Máster Universitario en

Ingeniería Computacional y Sistemas Inteligentes

Universidad del País Vasco Euskal Herriko Unibertsitatea

Konputazio Zientziak eta Adimen Artifiziala Saila –
Departamento de Ciencias de la Computación e Inteligencia Artificial

K I S A I C S I

Facultad de Informática

KZAA /CCIA

Tesis de Máster

Construcción de un modelo predictivo de tipo Gradrient boosting para ventas online

Javier Basauri Olabarria

Tutor(a/es)

Iñaki Inza
Departamento de Ciencia de la Computación e Inteligencia Artificial
Facultad de Informática

Septiembre 2015

Resumen

En este proyecto se describirá como construir un modelo predictivo de tipo gradient boosting para predecir el número de ventas online de un producto X del cual solo sabremos su número de identificación, teniendo en cuenta las campañas publicitarias y las características tanto cualitativas y cuantitativas de este. Para ello se utilizarán y se explicaran las diferentes técnicas utilizadas, como son: la técnica de la validación cruzada y el Blending.

El objetivo del proyecto es implementar el modelo así como explicar con exactitud cada técnica y herramienta utilizada y obtener un resultado válido para la competición propuesta en *Kaggle* con el nombre de *Online Product Sales*.

Índice

Índice de figuras

1. Introducción

1.1. Presentación del Problema

En los últimos años, ha existido un gran crecimiento en nuestras capacidades de generar y coleccionar datos, debido básicamente al gran poder de procesamiento de las máquinas como a su bajo costo de almacenamiento.

Sin embargo, dentro de estas enormes masas de datos existe una gran cantidad de información oculta, de gran importancia estratégica, a la que no se puede acceder por las técnicas clásicas de recuperación de la información. El descubrimiento de esta información "oculta" es posible gracias a la Minería de Datos (Data Mining)[1], que entre otras sofisticadas técnicas aplica la inteligencia artificial para encontrar patrones y relaciones dentro de los datos permitiendo la creación de modelos, es decir, representaciones abstractas de la realidad, pero es el descubrimiento del conocimiento (KDD, por sus siglas en inglés) que se encarga de la preparación de los datos y la interpretación de los resultados obtenidos, los cuales dan un significado a estos patrones encontrados.

Así el valor real de los datos reside en la información que se puede extraer de ellos, información que ayude a tomar decisiones o mejorar nuestra comprensión de los fenómenos que nos rodean. Hoy, más que nunca, los métodos analíticos avanzados son el arma secreta de muchos negocios exitosos. Empleando métodos analíticos avanzados para la explotación de datos, los negocios incrementan sus ganancias, maximizan la eficiencia operativa, reducen costos y mejoran la satisfacción del cliente.

Descubrimiento del conocimiento en bases de datos (KDD)

Según Molina [2] lo define como "la extracción no trivial de información potencialmente útil a partir de un gran volumen de datos, en el cual la información está implícita, donde se trata de interpretar grandes cantidades de datos y encontrar relaciones o patrones para conseguirlo harán falta técnicas de aprendizaje, estadística y bases de datos".

Las tareas comunes en KDD son la inducción de reglas, los problemas de clasificación y clustering, el reconocimiento de patrones, el modelado predictivo, la detección de dependencias, etc.

Los datos recogen un conjunto de hechos (una base de datos) y los patrones son expresiones que describen un subconjunto de los datos (un modelo aplicable a ese subconjunto), tal como se muestra en la Figura 1. El KDD involucra un proceso iterativo e interactivo de búsqueda de modelos, patrones o parámetros, los cuales descubiertos han de ser válidos, novedosos para el sistema y potencialmente útiles.

Kaggle

Los datos que usaremos serán extraídos de una plataforma web llamada *Kaggle*. *Kaggle* es una plataforma donde diferentes compañías publican competiciones sobre modelos predictivos y análisis. Estas competiciones están abiertas a todo aquel que quiera trabajar en ellas, con el aliciente de que si se resuelven en el tiempo requerido se puede obtener un premio. Puede ser dinero o un puesto de trabajo en dicha empresa. El éxito de esta plataforma se basa en las diferentes maneras que hay para afrontar un problema de minería de datos. Dado que hay miles de técnicas a la hora de trabajar con un conjunto de datos, el hecho de que una infinidad de personas puedan trabajar de manera diferente sobre el mismo tema genera más de una solución válida, a cada cual mejor. Las grandes empresas como: Facebook, Twitter, Google, Amazon... Son muy activas en este tipo de competiciones y podremos encontrar retos desde generar un algoritmo capaz de diferencias caras en una foto, hasta competiciones donde se pide construir un modelo para predecir si una persona comprara cierto producto o no. Existe mucha controversia respecto a este tipo de competiciones, ya que los premios monetarios son más que generosos normalmente, se dice que se esta fomentando el termino *freelance* o autónomo en castellano y esto crea un mercado libre del que las compañías salen beneficiadas. Como dato, y para hacerse una idea, la persona que consiguió el primer puesto en el modelo que vamos a crear en esta tesis tuvo un premio de 22 mil dolares. [3]

1.2. La competición: Online Product Sales

La competición que se propone en la página web de *Kaggle* es construir un modelo predictivo capaz de predecir el número de ventas que tendrá un producto el año siguiente basándonos en los datos recogidos durante el año actual. Esta competición fue propuesta en Mayo del 2012 y se cerró en Julio del mismo año. La persona que construyera el mejor modelo ganó un premio de 22mil dolares estadounidenses.

Para ello contaremos con diferente información para cada producto. Los productos vendrán identificados con un número de identificación o *id*. Lo cual dificultará el trabajo a la hora de reconocer los productos y analizar los resultados. Debido a que las variables que describen el producto están descritas de una manera que no reconoceremos, como podría ser color: rojo, el trabajo de preprocesado de datos será mas tedioso. Cada producto estará descrito por diferentes variables, cualitativas o cuantitativas, pero la única información que tendremos sobre ellas sera el nombre de la variable que estará descrita como la siguiente: *Quan_1*.

Tal y como se describe en la competición, deberemos basar nuestro mo-

delo en dos diferentes acontecimientos de cada producto. Por cada producto tendremos dos variables de tipo fecha que se referirán a las diferentes campañas publicitarias hechas para el producto. La primera variable de fecha nos dirá cuando empezó la campaña publicitaria del producto antes de que éste fuera lanzado. La segunda, nos dirá cuando empezó la campaña publicitaria una vez el producto fue lanzado. Basándonos en estas dos fechas deberemos de buscar efectos estacionarios en las ventas de cada producto para que nos pueda ayudar a la hora de predecir las ventas para el año siguiente.

El objetivo de la competición es construir un modelo predictivo que sea capaz de predecir las ventas de cada producto para los doce meses del año. El modelo se evaluara usando una formula que encontramos en la página de la competición en un apartado llamado *evaluation*. A cuanto menor valor en la formula, mejor será el modelo construido.

1.3. Objetivos

En este trabajo de fin de máster se intentara construir un modelo capaz de predecir el numero de ventas de un producto en los diferentes meses del año. Para ello usaremos diferentes técnicas de minería de datos como los análisis supervisados o diferentes regresiones con el fin de crear una predicción válida.

Como objetivos generales de este trabajo de fin de Máster podríamos enumerar los siguientes:

- Ahondar en el conjunto de datos y tratar de entenderlo.
- Aprender a usar herramientas para la minería de datos.
- Aprender nuevas técnicas efectivas para este problema.
- Describir y explicar las técnicas que se utilizaran.
- Programar un modelo que dé un resultado aceptable para las ventas online.

Apartando lo académico de los objetivos, no podemos olvidar que nuestro objetivo es construir un modelo que funcione. Ayudándonos así a obtener cierta competencia profesional en el ámbito de la minería de datos. Por ello, nuestros objetivos más concretos deberán ser los siguientes:

- Preprocesar los datos
- Construcción del modelo predictivo

- Análisis del Resultado
- Calculo del *Score* de nuestro modelo

A la hora de trabajar con un conjunto de datos es muy importante preprocesar los datos. Esto significa ver si los valores que aparecen en nuestro conjunto de datos son válidos para los algoritmos a usar. Normalmente, en un conjunto de datos tan grande, encontramos filas en las que faltan datos, numeros que no son relevantes, columnas que no sirven para nuestro cometido... Por ello, es importante que tratemos los datos antes de trabajar con ellos.

Una vez tratados los datos, podremos proceder a la construcción de nuestro modelo. Para ello usaremos diferentes técnicas de la minería de datos: validación cruzada, regresiones... Este sera lo que denominaremos el apartado técnico del trabajo.

Finalmente, nos quedara examinar el resultado obtenido por nuestro modelo. Es importante saber diferenciar entre un resultado cualquiera y un resultado que se adapte a nuestras necesidades. Debido a que, resultados obtendremos siempre pero probablemente no se acerquen a lo necesario y sus predicciones no serán nada certeras.

Ya que es un reto de *Kaggle*, podremos usar un algoritmo llamado *RMSLE*[4] o *root mean square logarithmic error* para calcular el *score* o puntuación de nuestro modelo. A menor valor en este *score* mejor sera nuestro modelo predictivo.[5]

2. Estado del Arte

El concepto de minería de datos apareció hace más de 10 años. El interés en este campo y su explotación en diferentes especialidades (negocios, finanzas, ingeniería, banca, salud, sistemas de energía, meteorología....), se ha incrementado recientemente debido a la combinación de diferentes factores, los cuales incluyen:

- El surgimiento de gran cantidad de datos (terabytes – $10^{12} bytes$ – de datos) debido a la medición y/o recopilación de datos automática, registros digitales, archivos centralizados de datos y simulaciones de software y hardware.
- El abaratamiento de los costos de los medios de almacenamiento.
- El surgimiento y rápido crecimiento del manejo de sistemas de bases de datos.
- Los avances en la tecnología computacional tal como los computadores rápidos y las arquitecturas paralelas.
- Los desarrollos continuos en técnicas de aprendizaje automático.
- La posible presencia de incertidumbre en los datos (ruido, outliers, información perdida).

El propósito general de la minería de datos es procesar la información de la gran cantidad de datos almacenados o que se puedan generar, y desarrollar procedimientos para manejar los datos y tomar futuras decisiones. Generalmente, una de las primeras tareas en el proceso de la minería de datos consiste en resumir la información almacenada en la base de datos, con el fin de comprender bien su contenido. Esto se realiza por medio de análisis estadísticos o técnicas de búsqueda y reporte. Las operaciones más complejas consisten en la identificación de modelos para predecir información acerca de objetos futuros. El término aprendizaje supervisado "*supervised learning*" (conocido como "*aprendizaje con profesor*") está implicado en el minado de datos, en el cual para cada entrada (input) de los objetos de aprendizaje, el objetivo de la salida (output) deseada es conocida e implicada en el aprendizaje. En los métodos de aprendizaje sin supervisión "*unsupervised learning*" ("aprendiendo por observación") el resultado no es suministrado o considerado del todo, y el método aprende por si solo de los valores de los atributos de entrada.

Pasos o procesos de la minería de datos

El proceso de minería involucra ajustar modelos o determinar patrones a partir de datos. Este ajuste normalmente es de tipo estadístico, en el sentido que se permite un cierto ruido o error dentro del modelo. En general el proceso de la minería de datos itera a través de cinco pasos básicos, tal como se muestra en la Figura 1:

- Selección de datos: consiste en buscar el objetivo y las herramientas del proceso de minería, identificando los datos a ser extraídos, buscando los atributos apropiados de entrada y la información de salida para representar la tarea. Las comprobaciones básicas deben incluir el tipo de consistencia, la validez de rangos, etc. Un sistema de minería de datos puede ser utilizado para este propósito, se pueden buscar patrones generales y reglas en las bases de datos que identifiquen valores irregulares que no cumplen las reglas establecidas.

- Trasformación de datos: las operaciones de transformación incluyen organizar los datos en la forma deseada, convirtiendo un tipo de datos en otro (por ejemplo de simbólico a numérico) definiendo nuevos atributos, reduciendo la dimensionalidad de los datos, removiendo ruidos, "*outliers*", normalizando, decidir estrategias para manejar datos perdidos.

- Minería de datos: los datos trasformados son analizados o coloquialmente hablando *minados*, utilizando una o más técnicas para extraer patrones de interés.

- Interpretación de resultados y validación: para comprender el significado del conocimiento extraído y su rango de validez, la aplicación de minería de datos prueba su robustez, utilizando métodos de validación establecidos y probándolo con datos diferentes a los utilizados para crear el modelo. Lo que se hace generalmente es dividir los datos en una serie para trabajo y otra para validación. Solo la serie de trabajo es utilizada para evaluar la habilidad del modelo desarrollado. La información extraída es también valorada (más subjetivamente) comparándola con experiencias anteriores.

- Incorporación del conocimiento descubierto: presentación de los resultados del modelo para poder comprobar o resolver conflictos con creencia o resultados anteriores y aplicar el nuevo modelo.

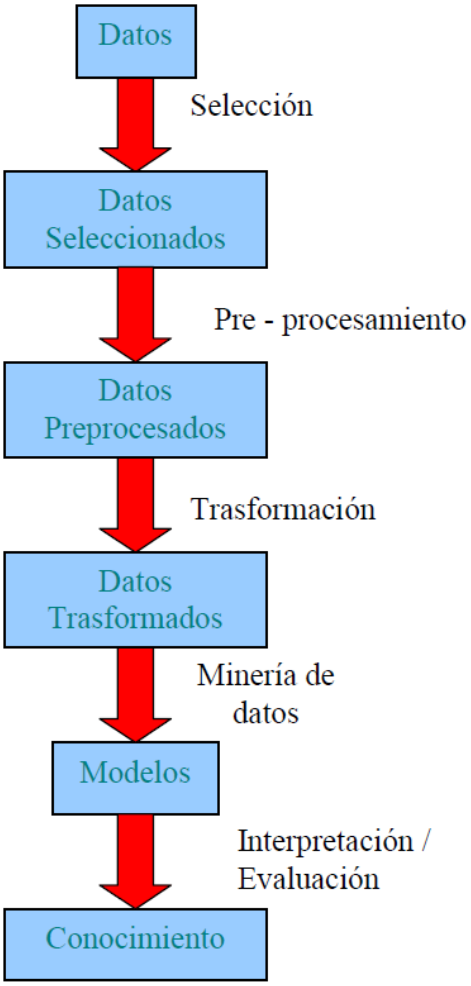

Figura 1: *Workflow* de la minería de datos

Las herramientas de minería de datos buscan dirigirse a dos requerimientos básicos:

- Descripción: descubriendo patrones, asociaciones y grupos de información. Puede incluir detección de desviaciones, segmentación de bases de datos, agrupamientos, asociaciones, reglas, resúmenes, visualización y minado de textos.
- Predicción: utilizando aquellos patrones para predecir tendencias y comportamientos futuros. La predicción incorpora tareas de clasificación, regresión y análisis de series temporales.

Técnicas usadas en la minería de datos

Dependiendo principalmente de la aplicación especifica y en el interés de la persona que trabaje los datos, se pueden identificar algunos tipos de tareas de minería de datos para las cuales se producen posibles respuestas. Algunas de las clases o categorías de minería de datos utilizados para la descripción y/o predicción son las siguientes:

- Presentación concisa de datos: apunta a producir descripciones compactas y características para un conjunto dado de datos. Pueden tomar múltiples formas: numérica (medidas simples de descripción estadística cómo medias, desviaciones estándar....), formas gráficas (histogramas, dispersiones), o en la forma de reglas "si-entonces". Se pueden realizar descripciones de la totalidad de los datos o seleccionando subconjuntos.

- Clasificación: desarrollando perfiles de grupos u objetos en términos de sus atributos. Un problema de clasificación es un aprendizaje supervisado, donde la información de salida es una clasificación discreta, es decir teniendo un objeto y sus atributos de entrada, el resultado de la clasificación es una de los posibles recíprocas clases exclusivas del problema. La tarea de clasificación tiende a descubrir alguna clase de relación entre los atributos de entrada y las clases de salida, tal que el conocimiento descubierto puede ser utilizado para predecir la clase de un nuevo objeto desconocido.

- Regresión: estableciendo relaciones entre series de objetos con el propósito de predecir. Un problema de regresión es un aprendizaje supervisado de construcción de un modelo más o menos trasparente, donde la información de salida es un valor numérico continuo o un vector de tales valores en vez de una clasificación discreta. Entonces, dando un objeto es posible predecir uno de sus atributos por medio de otros atributos, utilizando el modelo construido. La predicción de valores numéricos se puede realizar por métodos estadísticos clásicos o más avanzados, y por métodos simbólicos a menudo utilizados en las tareas de clasificación.

- Problemas temporales: es una regresión utilizando adicionalmente la información del tiempo. En ciertas aplicaciones es útil producir reglas que tengan en cuenta explícitamente el papel del tiempo. Las bases de datos que contienen información temporal pueden ser explotadas buscando patrones similares o aprendiendo a anticipar alguna situación anormal en los datos.

- Agrupamiento: fraccionando clases o ítems que presentan comportamientos o características similares en subconjuntos o grupos. El pro-

blema de agrupamiento, es un problema de aprendizaje sin supervisión, en el cual se busca encontrar en los datos grupos de objetos similares compartiendo un número de propiedades importantes. Se puede utilizar en la minería de datos para evaluar similitudes entre datos, construir un conjunto de prototipos representativos, analizar correlaciones entre atributos, o representar automáticamente un conjunto de datos por pequeños números de regiones, preservando las propiedades topológicas del espacio original de entrada.

- Modelos de causalidad: es un problema de descubrir relaciones de causa y efecto entre atributos. Una regla causal del tipo "si-entonces", indica no solo que existe una correlación entre la regla antecedente y la consecuente, sino que también la antecedente es causa de la consecuente.

- Análisis de asociación: reconociendo que la presencia de un grupo de ítems implica la presencia de otro grupo.

- Descubrimiento de secuencias: reconociendo que un grupo de ítems es seguido por otro grupo.

- Modelos de dependencia: consiste en descubrir un modelo que describe dependencias significantes entre atributos. Estas dependencias son generalmente expresadas como reglas "si-entonces" en la forma "si el antecedente es verdadero entonces la consecuencia es verdadera", donde tanto el antecedente como la consecuencia de la regla pueden ser una combinación de atributos.

- Detección de desviaciones: esta tarea esta enfocada a descubrir cambios significantes o desviaciones en los datos entre el contenido actual y el contenido esperado que puede ser previamente medido o de valores normalizados. Esto incluye la búsqueda de desviaciones en el tiempo y la búsqueda de diferencias inesperadas entre dos subconjuntos de datos.

La clasificación, regresión, y series temporales son utilizadas para predicción, mientras que el agrupamiento, la asociación y el descubrimiento de secuencias entre otras son más apropiados para describir relaciones existentes en los datos.

En nuestro caso tendremos en cuenta la clasificación, regresión y las series temporales, ya que nuestro objetivo es construir un modelo predictivo.

Técnicas de minería de datos

Como ya se ha comentado, las técnicas de la minería de datos provienen de la inteligencia artificial y de la estadística, dichas técnicas, no son más que algoritmos, más o menos sofisticados que se aplican sobre un conjunto de datos para obtener unos resultados.

Las técnicas más representativas son:

- Redes neuronales. Son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.
- Regresión lineal. Es la más utilizada para formar relaciones entre datos. Rápida y eficaz pero insuficiente en espacios multidimensionales donde puedan relacionarse más de 2 variables.
- Árboles de decisión. Un árbol de decisión es un modelo de predicción utilizado en el ámbito de la inteligencia artificial y el análisis predictivo, dada una base de datos se construyen estos diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que suceden de forma sucesiva, para la resolución de un problema.
- Modelos estadísticos. Es una expresión simbólica en forma de igualdad o ecuación que se emplea en todos los diseños experimentales y en la regresión para indicar los diferentes factores que modifican la variable de respuesta.
- Agrupamiento o Clustering. Es un procedimiento de agrupación de una serie de vectores según criterios habitualmente de distancia; se tratará de disponer los vectores de entrada de forma que estén más cercanos aquellos que tengan características comunes.
- Reglas de asociación.- Se utilizan para descubrir hechos que ocurren en común dentro de un determinado conjunto de datos.

Según el objetivo del análisis de los datos, los algoritmos utilizados se clasifican en supervisados y no supervisados

- Algoritmos supervisados (o predictivos): predicen un dato (o un conjunto de ellos) desconocido a priori, a partir de otros conocidos.
- Algoritmos no supervisados (o del descubrimiento del conocimiento): se descubren patrones y tendencias en los datos.

Aplicaciones comunes de la minería de datos

Muchas actividades o especialidades se pueden beneficiar del uso de la minería de datos, ya sea para realizar extracción de patrones o predecir comportamientos futuros. Algunas de las aplicaciones más frecuentes las encontramos en análisis de mercados para identificar afinidades entre productos y servicios adquiridos por el consumidor, segmentación de clientes para identificar características y comportamientos de clientes o consumidores en general que puedan ser explotados por el mercado, detección de fraudes en tarjetas de crédito, telecomunicaciones, sistemas de computo, detección de patrones en textos, imágenes o en la web, diagnósticos médicos etc.

El ejemplo clásico de aplicación de la minería de datos tiene que ver con la detección de hábitos de compra en supermercados. Un estudio muy citado detectó que los viernes había una cantidad inusualmente elevada de clientes que adquirían a la vez pañales y cerveza. Se detectó que se debía a que dicho día solían acudir al supermercado padres jóvenes cuya perspectiva para el fin de semana consistía en quedarse en casa cuidando de su hijo y viendo la televisión con una cerveza en la mano. El supermercado pudo incrementar sus ventas de cerveza colocándolas próximas a los pañales para fomentar las ventas compulsivas.

3. Conjunto de datos

En esta sección describiremos el conjunto de datos que se usará para este proyecto.

3.1. Datos

Tenemos dos archivos *.csv* diferentes. El primero es el denominado *train*; este es el que utilizaremos para entrenar nuestro modelo y aprender todo lo posible de él. El segundo es el denominado *test*. En este segundo es donde, una vez entrenado nuestro modelo, intentaremos predecir el numero de ventas del producto por cada mes.

Si echamos un vistazo rápido a los datos, podremos ver que muchas de las columnas tienen el valor *NaN*. Lo cual significa que un preprocesado de los datos va a ser más que suficiente si queremos obtener un resultado válido.

Los datos los obtenemos de la página del reto que esta *Kaggle* [6], con el nombre de *Online Product Sales*.

3.2. Descripción de variables

El objetivo de la competición, como hemos descrito antes, es construir el mejor modelo posible que sea capaz de predecir las ventas de un producto. Para ello, debemos imaginar que los productos son ayudados por campañas publicitarias de ayuda a la venta.

Cada línea dentro de el conjunto de datos representa un producto. Por lo tanto, si cargamos el conjunto de datos veremos que para la instancia de entrenamiento tenemos un conjunto de 751 productos y 558 variables. En cambio, para la instancia de test, veremos que el conjunto tiene 519 productos y 547 variables.

Si dedicamos algo de tiempo en investigar el conjunto de datos veremos que las primeras doce columnas denominadas *Outcome_M* contiene el número de ventas online. Cada una de las columnas representa las ventas de cada uno de los doce meses del año.

$$Outcome\_M1, Outcome\_M2, Outcome\_M3, Outcome\_M4,$$
$$Outcome\_M5, Outcome\_M6, Outcome\_M7, Outcome\_M8,$$
$$Outcome\_M9, Outcome\_M10, Outcome\_M11, Outcome\_M12$$

La columna Date_1 nos indica el día que comenzó la campaña publicitaria para el producto y el día que el producto fue lanzado al mercado.

La columna Date_2 nos indica el día que el producto fue anunciado, pero no puesto en venta, y la campaña prelanzamiento comenzó.

Las siguientes columnas del conjunto de datos nos ofrecen características del producto y de su campaña publicitarias. Así, encontraremos tres tipos de columnas:

- *Quan_X*
- *Cat_X*

Las columnas que empiecen con *Quan_x* nos indicarán columnas con variables cuantitativas. Las variables cuantitativas son aquellas características de los individuos de la muestra que puede medirse con un instrumento y lleva asociada una unidad de medida.

Sin embargo, las columnas que empiecen con *Cat_x* nos indicarán columnas con variables categóricas o cualitativas. Este tipo de variables nos describirán el producto. Encontraremos algunas de ellas que sean binarias, las cuales nos indicaran por ejemplo si un producto posee cierta cualidad o no. Se utilizará 1 si el producto posee la característica señalada o 0 si el producto no la posee.

4. Material y Métodos

4.1. Introducción

Para llevar a cabo este proyecto se han utilizado diferentes herramientas, desde las mas globales como pueden ser *Python* , un lenguaje de programación, hasta módulos específicos utilizados para *machine learning*. Por ello, en este apartado se detallarán las herramientas utilizadas y las razones por las que se han utilizado.

Las tareas que se llevan a cabo con la minería de datos se pueden clasificar como predictivas y descriptivas. En las tareas predictivas cada elemento de la base de datos se caracteriza por tener unos parámetros de entrada y un parámetro de salida. El objetivo es predecir el valor del parámetro de salida utilizando la información proporcionada por los parámetros de entrada. Dentro de las tareas predictivas existen dos tipos:

- Clasificación. Cada elemento de la base de datos tiene asociado un valor discreto (clase) y el objetivo que se persigue es maximizar el poder de predicción de la clasificación de nuevos elementos para los cuales la clase es desconocida.

- Regresión. En este caso el valor asociado a cada elemento es un número real. El objetivo es el mismo, maximizar la capacidad predicción de este valor para un elemento nuevo a través de una función real que se debe aprender.

En cuanto a las tareas descriptivas, los elementos de la base de datos sólo tienen atributos de entrada y el objetivo es el de agruparlos maximizando la similitud entre los elementos de un mismo grupo y minimizando dicha similitud entre los diferentes grupos. Un ejemplo claro de esto es el clustering.

4.2. Python

El lenguaje de programación usado para este proyecto ha sido Python. Concretamente la version 2.7 de Python. Se ha decidido usar esta version aún sabiendo que no es la última para no encontrar ningún error de compatibilidad con los módulos o paquetes que se usarán para los análisis de datos. Existen muchos otros programas o lenguajes de programación válidos para este tipo de proyectos: R, Weka ... A continuación explicaremos las razones por las que usaremos Python en este proyecto.

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un

Figura 2: *Python2*

lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Una de sus mayores características es que posee una licencia de código abierto.

Python fue diseñado para ser leído con facilidad. Una de sus características es el uso de palabras donde otros lenguajes utilizarían símbolos. Por ejemplo, los operadores lógicos !, | y & en Python se escriben not, or y and, respectivamente.

El contenido de los bloques de código (bucles, funciones, clases, etc.) es delimitado mediante espacios o tabuladores, conocidos como indentación [7]. Python se diferencia así de otros lenguajes de programación que para declarar los bloques usan un conjunto de caracteres, normalmente entre llaves.[8]

Para este proyecto se ha utilizado la distribución de Python llamada Anaconda. Anaconda es un paquete que posee Python y que se utiliza mayormente para análisis predictivos, computación científica y procesamiento de datos. Se eligió esta distribución por su fácil instalación y su potencia computacional. También por ser una distribución gratuita si se usa para fines académicos[9].

Figura 3: *Anaconda* logo de la distribución usada

4.3. Scikit Learn

El proyecto scikit-learn fue creado por David Cournepeau como un proyecto dentro de el Google Summer of code. Su nombre proviene de *Scipy Toolkit*, es una extensión desarrollada separadamente de la extension SciPy[10].

Figura 4: *Scikit-learn:* logo del modulo

Hoy en día Scikit-learn esta siendo desarrollado a una velocidad altísima. Si buscamos empresas de renombre que puedan usar esta herramienta encontramos Evernote, que utilizan las librerias de Scikit-learn para diferenciar recetas de otros posts hechos por usuarios usando un clasificador Naive Bayes.[11]. También encontramos Mendley, empresa que ofrece recomendaciones sobre productos usando un algoritmo de regresión llamado SGD.[12].

Para este proyecto se han usado diferentes utilidades dentro de este módulo de funciones. Una de las primeras es la función para hacer validación cruzada, más conocida por su nombre en ingles, *cross validation*.

Ya que queremos construir un modelo predictor se ha optado por usar un modelo lineal. Dentro de las diferentes funciones que ofrece Scikit-learn se ha probado el *gradient tree boosting* [13].

4.4. Cross Validation

La validación cruzada o *cross validation* es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar cómo de preciso es un modelo que se llevará a cabo a la práctica. Es una técnica muy utilizada en proyectos de inteligencia artificial para validar modelos generados.

La validación cruzada proviene de la mejora del método de retención o *holdout method* [14]. Este consiste en dividir en dos conjuntos complementarios los datos de muestra, realizar el análisis de un subconjunto (denominado datos de entrenamiento o training set), y validar el análisis en el otro subconjunto (denominado datos de prueba o test set), de forma que la función

de aproximación sólo se ajusta con el conjunto de datos de entrenamiento y a partir de aquí calcula los valores de salida para el conjunto de datos de prueba (valores que no ha analizado antes). La ventaja de este método es que es muy rápido a la hora de computar. Sin embargo, este método no es demasiado preciso debido a la variación del resultados obtenidos para diferentes datos de entrenamiento. La evaluación puede depender en gran medida de cómo es la división entre datos de entrenamiento y de prueba, y por lo tanto puede ser significativamente diferente en función de cómo se realice esta división. Debido a estas carencias aparece el concepto de validación cruzada.

Para este proyecto utilizaremos la validación cruzada de K iteraciones. En la validación cruzada de K iteraciones o K-fold cross-validation los datos de muestra se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y el resto (K-1) como datos de entrenamiento. El proceso de validación cruzada es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba, pero aun así tiene una desventaja, y es que, a diferencia del método de retención, es lento desde el punto de vista computacional. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos. Lo más común es utlizar la validación cruzada de 10 iteraciones (10-fold cross-validation). Como podemos observar en la figura 5, usamos la técnica 4-fold para estimar la precisión de los diferentes modelos predictivos.

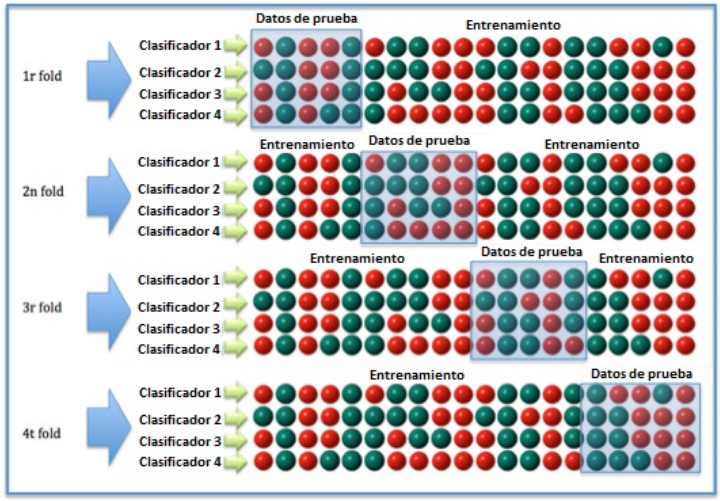

Figura 5: *Cross Validation:* ejemplo cuando K=4

4.5. Preprocesado de datos

Como hemos comentado en la sección de el conjunto de datos, nos encontramos ante un gran conjunto de datos en el que tenemos muchos valores perdidos, *NaN*. Los valores *NaN* (*Non Available Number*) son tediosos a la hora de ejecutar funciones como regresiones o clustering ya que son tratados como números infinitos. Para ello se ha implementado un programa en *Python* con el nombre de *explorar.py* en el que hacemos una rápida exploración del conjunto de datos.

Primero de todo, juntamos los dos conjuntos de datos, el de entrenamiento y el de test. Una vez creado el conjunto de datos global, seleccionamos las variables cuantitativas que mostraremos en escala logarítmica. Una vez decididas las variables que guardaremos en un vector llamado *to_log* procederemos a dibujar los histogramas de las variables seleccionadas. Como dato a añadir, se calcularán el numero de valores NaN por variable que mostraremos en la gráfica.

El resultado de los histogramas se puede observar en la Figura 6

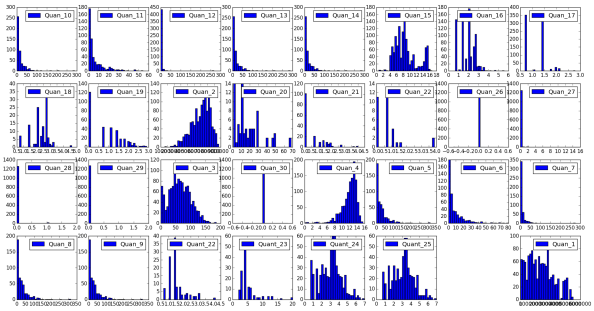

Figura 6: *Histograma:* representación gráfica de variables to_log

Como podemos observar, de 31 variables escogidas, ciertas muestran una dispersión más uniforme que otras. Si tenemos en cuenta el número de valores *NaN* de cada variable, podremos ver que en algunas de ellas no hay ningún valor *NaN* pero que en la mayoría el número oscila entre 811 y 1159. Debido a esto, habrá que tratar con especial dedicación estos valores ya que eliminarlos implicaría perder una cantidad inmensa de información.

Una vez analizado el conjunto de datos nos dispondremos a crear el *dataset* con el que trabajaremos para crear nuestro modelo. Para ello se ha implementado una función llamada *create_dataset* dentro del archivo *create_dataset.py*.

Lo primero que haremos será juntar los diferentes conjuntos de datos, co-

mo se ha hecho en *explore.py*. Una vez hecho esto crearemos cuatro diferentes vectores donde guardaremos los diferentes valores de las variables:

- X_categorical, vector donde se guardarán las columnas cualitativas.
- X_quantitative, vector donde se guardarán las variables cuantitativas.
- X_date, vector donde se guardará la diferencia entre fechas de prelanzamiento y lanzamiento del producto.
- X_id, vector para el id del producto

Como hemos descrito en el apartado *Conjunto de datos*, existen cuatro tipos de columnas: *Cat, Quan, Date y Outcome*. Para cada una de ellas deberemos de ejecutar un diferente tratamiento. Empezaremos con las columnas que empiecen con las letras "cat". Una vez que tenemos los datos de la variable, buscamos por los valores únicos, es decir, los tipos diferentes de cualidades o *levels* de la variable. A continuación binarizamos [15] la variable, consiguiendo así una variable tipo *dummy* que nos sera de gran ayuda a la hora de construir el modelo predictivo[16].

Para las columnas que empiecen con las cadena "Quant" el proceso será diferente. Primero de todo verificaremos si la columna esta en nuestra variable global *to_log* anteriormente explicada como las variables que podemos pasar a una escala logarítmica para obtener mejores resultados. En el caso de que este en este vector, ejecutaremos el logaritmo en cada uno de sus valores y lo guardaremos en el vector X_quantitave. De no ser así, desecharemos la columna llenándola de valores cercanos a cero, haciendo que tengan la mínima repercusión en el modelo.

Seguiremos con las variables con las cadenas "Date". Para estas variables la metodología a seguir será diferente en comparación a las anteriores. Dado que son fechas, nos sería de gran ayuda encontrar algún que otro efecto estacional. Esto nos ayudaría muchísimo a la hora de construir el modelo. El efecto estacional es cuando se determina unas pautas muy semejantes a lo largo del año de una determinada variable. Un ejemplo casi extremo, la venta de helados. En verano se puede cuadruplicar al invierno, por eso hay que corregir el efecto estacional si queremos hacer un análisis de regresión de la variable [17]. Basándonos en este tipo de fenómenos intentaremos encontrar algunos productos en los que, según la época del año sean más vendidos que otros.

Por último trataremos las columnas que contengan la cadena "outcome". Estas columnas representan el número de ventas del producto, por lo cual directamente las pasamos a nuestro conjunto de datos final.

Como función adicional se ha implementado una función para encontrar columnas redundantes y quitarlas de nuestro conjunto de datos final. La razón por la que hacemos esto pueden ser diferentes. La principal razón a tener en cuenta es que cuando existe un producto varias veces repetido, nuestro modelo podría "aprendérselo" de memoria y obtener peores resultados a la larga. La segunda razón es que al tener columnas redundantes solo estamos añadiendo información no necesaria al algoritmo. Por ello, se decide implementar esta función y eliminar aquellos datos repetidos.

Los datos se guardarán comprimidos en un archivo tipo *pickle.gz* para una fácil utilización. Usando el modulo *pickle* de Python comprimiremos toda la información en un archivo que llamaremos all_data. De esta manera nos será cómodo acceder a los datos desde diferentes módulos del programa sin tener que empezar desde el tratado de los *csv*. Una de las ventajas al usar este tipo de compresión es que no perdemos los vectores guardados en él. Es decir, a la hora de cargar este archivo podremos usar directamente los vectores que tienen toda la información tratada en el preprocesado anteriormente explicado.

4.6. Gradient Boosting

El Gradient Boosting es una técnica usada en *machine learning* para regresiones y problemas de clasificación, el cual produce un modelo predictivo uniendo de una manera efectiva diferentes modelos débiles de predicción, normalmente árboles de decisión. Construye el modelo en diferentes etapas, como hacen los diferentes métodos que usan *boosting*, y los generaliza permitiendo una optimización basada en una función de pérdida diferencial arbitraria.

La idea del Gradient Boosting nació por las observaciones de Leo Breiman [18] que se basa en la optimización de los algoritmos basándonos en un coste computacional aceptable. Los algoritmos de regresión gradient boosting fueron desarrollados basandonse en la idea de Breiman por Jerome H. Friedman [19][20]. Artículos posteriores a este desarrollo demostraron el punto de vista abstracto de un algoritmo boosting donde se observaba una iteracion funcional de un algoritmo tipo gradiente descendiente, como podemos observar en la Figura 7.

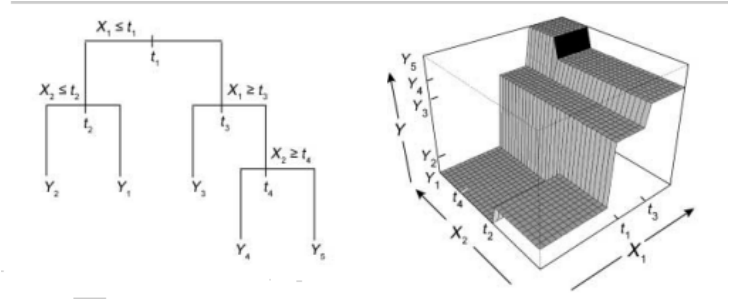

Figura 7: *Un simple árbol de decisión:* representación gráfica de un modelo de predicción usando Gradient Boosting

Así los algoritmos optimizan su coste computacional sobre el espacio, escogiendo la función (hipótesis más débil)que apunte en una dirección negativa gradiente de una manera iterativa. Esta manera de optimizar los algoritmos usando la gradiente en técnicas de *boosting* ha ayudado a que se usen de una manera muy activa en problemas de minería de datos y estadística mas allá de la regresión y la clasificación

Una breve descripción de los métodos *Boosting*

Los métodos adaptativos basados en *boosting* son simple algoritmos modulares que se crean de la siguiente manera. Supongamos que $g : X \to Y$ es la función que sera aprendida, donde Y es una variable finita, normalmente binaria aunque no tiene por que ser así. El algoritmo usa un proceso de aprendizaje, en el que tiene acceso a un número n de ejemplos de entrenamiento, $(x_1; y_1)...(x_n, y_n)$, ordenados aleatoriamente para XeY siguiendo

una distribución D. Como resultado generará una hipótesis $f : X \to Y$, de la cual el error es el valor esperado de una función de perdida en $f(x), g(x)$ donde x es seccionada dependiendo de D. Dado $\epsilon, \delta > 0$ y acceso a ejemplos aleatorios, un proceso de aprendizaje de tipo *strong* nos devolverá con probabilidad $1-\delta$ una hipótesis con un error como máximo de ϵ, con un tiempo de ejecución polinómico entre $1/\delta, 1/\epsilon$ y el número de ejemplos. Sin embargo, lo que un modelo de aprendizaje de tipo *weak* o débil nos devolvería satisface las mismas condiciones que un modelo con un aprendizaje *strong*. Con la diferencia que necesita de un ϵ mejor que el de un pronostico aleatorio.

Teniendo en cuenta el articulo de Schapire [21] se puede demostrar que cualquier método de aprendizaje de tipo débil se puede transformar eficientemente (*boosted*) en un procedimiento de aprendizaje de tipo *strong*. Por ejemplo, el algoritmo *AdaBoost* [22] consigue esto llamando múltiples veces a la función *WeakLearn*, en una secuencia de T etapas, cada una de ellas con diferentes distribuciones fijadas a un conjunto de datos y finalmente combinando todas las hipótesis. El algoritmo mantiene un *weight* w_t^i for cada ejemplo de entrenamiento i y etapa t, y la distribución D_t se consigue normalizando dichos *weights*. El algoritmo sigue los siguientes pasos en cada loop:

1. En la etapa t, se pasa la distribución D_t al algoritmo WeakLearn, la cual devuelve una hipótesis f_t.

2. La nueva distribución de entrenamiento es calculada de los nuevos *weight*.

Una vez dadas T iteraciones, un ejemplo de tipo test x sera clasificado por una combinación de hipótesis ordenadas por sus diferentes *weight*:

$$Y = sgn(\sum_{t=1}^{T} c_t f_t(x))$$

Cada coeficiente de combinación $c_t = log((1-\epsilon_t)/\epsilon_t)$ se basa en la precisión de la hipótesis f_t con respecto a la distribución.

A la hora de usar este método en nuestro programa hay que tener en cuenta los diferentes parámetros que le pasaremos para obtener un resultado óptimo. Los parámetros que hemos utilizado son los siguientes:

- Learning_rate: este atributo, que se refiere a la tasa de aprendizaje, tiene un valor de 0,1. Mediante este atributo podremos controlar la tasa de aprendizaje de cada árbol de decisión que forme nuestro modelo. Normalmente el valor del *learning_rate* va en concordancia con el de *n_stimators*. Debido a que, a cuanto menor sea este atributo, mayor tendrá que ser el número de estimadores para obtener un modelo robusto.

- Subsample: atributo que por defecto esta definido con un valor de 1. Si usamos un valor menor que 1, significará que usaremos el método de *Stochastic Gradient Boosting*[23] diciendo al algoritmo que interaccione directamente con el atributo *n_estimators*. En cambio, si usamos un valor mayor que 1, estaremos forzando una reducción de la varianza pero incrementaremos el margen de error.

- Max_depth: Atributo que nos indica la profundidad máxima de cada estimador del regresor. Lo que se traduce a el máximo de nodos dentro de cada árbol de decisión dentro del modelo. El valor óptimo para este atributo depende de la interacción entre los anteriormente descritos.

- N_estimators: Atributo que nos marcará el numero de veces que ejecutaremos la técnica *boosting* por cada árbol de decisión. Los métodos de *Gradient Boosting* son dados al sobreajuste [24], lo cual se reduce a que cuanto mayor sea el valor de este atributo mejor sera el resultado.

4.7. Blending

Por último nos queda explicar que es el apartado Blending de nuestro código. El *Blending* o lo que también se denomina *Ensemble learning* es una técnica usada en *machine learning* donde la finalidad es juntar diferentes modelos predictivos y combinarlos para obtener un resultado mejor. De esta manera permitimos que existen muchas más estructuras flexibles entre las diferentes alternativas [25]. La siguiente figura 8 nos puede ayudar a entender la flexibilidad de el modelo:

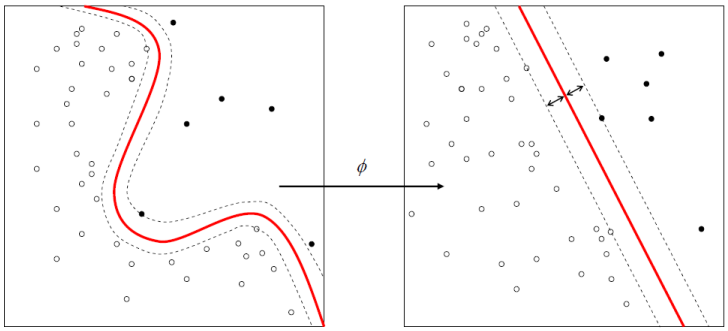

Figura 8: *Blending:* representación gráfica de un modelo flexible contra uno rígido

Los algoritmos de aprendizaje supervisados se describen comúnmente como funciones que tienen como objetivo buscar una hipótesis valida entre un espacio de diferentes hipótesis para hacer una buena predicción de un problema particular. Incluso si el espacio de hipótesis contiene una hipótesis realmente eficaz para dicho problema puede ser muy difícil y costoso encontrarla. Debido a esa dificultad si se combinan diferentes hipótesis entre ellas sería posible encontrar una mejor hipótesis. El método *Blending* es normalmente usado para métodos que devuelven más de una hipótesis basándose en un mismo conjunto de datos de entrenamiento.

Evaluar la predicción de un modelo que ha usado la técnica *Blending* siempre requiere de un mayor valor computacional que el de un modelo simple. Por ello estos métodos se utilizan para compensar modelos en los que el aprendizaje no ha sido del todo bueno, es decir, una vez aplicado el modelo y analizamos el resultado viéramos que la confianza de este es baja. Algoritmos eficientes como los arboles de decisión son normalmente reforzados mediante esta técnica aunque algoritmos más lentos se pueden beneficiar del *Blending* igualmente.

En nuestro caso hemos optado por construir diferentes modelos por cada mes. Esto significa que si para la validación cruzada hemos elegido un

valor de $K = 3$ por cada mes entrenamos 3 veces nuestro modelo. Las predicciones sobre el conjunto de datos de entrenamiento las guardaremos un la variable *dataset_blend_train*. Las predicciones sobre el conjunto test de los modelos creados los guardaremos en una variable que la llamaremos *dataset_blend_submission* para posteriormente tratarlos en el apartado *blending* del código. Una vez que hemos hecho las predicciones para cada mes del año nos dispondremos ha hacer el *blending* de nuestras predicciones y obtener los valores óptimos de predicción.

En este caso trataremos los diferentes resultados filtrados por mes, es decir, las predicciones de enero se combinarán entre ellas y a continuación las de febrero y así sucesivamente hasta llegar a diciembre. Para cada mes haremos lo mismo:

1. Creamos la variable *reg* usando la función *RidgeCV* del paquete *scikitLearn*. Por defecto esta función ejecuta una validación cruzada basándose en el tipo: *Leave one out*. Como input deberemos pasar un vector de valores *alpha* que probará. En nuestro caso se opta por un espacio logarítmico de tamaño 40.

2. Ejecutamos la función *fit* usando el vector *dataset_blend_train* y la columna de predicciones y.

3. Imprimimos en pantalla el mejor valor *alpha* del modelo para dicho mes.

4. Por ultimo ejecutamos el predictor y guardamos los valores en la variable *y_submission*

5. Resultados

Tal y como hemos comentado, el resultado de nuestro modelo será guardado en un archivo *test.csv* donde aparecerá por cada producto el número de ventas en los diferentes doce meses del año. En total tenemos un número de 519 productos. Se ha decidido mostrar los resultado de esta manera ya que era un requisito del reto de *Kaggle*. El resultado obtenido será el siguiente:

Cuadro 1: Resultado del modelo

id	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto
1	8349	4278	1735	1157	770	996	2305	1040
2	4893	3543	3281	1764	1747	1579	1037	1087
3	4434	5325	1560	1026	947	1040	735	709
4	7300	6483	2941	2880	2472	1968	1492	1479
5	3484	1509	847	714	800	646	615	577
6	6965	2067	1000	991	887	905	928	678
7	10501	4565	2687	2423	1917	2119	1589	1217
8	2284	1087	899	695	604	573	545	487
9	12719	8222	2465	2119	1678	1234	1126	977
10	6647	2405	1209	1413	1055	873	834	798
11	467588	230261	54522	44467	12821	7905	6504	2562
12	5278	5384	2252	1534	1331	1338	1000	1269
13	38725	18895	13477	4173	3240	2103	1967	948
14	3595	2563	1864	1725	1481	1254	755	1134
15	2036	840	557	590	548	583	566	552
16	3203	1597	1005	1007	727	589	513	459
17	2415	1292	1035	1338	848	765	626	586
18	3506	1615	1121	959	819	1110	1723	872

Como podemos observar en la tabla, tenemos las predicciones para 18 productos desde el mes de Enero hasta el mes de Agosto. Evidentemente esto solo es un pequeño ejemplo de el resultado para que podamos hacernos a la idea del *output* o resultado de nuestro modelo. Debido a que no sabemos a que tipo de producto ser refiere cada línea no podemos sacar grandes conclusiones de aquí. De todas formas, podemos encontrar relaciones interesantes en algunos de los productos. Por ejemplo, si nos fijamos en el producto número 11, veremos que tiene un alto número de ventas en los meses de Enero, Febrero y marzo; pero que decaen en los meses de verano. Basándonos en este dato, se podría decir que el producto con *id* 11 es un producto relacionado con el invierno, imaginemos que el producto es algún tipo de ropa de abrigo.

Tendría sentido que durante los meses fríos se vendieran más que durante el verano. El hecho de no tener información especifica de cada producto dificulta mucho el análisis de el resultado, si no sabemos a que producto se refiere cada *id* solo podremos lanzar hipótesis.

Si calculamos el RMSLE anteriormente explicado y obtenemos un valor de 6,502612.

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(p_i - 1) - log(a_i - 1))^2}$$

Teniendo en cuenta este valor no podemos decir que el modelo sea el más preciso que se podría haber conseguido, ya que cuanto mas cercano al cero mejor y mas preciso sería nuestro modelo. A decir verdad no sabemos los *scores* de los otros participantes de la competición en *kaggle* por lo cual no podemos hacer ningún tipo de comparación con otros modelos.

Se han hecho pruebas con diferentes parámetros en el regresor para verificar la calidad del modelo. Los parámetros utilizados para obtener el RMSLE anterior son los siguientes:

$$GradientBoostingRegressor(learning\_rate = 0{,}001,$$
$$subsample = 0{,}5, max\_depth = 6, n\_estimators = 10000)$$

Si cambiamos el valor de n_estimators, de mil a un número mayor, obtendremos un modelo más robusto pero a su vez de mayor coste computacional. Debido a que cuanto más alto sea este número más etapas de *boosting* tendrá que calcular el algoritmo. Se ha probado con un modelo con las siguientes características:

$$GradientBoostingRegressor(learning\_rate = 0{,}001,$$
$$subsample = 0{,}5, max\_depth = 6, n\_estimators = 20000)$$

El resultado ha sido un modelo con un valor $RMSLE = 2{,}3654$, que como podemos observar, es considerablemente menor que el anterior. Hay que destacar que el modelo con un valor en $n\_estimators = 10000$ tardó 15 minutos en ejecutarse y el segundo modelo con $n\_estimators = 20000$ tardo 6 horas en ejecutarse. De estos datos podemos observar el coste computacional de un modelo más robusto contra uno menos robusto. La mejora es sustancial pero el coste computacional del segundo es 24 veces mayor que el primero. Para un conjunto de datos que no es de un tamaño considerable es un tiempo demasiado elevado. Es verdad que en la competición de *Kaggle* en ningún momento se tiene en cuenta el coste computacional pero es un dato muy a tener en cuenta si pensamos en el uso de este segundo modelo para un

problema de *big-data*, donde los conjuntos de datos son inmensos y este tipo de algoritmos tardarían demasiado tiempo en ofrecernos un resultado.

6. Conclusión

En este apartado explicaremos las conclusiones después de haber finalizado el trabajo.

El principal objetivo de este trabajo era construir y comprender cómo funcionan los modelos predictivos en un escenario real. Es decir, saliendo del ámbito académico ver si seríamos capaces de construir un modelo predictivo que sea valido para una situación real. Por ello se eligió este tema para el trabajo de fin de máster. El hecho de que se haya podido entender un conjunto de datos real, con variables sin estar totalmente descritas, lo cual nos ha hecho trabajar en momentos con cierta incertidumbre ya que no sabíamos si estábamos haciendo un uso correcto de la variable, ha sido verdaderamente satisfactorio.

Es una lástima que no podamos comparar nuestro modelo con otros modelos que hayan creado los diferentes participantes. Ya que, si esto fuera posible, podríamos haber sabido en qué situación se encontraría en la competición. Para ser sincero, el hecho de haber trabajado "a ciegas" durante gran parte del proyecto ha limitado mucho la calidad de este. Al tratarse de una competición que ademas tenía como premio una importante cantidad económica, la información facilitada era escasísima. Aún así, creo que el trabajo de investigación y aprendizaje que se ha efectuado durante este proyecto ha sido más que interesante.

Si tengo en cuenta lo que sabía al empezar el proyecto y ahora que lo he terminado veo una sustancial mejora. Durante el máster en las asignaturas de minería de datos aprendimos las bases de esta tecnología. Bases que de no tenerlas bien afincadas no podría haber terminado este proyecto. Dado que estas técnicas son increíblemente abstractas y muchas veces dependen del punto de vista que tiene uno sobre el problema hay más de una manera de afrontar el problema de manera correcta. Ese fue uno de los primeros problemas que se encontró a la hora de hacer frente a este proyecto. El hecho de que haya más de una manera de hacer lo "correcto" puede llevar a momentos en los que se hacen cosas sin saber muy bien por que y terminando en un punto en el que se han dado palos de ciego. Hubo que eliminar todo aquello que se basara en prueba y resultado e investigar sobre métodos que realmente son usados hoy en día para un problema como este.

Para conseguir este objetivo hubo que aprender tecnologías nuevas como son: el lenguaje de programación python y dentro de este los diferentes módulos usados. La curva de aprendizaje, viniendo de una ingeniería informática, no es que sea muy dura. El problema es, que para este tipo de problemas, hay que pensar de una manera más global y no tan ceñida a lo que la informática se refiere. Lo cual hizo que se necesitara más tiempo del previsto

para obtener un resultado válido.

Como conclusión final, me gustaría recalcar el hecho de que se ha trabajado sobre un problema real y que se ha conseguido interiorizar nuevos métodos de *machine learning* que serán de gran ayuda y uso para un futuro profesional. También hay que recalcar que se ha creado un modelo predictivo que no supone un gran valor computacional y que se podría haber usado para formar parte en la competición propuesta en *kaggle*.

7. Bibliográfia

Referencias

[1] Wikipedia, *Data Mining.* https://en.wikipedia.org/wiki/Data_mining

[2] Molina, Luis Carlos., *Torturando los Datos Hasta que Confiesen.* Departamento de Lenguajes y Sistemas Informáticos (2000),Universidad Politécnica de Cataluña. Barcelona, España.

[3] Wikipedia, *Kaggle*, última actualización 7 de septiembre de 2015. https://en.wikipedia.org/wiki/Kaggle

[4] Kaggle, *Root Mean Square Logarithmic Error.* https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError

[5] Wikipedia, *Root Mean Square Logarithmic Error.*

https://en.wikipedia.org/wiki/Root-mean-square_deviation

[6] Online Sales, *Predict the online sales of a consumer product based on a data set of product features..*

https://www.kaggle.com/c/online-sales

[7] David Goodger., *Python v2.7.8 Documentation (en inglés).* Consultado el 20 de julio de 2014.

[8] Python Software Foundation., *Code Like a Pythonista: Idiomatic Python.*Consultado el 20 de julio de 2014.

[9] Continuum Analytics, *Anaconda distribution.*

https://store.continuum.io/cshop/anaconda/

[10] Dreijer, Janto, *Scipy.*

https://scikits.appspot.com/scikit-learn

[11] Mark Ayzenshtat, *Stay classified*Evernote Techblog.

http://blog.evernote.com/tech/2013/01/22/stay-classified/

[12] Mark Levy2013, *Efficient Top-N Recommendation by Linear Regression*ACM RecSys Large Scale Recommender System workshop.

http://www.slideshare.net/MarkLevy/efficient-slides

[13] Gradient tree boosting, *Ensemble methods.*

http://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting

[14] Ron Kohavi, *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection* , Computer Science Department. Stanford University

http://frostiebek.free.fr/docs/Machine%20Learning/validation-1.pdf

[15] Smita Skrivanek, *The Use of Dummy Variables in Regression Analysis.* Principal Statistician, MoreSteam.com LLC .

https://www.moresteam.com/whitepapers/download/dummy-variables.pdf

[16] Wikipedia, *Dummy variable (statistics).*

https://en.wikipedia.org/wiki/Dummy_variable_(statistics)

[17] Series Temporales,

http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/EDescrip/tema7.pdf

[18] Breiman, L., *Arcing The Edge.*

http://statistics.berkeley.edu/sites/default/files/tech-reports/486.pdf

[19] Friedman, J. H., *Greedy Function Approximation:*A Gradient Boosting Machine.

http://www-stat.stanford.edu/~jhf/ftp/trebst.pdf

[20] Friedman, J. H., *Stochastic Gradient Boosting.*

https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

[21] ROBERT E. SCHAPIRE, *The Strength of Weak Learnability* .

http://rob.schapire.net/papers/strengthofweak.pdf

[22] Wikipedia, *AdaBoost Algorithm.*

https://en.wikipedia.org/wiki/AdaBoost

[23] Jerome H. Friedman, *Stochastic gradient boosting*, Computational Statistics and Data Analysis (2002) .

http://www.sciencedirect.com/science/article/pii/S0167947301000652

[24] Wikipedia, *Sobreajuste*.

https://es.wikipedia.org/wiki/Sobreajuste

[25] Polikar, R., *Ensemble based systems in decision making*, IEEE Circuits and Systems Magazine (2006) .

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1688199

8. Anexo

Todo el código usado para generar el modelo como los resultados y los conjuntos de datos se pueden encontrar en la siguiente url:

`https://github.com/basauri89/TFM-online-sales`