nombre]= predicciones
    return mae
```

Como podemos ver, se le pasan diferentes parámetros, de los cuales voy a destacar model, X_test e y_test, ya que los otros dos parámetros que se pasan me servirán únicamente para representaciones posteriores.

Lo que hace evaluar, es obtener las predicciones que hace nuestro modelo con el dataset de validación. Un paso adicional ha sido convertir los valores predichos negativos a 0. Esto se debe a que cuando obtenía la predicción sin hacer este cambio, observaba valores negativos del tiempo; esto no puede ser posible y es por ello que se hace este cambio.

Por último, calculamos el error MAE y lo devolvemos con return.

A continuación, en las Figuras 9, 10, 11, se muestran los distintos modelos lineales que se hicieron para las diferentes variables tiempo.

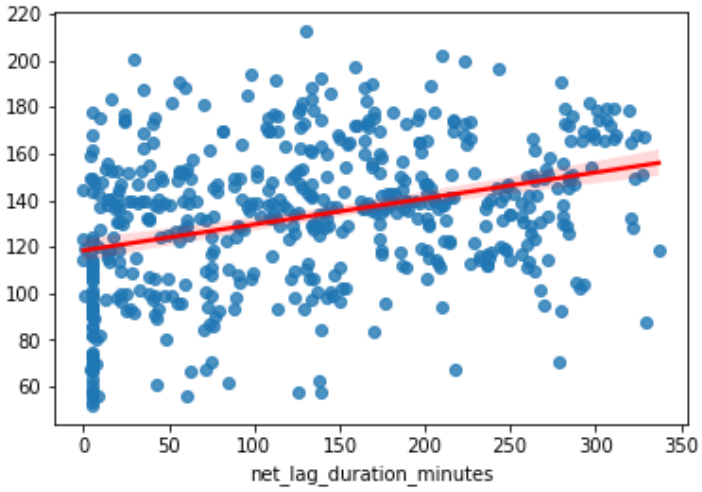


Figura 9. Gráfica del modelo lineal net_lag_duration_minutes

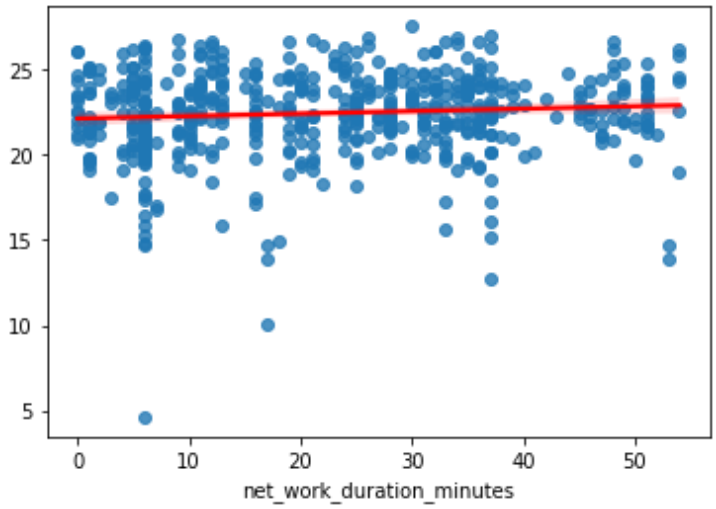


Figura 10. Gráfica del modelo lineal net_work_duration_minutes

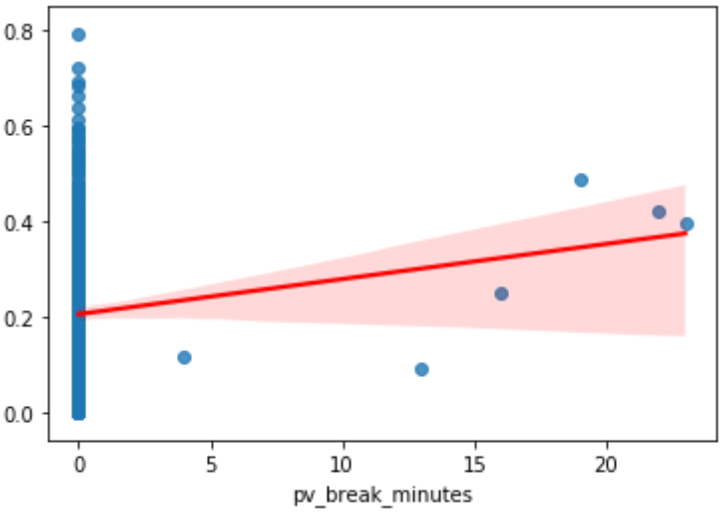


Figura 11. Gráfica del modelo lineal pv_break_minutes

4.1.1 VENTAJAS

Se puede comprobar por lo comentado, que las mayores ventajas de este tipo de modelo son la linealidad, que hace de la estimación un procedimiento simple, y sobre todo el fácil entendimiento que poseen las ecuaciones lineales (26).

Entre las diferentes ventajas que posee este modelo, es que puede ser entrenado con cualquier tamaño de muestra y se nos informa sobre la relevancia de las variables predictores.

4.1.2 DESVENTAJAS

Este tipo de modelo es muy sencillo porque modela las relaciones lineales entre las variables, siempre suponiendo y muchas veces de manera errónea que existe este tipo de relación (26).

Otra gran desventaja de la regresión lineal, es que es muy sensible a los valores atípicos (anomalías o outliers).

4.2 RANDOM FOREST

El Random Forest o bosque aleatorio en español, es una combinación de árboles de predicción, de modo que cada árbol depende del valor del vector aleatorio probado de forma independiente, y la distribución de cada árbol es la misma. Esta es una modificación importante de bagging: construye un conjunto largo de árboles no relacionados y luego los promedia (27).

Es un método versátil de aprendizaje, capaz de adaptarse tanto a problemas de clasificación como a problemas de regresión. Este método, realiza una reducción dimensional, trata valores perdidos, valores atípicos y otros pasos esenciales de exploración de datos. Es un tipo de método de aprendizaje por conjuntos, donde un grupo de modelos débiles se combinan para formar un modelo más potente (28).

En muchos problemas, el rendimiento del algoritmo de bosque aleatorio es muy similar al algoritmo de refuerzo, y el entrenamiento y el ajuste son más simples. Por lo tanto, los random forest son muy populares y se utilizan ampliamente (28).

La idea esencial de bagging, es promediar muchos modelos ruidosos pero poco sesgados, reduciendo así la varianza.

Los árboles pueden registrar estructuras interactivas complejas en los datos y, si crece lo suficiente, sus desviaciones son relativamente bajas, es por ello, que se adaptan de manera óptima a la utilización de bagging. La manera para construir cada árbol es la siguiente:

1. Sea N el número de casos de prueba y M el número de variables en el clasificador.
2. Sea m el número de variables de entrada utilizadas para determinar la decisión de un nodo dado; m debe ser mucho menor que M.
3. Elija un conjunto de entrenamiento para este árbol y use los casos de prueba restantes para estimar el error.
4. Para cada nodo del árbol, se seleccionan aleatoriamente m variables como base para la toma de decisiones. Calcule la mejor partición del conjunto de entrenamiento a partir de m variables (28).

Para predecir, se empuja un nuevo caso por el árbol hasta el final de este. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso recorre todos los árboles del ensamblaje y las etiquetas que ocurren con más frecuencia se informan como predicciones.

En cuanto a la implementación de este modelo en Python, se muestra en el siguiente código:

```
# Grid de hiperparámetros evaluados
# ==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }

# Búsqueda por grid search con validación cruzada
# ==============================================================================
grid = GridSearchCV(
        estimator  = RandomForestRegressor(random_state = 1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1998),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train, y = y_train)
```

Se ha hecho uso, como se puede apreciar, de la función de GridSearchCV(), que ayuda mediante validación cruzada a la estimación de los parámetros que mejor se adapten al modelo.

En las Figuras 12, 13, 14, muestran la importancia de las variables en los distintos modelos lag, work y break respectivamente.

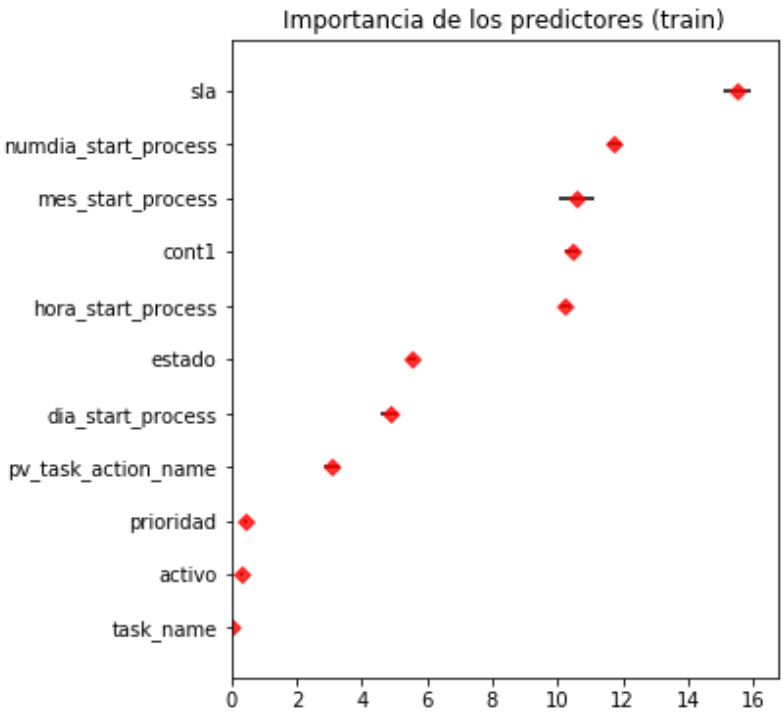


Figura 12. Importancia de las variables del modelo lag random forest

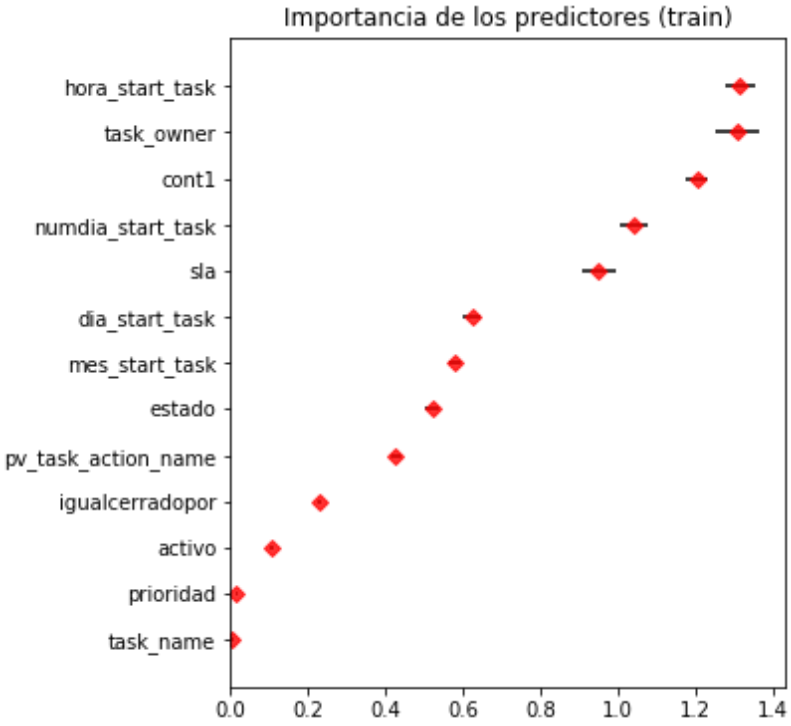


Figura 13. Importancia de las variables del modelo work random forest

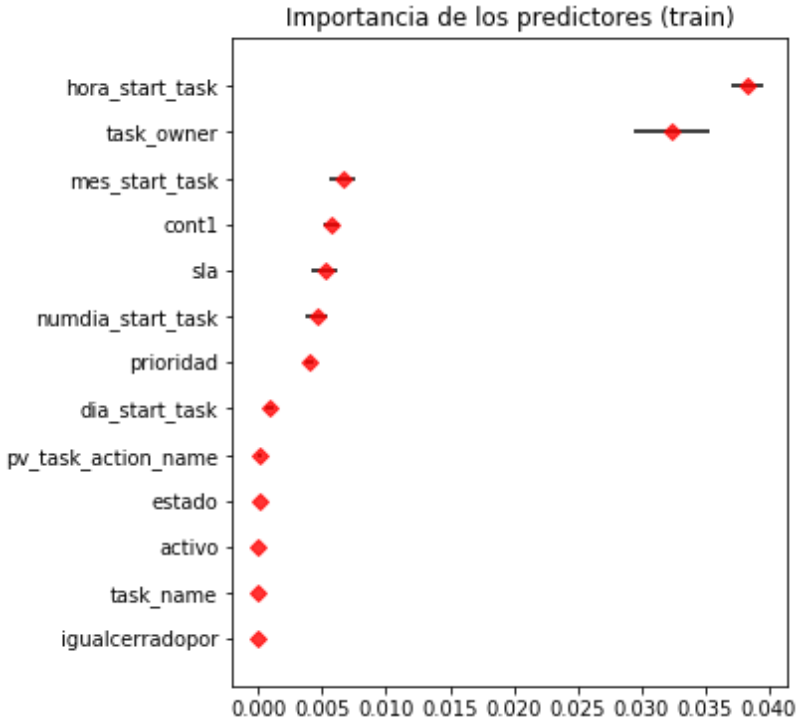


Figura 14. Importancia de las variables del modelo break random forest

4.2.1 VENTAJAS

Se puede observar, que este tipo de modelo, es muy potente y a la vez muy preciso debido al ajuste que realizan los distintos árboles. Cabe destacar, el buen rendimiento en muchos problemas, incluidos aquellos que se identifican como no lineales.

Para este modelo, la preparación de los datos es más escasa que para otros, pues hay pocas suposiciones. Este modelo a la vez, es capaz de identificar de entre miles de variables de entrada, aquellas más importantes para la explicación de nuestros datos.

Quizás, una de las ventajas más llamativas de los modelos Random Forest, son la reducción de la dimensionalidad que hacen. Además, se puede utilizar como método no supervisado (agrupamiento) y detección de valores atípicos (29).

4.2.2 DESVENTAJAS

Una de las grandes desventajas, es que el número de árboles a entrenar, debe ser seleccionado.

Al ser un modelo que internamente entrena muchos árboles, no permite interpretabilidad, y se puede caer de manera sencilla en el sobreajuste.

Si bien es cierto que para clasificación, es un modelo adecuado, en cambio, para hacer frente a problemas de regresión, no tanto, debido a que no puede predecir más allá del rango de valores del conjunto de entrenamiento (29).

4.3 K-NEAREST-NEIGHBORS

El K vecino más cercano, conocido como KNN, es un algoritmo de clasificación muy simple y sencillo de entender, pero a la vez importante en el aprendizaje automático.

Este algoritmo, se clasifica dentro del aprendizaje supervisado. Se trata de un clasificador que se basa en instancias y no en parámetros.

Un método no paramétrico significa que no hace suposiciones explícitas sobre la forma funcional de los datos, de esta manera, se evita así un modelado incorrecto. Si por ejemplo, si nuestro modelo contiene datos no gaussianos, pero automáticamente se elige un modelo que es gaussiano, nuestro algoritmo hará predicciones muy pobres (30).

Un aprendizaje está basado en instancias cuando, en vez de aprender un modelo, se aprende los datos de entrenamiento de memoria que luego en la fase de predicción, esta memorización la utiliza como si fuera el conocimiento. Esto quiere decir que, el algoritmo hará consultas a la base de datos, es decir, a nuestros datos de entrenamiento, cuando se le pida etiquetar una entrada, y es entonces cuando utilizará los datos de entrenamiento para ofrecernos una predicción que será más o menos acertada (30).

Hay que tener en cuenta que la fase de entrenamiento mínimo de KNN implica tanto una sobrecarga de memoria, ya que necesitamos almacenar un conjunto de datos potencialmente enorme, como una sobrecarga computacional durante la fase de validación, ya que clasificar una observación requiere agotar todo el conjunto de datos. Esto nos puede ofrecer tiempo de demora a la hora de obtener respuestas, luego en la práctica, es un gran inconveniente a tener en cuenta (30).

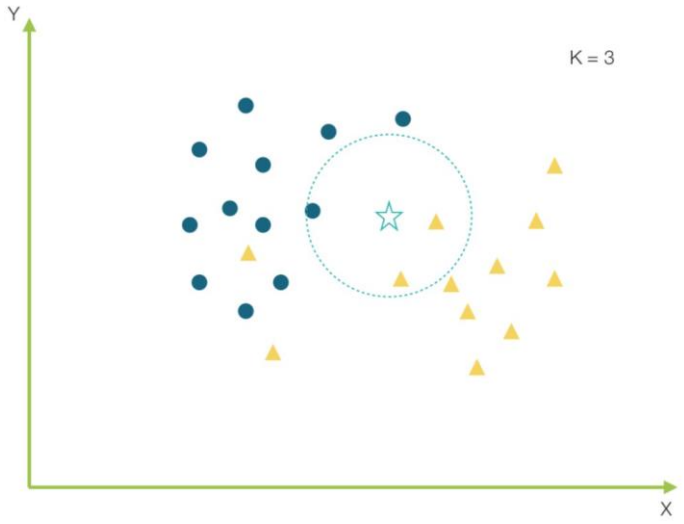


Figura 15. Ejemplo de funcionamiento del modelo KNN

En la Figura 15, la estrella imaginemos que es el punto Z que queremos predecir. Lo que internamente realiza el algoritmo KNN, es buscar los K puntos más cercanos al Z que queremos predecir. Cada K punto que pertenece a la fase de entrenamiento, se le tiene asociada una clase que será por la cual vote. Se hace un recuento en las votaciones obtenidas por los Ks y la clase con más votos será nuestra predicción del punto Z.

Para encontrar los K puntos más cercanos, se utilizan medidas de distancias.

La distancia más utilizada para hallar los K puntos, es la distancia euclidiana, pero también hay que tener en cuenta y conocer otro tipo de distancias, ya que podrían adaptarse mejor a nuestro modelo, entre ellas se incluyen la distancia de Mahattan y Minkowski cuya fórmula se muestra en la Figura 16.

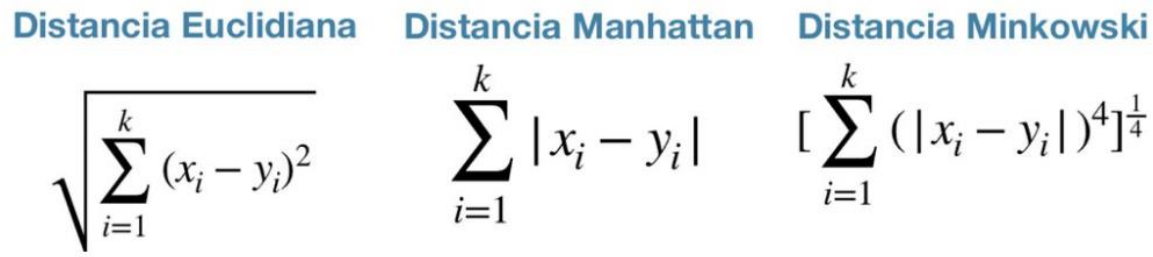


Figura 16. Fórmulas de las distancias

Se podría resumir el proceso del algoritmo KNN en los siguientes 4 pasos:

1. Calcular la distancia
2. Encontrar los K vecinos más cercanos al punto a predecir
3. Votar por las etiquetas
4. Hacer un recuento de las etiquetas y asignar la mayor votación

K, es un hiperparámetro elegido a la hora de construir el modelo. No existe un número K (vecinos) óptimo que sea capaz de adaptar a todos nuestros datos. Para un K pequeño, se tendrá un bajo sesgo y una alta varianza, y se tendrá más flexibilidad y un número grande de vecinos tendrán un límite de decisión más suave, lo que implica que tendrá una varianza más baja pero un sesgo más alto (30).

Un truco que suele darse para elegir el número de vecinos K, es fijarse en el número de clases que tiene nuestro modelo, en caso de que este número sea par, es recomendable tomar un K impar para que no haya problemas a la hora del recuento de votos y asignación de predicción.

El código utilizado para implementar KNN en Python ha sido el siguiente:

```
grid_params = {
    'n_neighbors': [3,5,11,19],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
}
gs = GridSearchCV(
    KNeighborsRegressor(),
    grid_params,
    verbose = 1,
    cv = 5,
    n_jobs = -1
```

```
    )

gs_results = gs.fit(x_train_scaled, y_train)
```

Podemos ver, que al igual que en el algoritmo anterior, se ha utilizado la función GridSearchCV() para el ajuste y selección de los mejores parámetros.

4.3.1 VENTAJAS

Como ventaja principal, destacar que es un algoritmo muy simple que puede manejar problemas cotidianos de clasificación e incluso aquellos que son de clasificación múltiple, sin olvidar, que es capaz de manejar problemas de regresión como es la estimación de tiempos de este proyecto.

A parte de ser simple, lo cual lo hace más sencillo de entender, es potente incluso para el reconocimiento de manuscritos de dígitos (31).

4.3.2 DESVENTAJAS

KNN es un algoritmo que no es bueno a la hora de procesar datos multidimensionales y además es sensible a atributos que resultan irrelevantes. El ruido afecta enormemente a este algoritmo.

Es muy lento cuando hay gran cantidad de datos de entrenamiento y depende mucho dela distancia elegida. Para ello, se normalizan las variables, para que los atributos con mayor rango, no tengan más peso que el resto (32).

4.4 SVM

SVM es un modelo lineal para problemas de clasificación y regresión. Puede resolver problemas lineales y no lineales y funciona bien para muchos problemas prácticos. La idea de la SVM es simple: El algoritmo crea una línea o un hiperplano que separa los datos en clases. La SVM es un algoritmo que toma los datos como entrada y produce una línea que separa esas clases si es posible (33).

Según el algoritmo SVM, encontramos los puntos más cercanos a la línea de ambas clases, que se denominan vectores de soporte. Calculamos la distancia entre la línea y los vectores de soporte. Esta distancia se llama margen. Nuestro objetivo es maximizar el margen. El hiperplano para el que el margen es máximo es el hiperplano óptimo.

Así, SVM trata de hacer una frontera de decisión de tal manera que la separación entre las dos clases (esa calle) sea lo más amplia posible.

Se puede dar el caso en que los datos no son linealmente separables. No podemos trazar una línea recta que pueda clasificar estos datos. Sin embargo, estos datos pueden convertirse en datos linealmente separables en una dimensión superior (33).

De esta manera, los datos son claramente separables linealmente (en otras dimensiones).

Por lo tanto, podemos proyectar el separador lineal en la dimensión superior de nuevo en las dimensiones originales utilizando una transformación.

Podemos entonces clasificar los datos añadiendo una dimensión adicional para que se conviertan en linealmente separables y, a continuación, proyectando el límite de decisión de vuelta a las dimensiones originales utilizando una transformación matemática. Pero encontrar la transformación correcta para cualquier conjunto de datos no es tan fácil (33). Podemos utilizar los kernels en la implementación de SVM de sklearn para hacer este trabajo.

Para este proyecto, los kernels utilizados, hay sido lineal y radial, y el código de ambos se muestra a continuación:

```
modelo = SVR(kernel = 'linear')
modelo.fit(X_train, y_train)

modelo  = SVR(kernel= "rbf", gamma='scale')
modelo.fit(X_train, y_train)
```

4.4.1 VENTAJAS

Es un algoritmo eficaz en espacios de grandes dimensiones, incluso es eficaz en casos donde el número de dimensiones es mayor que el número de muestras.

Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamada vectores de soporte), por lo que también es eficiente en memoria.

Es versátil, se pueden especificar diferentes funciones del núcleo para la función de decisión. Se proporcionan kernels comunes, pero también es posible especificar kernels personalizados (34).

4.4.2 DESVENTAJAS

El algoritmo SVM no es adecuado para grandes conjuntos de datos.

SVM no funciona muy bien cuando el conjunto de datos tiene más ruido, es decir, cuando las clases objetivo se solapan.

En los casos en los que el número de características para cada punto de datos supera el número de muestras de datos de entrenamiento, SVM tendrá un rendimiento inferior.

Como el clasificador de vectores de soporte funciona poniendo puntos de datos, por encima y por debajo del hiperplano de clasificación, no hay una explicación probabilística para la clasificación (35).

4.5 XGBOOST

XGBoost se ha convertido en una herramienta muy utilizada y realmente popular, ya que ha sido probada en producción en problemas a gran escala.

Es una herramienta flexible y versátil que puede trabajar con la mayoría de los problemas de regresión, clasificación y ranking, así como con funciones objetivo construidas por el usuario.

Su nombre significa eXtreme Gradient Boosting, fue desarrollado por Tianqi Chen y ahora forma parte de una colección más amplia de bibliotecas de código abierto desarrolladas por la Distributed Machine Learning Community (DMLC) (36).

XGBoost es una implementación escalable y precisa de las máquinas de refuerzo de gradiente y ha demostrado que supera los límites de la potencia de cálculo de los algoritmos de árboles reforzados, ya que se construyó y desarrolló con el único propósito de mejorar el rendimiento del modelo y la velocidad de cálculo. En concreto, se diseñó para explotar todos los recursos de memoria y hardware de los algoritmos de refuerzo de árboles.

La implementación de XGBoost es capaz de realizar las tres formas principales de gradient boosting (Gradient Boosting (GB), Stochastic GB y Regularized GB) y es lo suficientemente robusto como para soportar el ajuste fino y la adición de parámetros de regularización (36).

El algoritmo se ha desarrollado para reducir eficazmente el tiempo de cálculo y asignar un uso óptimo de los recursos de memoria. Entre las características importantes de la implementación se encuentran el manejo de los valores que faltan (Sparse Aware), la estructura de bloques para soportar la paralelización en la construcción del árbol y la capacidad de ajustar y potenciar en los nuevos datos añadidos a un modelo entrenado (Continued Training) (36).

En primer lugar, trataré el concepto de boosting. Se trata de un método de conjunto que busca crear un clasificador (modelo) fuerte a partir de clasificadores "débiles". En este contexto, débil y fuerte se refieren a una medida de la correlación de los aprendices con la variable objetivo real. Al añadir modelos uno encima de otro de forma iterativa, los errores del modelo anterior son corregidos por el siguiente predictor, hasta que los datos de entrenamiento son predichos o reproducidos con precisión por el modelo (36).

El boosting de gradiente también comprende un método de conjunto que añade secuencialmente predictores y corrige los modelos anteriores. Sin embargo, en lugar de asignar diferentes pesos a los clasificadores después de cada iteración, este método ajusta el nuevo modelo a los nuevos residuos de la predicción anterior y luego minimiza la pérdida al

añadir la última predicción. Al final, está actualizando su modelo utilizando el descenso de gradiente. Esto es compatible tanto con problemas de regresión como de clasificación. XGBoost, se implementa para el refuerzo de árboles de decisión con un término de regularización adicional en la función objetivo (36).

En el siguiente código, se muestran los parámetros utilizados y ajustados previamente con GridSearch():

```
xgb1 = XGBRegressor(
 objective='reg:squarederror',
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=3,
 min_child_weight=4,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.7,
 reg_alpha=1,
 nthread=4,
 scale_pos_weight=1,
 eval_metric= "mae",
 seed=27)
```

Tras el entrenamiento del modelo, se muestran las variables más importantes para el modelo en las Figuras 17, 18, 19 que corresponden con lag, work y break respectivamente.

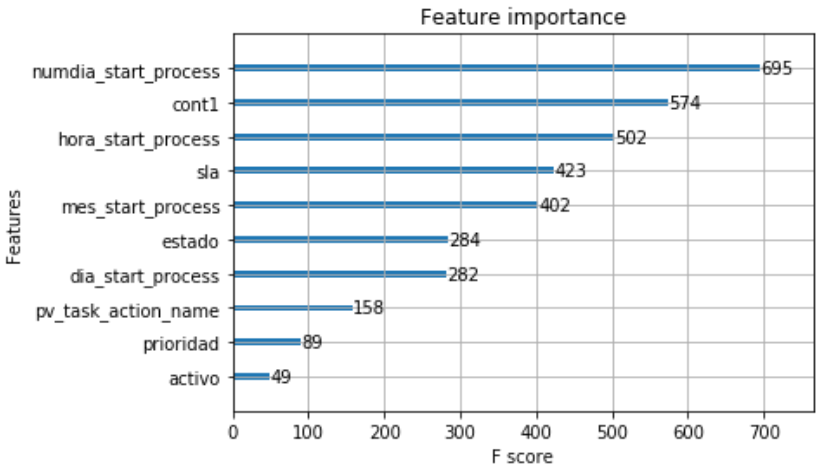


Figura 17. Importancia de las variables modelo lag XGBoost

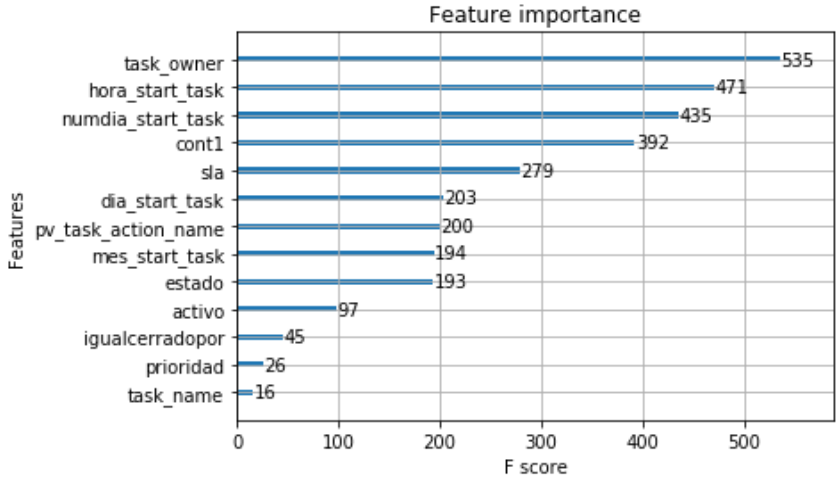


Figura 18. Importancia de las variables modelo work XGBoost

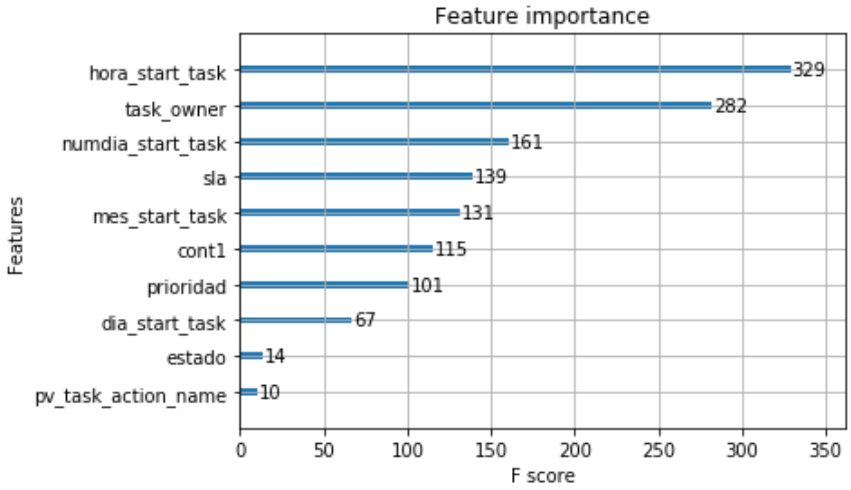


Figura 19. Importancia de las variables modelo break XGBoost

También, se ha mostrado el árbol realizado para cada uno de los modelos lag, work y break como muestran las Figuras 20, 21, 22 respectivamente.

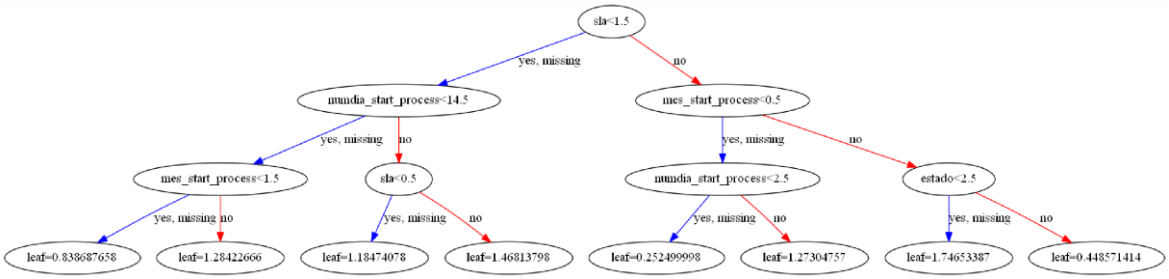


Figura 20. Árbol modelo lag XGBoost

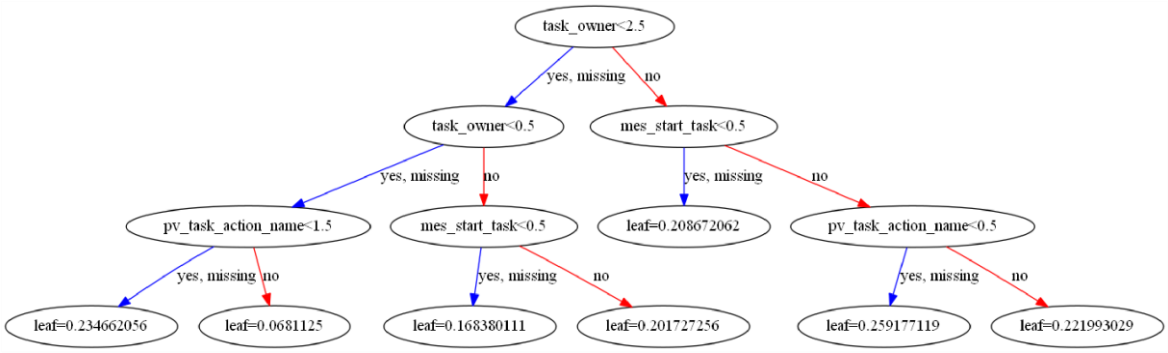


Figura 21. Árbol modelo work XGBoost

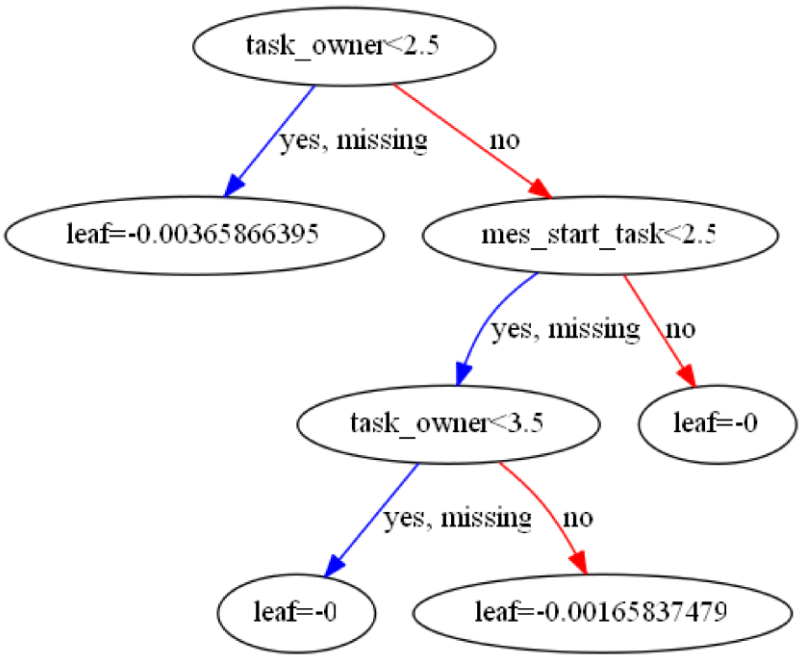


Figura 22. Árbol modelo break XGBoost

4.5.1 VENTAJAS

1. Regularización: XGBoost lleva incorporada la regularización L1 (Lasso Regression) y L2 (Ridge Regression) que evita que el modelo se ajuste en exceso.

2. Procesamiento paralelo: XGBoost utiliza la potencia del procesamiento paralelo y por eso es mucho más rápido que GBM. Utiliza múltiples núcleos de CPU para ejecutar el modelo.

3. Manejo de valores perdidos: XGBoost tiene una capacidad incorporada para manejar valores perdidos. Cuando XGBoost encuentra un valor faltante en un nodo, intenta tanto la división a la izquierda como a la derecha y aprende el camino que lleva a una mayor pérdida para cada nodo. Hace lo mismo cuando trabaja con los datos de prueba.

4. Validación cruzada: XGBoost permite al usuario ejecutar una validación cruzada en cada iteración del proceso de boosting y, por tanto, es fácil obtener el número óptimo exacto de iteraciones de boosting en una sola ejecución.

5. Poda efectiva del árbol: XGBoost realiza divisiones hasta la profundidad máxima especificada y luego comienza a podar el árbol hacia atrás y elimina las divisiones más allá de las cuales no hay ganancia positiva (37).

4.5.2 DESVENTAJAS

La interpretación es muy difícil al igual que la visualización. La posibilidad de sobreajuste si los parámetros no están bien ajustados es muy alta. Y del mismo modo, es un algoritmo difícil de ajustar, ya que tiene un gran número de hiperparámetros (38).

4.6 LIGHTGBM

LightGBM es un marco de refuerzo de gradiente que hace uso de algoritmos de aprendizaje basados en árboles que se considera un algoritmo muy potente cuando se trata de computación. Es un algoritmo de procesamiento rápido (39).

Mientras que los árboles de otros algoritmos crecen horizontalmente, el algoritmo LightGBM crece verticalmente, lo que significa que crece por hojas y otros algoritmos crecen por niveles. LightGBM elige la hoja con mayor pérdida para crecer. Puede reducir más pérdidas que un algoritmo por niveles al crecer la misma hoja.

LightGBM se llama "Light" por su capacidad de cálculo y por dar resultados más rápidos. Requiere menos memoria para ejecutarse y es capaz de manejar grandes cantidades de datos.

Se puede utilizar para grandes volúmenes de datos, especialmente cuando se necesita alcanzar una gran precisión (39).

LightGBM se considera un algoritmo realmente rápido y el más utilizado en el aprendizaje automático cuando se trata de obtener resultados rápidos y de alta precisión. Hay más de 100 parámetros en la documentación de LightGBM (39).

A continuación, se muestra una pieza de código usada para ajustar LightGBM:

```
param_grid = {'n_estimators'     : [100, 500, 1000, 5000],
              'max_depth'        : [-1, 1, 3, 5, 10, 20],
              'subsample'        : [0.5, 1],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'boosting_type'    : ['gbdt']
             }
grid = GridSearchCV(
        estimator  = LGBMRegressor(random_state=1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1998),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train, y = y_train)
```

En las Figuras 23, 24, 25, se muestran representadas de manera visual la importancia de las variables para los modelos lag, work y break respectivamente.

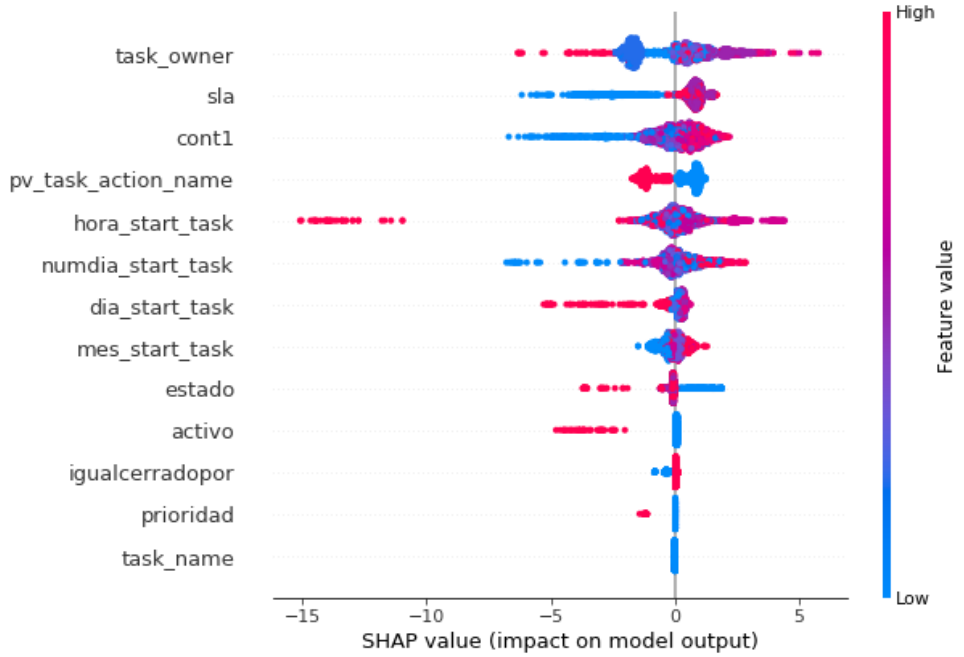


Figura 23. Importancia de las variables modelo lag LightGBM

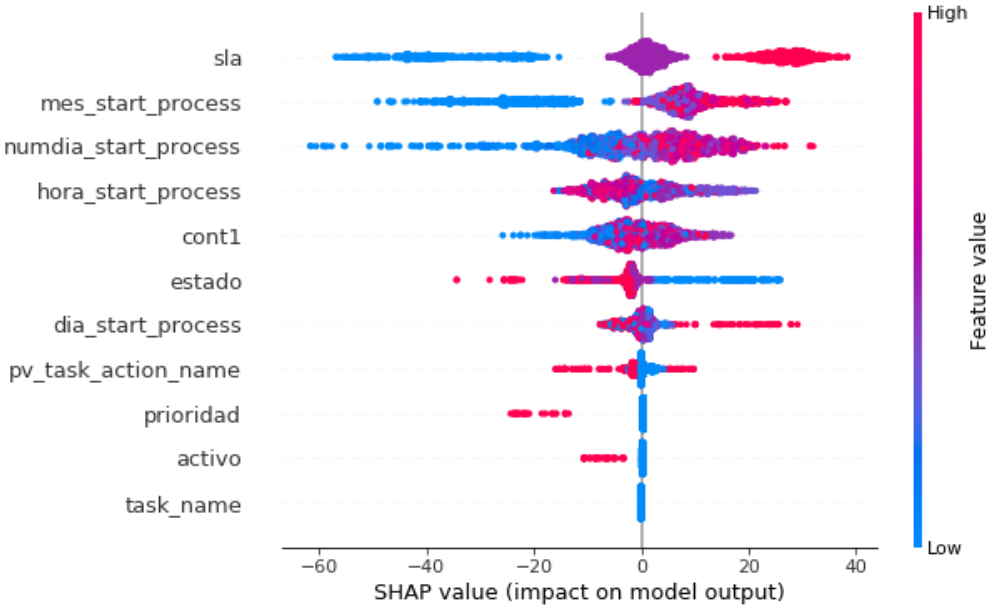


Figura 24. Importancia de las variables modelo work LightGBM

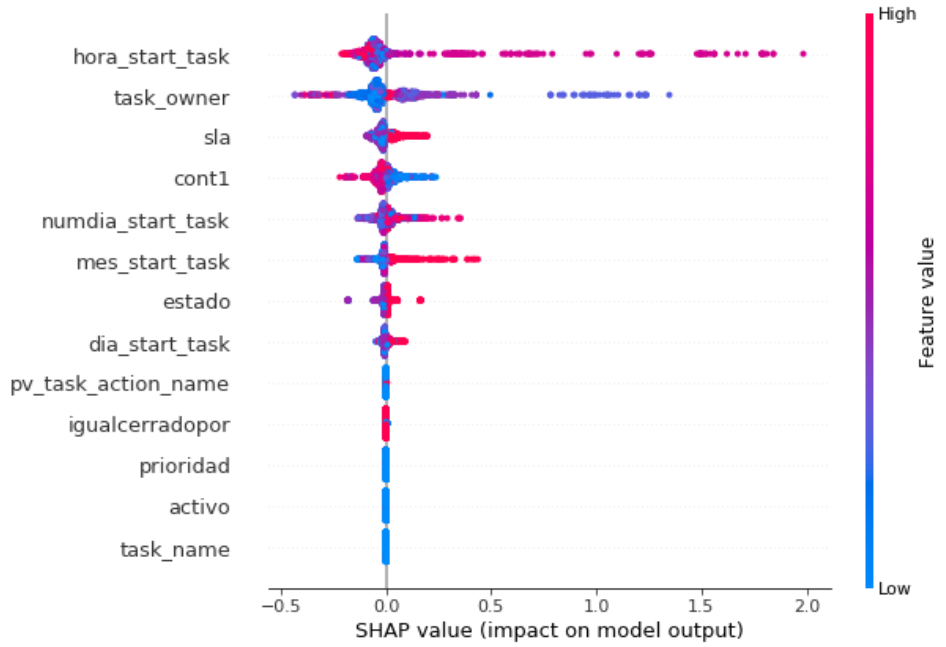


Figura 25. Importancia de las variables modelo break LightGBM

4.6.1 VENTAJAS

LightGBM tiene una mayor velocidad de entrenamiento y a la vez una mayor eficiencia.

Hace un menor uso de memoria ya que, sustituye los valores continuos por intervalos discretos, lo que permite un menor uso de la memoria.

Tiene una mejor precisión que cualquier otro algoritmo de refuerzo. Produce árboles mucho más complejos al seguir un enfoque de división por hojas en lugar de un enfoque por niveles, que es el principal factor para lograr una mayor precisión. Sin embargo, a veces puede llevar a un sobreajuste que puede evitarse estableciendo el parámetro max_depth.

Es capaz de funcionar igual de bien con grandes conjuntos de datos con una reducción significativa del tiempo de entrenamiento en comparación con XGBOOST.

Destacar que es un algoritmo que soporta el aprendizaje en paralelo (40).

4.6.2 DESVENTAJAS

LightGBM no es para un pequeño volumen de datos. Puede sobreajustar fácilmente los datos pequeños debido a su sensibilidad. Se puede utilizar para datos que tengan más de 10.000 filas. No hay un umbral fijo que ayude a decidir el uso de LightGBM.

Una de las desventajas de utilizar este algoritmo actualmente es su reducida base de usuarios, pero eso está cambiando rápidamente. Este algoritmo, además de ser más preciso y de ahorrar tiempo que el XGBOOST, ha tenido un uso limitado debido a la menor documentación disponible (41).

4.7 ENSEMBLE

El aprendizaje conjunto es un paradigma de aprendizaje automático en el que se entrenan múltiples modelos (a menudo denominados "aprendices débiles") para resolver el mismo problema y se combinan para obtener mejores resultados. La hipótesis principal es que cuando los modelos débiles se combinan correctamente podemos obtener modelos más precisos y/o robustos (42).

Para configurar un método de aprendizaje ensemble, primero tenemos que seleccionar nuestros modelos base para ser agregados. La mayoría de las veces, se utiliza un único algoritmo de aprendizaje de base, de modo que tenemos aprendices débiles homogéneos que se entrenan de diferentes maneras. Se dice entonces que el modelo de conjunto que obtenemos es "homogéneo". Sin embargo, también existen algunos métodos que utilizan diferentes tipos de algoritmos de aprendizaje de base: algunos aprendices débiles heterogéneos se combinan entonces en un "modelo de conjunto heterogéneo", como ha sido el caso de este proyecto (42).

Un punto importante es que la elección de aprendices débiles debe ser coherente con la forma en que agregamos estos modelos. Si elegimos modelos base con bajo sesgo pero alta varianza, debe ser con un método de agregación que tienda a reducir la varianza, mientras

que si elegimos modelos base con baja varianza pero alto sesgo, debe ser con un método de agregación que tienda a reducir el sesgo.

Esto nos lleva a la cuestión de cómo combinar estos modelos. Podemos mencionar tres tipos principales de meta-algoritmos que tienen como objetivo combinar aprendices débiles:

- **Bagging**, que a menudo considera aprendices débiles homogéneos, los aprende independientemente unos de otros en paralelo y los combina siguiendo algún tipo de proceso de promediación determinista.
- **Boosting**, que suele considerar aprendices débiles homogéneos, los aprende secuencialmente de forma muy adaptativa (un modelo base depende de los anteriores) y los combina siguiendo una estrategia determinista.
- **Stacking**, que suele considerar aprendices débiles heterogéneos, los aprende en paralelo y los combina entrenando un metamodelo para obtener una predicción basada en las predicciones de los diferentes modelos débiles.

A grandes rasgos, podemos decir que el bagging se centrará principalmente en obtener un modelo de conjunto con menos varianza que sus componentes, mientras que el boosting y el stacking tratarán principalmente de producir modelos fuertes menos sesgados que sus componentes (aunque la varianza también pueda reducirse) (42).

En Figura 26, se muestra de manera ilustrativa el funcionamiento que se ha llevado a cabo, que según lo explicado, ha sido Stacking, un conjunto de aprendices heterogéneos. Que ha sido una mezcla de todos los algoritmos explicados.

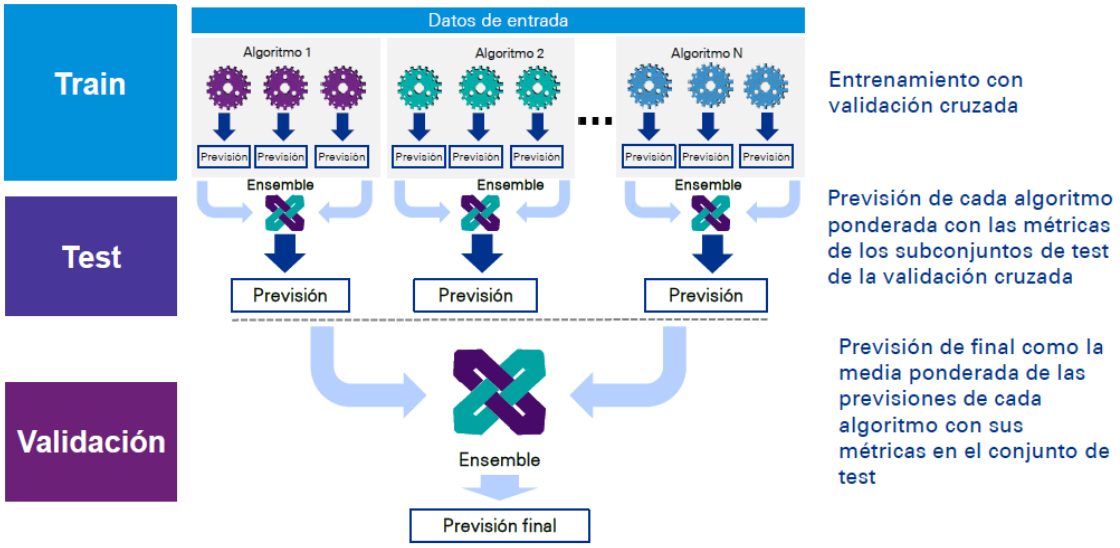


Figura 26. Funcionamiento del modelo empleado

El código utilizado para realizar un aprendizaje por conjunto ha sido el siguiente:

```
# Voting Ensemble for Classification
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
# create the sub models
estimators = []
model1 = LinearRegression()
estimators.append(('LR', model1))
model2 = RandomForestRegressor()
estimators.append(('RF', model2))
model3 = KNeighborsRegressor()
estimators.append(('KNN', model3))
model4 = SVR(kernel = 'linear')
estimators.append(('SVML', model4))
model5 = SVR(kernel= "rbf", gamma='scale')
estimators.append(('SVMR', model5))
model6 = XGBRegressor()
estimators.append(('XGBoost', model6))
model7 = LGBMRegressor()
estimators.append(('LightGBM', model7))
# create the ensemble model
ensemble = VotingRegressor(estimators)

predicciones_encemble = ensemble.fit(X,Y).predict(X)
```

4.7.1 VENTAJAS

Un conjunto puede crear una menor varianza y un menor sesgo. Además, un conjunto crea una comprensión más profunda de los datos. Los patrones de datos subyacentes quedan ocultos.

En general, los conjuntos tienen una mayor precisión predictiva. Los resultados de las pruebas mejoran con el tamaño del conjunto.

Con un enfoque de "bagging", cada modelo debe ajustarse a un sobreajuste. La independencia de los modelos se aprovecha porque el ensacado es una técnica de reducción de la varianza. Las predicciones pueden suavizarse para mejorar la estabilidad. Estos modelos se ejecutan en paralelo y se promedian. Con el boosting, los modelos se utilizan de forma secuencial y se da más peso a las clasificaciones erróneas de ejecuciones anteriores. El refuerzo es una técnica de reducción del sesgo. El apilamiento puede realizarse con bosques aleatorios. El apilamiento mejora la precisión y mantiene la varianza y el sesgo bajos (43).

4.7.2 DESVENTAJAS

Los conjuntos de modelos no siempre son mejores. Las nuevas observaciones pueden seguir confundiendo. Es decir, los conjuntos no pueden ayudar a las diferencias desconocidas entre la muestra y la población. Los conjuntos deben utilizarse con cuidado.

Los conjuntos pueden ser más difíciles de interpretar.

Por último, los conjuntos son más costosos de crear, entrenar y desplegar. El retorno de la inversión de un enfoque de conjunto debe considerarse cuidadosamente. Por lo general, una mayor complejidad no es buena (43).

Capítulo 5. EVALUACIÓN Y RESULTADOS

5.1 EVALUACIÓN

Para evaluar el desempeño de los modelos, se ha realizado la predicción con el modelo creado tanto en los datos de entrenamiento como en los datos de prueba. Con esto se ha comprobado si existe sobreajuste del modelo a los datos comparando los MAE de ambos datasets.

Existe sobreajuste en el modelo cuando el algoritmo aprende los patrones del conjunto de entrenamiento pero no logra los mismos resultados en el set de prueba. Esto suele suceder cuando se permite a los algoritmos de árboles de decisión crecer sin límites, hasta el punto que aprende de memoria los datos de entrenamiento. La técnica de optimización de hiperparámetros (hyperparameter tuning) sirve para evitar este inconveniente.

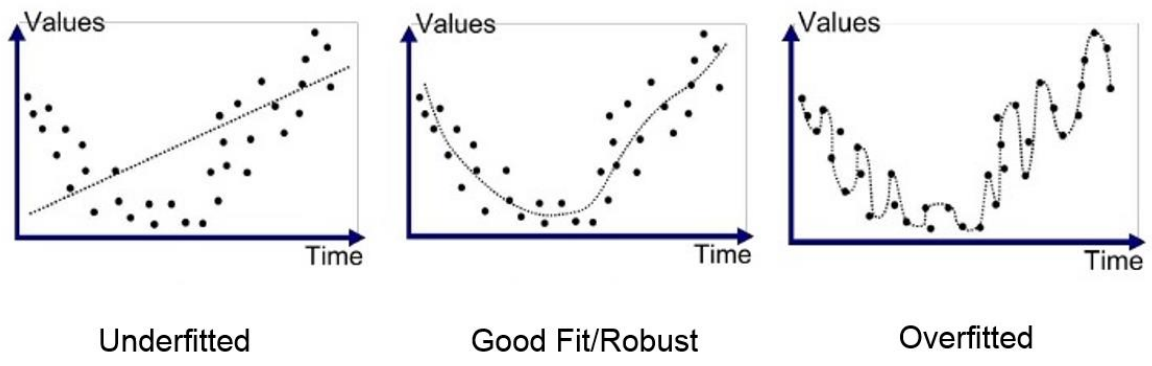


Figura 27. Casos de modelos

Tanto en la primer gráfica de la Figura 27 como la tercera muestran casos donde el modelo es o muy simple o extremadamente ajustado a los datos de entrenamiento. Es por ello, que el objetivo es encontrar un modelo como el del medio, bueno y robusto, que no se aprenda los datos de entrenamiento de memoria, ni que intente generalizar tanto que no consiga adaptarse al conjunto de los datos.

Los resultados promedio obtenidos para los mejores modelos se ven en la tabla 4 mostrada a continuación. Se comprobó que los modelos no estuvieran sobreajustado, tras esa comprobación, se obtuvieron los siguientes resultados en el conjunto de validación:

	Algoritmo empleado	**MAE (test)**
Net_lag_duration_minutes	LR	75.8 (149.86)
Net_work_duration_minutes	LR	12.7 (25.30)
Pv_break_minutes	LR	0.37 (0.66)
Net_lag_duration_minutes	RF	72.8 (149.86)
Net_work_duration_minutes	RF	12.1 (25.30)
Pv_break_minutes	RF	0.35 (0.66)
Net_lag_duration_minutes	KNN	83.4 (149.86)
Net_work_duration_minutes	KNN	12.6 (25.30)
Pv_break_minutes	KNN	0.45 (0.66)
Net_lag_duration_minutes	SVM Lineal	74.7 (149.86)

Net_work_duration_minutes	SVM Lineal	12.7 (25.30)
Pv_break_minutes	SVM Lineal	0.27 (0.66)
Net_lag_duration_minutes	SVM Radial	78.0 (149.86)
Net_work_duration_minutes	SVM Radial	12.9 (25.30)
Pv_break_minutes	SVM Radial	0.27 (0.66)
Net_lag_duration_minutes	XGBoost	72.9 (149.86)
Net_work_duration_minutes	XGBoost	12.1 (25.30)
Pv_break_minutes	XGBoost	0.39 (0.66)
Net_lag_duration_minutes	LightGBM	72.6 (149.86)
Net_work_duration_minutes	LightGBM	12.0 (25.30)
Pv_break_minutes	LightGBM	0.34 (0.66)
Net_lag_duration_minutes	Ensemble	58.9 (149.86)
Net_work_duration_minutes	Ensemble	9.9 (25.30)

Pv_break_minutes	Ensemble	0.25 (0.66)

Tabla 4. Resumen de las métricas MAE(Media del fichero)

En la tabla se observan los resultados de los distintos modelos y los resultados del modelo ensamblado indicado como óptimo. Para cada celda con resultados en primer lugar se encuentra el MAE del conjunto de evaluación, que indica el error cometido en promedio, y entre paréntesis se encuentra la media en minutos de la variable respuesta que se estudia en cada modelo. Por ejemplo, en el modelo lineal para la variable Net_lag_duration_minutes, en media se tarda 150 minutos hasta que se resuelve la tarea y en promedio el error es de casi 76 minutos.

Como el modelo ensamblado proporciona en promedio un error menor, véase en las Figuras 28, 29, 30, se selecciona como modelo óptimo. Por tanto, en el siguiente apartado se comentan únicamente los resultados del modelo ensamblado.

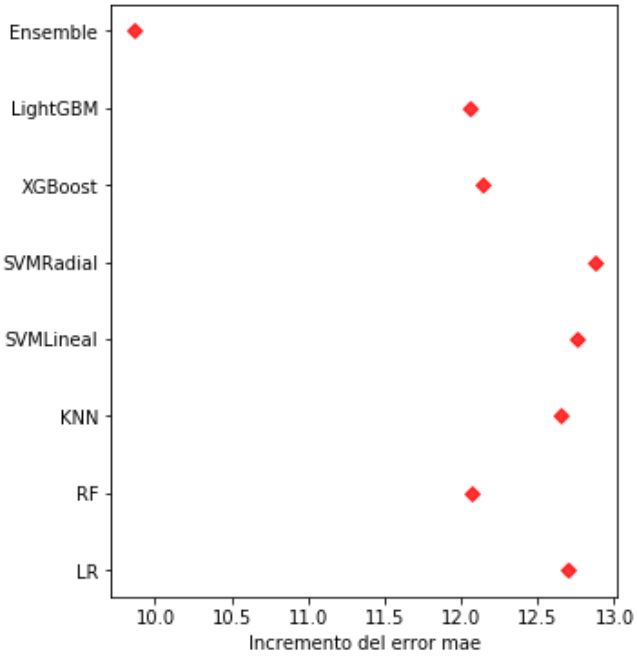


Figura 28. Incremento del error MAE en los modelos lag

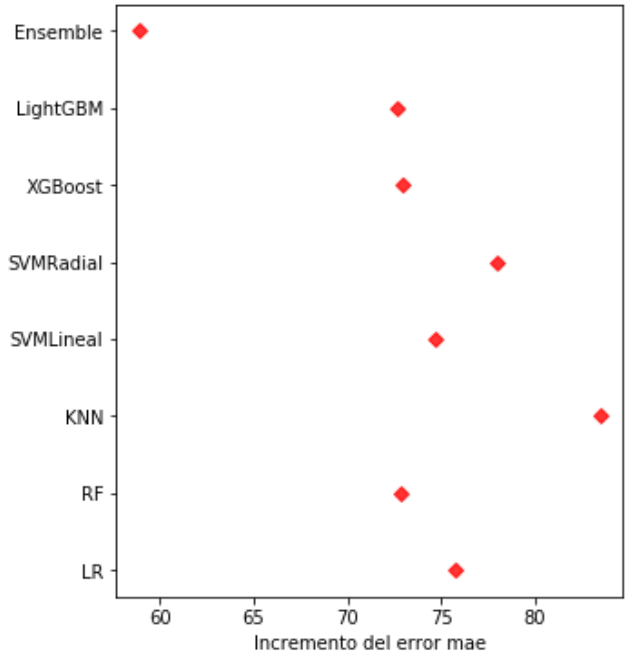


Figura 29. Incremento del error MAE en los modelos work

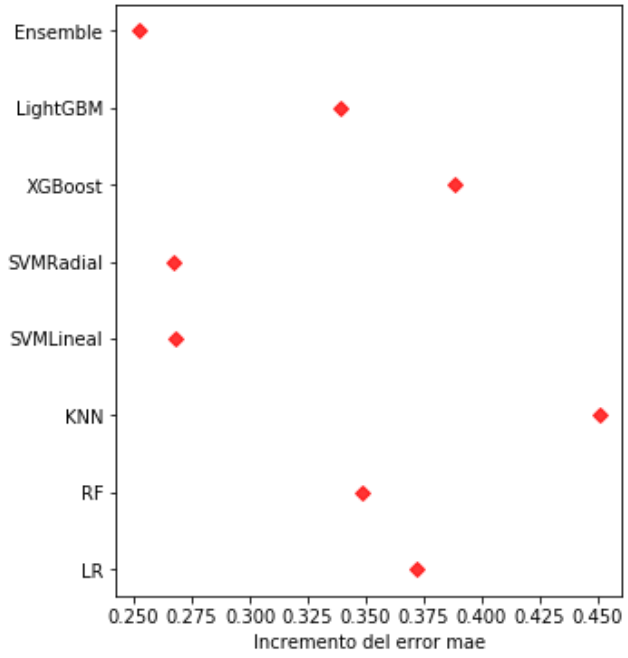


Figura 30. Incremento del error MAE en los modelos break

Estos números a simple vista son difíciles de interpretar como buenos o malos. Una mejor idea es analizar el error distribuido por duración de la tarea, y eso es lo que presento en la siguiente sección.

5.2 RESULTADOS

A continuación se presentan los resultados obtenidos en función del valor real de la variable respuesta. Para esto, se va a separar el número de intervalos que se vea conveniente para cada una de las variables y se va a mostrar en gráficos la diferencia en valor absoluto, la diferencia entre la media de la predicción en el intervalo y el valor real.

Por un lado, se van a comentar los resultados de los modelos obtenidos para las variables lag y work, posteriormente se comentarán los resultados de la variable break.

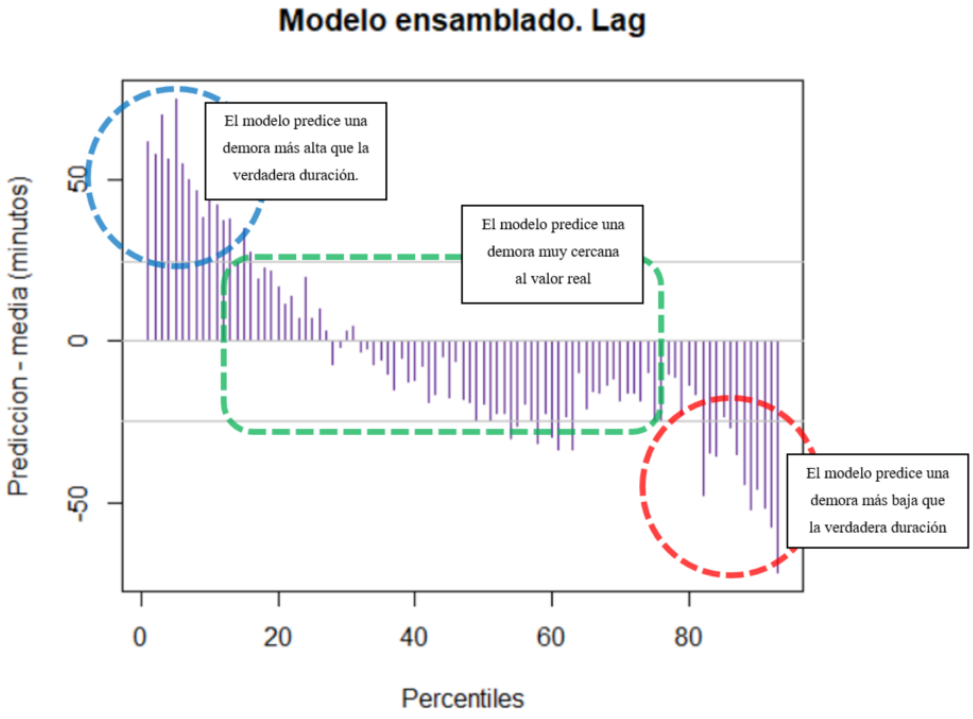


Figura 31. Error en minutos por percentiles de la variable Lag.

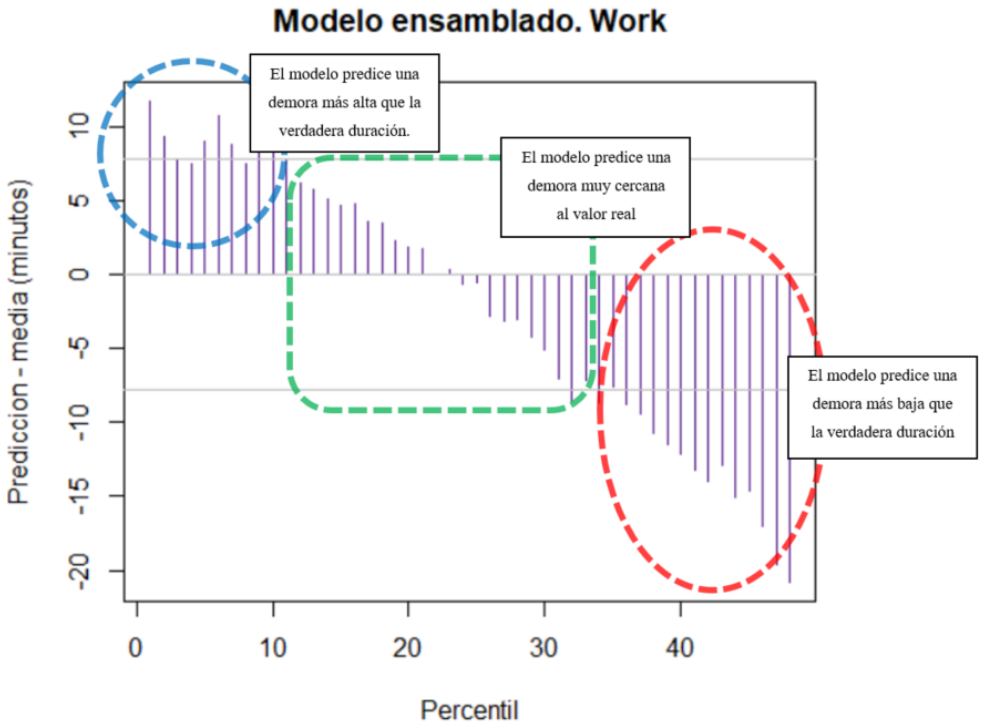


Figura 32. Error en minutos por percentiles de la variable Work.

Se recuerda que lag hace referencia al tiempo entre que se genera la tarea y es aceptada por el usuario, mientras que work es el tiempo entre que el analista acepta la tarea y la resuelve.

En las Figura 31 y 32, se recogen dos gráficos cuya interpretación es similar. Se recoge la diferencia, por grupos, entre la media de la estimación y la media de lo observado. Las líneas grises horizontales indican: equidad entre lo estimado y lo observado y la media del MAE tanto en positivo como negativo. Para la variable lag, se ha agrupado por percentiles mientras que work se ha agrupado por los percentiles de dos en dos. En ambos gráficos de la figura se observa que en los extremos es donde peor se predice, pero que en los percentiles de en medio, el error es menor que el error promedio.

Lo que podemos sacar de resultado de nuestros modelos lag y work, es que es capaz de adaptarse muy bien en tareas de duración media, sin embargo, las tareas de duración breve, son estimadas con valores altos, lo cual no nos supone mucho problema, ya que el cliente esperará un tiempo alto y la tarea será realizada de manera más rápida. El problema que se

puede ver en las gráficas, está a la hora de predecir las tareas de duración por encima de la media, ya que el valor que se predice es bajo. Esto supone un problema para nuestro cliente, ya que puede afectar gravemente la satisfacción del usuario, ya que se le estima una duración corta y la demora de la tarea aumentará a más de la media estimada de todas las tareas; en otras palabras, tiene un mayor impacto al usuario.

Por otro lado, está el modelo relativo a los descansos. En esta variable el 97% de los datos tiene valor 0, por lo que el modelo va a tender a predecir mejor cuando no hay descansos que cuando los hay como se puede ver en los errores de la tabla.

Los modelos empleados, al no ser paramétricos, no permiten extraer coeficientes como una típica regresión lineal, pero sí nos permiten mostrar qué variables son las más importantes para predecir el tiempo de duración de cada una de las etapas. Se ha extraído dicha importancia y se muestra en la siguiente Tabla 5:

Lag		Work		Break	
Variable	Importancia (%)	Variable	Importancia (%)	Variable	Importancia (%)
sla1	100	hora_start_task	100	task_owner[redacted]	100
numdia_start_process	20	activo1	43	hora_start_task	46
hora_start_process	19	sla2	37	numdia_start_task	25
cont1	18	task_owne[redacted]	18	sla1	10
estado5	12	pv_task_action_nameOK Pago	16	task_owne[redacted]	8
sla2	11	numdia_start_task	11	estado2	7
mes_start_process5	4	estado5	10	cont1	7
mes_start_process4	4	task_owner[redacted]	10	mes_start_task4	6
prioridad1	4	cont1	8	igualcerradopor1	5
mes_start_process7	3	task_owne[redacted]	8	dia_start_task4	3
Pv_task_action_nameOK Pago	3	task_owne[redacted]	7	estado3	3
Dia_start_process5	2	task_owne[redacted]	7	mes_start_task7	2

Tabla 5. Importancia de las variables en el modelo empleado

En el modelo desarrollado para el tiempo lag, la variable sla tiene mucha influencia en el tiempo hasta que se asigna la tarea. El momento del mes y la hora en el que se reciba el email que abre la tarea importa también en este modelo. La carga de trabajo se puede observar que también influye. Y el mes en el que se reciba el trabajo influirá para nuestra predicción.

En el modelo desarrollado para el tiempo work, la hora a la que se recibe el email influye en el tiempo de trabajo del empleado, la variable activo tiene relevancia como podemos ver.

Sla está relacionado con el tiempo de trabajo y el trabajador que gestione la tarea influye a la hora de determinar el tiempo.

El modelo desarrollado para el tiempo break, el trabajador que esté llevando a cabo la tarea es muy importante. La hora y el momento del mes influyen en los tiempos de descanso de los trabajadores y en función del sla el trabajador toma más o menos descanso.

Capítulo 6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1 CONCLUSIONES

Los resultados que presento en este proyecto, demuestran que con unos pocos datos obtenidos directamente de las tablas disponibles actualmente, somos capaces de predecir la demora de una tarea con diferentes niveles de precisión, a través del empleo de técnicas de Machine Learning.

El modelo final empleado presenta dificultades en predecir las tareas ubicadas en los percentiles 1-10 y 80-100. Puntualmente, para aquellas tareas que normalmente resultan en una rápida resolución, el modelo predecirá demoras de entre 20 y 50 minutos. En términos de impacto a la experiencia de usuario, se podría relativizar este impacto en el sentido de que el usuario verá que su tarea se resuelve aún más rápido.

Para los percentiles 80 en adelante, es decir las tareas de mayor duración el modelo tiene una tendencia a subestimar la duración. Estas serían los casos más extremos.

Por otro lado, todo lo ubicado entre el percentil 10 y el 80 demuestra un muy buen desempeño entre lo predicho y lo real.

La dificultad en predecir los valores más extremos (los de resolución más rápida y los de mayor duración) demuestran que hay datos que el modelo no ha utilizado y que son claves para pulir estos detalles.

Entiendo estos resultados como una buena base para plantear un proyecto que abarque con mayor profundidad esta conceptualización y se plantee llevar estos modelos a producción.

Resulta esencial, la incorporación de nuevos datos que no han sido utilizados en esta prueba de concepto, fundamentalmente la información de correos electrónicos ya mencionada.

6.2 TRABAJO FUTURO

Los resultados del pequeño proyecto realizado, nos indican que es posible predecir los tiempos de resolución de las operaciones de financiación aplicando técnicas de Machine Learning, y creo que en una fase 2 se puede mejorar la precisión de los modelos y llevarlos a producción. En la siguiente Tabla 6 se muestra como lo planteo:

Conceptualización Fase 2	Toma de requerimientos funcionales
	Pedidos de datos actualizados
	Extracción de datos de correos electrónicos
Reentrenamiento del modelo con más datos	Reexploración de datos
	Incorporación de variables nuevas y correos electrónicos
	Reentrenamiento de modelos
	Evaluación de modelos
	Modelo final
Productivización y despliegue de la solución	Lector de correos electrónicos
	Despliegue y automatización de la solución
Optimización de asignaciones	Modelado de optimización de asignaciones

Tabla 6. Secciones del trabajo futuro

En la etapa de conceptualización Fase 2, sería necesaria la recepción actualizada de los datos de los últimos meses y por otro lado, fundamental para mejorar los modelos en fase 2 la extracción de datos de correos electrónicos.

En la etapa de reentrenamiento del modelo con más datos, se volvería a hacer una exploración de los datos para la incorporación de nuevas variables en caso de que se necesitasen ya que dispondríamos de nuevos datos proporcionados como son los correos electrónicos. La intención en esta fase, sería lograr una reducción del error de predicción del 20% - 30% con respecto al proyecto actual.

Para la productivización y despliegue de la solución, sería necesario un lector de correos electrónicos, que fuese el motor de extracción de metadatos de emails. Y para el despliegue y automatización, sería necesaria una integración de los modelos con los sistemas para la visualización del usuario o desarrollo propio.

Por último, se cree que los modelos lograrían además generar un sistema de asignación de tareas inteligente para optimizar tiempos aún más.

Capítulo 7. BIBLIOGRAFÍA

[1] B. (2019, 28 agosto). *CRISP-DM: La metodología para poner orden en los proyectos*. Sngular. https://www.sngular.com/es/data-science-crisp-dm-metodologia/

[2] *Appian*. (2021). [Software]. https://appian.com/es/resources/resource-center/google/appian-obtiene-la-distincion-customers-choice-como-la-plataforma-google.html

[3] Ferrer, F. R. (2019). *Descubriendo Appian, la plataforma de desarrollo de software de moda*. https://blog.softtek.com/es/appian.

[4] Robledano, A. (2020, 3 julio). *Qué es Python: Características, evolución y futuro*. OpenWebinars.net. https://openwebinars.net/blog/que-es-python/

[5] Gonzalez, L. (2020, 20 agosto). *Introducción a la Librería Pandas de Python - Parte 1*. Aprende IA. https://aprendeia.com/introduccion-a-la-libreria-pandas-de-python-parte-1/

[6] Alberca, A. S. (2021, 14 mayo). *La librería Pandas*. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/pandas/

[7] Alberca, A. S. (2020, 4 octubre). *La librería Numpy*. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/numpy/

[8] Alberca, A. S. (2020, 4 octubre). *La librería Numpy*. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/numpy/

[9] colaboradores de Wikipedia. (2021, 17 febrero). *NumPy*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/NumPy

[10] Alberca, A. S. (2020, 15 septiembre). *La librería Datetime*. Aprende con Alf. https://aprendeconalf.es/docencia/python/manual/datetime/

[11] *warnings — Warning control — Python 3.9.5 documentation*. (2001). https://docs.python.org/3/library/warnings.html.

[12] Gonzalez, L. (2020, 20 agosto). *Introducción a la Librería Matplotlib de Python – Parte 1*. Aprende IA. https://aprendeia.com/libreria-pandas-de-matplotlib-tutorial/

[13] Rodríguez, D. (2018, 1 diciembre). *Visualización de datos en Python con Seaborn*. Analytics Lane. https://www.analyticslane.com/2018/07/20/visualizacion-de-datos-con-seaborn/

[14] *Presentación | Interactive Chaos*. (s. f.). interactivechaos. Recuperado 10 de junio de 2021, de https://interactivechaos.com/es/manual/tutorial-de-seaborn/presentacion

[15] Carmona, P. (2020, 21 abril). *Data Visualization con pandas y seaborn - Ironhack*. Medium. https://medium.com/ironhack/data-visualization-con-pandas-y-seaborn-1044906af34f

[16] *random —Generar números pseudoaleatorios — documentación de Python - 3.9.5*. (s. f.). python. Recuperado 10 de junio de 2021, de https://docs.python.org/es/3/library/random.html

[17] *La librería random | Interactive Chaos*. (s. f.). Recuperado 10 de junio de 2021, de https://interactivechaos.com/es/manual/tutorial-de-python/la-libreria-random

[18] *math — Funciones matemáticas — documentación de Python - 3.10.0b2*. (s. f.). python. Recuperado 10 de junio de 2021, de https://docs.python.org/es/3.10/library/math.html

[19] Gonzalez, L. (2020, 20 agosto). *Introducción a la librería Scikit-Learn de Python*. Aprende IA. https://aprendeia.com/libreria-scikit-learn-de-python/

[20] colaboradores de Wikipedia. (2021, 2 mayo). *SciPy*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/SciPy

[21] *XGBoost Documentation — xgboost 1.5.0-dev documentation*. (s. f.) Recuperado 13 de junio de 2021, de https://xgboost.readthedocs.io/en/latest/

[22] Bagnato, J. I. (2020, 25 junio). *10 Librerías Python para Data Science y Machine Learning*. SolidQ Blogs. https://blogs.solidq.com/es/poder-del-dato/10-librerias-python-para-data-science-y-machine-learning/

[23] *Librerías en Data Science | Interactive Chaos*. (s. f.). Recuperado 13 de junio de 2021, de https://interactivechaos.com/es/manual/tutorial-de-python/librerias-en-data-science

[24] fisterra.com. (s. f.). *Técnicas de regresión: Regresión Lineal Simple*. Recuperado 13 de junio de 2021, de https://www.fisterra.com/mbe/investiga/regre_lineal_simple/regre_lineal_simple.asp

[25] Rodrigo, J. A. (s. f.). *Introducción a la Regresión Lineal Múltiple*. Ciencia de datos. Recuperado 13 de junio de 2021, de https://www.cienciadedatos.net/documentos/25_regresion_lineal_multiple.html

[26] N. (2021, 30 enero). *Pros y Contras de Regresión Lineal*. Enciclopedia de ventajas y desventajas. https://prosycontras.net/educacion/pros-y-contras-de-regresion-lineal/

[27] colaboradores de Wikipedia. (2020, 5 octubre). *Random forest*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Random_forest

[28] Gonzalez, L. (2020, 19 agosto). *Aprendizaje Supervisado: Random Forest Classification*. Aprende IA. https://aprendeia.com/aprendizaje-supervisado-random-forest-classification/

[29] Johanna Orellana Alvear - johanna.orellana@ucuenca.edu.ec. (s. f.). *Arboles de decision y Random Forest*. Recuperado 13 de junio de 2021, de https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html

[30] Gonzalez, L. (2020, 21 agosto). *K Vecinos más Cercanos - Teoría*. Aprende IA. https://aprendeia.com/k-vecinos-mas-cercanos-teoria-machine-learning/

[31] *Resumen de las ventajas y desventajas del algoritmo KNN y resumen del proceso de aprendizaje automático - programador clic*. (s. f.). programmer click. Recuperado 13 de junio de 2021, de https://programmerclick.com/article/24621148533/

[32] M. (2018, 7 julio). *KNN Ventajas y Desventajas-knn*. pdfslide.net. https://pdfslide.net/documents/knn-ventajas-y-desventajas-knn.html

[33] Pupale, R. (2019, 11 febrero). *Support Vector Machines(SVM) — An Overview - Towards Data Science*. Medium. https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

[34] Fernandez, R. (2019, 5 septiembre). *Support Vector Machines (SVM)*. ▷ Cursos de Programación de 0 a Experto © Garantizados. https://unipython.com/support-vector-machines-svm/

[35] K, D. (2020, 26 diciembre). *Top 4 advantages and disadvantages of Support Vector Machine or SVM*. Medium. https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107

[36] Brownlee, J. (2021, 16 febrero). *A Gentle Introduction to XGBoost for Applied Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[37] Kumar, N., Kumar, N., & Profile, V. M. C. (s. f.). *Advantages of XGBoost Algorithm in Machine Learning*. Recuperado 13 de junio de 2021, de http://theprofessionalspoint.blogspot.com/2019/03/advantages-of-xgboost-algorithm-in.html

[38] Gupta, S. (2020, 23 junio). *Pros and cons of various Machine Learning algorithms*. Medium. https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6

[39] Dwivedi, R. (s. f.). *What is LightGBM Algorithm, How to use it? | Analytics Steps*. Recuperado 13 de junio de 2021, de https://www.analyticssteps.com/blogs/what-light-gbm-algorithm-how-use-it

[40] *what are the Pros and cons of LGB model? | Data Science and Machine Learning*. (s. f.). Kaggle. Recuperado 13 de junio de 2021, de https://www.kaggle.com/questions-and-answers/103834

[41] Khandelwal, P. (2020, 27 marzo). *Which algorithm takes the crown: Light GBM vs XGBOOST?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

[42] Rocca, J. (2021, 7 abril). *Ensemble methods: bagging, boosting and stacking - Towards Data Science*. Medium. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

[43] Kapalko, R. (2019, 7 noviembre). *Predictive Model Ensembles: Pros and Cons*. Perficient Blogs. https://blogs.perficient.com/2019/11/07/predictive-model-ensembles-pros-and-cons/

CÓDIGO

In [1]:
```
import pandas as pd
import numpy as np
import datetime
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
```

CARGAMOS LOS DATOS EN DATAFRAMES

In [2]:
```
dataUrdexpediente = pd.read_csv("urdexpediente.txt",sep=";")
```

In [3]:
```
urdhistoricosistema = pd.read_csv("urdhistoricosistema.txt", sep=";",encoding
='latin1')
```

CARGAMOS LA TABLA PRINCIPAL ELIMINANDO LOS ESPACIOS EN BLANCO

In [4]:
```
dataurdm_process_task_metric = pd.read_csv("urdm_process_task_metric.txt",sep
="\s*;",encoding='latin_1')
```

CRUZAMOS LA TABLA PRINCIPAL CON dataUrdexpediente EN EL NUEVO DATAFRAME data

In [5]:
```
data = dataurdm_process_task_metric.merge(dataUrdexpediente, left_on='pv_oper
acion',right_on='operacion',how='inner')
data.columns
```

Out[5]:
```
Index(['task_metric_id', 'task_uuid', 'task_id', 'task_display_name',
       'task_name', 'task_owner', 'task_assignees', 'task_status',
       'task_status_name', 'task_start_time', 'task_assignment_time',
       'task_completion_time', 'task_description', 'task_priority',
       'task_priority_name', 'task_deadline', 'process_display_name',
       'process_id', 'process_model_uuid', 'process_model_id',
       'process_model_name', 'completion_duration', 'lag_duration',
       'work_duration', 'net_completion_duration', 'net_lag_duration',
       'net_work_duration', 'completion_duration_minutes',
       'lag_duration_minutes', 'work_duration_minutes',
       'net_comp_duration_minutes', 'net_lag_duration_minutes',
       'net_work_duration_minutes', 'last_updated_at', 'created_at',
       'application_code', 'ignore_ind', 'custom_field_1', 'custom_field_2',
       'custom_field_3', 'calc_task_acceptance_time',
       'calc_last_user_lag_duration', 'calc_last_user_lag_duration_minutes',
       'pv_break_minutes', 'pv_task_action_code', 'pv_task_action_name',
       'user_rol', 'calc_task_start_time', 'pv_operacion',
       'pv_tipo_operacion_code', 'pv_tipo_operacion_name',
       'process_start_time', 'effectiveworktime', 'id', 'operacion',
       'tipooperacion', 'fecharecepcion', 'estado', 'prioridad', 'sla',
       'fechafinsla', 'fechacierre', 'cerradopor', 'usuarioprioridad',
       'activo', 'finplazogestiones'],
      dtype='object')
```

IDENTIFICAR MISSING VALUES

In [6]:
```
data = data.replace('', np.nan)
data = data.replace('Not Found', np.nan)
```

ELIMINAS COLUMNAS CON INFO DUPLICADA

In [7]:
```
data = data.loc[:, ["pv_operacion",
                    "pv_tipo_operacion_name",
                    "pv_break_minutes",
                    "pv_task_action_name",
                    "task_name",
                    "task_owner",
                    "task_start_time",
                    "task_assignment_time",
                    "net_lag_duration_minutes",
                    "net_work_duration_minutes",
                    "task_id",
                    "user_rol",
                    "process_start_time",
                    "tipooperacion",
                    "estado",
                    "prioridad",
                    "sla",
                    "cerradopor",
                    "activo"]]
```

CAMBIAMOS AL FORMATO FECHA CORRECTO

In [8]:
```
data['task_start_time'] = pd.to_datetime(data['task_start_time'], format="%Y-
%m-%d %H:%M:%S")
data['process_start_time'] = pd.to_datetime(data['process_start_time'], forma
t="%Y-%m-%d %H:%M:%S")
```

In [9]:
```
data.dtypes
```

Out[9]:
```
pv_operacion                        object
pv_tipo_operacion_name              object
pv_break_minutes                   float64
pv_task_action_name                 object
task_name                           object
task_owner                          object
task_start_time             datetime64[ns]
task_assignment_time                object
net_lag_duration_minutes           float64
net_work_duration_minutes          float64
task_id                              int64
user_rol                            object
process_start_time          datetime64[ns]
tipooperacion                      float64
estado                               int64
prioridad                            int64
sla                                  int64
cerradopor                          object
activo                               int64
dtype: object
```

CREAMOS VARIABLE TIPOUSUARIO

In [10]:
```
descansos = data.groupby(by=['task_owner']).mean().iloc[:,0:1]
descansos.columns = ['descanso']
```

In [11]:
```
data = data.merge(descansos, left_on='task_owner',right_on='task_owner',how=
'inner')
```

In [12]:
```
data["tipousuario"] = np.where(data["descanso"] > 0, 1, 0)
data = data.drop(['descanso'],axis=1)
```

CREAMOS LA VARIABLE IGUALCERRADOPOR

In [13]:
```
data['igualcerradopor']=0
for i in range(0,len(data['task_owner'])):
    if ((data['task_owner'][i] is not None) and (data['cerradopor'][i] is not
None)):
        if (data['task_owner'][i] == data['cerradopor'][i]):
              data['igualcerradopor'][i] = 1
```

In [14]:
```
data = data.drop(['cerradopor'],axis=1)
```

RECODIFICACIÓN DE TASK_STAR_TIME EN DIA DE LA SEMANA, DÍA, MES, AÑO Y HORA

In [15]:
```
data["dia_start_task"] = data["task_start_time"].dt.strftime("%w")
data["mes_start_task"] = data["task_start_time"].dt.strftime("%m")
data["hora_start_task"] = data["task_start_time"].dt.strftime("%H")
data["ano_start_task"] = data["task_start_time"].dt.strftime("%Y")
data["numdia_start_task"] = data["task_start_time"].dt.strftime("%d")
```

RECODIFICACION DE PROCESS_START_TIME EN DIA DE LA SEMANA, DÍA, MES, AÑO Y HORA

In [16]:
```
data["dia_start_process"] = data["process_start_time"].dt.strftime("%w")
data["mes_start_process"] = data["process_start_time"].dt.strftime("%m")
data["hora_start_process"] = data["process_start_time"].dt.strftime("%H")
data["ano_start_process"] = data["process_start_time"].dt.strftime("%Y")
data["numdia_start_process"] = data["process_start_time"].dt.strftime("%d")
```

TRATAMOS FILAS CON MISSING. Los modelos no utilizan datos missing

In [17]:
```
any(data['task_owner'].isna())
any(data['igualcerradopor'].isna())
```

Out[17]:
```
False
```

In [18]:
```
data.isna().sum()
```

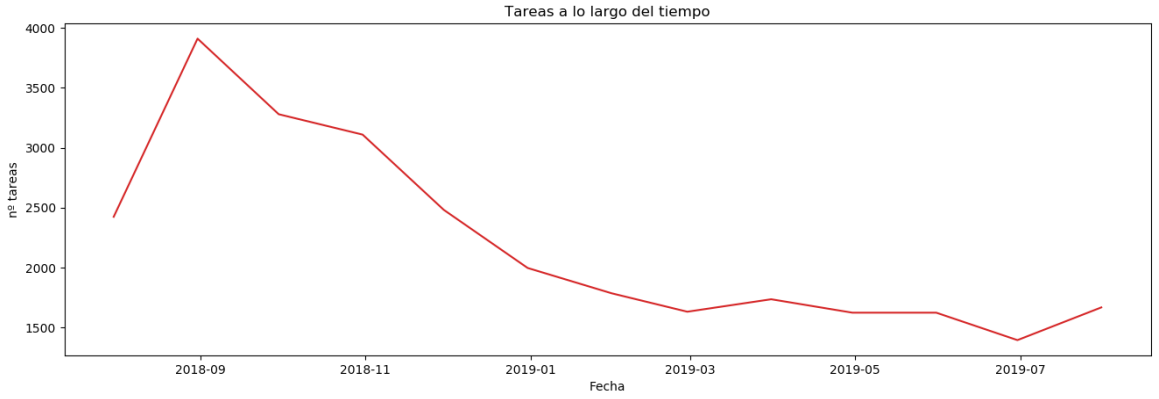
Out[18]:
```
pv_operacion                    0
pv_tipo_operacion_name        275
pv_break_minutes                0
pv_task_action_name         13575
task_name                       0
task_owner                      0
task_start_time                 0
task_assignment_time            0
net_lag_duration_minutes        0
net_work_duration_minutes       0
task_id                         0
user_rol                        0
process_start_time              0
tipooperacion                 302
estado                          0
prioridad                       0
sla                             0
activo                          0
tipousuario                     0
igualcerradopor                 0
dia_start_task                  0
mes_start_task                  0
hora_start_task                 0
ano_start_task                  0
numdia_start_task               0
dia_start_process               0
mes_start_process               0
hora_start_process              0
ano_start_process               0
numdia_start_process            0
dtype: int64
```

In [19]:
```
data = data.dropna()
```

REPRESENTAMOS LAS TAREAS A LO LARGO DEL TIEMPO

In [20]:
```
datosGrafico = data
datosGrafico['task_start_time'] = pd.to_datetime(datosGrafico['task_start_tim
e'], format="%Y-%m-%d")
datosGrafico = datosGrafico.set_index('task_start_time')
datosGrafico=datosGrafico.resample('m').count()
```

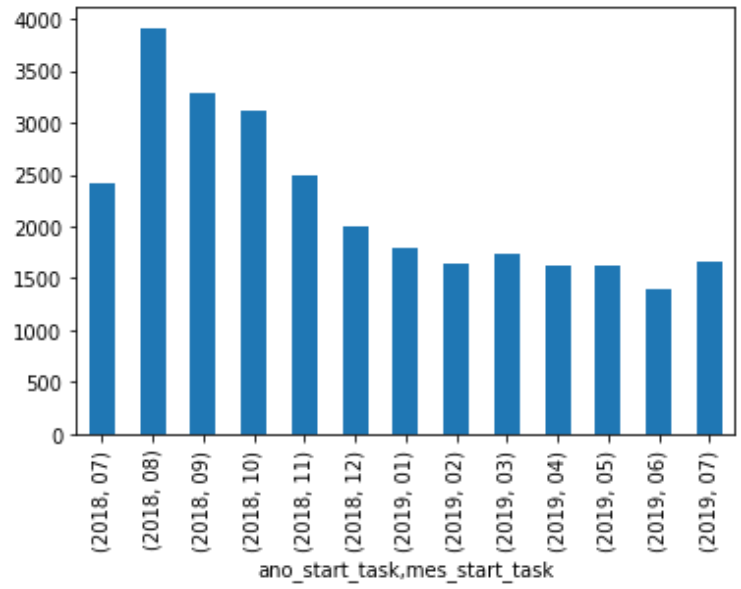
In [21]:
```
def plot_df(df, x, y, ylabel="" ,title="", xlabel='Fecha', dpi=100):
    plt.figure(figsize=(16,5), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()

plot_df(datosGrafico, x=datosGrafico.index, y= datosGrafico['pv_operacion'],
ylabel='nº tareas', title='Tareas a lo largo del tiempo')
```



In [22]:
```
datosMes = data.groupby(by=['ano_start_task','mes_start_task']).count()
datosMes = datosMes['pv_operacion']
tareasMes = data.groupby(by=['ano_start_process','mes_start_process']).count
()
tareasMes = tareasMes['pv_operacion']
```

In [23]:
```
datosMes.plot.bar(x=datosMes.index, y='pv_operacion')
```

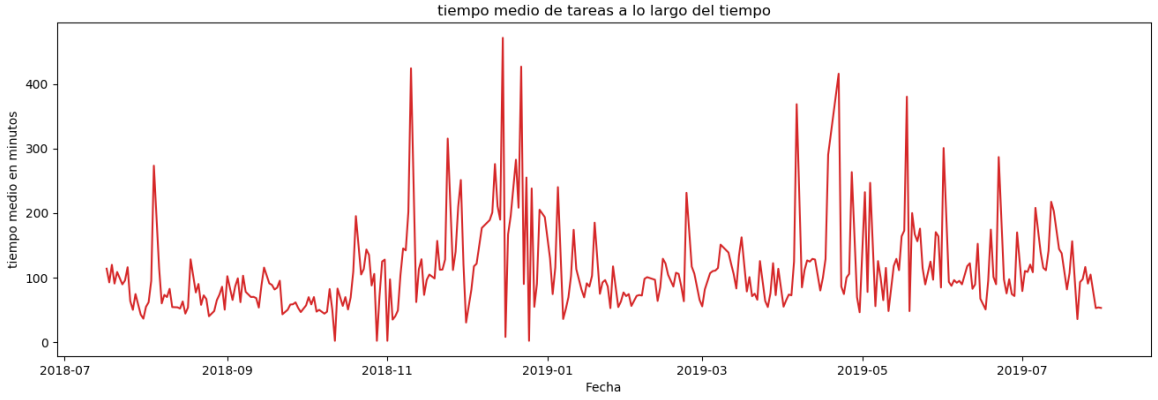
Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e8c71edac8>
```



REPRESENTAMOS EL TIEMPO MEDIO EN RESOLVER UNA TAREA A LO LARGO DEL TIEMPO

In [24]:
```
datosNuevoGrafico = data
datosNuevoGrafico['total_minutos'] = datosNuevoGrafico['pv_break_minutes'] +
datosNuevoGrafico['net_lag_duration_minutes'] + datosNuevoGrafico['net_work_d
uration_minutes']
datosNuevoGrafico = datosNuevoGrafico.loc[:,['task_start_time','total_minuto
s']]
datosNuevoGrafico['task_start_time'] = datosNuevoGrafico['task_start_time'].d
t.date
datosNuevoGrafico = datosNuevoGrafico.groupby(by=['task_start_time']).mean()
```

In [25]:
```
plot_df(datosNuevoGrafico, x=datosNuevoGrafico.index, y= datosNuevoGrafico['t
otal_minutos'], ylabel='tiempo medio en minutos', title='tiempo medio de tare
as a lo largo del tiempo')
```

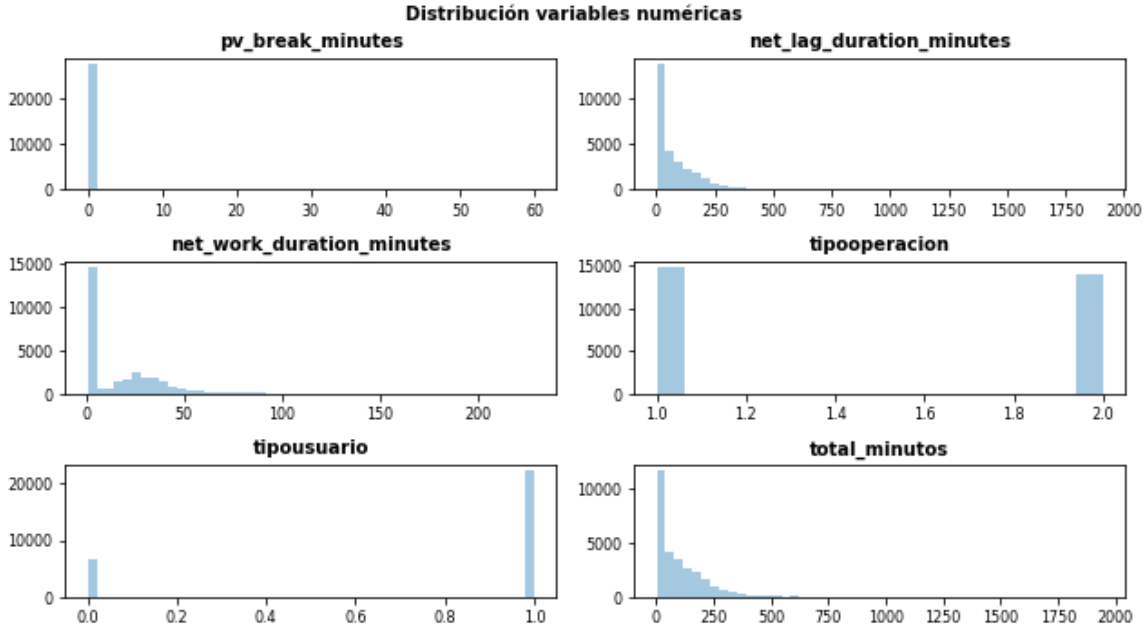


In [26]:
```
datosNuevoGrafico['total_minutos'].mean()
```

Out[26]: 110.42782182634687

REPRESENTAMOS LA DISTRIBUCIÓN DE CADA VARIABLE NUMÉRICA

In [27]:
```
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(9, 5))
axes = axes.flat
columnas_numeric = data.select_dtypes(include=['float64', 'int']).columns

for i, colum in enumerate(columnas_numeric):
    sns.distplot(
        data[colum],
        kde = False,
        hist=True,
        ax = axes[i]
    )
    axes[i].set_title(colum, fontsize = 10, fontweight = "bold")
    axes[i].tick_params(labelsize = 8)
    axes[i].set_xlabel("")

fig.tight_layout()
plt.subplots_adjust(top = 0.9)
fig.suptitle('Distribución variables numéricas', fontsize = 10, fontweight =
"bold");
```



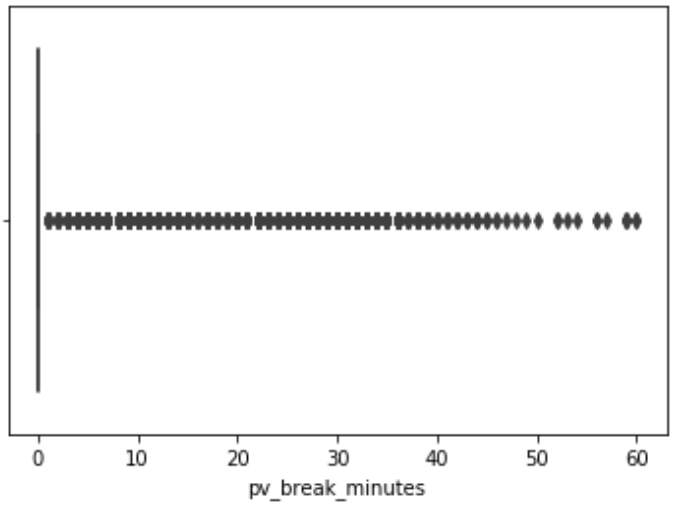
REPRESENTAMOS LOS DATOS CON BOXPLOT

In [28]:
```
data.columns
```

Out[28]:
```
Index(['pv_operacion', 'pv_tipo_operacion_name', 'pv_break_minutes',
       'pv_task_action_name', 'task_name', 'task_owner', 'task_start_time',
       'task_assignment_time', 'net_lag_duration_minutes',
       'net_work_duration_minutes', 'task_id', 'user_rol',
       'process_start_time', 'tipooperacion', 'estado', 'prioridad', 'sla',
       'activo', 'tipousuario', 'igualcerradopor', 'dia_start_task',
       'mes_start_task', 'hora_start_task', 'ano_start_task',
       'numdia_start_task', 'dia_start_process', 'mes_start_process',
       'hora_start_process', 'ano_start_process', 'numdia_start_process',
       'total_minutos'],
      dtype='object')
```

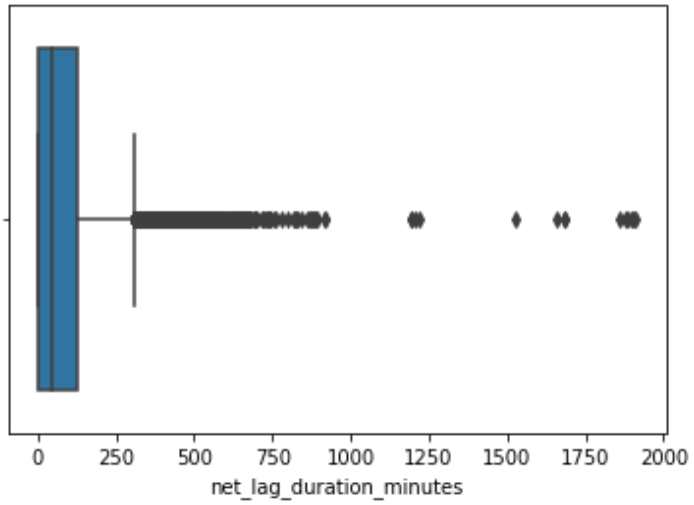
In [29]:
```
sns.boxplot(data=data, x='pv_break_minutes')
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c70d5d48>



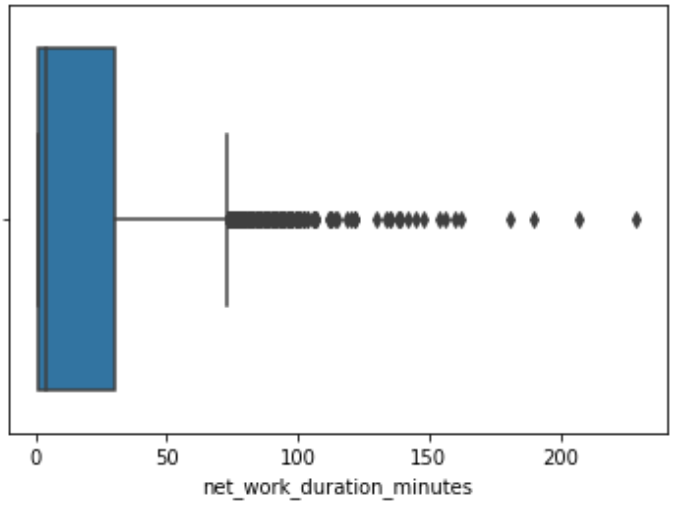
In [30]:
```
sns.boxplot(data=data, x='net_lag_duration_minutes')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6b57448>



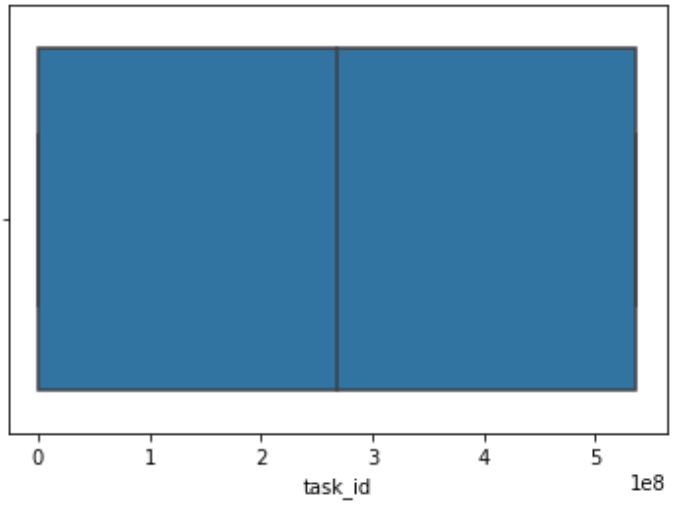
In [31]:
```
sns.boxplot(data=data, x='net_work_duration_minutes')
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6b1de08>



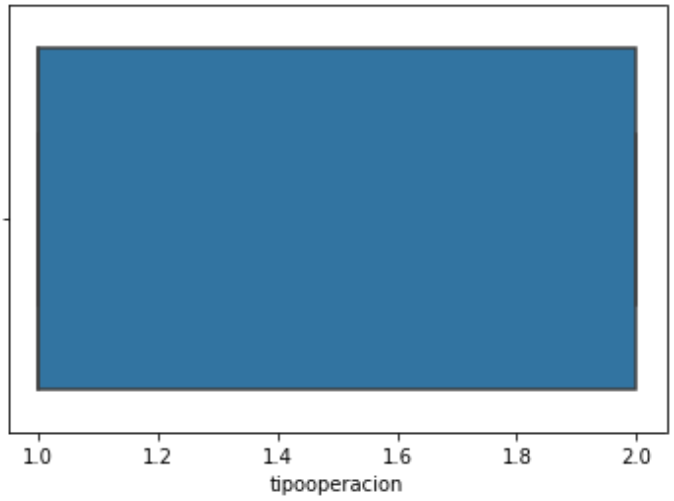
In [32]:
```
sns.boxplot(data=data, x='task_id')
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6e4a708>



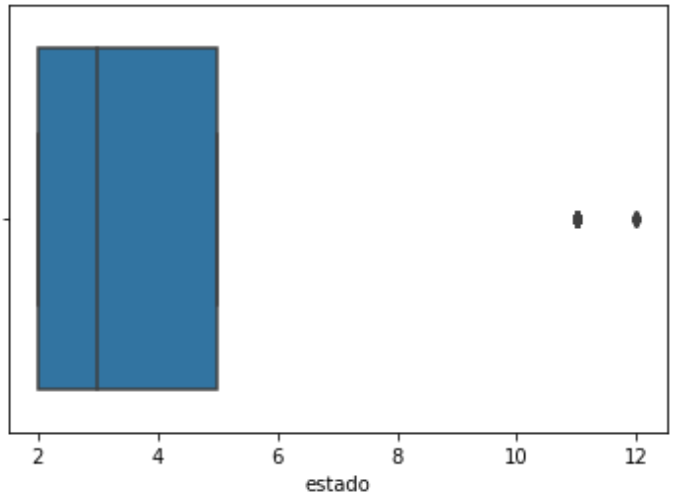
In [33]:
```
sns.boxplot(data=data, x='tipooperacion')
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6957208>



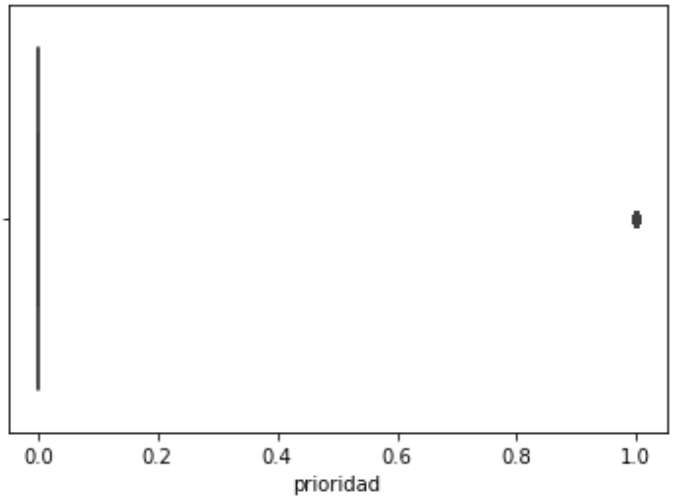
In [34]:
```
sns.boxplot(data=data, x='estado')
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8be5d5048>



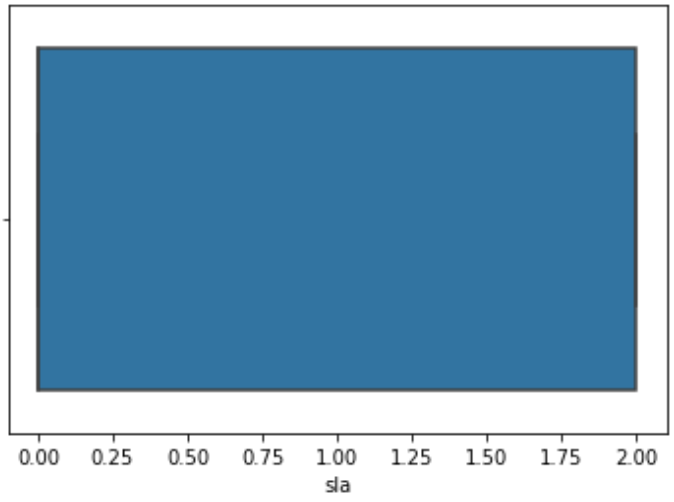
In [35]:
```
sns.boxplot(data=data, x='prioridad')
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6913648>



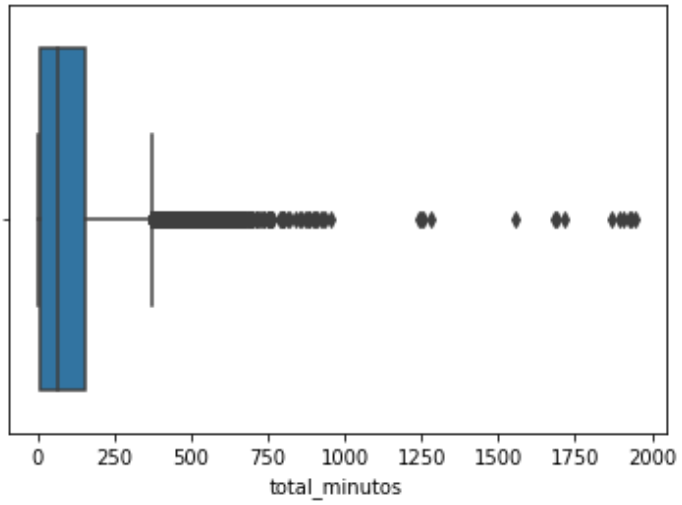
In [36]:
```
sns.boxplot(data=data, x='sla')
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6871688>



In [37]:
```
sns.boxplot(data=data, x='total_minutos')
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1e8c6835248>



SE ELIMINAN LOS OUTLIERS DE net_lag_duration_minutes Y net_work_duration_minutes

In [38]:
```
data = data[data.net_lag_duration_minutes < 1000]
data = data[data.net_work_duration_minutes < 175]
```

SELECCION DE LOS DATOS ADECUADOS SIN 2018

In [39]:
```
data = data[data["pv_tipo_operacion_name"]!= "Empresa"]
data = data[data["ano_start_process"] == "2019"]
data = data[data["mes_start_process"]!= "03"]
data = data[data["mes_start_process"]!= "02"]
data = data[data["mes_start_process"]!= "01"]
```

CREACIÓN DE LA VARIABLE AUXILIAR NÚMERO DE TAREAS RECIBIDAS POR DÍA

In [40]:
```
data_aux1 = data.groupby(['ano_start_process', 'mes_start_process','numdia_st
art_process','dia_start_process']).count()

data_aux1['cont1']= data_aux1['pv_operacion']

data_aux1['id']= data_aux1.index

data_aux1 = data_aux1.iloc[:, [-1,-2]]
```

In [41]:
```
data_aux2 = data.groupby(['ano_start_task', 'mes_start_task','numdia_start_ta
sk','dia_start_task']).count()

data_aux2['cont2']= data_aux2['pv_operacion']

data_aux2['id']= data_aux2.index

data_aux2 = data_aux2.iloc[:, [-1,-2]]
```

In [42]:
```
data_aux = data_aux1.merge(data_aux2, left_on='id',right_on='id',how='inner')
```

In [43]:
```
urdhistoricosistema = urdhistoricosistema[urdhistoricosistema['tipo'] == 1]
```

In [44]:
```
urdhistoricosistema['fecha'] = pd.to_datetime(urdhistoricosistema['fecha'], f
ormat="%Y-%m-%d %H:%M:%S")
urdhistoricosistema["dia"] = urdhistoricosistema["fecha"].dt.strftime("%w")
urdhistoricosistema["mes"] = urdhistoricosistema["fecha"].dt.strftime("%m")
urdhistoricosistema["numdia"] = urdhistoricosistema["fecha"].dt.strftime("%d"
)
urdhistoricosistema["ano"] = urdhistoricosistema["fecha"].dt.strftime("%Y")
urdhistoricosistema = urdhistoricosistema.drop(['id','fecha'],axis=1)
```

In [45]:
```
urdhistoricosistema = urdhistoricosistema.drop_duplicates()
```

In [46]:
```
data_numtrabajadores = urdhistoricosistema.groupby(['ano', 'mes','numdia','di
a']).count()
```

In [47]:
```
data_numtrabajadores['cont']= data_numtrabajadores['tipo']
data_numtrabajadores = data_numtrabajadores.drop(['tipo','valor','usuario'],a
xis=1)
```

In [48]:
```
data_numtrabajadores['id']= data_numtrabajadores.index
```

In [49]:
```
data_aux = data_aux.merge(data_numtrabajadores, left_on='id',right_on='id',ho
w='inner')
data_aux = data_aux.sort_values('id')
```

In [50]:
```
data['nuevo1'] = data['ano_start_process']
data['nuevo2'] = data['mes_start_process']
data['nuevo3'] = data['numdia_start_process']
data['nuevo4'] = data['dia_start_process']
data = data.set_index(['nuevo1','nuevo2','nuevo3','nuevo4'])
data['id']= data.index
```

In [51]:
```
data = data.merge(data_aux, left_on='id',right_on='id',how='inner')
```

In [52]:
```
data = data.sort_values('id')
```

In [53]:
```
data = data.drop(['id','task_start_time','task_assignment_time','process_star
t_time','ano_start_task','ano_start_process'],axis=1)
```

HACEMOS UN DESCRIBE E INFO DE LA TABLA FINAL

In [54]:
```
data.describe()
```

Out[54]:

	pv_break_minutes	net_lag_duration_minutes	net_work_duration_minutes	task_id
count	2848.000000	2848.000000	2848.000000	2.848000e+03
mean	0.433287	164.885534	28.048104	2.659052e+08
std	3.956087	115.443937	16.004021	2.200259e+08
min	0.000000	1.000000	1.000000	5.000000e+00
25%	0.000000	85.000000	19.000000	2.178700e+04
50%	0.000000	164.000000	27.000000	2.684487e+08
75%	0.000000	228.000000	37.000000	5.368778e+08
max	59.000000	916.000000	102.000000	5.369052e+08

In [55]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2848 entries, 2393 to 2366
Data columns (total 29 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   pv_operacion               2848 non-null   object 
 1   pv_tipo_operacion_name     2848 non-null   object 
 2   pv_break_minutes           2848 non-null   float64
 3   pv_task_action_name        2848 non-null   object 
 4   task_name                  2848 non-null   object 
 5   task_owner                 2848 non-null   object 
 6   net_lag_duration_minutes   2848 non-null   float64
 7   net_work_duration_minutes  2848 non-null   float64
 8   task_id                    2848 non-null   int64  
 9   user_rol                   2848 non-null   object 
 10  tipooperacion              2848 non-null   float64
 11  estado                     2848 non-null   int64  
 12  prioridad                  2848 non-null   int64  
 13  sla                        2848 non-null   int64  
 14  activo                     2848 non-null   int64  
 15  tipousuario                2848 non-null   int32  
 16  igualcerradopor            2848 non-null   int64  
 17  dia_start_task             2848 non-null   object 
 18  mes_start_task             2848 non-null   object 
 19  hora_start_task            2848 non-null   object 
 20  numdia_start_task          2848 non-null   object 
 21  dia_start_process          2848 non-null   object 
 22  mes_start_process          2848 non-null   object 
 23  hora_start_process         2848 non-null   object 
 24  numdia_start_process       2848 non-null   object 
 25  total_minutos              2848 non-null   float64
 26  cont1                      2848 non-null   int64  
 27  cont2                      2848 non-null   int64  
 28  cont                       2848 non-null   int64  
dtypes: float64(5), int32(1), int64(9), object(14)
memory usage: 656.4+ KB
```

LO PASAMOS A .CSV

In [56]:
```
data.to_csv('datosDepurados.csv',encoding='latin1')
```

In [1]:

```
import pandas as pd
import numpy as np
import random
import math
from random import sample
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVR
from sklearn import model_selection
from sklearn.utils import class_weight
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt
import xgboost as xgb
from numpy import array
from sklearn.metrics import classification_report
from urllib.request import urlretrieve
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.model_selection import StratifiedKFold
from scipy.stats import uniform, randint
from sklearn.metrics import auc, accuracy_score, confusion_matrix, mean_absol
ute_error
from sklearn.model_selection import cross_val_score, KFold
from xgboost import plot_importance
from matplotlib import pyplot
from xgboost import plot_tree
from xgboost.sklearn import XGBRegressor
from numpy import mean
from numpy import std
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score
import lightgbm as lgb
import shap
from sklearn.ensemble import RandomForestRegressor
from lightgbm.sklearn import LGBMRegressor
import multiprocessing
from sklearn.model_selection import RepeatedKFold
from sklearn.inspection import permutation_importance
from sklearn import metrics
from sklearn.neighbors import KNeighborsRegressor
from sklearn import neighbors
from sklearn import model_selection
from sklearn.ensemble import VotingRegressor
```

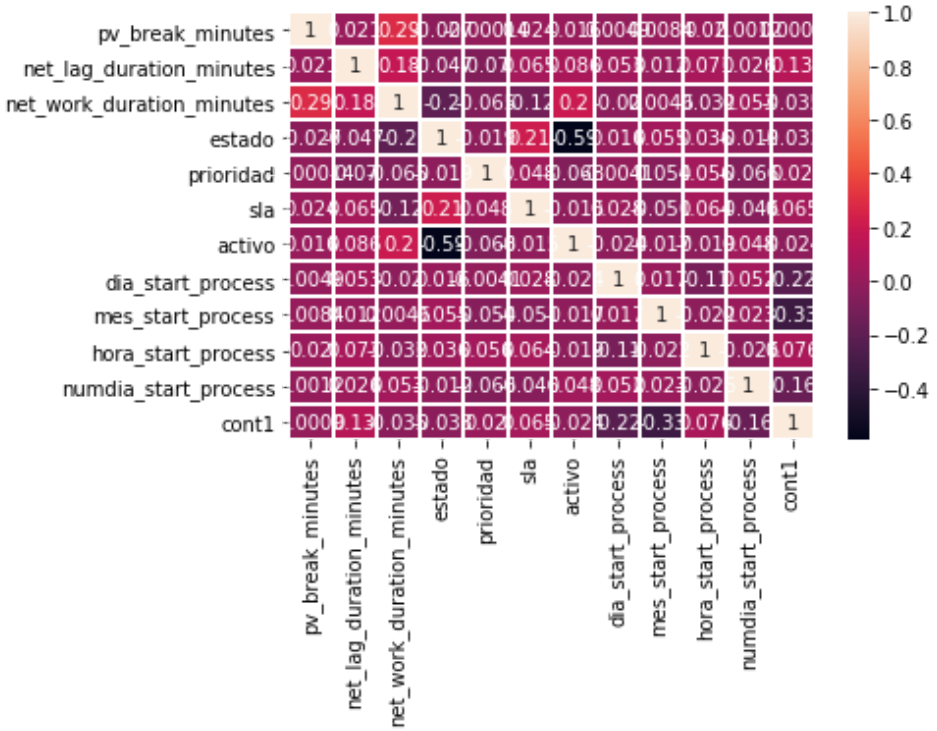
Leemos los datos y preparamos los datos para los modelos

In [2]:
```
data = pd.read_csv("./datosDepurados.csv",encoding='latin1',index_col= 0)
data = data.drop(['pv_operacion','tipooperacion','task_id','cont','cont2','to
tal_minutos'],axis=1)

datalag = data.drop(['dia_start_task','hora_start_task','mes_start_task','num
dia_start_task','pv_tipo_operacion_name','user_rol','task_owner','tipousuari
o','igualcerradopor'],axis=1)
datawork = data.drop(['dia_start_process','hora_start_process','mes_start_pro
cess','numdia_start_process','pv_tipo_operacion_name','user_rol'],axis=1)
databreak = datawork

datalag = datalag.loc[datalag['net_lag_duration_minutes'] < np.quantile(datal
ag['net_lag_duration_minutes'], 0.95),:]
datawork = datawork.loc[datawork['net_work_duration_minutes'] < np.quantile(d
atawork['net_work_duration_minutes'], 0.95),:]
#databreak = databreak.loc[databreak['pv_break_minutes'] < np.quantile(databr
eak['pv_break_minutes'], 0.95),:]
```

Vemos la correlación entre variables

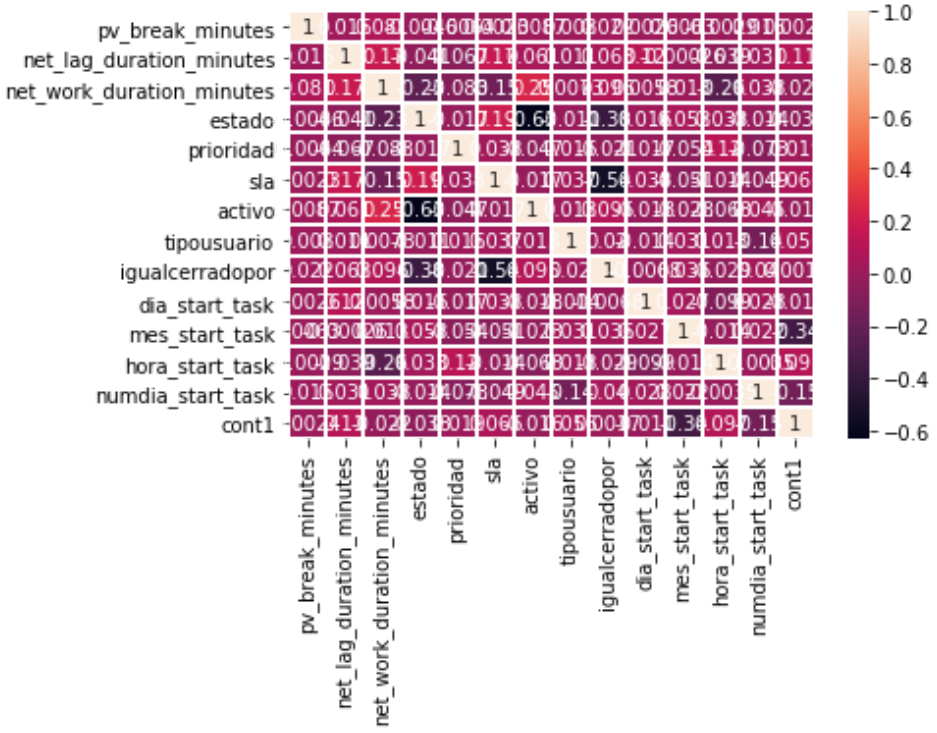
In [3]:
```
sns.heatmap(datalag.corr(),annot=True,lw=1)
```

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x161775745c8>



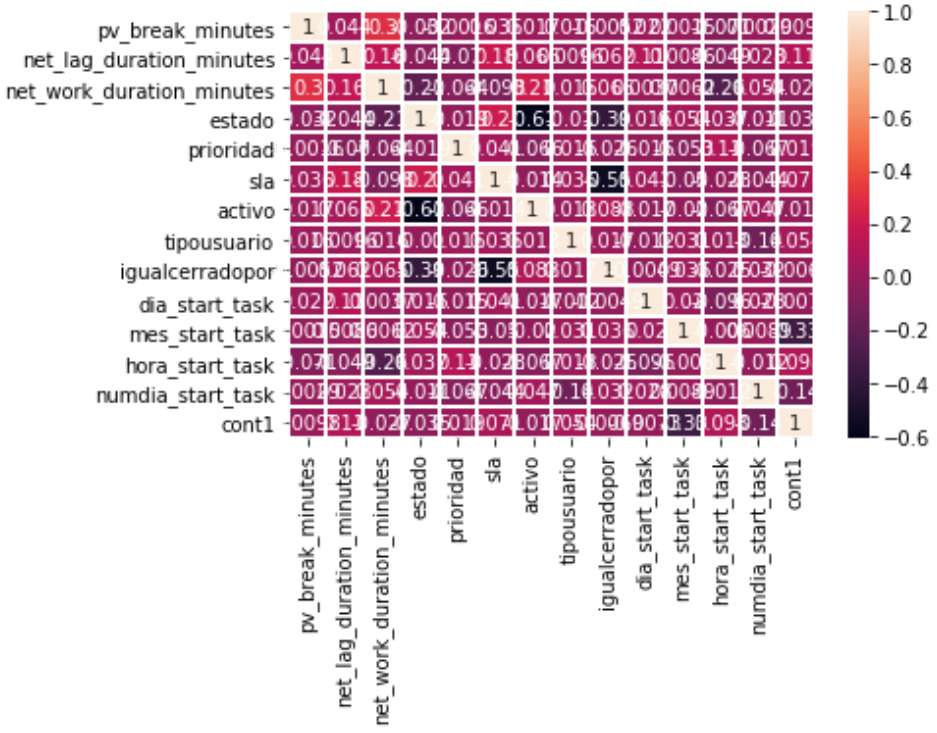
In [4]:
```
sns.heatmap(datawork.corr(),annot=True,lw=1)
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x161778749c8>



In [5]:
```
sns.heatmap(databreak.corr(),annot=True,lw=1)
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x16177abe4c8>



Funciones auxiliares

In [6]:

```
def modelfit(dataset,alg,X_train,y_train, dataXGB_Lag_train_mat,X_test,y_test
, useTrainCV=True, cv_folds=5):
    if useTrainCV:
        xgb_param = alg.get_xgb_params()
        cvresult = xgb.cv(xgb_param, dataXGB_Lag_train_mat, num_boost_round=a
lg.get_params()['n_estimators'], nfold=cv_folds,
            metrics={'mae'}, early_stopping_rounds=10)
        alg.set_params(n_estimators=cvresult.shape[0])

    clf = alg.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    predicted_y_corregido = [0 if i < 0 else i for i in y_pred] # sustitución
por 0
    dataset['XGBoost'] = predicted_y_corregido
    print('MAE')
    print(mean_absolute_error(y_test, predicted_y_corregido))
    return mean_absolute_error(y_test, predicted_y_corregido)

def str_a_num(df):
    for col in df:
        original = df[col].unique()
        reemplazo = list(range(len(original)))
        mapa = dict(zip(original, reemplazo))
        df[col] = df[col].replace(mapa)
    return(df)

def evaluar(nombre,dataset,model,X_test,y_test):
    predicciones = model.predict(X = X_test)
    predicciones = [0 if i < 0 else i for i in predicciones] # sustitución po
r 0
    mae = mean_absolute_error(
            y_true  = y_test,
            y_pred  = predicciones
           )
    print(f"El error (mae) de test es: {mae}")
    dataset[nombre]= predicciones
    return mae

def repRF(modelo_final,X_train,y_train):
    importancia = permutation_importance(
                estimator    = modelo_final,
                X            = X_train,
                y            = y_train,
                n_repeats    = 5,
                scoring      = 'neg_mean_absolute_error',
                n_jobs       = multiprocessing.cpu_count() - 1,
                random_state = 123
             )

    # Se almacenan los resultados (media y desviación) en un dataframe
    df_importancia = pd.DataFrame(
                        {k: importancia[k] for k in ['importances_mean', 'imp
ortances_std']}
                    )
    df_importancia['feature'] = X_train.columns

    # Gráfico
    fig, ax = plt.subplots(figsize=(5, 6))
    df_importancia = df_importancia.sort_values('importances_mean', ascending
=True)
    ax.barh(
        df_importancia['feature'],
```

```
        df_importancia['importances_mean'],
        xerr=df_importancia['importances_std'],
        align='center',
        alpha=0
    )
    ax.plot(
        df_importancia['importances_mean'],
        df_importancia['feature'],
        marker="D",
        linestyle="",
        alpha=0.8,
        color="r"
    )
    ax.set_title('Importancia de los predictores (train)')
    ax.set_xlabel('Incremento del error tras la permutación');
```

División para modelo XGBOOST

In [7]:
```
dataXGB_Lag = datalag
dataXGB_Work = datawork
dataXGB_Break = databreak

### Conversión a numérico

dataXGB_Lag = str_a_num(dataXGB_Lag)
dataXGB_Work = str_a_num(dataXGB_Work)
dataXGB_Break = str_a_num(dataXGB_Break)

### Sets de entrenamiento y prueba

dataXGB_Lag_train, dataXGB_Lag_test = train_test_split(dataXGB_Lag, test_size
=.2, random_state=1998)
dataXGB_Work_train, dataXGB_Work_test = train_test_split(dataXGB_Work, test_s
ize=.2, random_state=1998)
dataXGB_Break_train, dataXGB_Break_test = train_test_split(dataXGB_Break, tes
t_size=.2, random_state=1998)

### Convertir a DMatrix

dataXGB_Lag_train_mat = xgb.DMatrix(dataXGB_Lag_train.drop(['net_work_duratio
n_minutes','pv_break_minutes',"net_lag_duration_minutes"], 1), label=dataXGB_
Lag_train["net_lag_duration_minutes"])
dataXGB_Lag_test_mat = xgb.DMatrix(dataXGB_Lag_test.drop(['net_work_duration_
minutes','pv_break_minutes',"net_lag_duration_minutes"], 1), label=dataXGB_La
g_test["net_lag_duration_minutes"])

dataXGB_Work_train_mat = xgb.DMatrix(dataXGB_Work_train.drop(['net_lag_durati
on_minutes','pv_break_minutes','tipousuario',"net_work_duration_minutes"], 1
), label=dataXGB_Work_train["net_work_duration_minutes"])
dataXGB_Work_test_mat = xgb.DMatrix(dataXGB_Work_test.drop(['net_lag_duration
_minutes','pv_break_minutes','tipousuario',"net_work_duration_minutes"], 1),
label=dataXGB_Work_test["net_work_duration_minutes"])

dataXGB_Break_train_mat = xgb.DMatrix(dataXGB_Break_train.drop(['net_lag_dura
tion_minutes','net_work_duration_minutes','tipousuario',"pv_break_minutes"],
1), label=dataXGB_Break_train["pv_break_minutes"])
dataXGB_Break_test_mat = xgb.DMatrix(dataXGB_Break_test.drop(['net_lag_durati
on_minutes','net_work_duration_minutes','tipousuario',"pv_break_minutes"], 1
), label=dataXGB_Break_test["pv_break_minutes"])
```

MODELOS para Lag

In [8]:
```
datalag1 = datalag.drop(['net_work_duration_minutes','pv_break_minutes'],1)
dataejemplo = datalag1.drop(['net_lag_duration_minutes'],1)

X = pd.get_dummies(data=dataejemplo, drop_first=False)
Y = datalag1['net_lag_duration_minutes']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rand
om_state=101)
y_test_Lag = y_test.to_frame()
```

LR

In [9]:
```
modelo = LinearRegression()
modelo.fit(X = X_train, y = y_train)
```

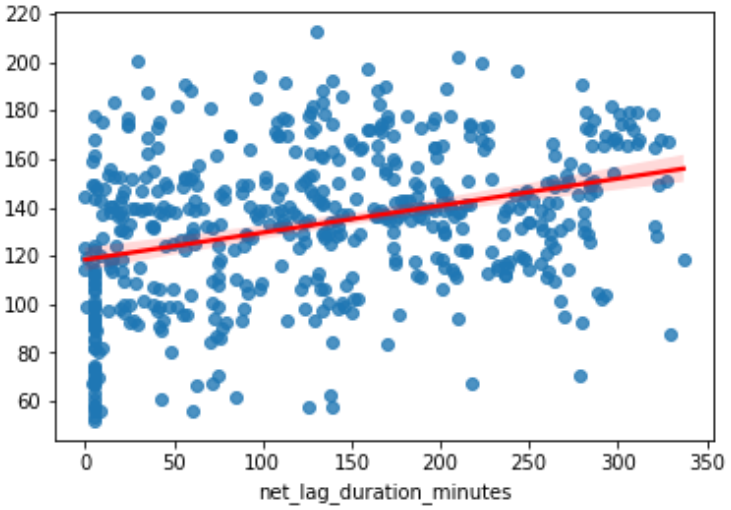
Out[9]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [10]:
```
maeLR_Lag = evaluar('LR',y_test_Lag, modelo,X_test,y_test)
```

```
El error (mae) de test es: 75.76881129541366
```

In [11]:
```
predicciones = modelo.predict(X = X_test)
predicciones = [0 if i < 0 else i for i in predicciones]
sns.regplot(y_test,predicciones,line_kws={"color": "red"})
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16177d7c548>
```



Random Forest

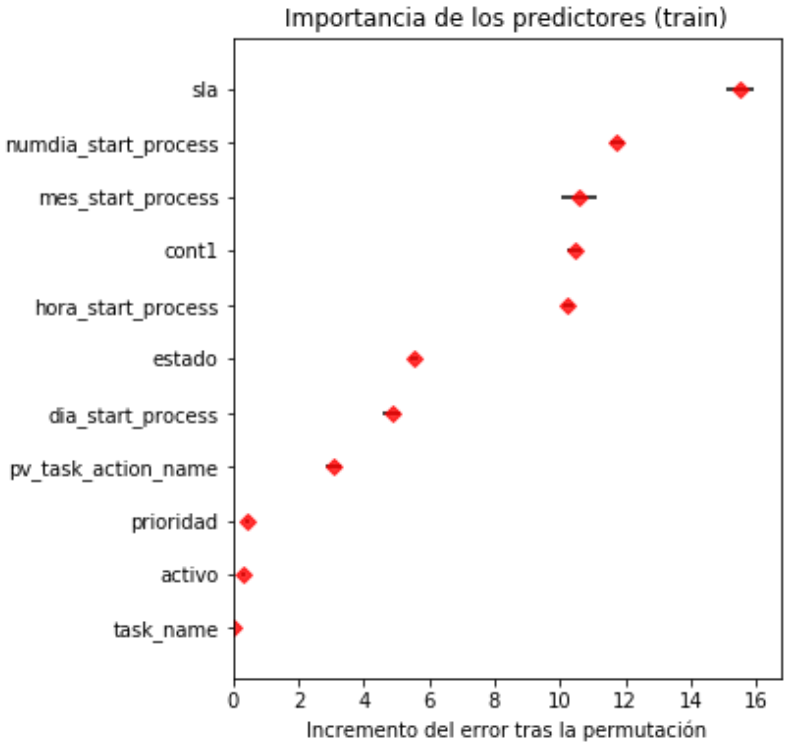
In [12]:
```
# Grid de hiperparámetros evaluados
# ==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }

# Búsqueda por grid search con validación cruzada
# ==============================================================================
grid = GridSearchCV(
        estimator  = RandomForestRegressor(random_state = 1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1998
),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train, y = y_train)
```

Out[12]:
```
GridSearchCV(cv=RepeatedKFold(n_repeats=3, n_splits=5, random_state=1998),
             error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=Non
e,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100, n_jobs=None,
                                             oob_score=False, random_state=1
998,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=7,
             param_grid={'max_depth': [None, 3, 10, 20],
                         'max_features': [5, 7, 9], 'n_estimators': [150]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_absolute_error', verbose=0)
```

In [13]:
```
modelo_final = grid.best_estimator_
maeRF_Lag = evaluar('RF',y_test_Lag, modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 72.79997116420208
```

In [14]:
```
repRF(modelo_final,X_train,y_train)
```



KNN

In [15]:
```
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(X_train)
x_train_scaled = pd.DataFrame(x_train_scaled)

x_test_scaled = scaler.fit_transform(X_test)
x_test_scaled = pd.DataFrame(x_test_scaled)
```

In [16]:
```
grid_params = {
    'n_neighbors': [3,5,11,19],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
}
gs = GridSearchCV(
    KNeighborsRegressor(),
    grid_params,
    verbose = 1,
    cv = 5,
    n_jobs = -1
    )

gs_results = gs.fit(x_train_scaled, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  80 out of  80 | elapsed:    0.6s finished
```

In [17]:
```
modelo_final = gs_results.best_estimator_
maeKNN_Lag = evaluar('KNN',y_test_Lag,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 83.43467263352466
```

SVM Linear

In [18]:
```
modelo = SVR(kernel = 'linear')
modelo.fit(X_train, y_train)
```

Out[18]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [19]:
```
maeSVMLinear_Lag = evaluar('SVM_Lineal',y_test_Lag,modelo,X_test,y_test)
```

```
El error (mae) de test es: 74.70939333643396
```

SVM Radial

In [20]:
```
modelo  = SVR(kernel= "rbf", gamma='scale')
modelo.fit(X_train, y_train)
```

Out[20]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [21]:
```
maeSVMRadial_Lag = evaluar('SVM_Radial',y_test_Lag,modelo,X_test,y_test)
```

```
El error (mae) de test es: 78.00439145837338
```

XGBOOST

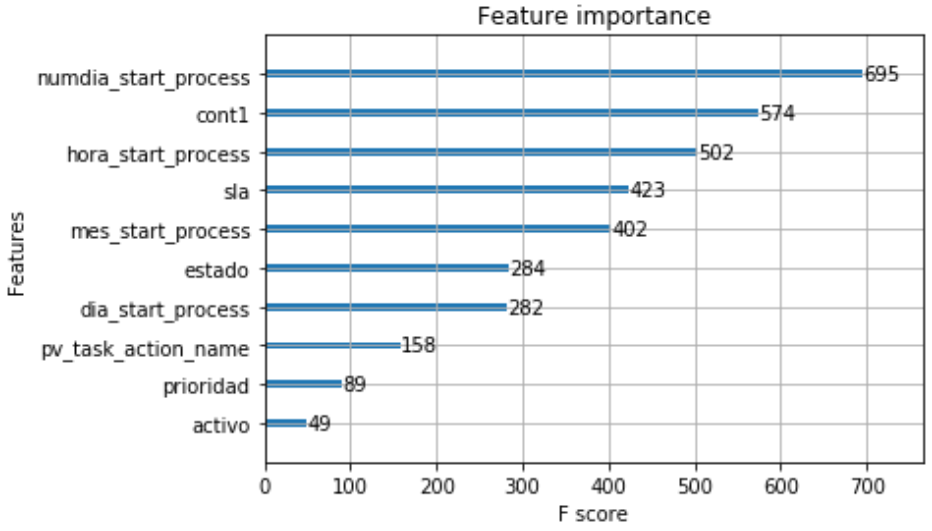
In [22]:
```
random.seed(1998)
```

In [23]:
```
xgb1 = XGBRegressor(
 objective='reg:squarederror',
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=3,
 min_child_weight=4,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.7,
 reg_alpha=1,
 nthread=4,
 scale_pos_weight=1,
 eval_metric= "mae",
 seed=27)
```

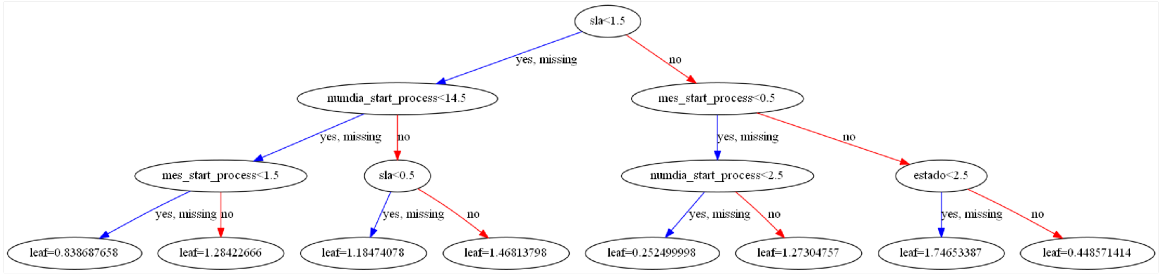
In [24]:
```
maeXGB_Lag = modelfit(y_test_Lag,xgb1,X_train,y_train, dataXGB_Lag_train_mat,
X_test,y_test)
```

```
MAE
72.89671439384136
```

In [25]:
```
plot_importance(xgb1)
pyplot.show()
```



In [26]:
```
fig, ax = plt.subplots(figsize=(50, 50))
plot_tree(xgb1, ax=ax)
plt.show()
```



LightGBM

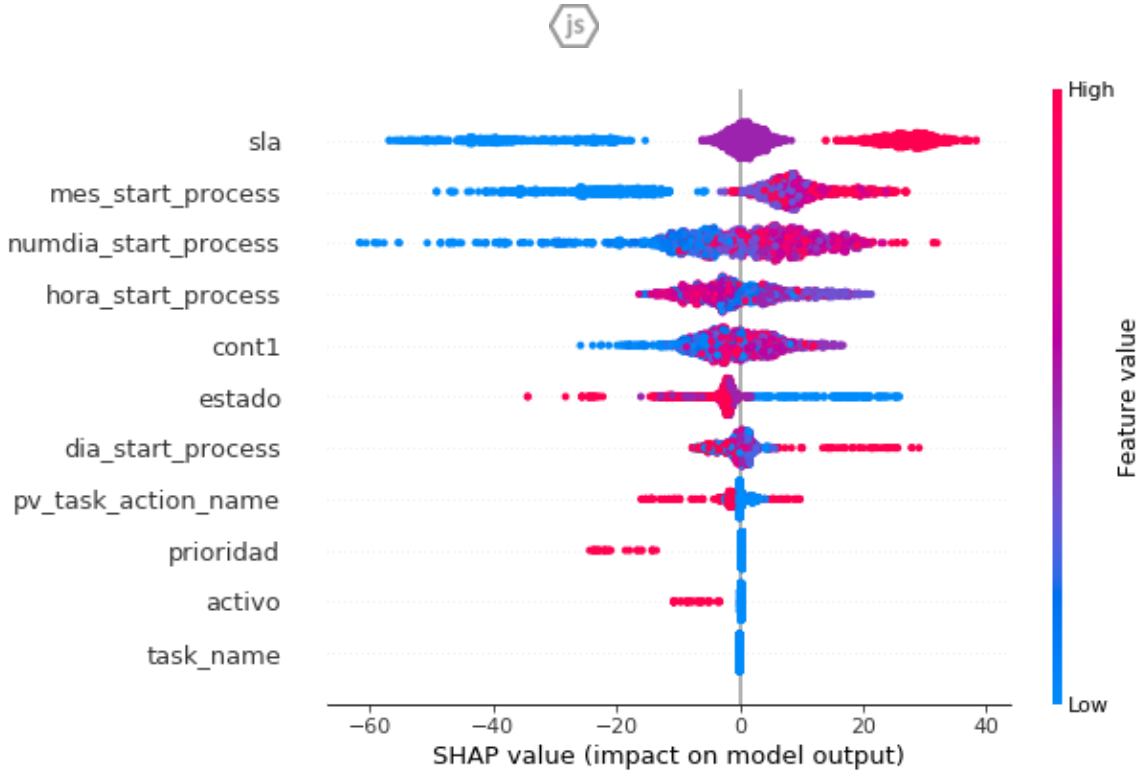
In [27]:
```
param_grid = {'n_estimators'     : [100, 500, 1000, 5000],
              'max_depth'        : [-1, 1, 3, 5, 10, 20],
              'subsample'        : [0.5, 1],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'boosting_type'    : ['gbdt']
             }
grid = GridSearchCV(
        estimator  = LGBMRegressor(random_state=1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1998
),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train, y = y_train)
```

Out[27]:
```
GridSearchCV(cv=RepeatedKFold(n_repeats=1, n_splits=5, random_state=1998),
             error_score=nan,
             estimator=LGBMRegressor(boosting_type='gbdt', class_weight=Non
e,
                                     colsample_bytree=1.0,
                                     importance_type='split', learning_rate=
0.1,
                                     max_depth=-1, min_child_samples=20,
                                     min_child_weight=0.001, min_split_gain=
0.0,
                                     n_estimators=100, n_jobs=-1, num_leaves
=31,
                                     objective=None, rand...
                                     reg_alpha=0.0, reg_lambda=0.0, silent=T
rue,
                                     subsample=1.0, subsample_for_bin=20000
0,
                                     subsample_freq=0),
             iid='deprecated', n_jobs=7,
             param_grid={'boosting_type': ['gbdt'],
                         'learning_rate': [0.001, 0.01, 0.1],
                         'max_depth': [-1, 1, 3, 5, 10, 20],
                         'n_estimators': [100, 500, 1000, 5000],
                         'subsample': [0.5, 1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_absolute_error', verbose=0)
```

In [28]:
```
modelo_final = grid.best_estimator_
maeLightGBM_Lag = evaluar('LightGBM',y_test_Lag,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 72.61633908335027
```

In [29]:
```
shap.initjs()
explainer = shap.TreeExplainer(modelo_final)
shap_vals = explainer.shap_values(X_train)
feature_names = X_train.columns.tolist()
shap.summary_plot(shap_vals, X_train, max_display=30, feature_names=feature_n
ames)
```



ENSEMBLE

In [30]:
```
# Voting Ensemble for Classification
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
# create the sub models
estimators = []
model1 = LinearRegression()
estimators.append(('LR', model1))
model2 = RandomForestRegressor()
estimators.append(('RF', model2))
model3 = KNeighborsRegressor()
estimators.append(('KNN', model3))
model4 = SVR(kernel = 'linear')
estimators.append(('SVML', model4))
model5 = SVR(kernel= "rbf", gamma='scale')
estimators.append(('SVMR', model5))
model6 = XGBRegressor()
estimators.append(('XGBoost', model6))
model7 = LGBMRegressor()
estimators.append(('LightGBM', model7))
# create the ensemble model
ensemble = VotingRegressor(estimators)

predicciones_encemble = ensemble.fit(X,Y).predict(X)
predicciones_encemble = [0 if i < 0 else i for i in predicciones_encemble] #
 sustitución por 0
mae = mean_absolute_error(
        y_true  = Y,
        y_pred  = predicciones_encemble
        )
print(f"El error (mae) de test es: {mae}")
maeEnsemble_Lag = mae
```

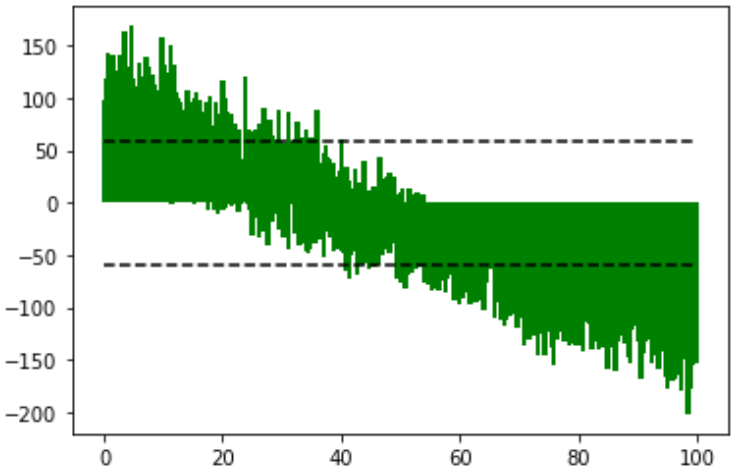
```
El error (mae) de test es: 58.90988258069065
```

In [31]:
```
LagPred_Real = Y.to_frame()
LagPred_Real['Prediccion'] = predicciones_encemble
LagPred_Real['Prediccion - Real'] = LagPred_Real['Prediccion'] - LagPred_Real
['net_lag_duration_minutes']
mediaErrLag = mean(abs(LagPred_Real['Prediccion - Real']))
```

In [32]:
```
LagPred_Real = LagPred_Real.sort_values('net_lag_duration_minutes')
LagPred_Real['porcentaje'] = LagPred_Real['net_lag_duration_minutes'].multipl
y(100).div(LagPred_Real.iloc[-1,0])
```

In [33]:
```
fig, ax = plt.subplots()
ax.bar(LagPred_Real['porcentaje'], LagPred_Real['Prediccion - Real'],color=
"g")
ax.plot([0, 100], [mediaErrLag, mediaErrLag], "k--")
ax.plot([0, 100], [-mediaErrLag, -mediaErrLag], "k--")
```

Out[33]:
```
[<matplotlib.lines.Line2D at 0x16100123c88>]
```



MODELOS para Work

In [34]:
```
datawork1 = datawork.drop(['net_lag_duration_minutes','pv_break_minutes','tip
ousuario'],1)
dataejemplowork = datawork1.drop(['net_work_duration_minutes'],1)

X = pd.get_dummies(data=dataejemplowork, drop_first=False)
Y = datawork1['net_work_duration_minutes']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rand
om_state=101)
y_test_Work = y_test.to_frame()
```

LR

In [35]:
```
modelo = LinearRegression()
modelo.fit(X = X_train, y = y_train)
```

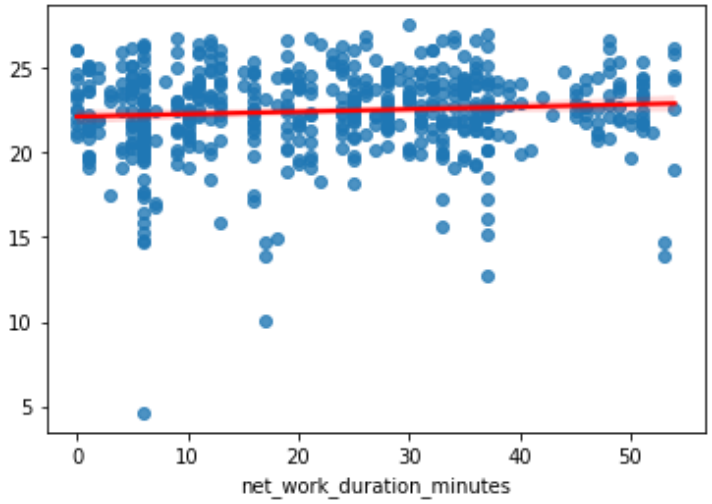
Out[35]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [36]:
```
maeLR_Work = evaluar('LR',y_test_Work,modelo,X_test,y_test)
```

```
El error (mae) de test es: 12.703493378982763
```

In [37]:
```
predicciones = modelo.predict(X = X_test)
predicciones = [0 if i < 0 else i for i in predicciones]
sns.regplot(y_test,predicciones,line_kws={"color": "red"})
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x16104c18288>



Random Forest

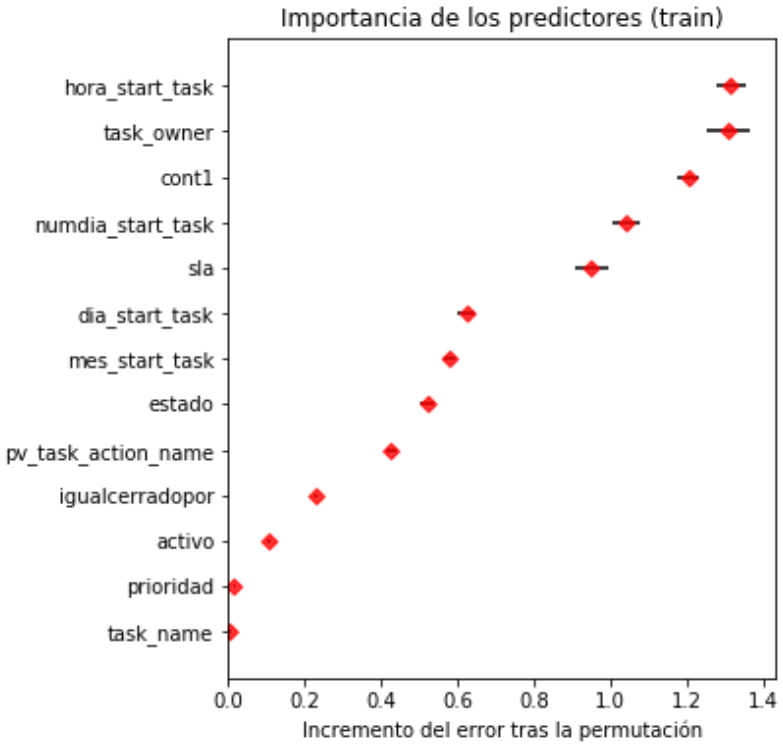
In [38]:
```
# Grid de hiperparámetros evaluados
# ==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }

# Búsqueda por grid search con validación cruzada
# ==============================================================================
grid = GridSearchCV(
        estimator  = RandomForestRegressor(random_state = 1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1998
),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train, y = y_train)
```

Out[38]:
```
GridSearchCV(cv=RepeatedKFold(n_repeats=3, n_splits=5, random_state=1998),
             error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=Non
e,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100, n_jobs=None,
                                             oob_score=False, random_state=1
998,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=7,
             param_grid={'max_depth': [None, 3, 10, 20],
                         'max_features': [5, 7, 9], 'n_estimators': [150]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_absolute_error', verbose=0)
```

In [39]:
```
modelo_final = grid.best_estimator_
maeRF_Work = evaluar('RF',y_test_Work,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 12.070338374981937
```

In [40]:
```
repRF(modelo_final,X_train,y_train)
```



KNN

In [41]:
```
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(X_train)
x_train_scaled = pd.DataFrame(x_train_scaled)

x_test_scaled = scaler.fit_transform(X_test)
x_test_scaled = pd.DataFrame(x_test_scaled)
```

In [42]:
```
grid_params = {
    'n_neighbors': [3,5,11,19],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
}
gs = GridSearchCV(
    KNeighborsRegressor(),
    grid_params,
    verbose = 1,
    cv = 5,
    n_jobs = -1
    )

gs_results = gs.fit(x_train_scaled, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  80 out of  80 | elapsed:    0.7s finished
```

In [43]:
```
modelo_final = gs_results.best_estimator_
maeKNN_Work = evaluar('KNN',y_test_Work,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 12.65006323572332
```

SVM Linear

In [44]:
```
modelo = SVR(kernel = 'linear')
modelo.fit(X_train, y_train)
```

Out[44]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [45]:
```
maeSVMLinear_Work = evaluar('SVM_Lineal',y_test_Work,modelo,X_test,y_test)
```

```
El error (mae) de test es: 12.75247788434682
```

SVM Radial

In [46]:
```
modelo = SVR(kernel= "rbf", gamma='scale')
modelo.fit(X_train, y_train)
```

Out[46]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [47]:
```
maeSVMRadial_Work = evaluar('SVM_Radial',y_test_Work,modelo,X_test,y_test)
```

```
El error (mae) de test es: 12.87000981726629
```

XGBOOST

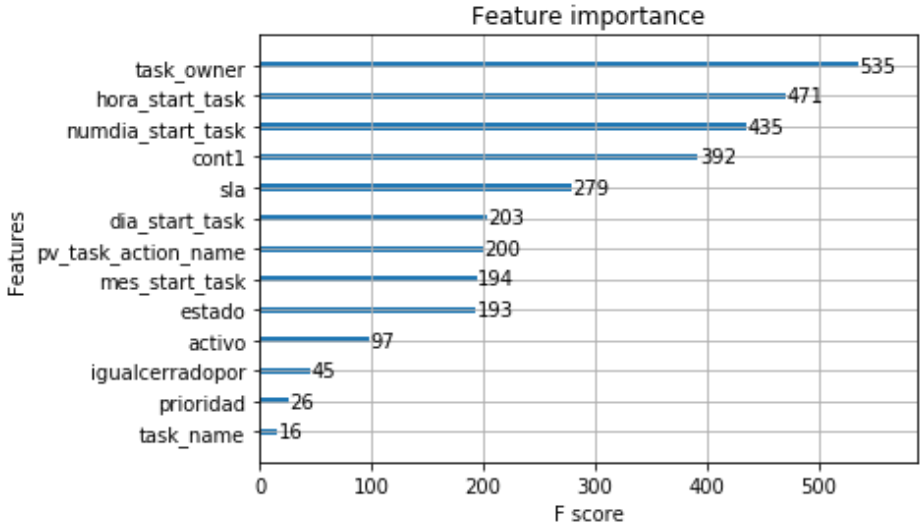
In [48]:
```
random.seed(1998)
```

In [49]:
```
xgb1 = XGBRegressor(
 objective='reg:squarederror',
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=3,
 min_child_weight=4,
 gamma=0.3,
 subsample=0.9,
 colsample_bytree=0.6,
 reg_alpha=0.01,
 nthread=4,
 scale_pos_weight=1,
 eval_metric= "mae",
 seed=27)
```

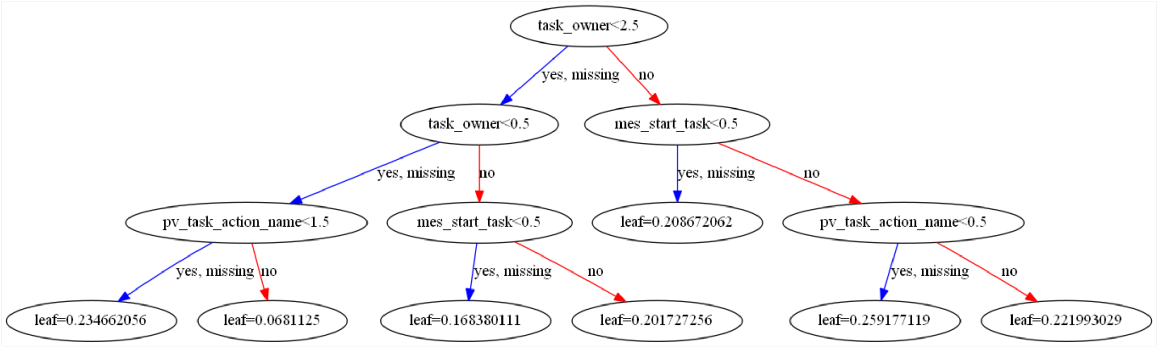
In [50]:
```
maeXGB_Work = modelfit(y_test_Work,xgb1,X_train,y_train, dataXGB_Work_train_mat,X_test,y_test)
```

```
MAE
12.139856447794521
```

In [51]:
```
plot_importance(xgb1)
pyplot.show()
```



In [52]:
```
fig, ax = plt.subplots(figsize=(50, 50))
plot_tree(xgb1, ax=ax)
plt.show()
```



LightGBM

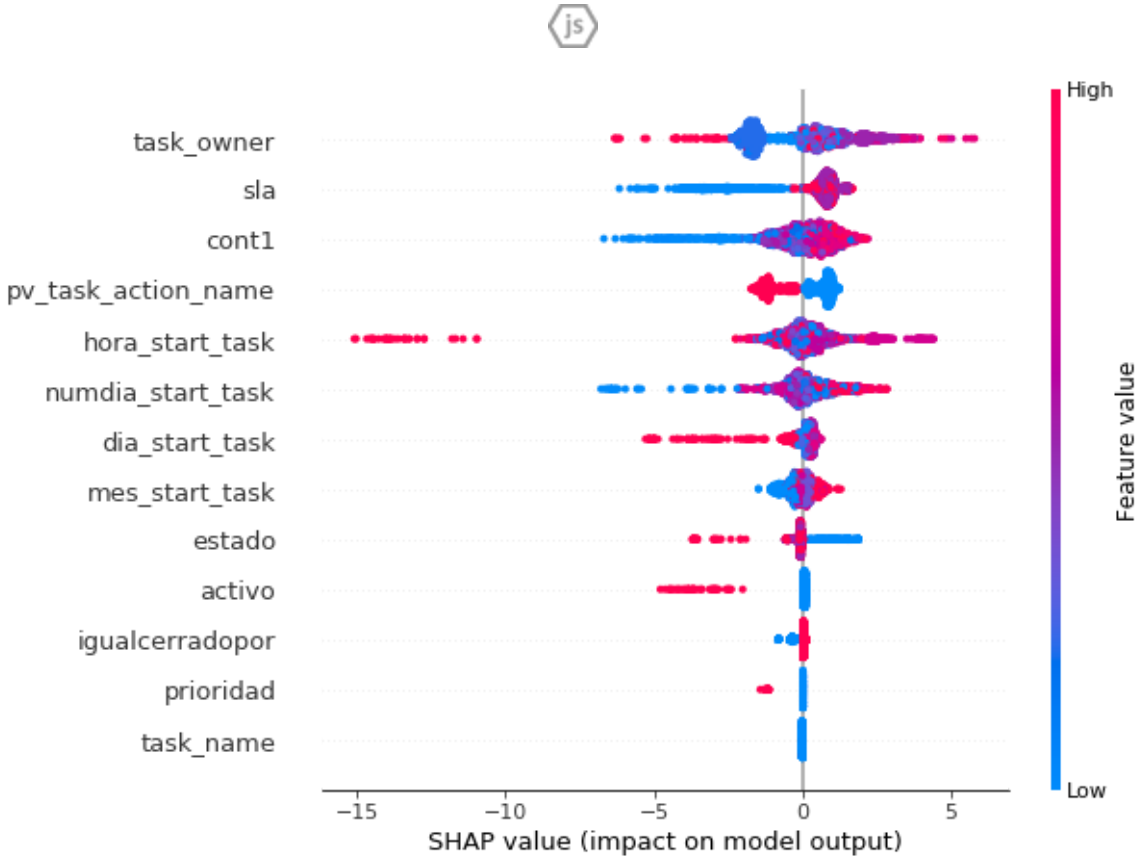
In [53]:
```
param_grid = {'n_estimators'     : [100, 500, 1000, 5000],
              'max_depth'        : [-1, 1, 3, 5, 10, 20],
              'subsample'        : [0.5, 1],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'boosting_type'    : ['gbdt']
             }
grid = GridSearchCV(
        estimator  = LGBMRegressor(random_state=1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1998
),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train, y = y_train)
```

Out[53]:
```
GridSearchCV(cv=RepeatedKFold(n_repeats=1, n_splits=5, random_state=1998),
             error_score=nan,
             estimator=LGBMRegressor(boosting_type='gbdt', class_weight=Non
e,
                                     colsample_bytree=1.0,
                                     importance_type='split', learning_rate=
0.1,
                                     max_depth=-1, min_child_samples=20,
                                     min_child_weight=0.001, min_split_gain=
0.0,
                                     n_estimators=100, n_jobs=-1, num_leaves
=31,
                                     objective=None, rand...
                                     reg_alpha=0.0, reg_lambda=0.0, silent=T
rue,
                                     subsample=1.0, subsample_for_bin=20000
0,
                                     subsample_freq=0),
             iid='deprecated', n_jobs=7,
             param_grid={'boosting_type': ['gbdt'],
                         'learning_rate': [0.001, 0.01, 0.1],
                         'max_depth': [-1, 1, 3, 5, 10, 20],
                         'n_estimators': [100, 500, 1000, 5000],
                         'subsample': [0.5, 1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_absolute_error', verbose=0)
```

In [54]:
```
modelo_final = grid.best_estimator_
maeLightGBM_Work = evaluar('LightGBM',y_test_Work,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 12.059533517936162
```

In [55]:
```
shap.initjs()
explainer = shap.TreeExplainer(modelo_final)
shap_vals = explainer.shap_values(X_train)
feature_names = X_train.columns.tolist()
shap.summary_plot(shap_vals, X_train, max_display=30, feature_names=feature_n
ames)
```



ENSEMBLE

In [56]:
```
# Voting Ensemble for Classification
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
# create the sub models
estimators = []
model1 = LinearRegression()
estimators.append(('LR', model1))
model2 = RandomForestRegressor()
estimators.append(('RF', model2))
model3 = KNeighborsRegressor()
estimators.append(('KNN', model3))
model4 = SVR(kernel = 'linear')
estimators.append(('SVML', model4))
model5 = SVR(kernel= "rbf", gamma='scale')
estimators.append(('SVMR', model5))
model6 = XGBRegressor()
estimators.append(('XGBoost', model6))
model7 = LGBMRegressor()
estimators.append(('LightGBM', model7))
# create the ensemble model
ensemble = VotingRegressor(estimators)

predicciones_encemble = ensemble.fit(X,Y).predict(X)
predicciones_encemble = [0 if i < 0 else i for i in predicciones_encemble] #
 sustitución por 0
mae = mean_absolute_error(
        y_true  = Y,
        y_pred  = predicciones_encemble
        )
print(f"El error (mae) de test es: {mae}")
maeEnsemble_Work = mae
```

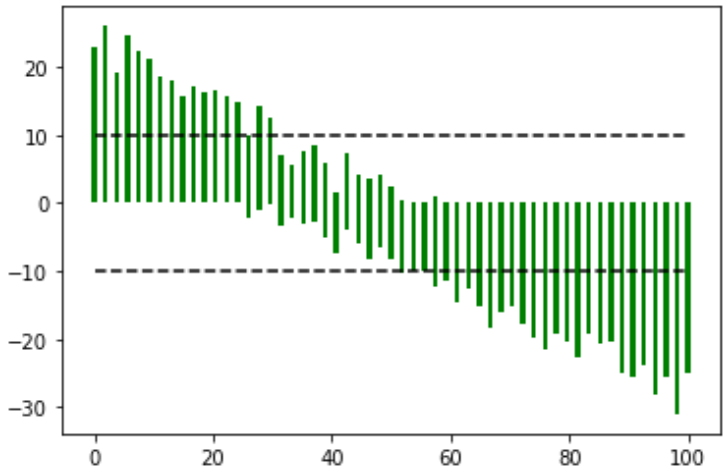
```
El error (mae) de test es: 9.862600568203732
```

In [57]:
```
WorkPred_Real = Y.to_frame()
WorkPred_Real['Prediccion'] = predicciones_encemble
WorkPred_Real['Prediccion - Real'] = WorkPred_Real['Prediccion'] - WorkPred_R
eal['net_work_duration_minutes']
mediaErrWork = mean(abs(WorkPred_Real['Prediccion - Real']))
```

In [59]:
```
WorkPred_Real = WorkPred_Real.sort_values('net_work_duration_minutes')
WorkPred_Real['porcentaje'] = WorkPred_Real['net_work_duration_minutes'].mult
iply(100).div(WorkPred_Real.iloc[-1,0])
```

In [60]:
```
fig, ax = plt.subplots()
ax.bar(WorkPred_Real['porcentaje'], WorkPred_Real['Prediccion - Real'],color=
"g")
ax.plot([0, 100], [mediaErrWork, mediaErrWork], "k--")
ax.plot([0, 100], [-mediaErrWork, -mediaErrWork], "k--")
```

Out[60]:
```
[<matplotlib.lines.Line2D at 0x16107c66cc8>]
```



MODELOS para Break

In [61]:
```
databreak1 = databreak.drop(['net_lag_duration_minutes','net_work_duration_mi
nutes','tipousuario'],1)
dataejemplobreak = databreak1.drop(['pv_break_minutes'],1)

X = pd.get_dummies(data=dataejemplobreak, drop_first=False)
Y = databreak1['pv_break_minutes']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, rand
om_state=101)
y_test_Break = y_test.to_frame()
```

LR

In [62]:
```
modelo = LinearRegression()
modelo.fit(X = X_train, y = y_train)
```

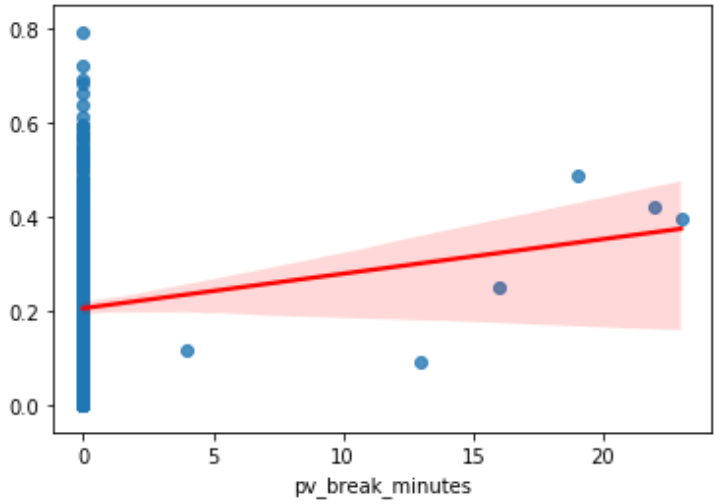
Out[62]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [63]:
```
maeLR_Break = evaluar('LR',y_test_Break,modelo,X_test,y_test)
```

```
El error (mae) de test es: 0.3719482933256809
```

In [64]:
```
predicciones = modelo.predict(X = X_test)
predicciones = [0 if i < 0 else i for i in predicciones]
sns.regplot(y_test,predicciones,line_kws={"color": "red"})
```

Out[64]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x16107bec708>
```



Random Forest

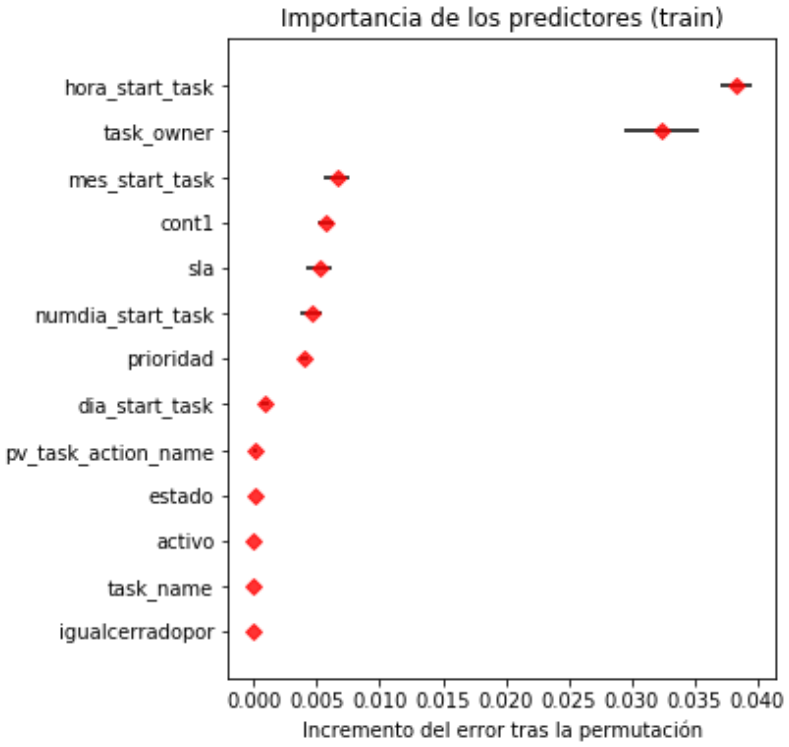
In [65]:
```
# Grid de hiperparámetros evaluados
# ==============================================================================
param_grid = {'n_estimators': [150],
              'max_features': [5, 7, 9],
              'max_depth'   : [None, 3, 10, 20]
             }

# Búsqueda por grid search con validación cruzada
# ==============================================================================
grid = GridSearchCV(
        estimator  = RandomForestRegressor(random_state = 1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1998
),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )

grid.fit(X = X_train, y = y_train)
```

Out[65]:
```
GridSearchCV(cv=RepeatedKFold(n_repeats=3, n_splits=5, random_state=1998),
             error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True, ccp_alpha=0.0,
                                             criterion='mse', max_depth=Non
e,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100, n_jobs=None,
                                             oob_score=False, random_state=1
998,
                                             verbose=0, warm_start=False),
             iid='deprecated', n_jobs=7,
             param_grid={'max_depth': [None, 3, 10, 20],
                         'max_features': [5, 7, 9], 'n_estimators': [150]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_absolute_error', verbose=0)
```

In [66]:
```
modelo_final = grid.best_estimator_
maeRF_Break = evaluar('RF',y_test_Break,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 0.3483682463748386
```

In [67]:
```
repRF(modelo_final,X_train,y_train)
```



KNN

In [68]:
```
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(X_train)
x_train_scaled = pd.DataFrame(x_train_scaled)

x_test_scaled = scaler.fit_transform(X_test)
x_test_scaled = pd.DataFrame(x_test_scaled)
```

In [69]:
```
grid_params = {
    'n_neighbors': [3,5,11,19],
    'weights': ['uniform','distance'],
    'metric': ['euclidean','manhattan']
}
gs = GridSearchCV(
    KNeighborsRegressor(),
    grid_params,
    verbose = 1,
    cv = 5,
    n_jobs = -1
    )

gs_results = gs.fit(x_train_scaled, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  80 out of  80 | elapsed:    0.6s finished
```

In [70]:
```
modelo_final = gs_results.best_estimator_
maeKNN_Break = evaluar('KNN',y_test_Break,modelo_final,X_test,y_test)
```

```
El error (mae) de test es: 0.4505078485687904
```

SVM Linear

In [71]:
```
modelo = SVR(kernel = 'linear')
modelo.fit(X_train, y_train)
```

Out[71]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [72]:
```
maeSVMLinear_Break = evaluar('SVM_Lineal',y_test_Break,modelo,X_test,y_test)
```

```
El error (mae) de test es: 0.26791102919636706
```

SVM Radial

In [73]:
```
modelo = SVR(kernel= "rbf", gamma='scale')
modelo.fit(X_train, y_train)
```

Out[73]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [74]:
```
maeSVMRadial_Break = evaluar('SVM_Radial',y_test_Break,modelo,X_test,y_test)
```

```
El error (mae) de test es: 0.26704077539724613
```

XGBOOST

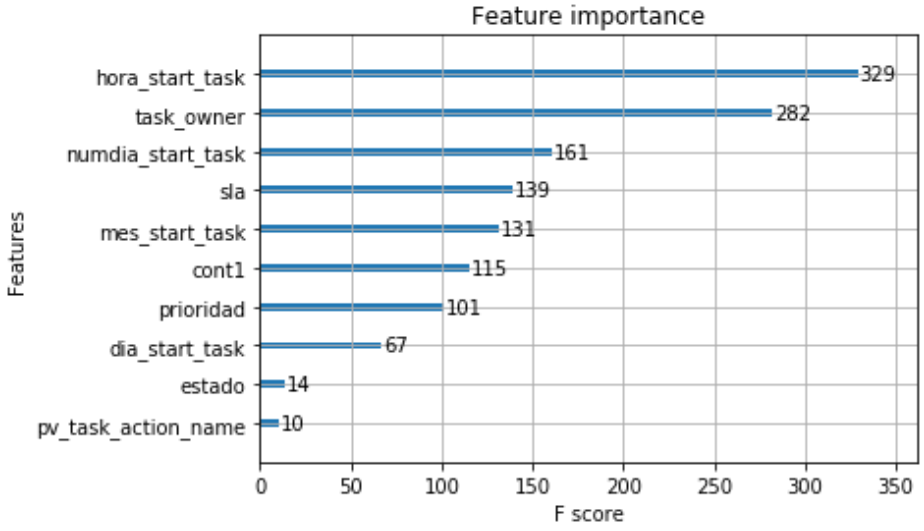
In [75]:
```
random.seed(1998)
```

In [76]:
```
xgb1 = XGBRegressor(
 objective='reg:squarederror',
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=3,
 min_child_weight=4,
 gamma=0,
 subsample=0.9,
 colsample_bytree=0.7,
 reg_alpha=100,
 nthread=4,
 scale_pos_weight=1,
 eval_metric= "mae",
 seed=27)
```

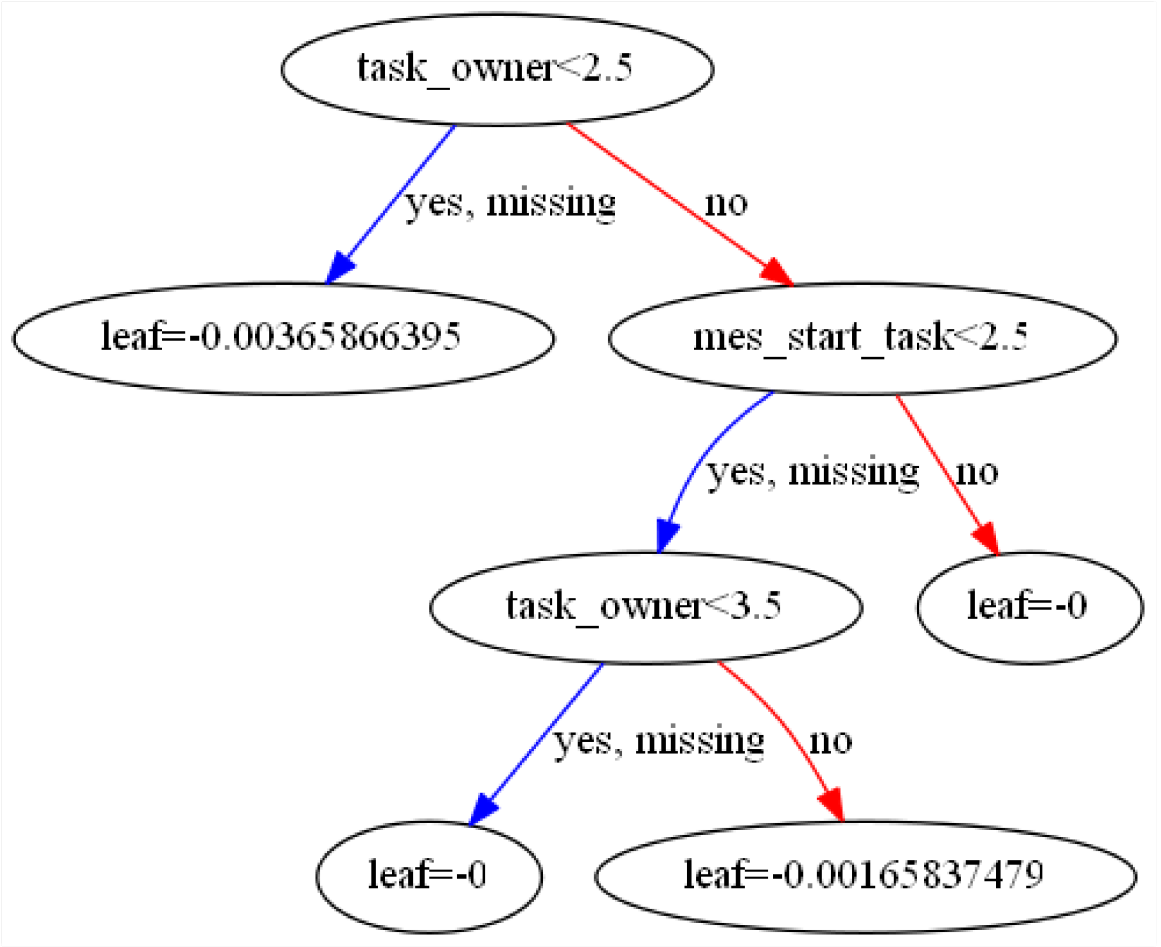
In [77]:
```
maeXGB_Break = modelfit(y_test_Break,xgb1,X_train,y_train, dataXGB_Work_train
_mat,X_test,y_test)
```

```
MAE
0.38869563299966486
```

In [78]:
```
plot_importance(xgb1)
pyplot.show()
```



In [79]:
```
fig, ax = plt.subplots(figsize=(50, 50))
plot_tree(xgb1, ax=ax)
plt.show()
```



LightGBM

In [80]:
```
param_grid = {'n_estimators'     : [100, 500, 1000, 5000],
              'max_depth'        : [-1, 1, 3, 5, 10, 20],
              'subsample'        : [0.5, 1],
              'learning_rate'    : [0.001, 0.01, 0.1],
              'boosting_type'    : ['gbdt']
             }
grid = GridSearchCV(
        estimator  = LGBMRegressor(random_state=1998),
        param_grid = param_grid,
        scoring    = 'neg_mean_absolute_error',
        n_jobs     = multiprocessing.cpu_count() - 1,
        cv         = RepeatedKFold(n_splits=5, n_repeats=1, random_state=1998
),
        refit      = True,
        verbose    = 0,
        return_train_score = True
       )
grid.fit(X = X_train, y = y_train)
```

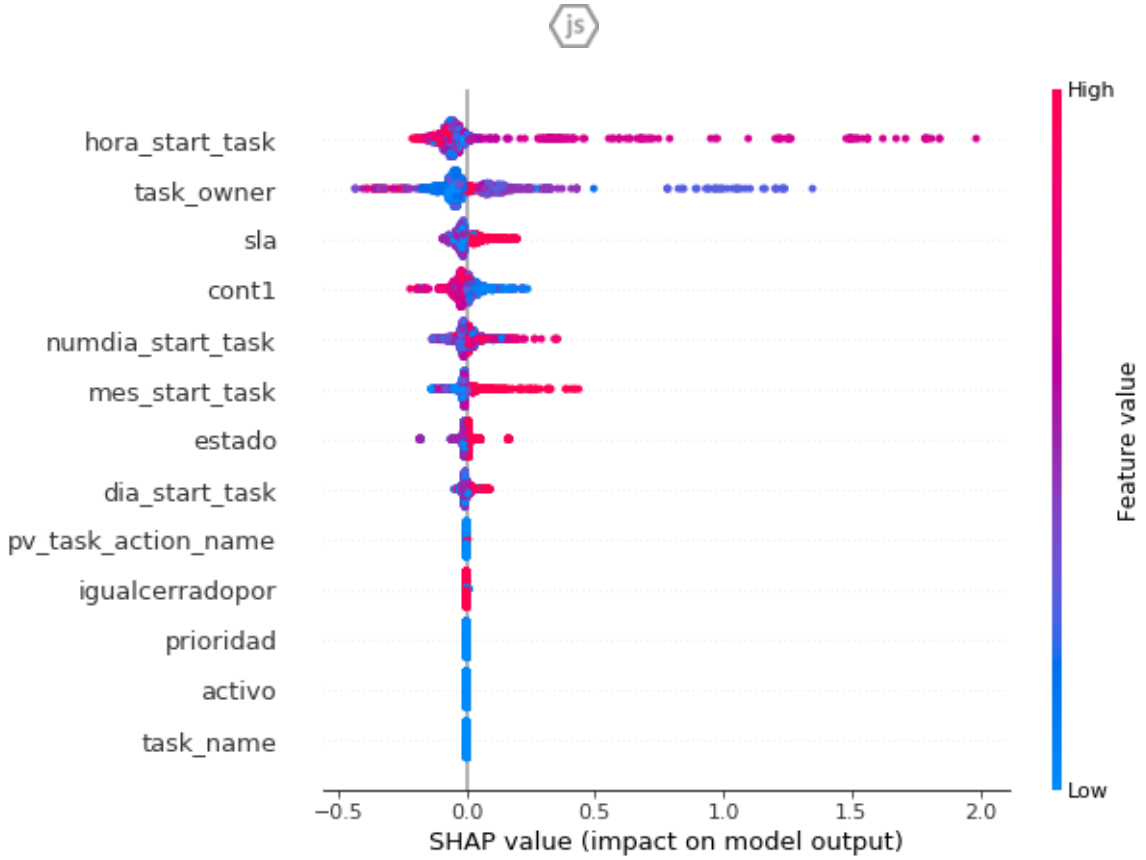
```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num
_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
```

Out[80]:
```
GridSearchCV(cv=RepeatedKFold(n_repeats=1, n_splits=5, random_state=1998),
             error_score=nan,
             estimator=LGBMRegressor(boosting_type='gbdt', class_weight=Non
e,
                                     colsample_bytree=1.0,
                                     importance_type='split', learning_rate=
0.1,
                                     max_depth=-1, min_child_samples=20,
                                     min_child_weight=0.001, min_split_gain=
0.0,
                                     n_estimators=100, n_jobs=-1, num_leaves
=31,
                                     objective=None, rand...
                                     reg_alpha=0.0, reg_lambda=0.0, silent=T
rue,
                                     subsample=1.0, subsample_for_bin=20000
0,
                                     subsample_freq=0),
             iid='deprecated', n_jobs=7,
             param_grid={'boosting_type': ['gbdt'],
                         'learning_rate': [0.001, 0.01, 0.1],
                         'max_depth': [-1, 1, 3, 5, 10, 20],
                         'n_estimators': [100, 500, 1000, 5000],
                         'subsample': [0.5, 1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_absolute_error', verbose=0)
```

In [81]:
```
modelo_final = grid.best_estimator_
maeLightGBM_Break = evaluar('LightGBM',y_test_Break,modelo_final,X_test,y_tes
t)
```

```
El error (mae) de test es: 0.33895044152864334
```

In [82]:
```
shap.initjs()
explainer = shap.TreeExplainer(modelo_final)
shap_vals = explainer.shap_values(X_train)
feature_names = X_train.columns.tolist()
shap.summary_plot(shap_vals, X_train, max_display=30, feature_names=feature_n
ames)
```



ENSEMBLE

In [83]:
```
# Voting Ensemble for Classification
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
# create the sub models
estimators = []
model1 = LinearRegression()
estimators.append(('LR', model1))
model2 = RandomForestRegressor()
estimators.append(('RF', model2))
model3 = KNeighborsRegressor()
estimators.append(('KNN', model3))
model4 = SVR(kernel = 'linear')
estimators.append(('SVML', model4))
model5 = SVR(kernel= "rbf", gamma='scale')
estimators.append(('SVMR', model5))
model6 = XGBRegressor()
estimators.append(('XGBoost', model6))
model7 = LGBMRegressor()
estimators.append(('LightGBM', model7))
# create the ensemble model
```

In [84]:
```
ensemble = VotingRegressor(estimators)
```

In [85]:
```
predicciones_encemble = ensemble.fit(X,Y).predict(X)
predicciones_encemble = [0 if i < 0 else i for i in predicciones_encemble] #
 sustitución por 0
mae = mean_absolute_error(
        y_true  = Y,
        y_pred  = predicciones_encemble
        )
maeEnsemble_Break = mae
print(f"El error (mae) de test es: {mae}")
```

```
El error (mae) de test es: 0.2521955534212511
```

ELECCIÓN DE MODELOS

In [86]:
```
modelosNombre = ['LR', 'RF', 'KNN', 'SVMLineal', 'SVMRadial', 'XGBoost', 'Lig
htGBM','Ensemble']
```

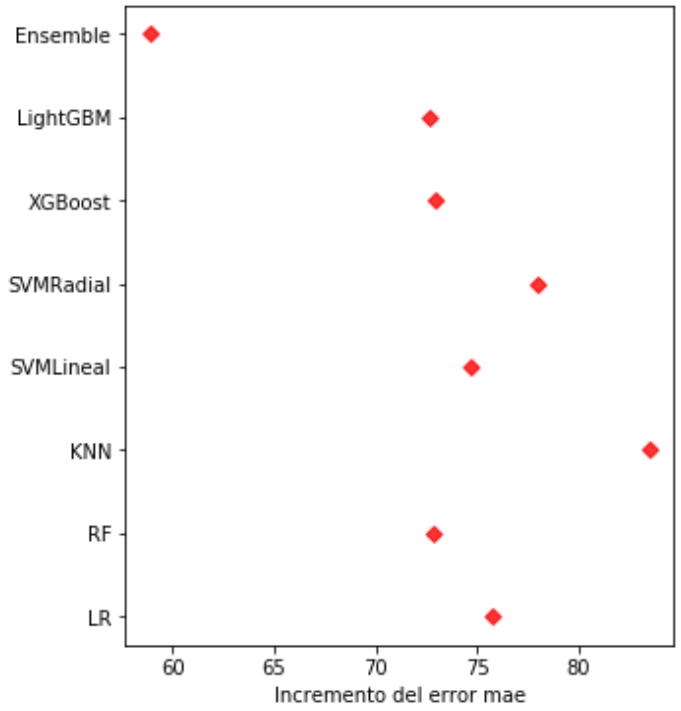
In [87]:
```
maesLag = [maeLR_Lag, maeRF_Lag, maeKNN_Lag, maeSVMLinear_Lag, maeSVMRadial_L
ag, maeXGB_Lag, maeLightGBM_Lag, maeEnsemble_Lag]
```

In [88]:
```
maesWork = [maeLR_Work, maeRF_Work, maeKNN_Work, maeSVMLinear_Work, maeSVMRad
ial_Work, maeXGB_Work, maeLightGBM_Work, maeEnsemble_Work]
```

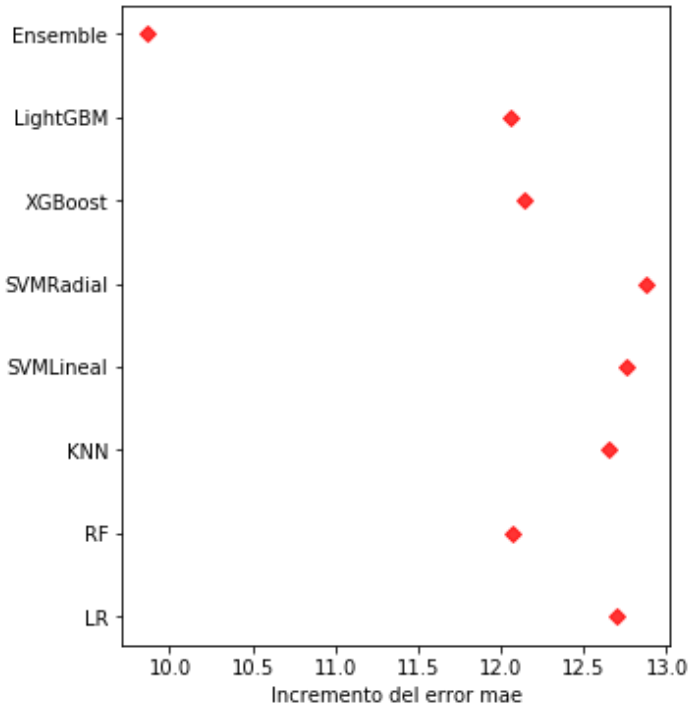
In [89]:
```
maesBreak =[maeLR_Break, maeRF_Break, maeKNN_Break, maeSVMLinear_Break, maeSV
MRadial_Break, maeXGB_Break, maeLightGBM_Break, maeEnsemble_Break]
```

In [90]:

```
fig, ax = plt.subplots(figsize=(5, 6))
ax.plot(
    maesLag,
    modelosNombre,
    marker="D",
    linestyle="",
    alpha=0.8,
    color="r"
)
ax.set_xlabel('Incremento del error mae');
```

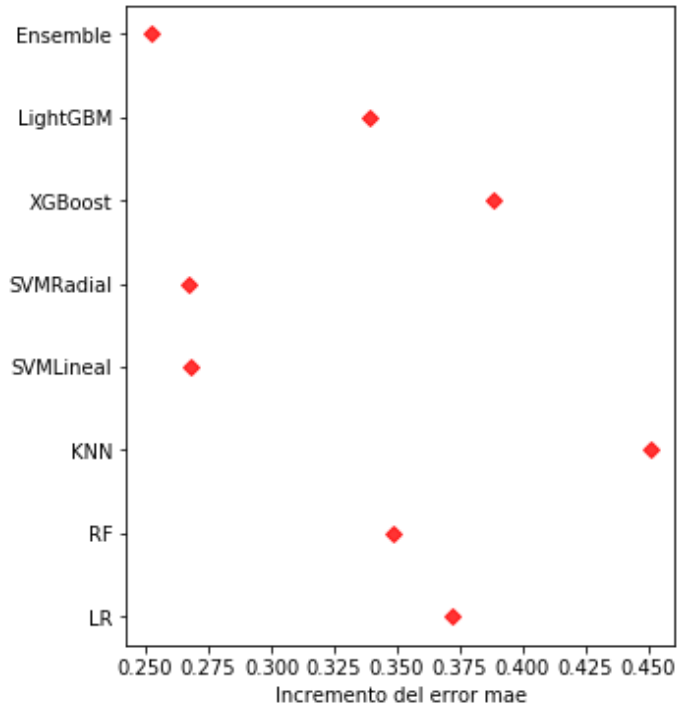


In [91]:
```
fig, ax = plt.subplots(figsize=(5, 6))
ax.plot(
    maesWork,
    modelosNombre,
    marker="D",
    linestyle="",
    alpha=0.8,
    color="r"
)
ax.set_xlabel('Incremento del error mae');
```



In [92]:

```
fig, ax = plt.subplots(figsize=(5, 6))
ax.plot(
    maesBreak,
    modelosNombre,
    marker="D",
    linestyle="",
    alpha=0.8,
    color="r"
)
ax.set_xlabel('Incremento del error mae');
```



ANEXO A

Anexo A

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
I.C.A.I.

PROYECTOS FIN DE MÁSTER
CURSO: 2020-2021

Ficha de proyecto fin de máster
(RELLENAR CON LETRAS DE IMPRENTA EN ORDENADOR)

Titulación y optatividad: Máster en Big Data, Tecnologías y Analítica Avanzada (MBD)
Alumno 1º Apellido: Gila
2º Apellido: Briñas
Nombre: Natalia

Teléfono de contacto: +34 627 86 01 47
e-mail: 202008739@alu.comillas.edu
natigila@gmail.com

Título del Proyecto Fin de Máster:

Algoritmos avanzados de Machine Learning en la estimacion de tiempos de resolucion de procesos bancarios

Director (nombre y dos apellidos): Gabriel Pierobon
Teléfono de contacto: +34 620 45 45 23
e-mail: gpierobon@kpmg.es

Breve descripción del proyecto (5 o 6 líneas)

El objetivo del proyecto, es la aplicación de algoritmos de machine learning avanzados en la predicción del tiempo que toma la resolución de procesos de un banco medido en minutos. Para la realización del proyecto, el análisis y el entrenamiento de los modelos, el lenguaje de programación que se utilizará será Python.
Se realizará en primer lugar un estudio de exploración y descripción de los datos proporcionados así como un proceso de depuración y la impieza. Tras esta fase, se ajustarán modelos sencillos como por ejemplo, regresiones lineales y SVM, para evaluar resultados y establecer un benchmark, y en función de estos, se procederá con modelos más complejos como por ejemplo XGBOOST o LightGBM.
Una vez obtenidos los resultados, se evaluarán utilizando diferentes métricas asociadas a predicciones numéricas

Aceptación del Director (firma y fecha)

16/02/2021

DATOS RELATIVOS AL PROYECTO FIN DE GRADO

Título del Proyecto Fin de Grado:

SISTEMA de ORGANIZACIÓN de PRÁCTICAS UNIVERSITARIAS

Director/es del Proyecto Fin de Grado:

Guillermo Román Díez

Curso Académico en el que se realizó:

2019-2020

Universidad (indicarla si no es Comillas):

Universidad Politécnica de Madrid

En el caso de realización en Comillas, indicar especialidad en el Grado: