Trabajo Final de Máster

Predicción de Ventas de Comestibles Corporación Favorita

Autor: **Gabriel Kreplak**

Plan: Máster en Inteligencia de Negocio y Big Data

Área: Análisis de Datos

Consultora: **Dra. Laia Subirats Maté**

Profesoras responsables de la asignatura: **Dra. Teresa Sancho Vinuesa y Dra. María Pujol Jover**

23 de enero de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Predicción de Ventas de Comestibles Corporación Favorita.
Nombre del autor:	Gabriel Kreplak
Nombre del consultor/a:	Dra. Laia Subirats Maté
Nombre del PRA:	Dra. Teresa Sancho Vinuesa y Dra. María Pujol Jover
Fecha de entrega (mm/aaaa):	Enero 2018
Titulación::	Máster Inteligencia de Negocio y Big Data
Área del Trabajo Final:	Análisis de Datos
Idioma del trabajo:	Español – Inglés
Palabras clave	análisis de datos, ventas, light-GBM, alimentación, predicción, regresión

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.

El trabajo escogido tiene por finalidad conseguir una puntuación relevante en la competición de Kaggle: **Corporación Favorita Grocery Sales Forecasting**. El objetivo de la competición es construir un modelo predictivo para pronosticar ventas futuras.

Los métodos de planificación de ventas de esta compañía actualmente están poco sustentados en datos y como consecuencia no están automatizados. Por este motivo, Corporación Favorita propone a la comunidad Kaggle el desarrollo te técnicas de Machine Learning para conseguir adecuar en lo posible la logística y oferta de productos a la demanda de éstos donde son requeridos.

Como se detallará en el apartado de descripción de los ficheros de datos que se ofrecen para la predicción, éstos incluyen información para entrenamiento y test de los algoritmos supervisados y otros ficheros de datos para poder poner en contexto los datos mencionados. Los ficheros aportados son: train.csv, test.csv, sample_submission.csv, stores.csv, ítems.csv, transactions.csv, oil.csv y holidays_events.csv . El trabajo lo estructuraré en 2 partes principales:

a) Análisis Exploratorio de Datos, que incluirá que estudiará detalladamente la información de ventas: en general, por tipo de tienda, geolocalización , ventas por estado, por ciudad, ventas en el tiempo, correlación ventas – precio del petróleo, análisis de los productos y sus familias, transacciones, etc.

b) Estudio Predictivo para lo que utilizaré los algoritmos con los que obtenga un resultado que me permitan obtener una buena clasificación en la tabla y recibir también comentarios positivos.

Abstract (in English, 250 words or less):

The goal of this final grade project is to earn a relevant score in the Kaggle's competition: Corporación Favorita Grocery Sales Forecasting through building a state of the art predictive model aimed to forecast future sales.

Corporación Favorita has challenged the Kaggle community to build a model that more accurately forecasts product sales. They currently rely on subjective forecasting methods with very little data to back them up and very little automation to execute plans. They're excited to see how machine learning could better ensure they please customers by having just enough of the right products at the right time.

As it will be detailed below, in section "Description of the Input Data", training and testing data files are supplied in order to develop predictive supervised algorithms as well as other data file to be able to put the aforementioned data in context. The files provided are: train.csv, test.csv, simple_submission, stores.csv, items.csv, transactions.csv, oil.csv and holidays_events.csv.

The work will be structured in 2 main parts:

a) Exploratory Data Analysis, which will include detailed study of sales information: in general, by type of store, geolocation, sales by state, by city, sales in time, sales correlation - price of oil, analysis of products and their families, transactions, etc.

b) Predictive study for which I will use the state of the art algorithms in order to obtain a relevant classification in the score table and to receive positive comments as well.

Índice

Lista de Figuras

Lista de Tablas

Lista de Ecuaciones

1 Introducción

1.1 Contexto y justificación del trabajo

Las tiendas de comestibles al detalle siempre se debaten entre la planificación de las ventas y las compras: predecir con exceso de stock puede conducir a problemas de sobre-almacenamiento de productos perecederos; predecir en defecto produce un rápido agotamiento de los productos mas populares con el consiguiente lucro cesante y consumidores insatisfechos.

El problema deviene mas complejo a medida que el minorista agrega nuevas ubicaciones con necesidades propias, productos especiales y gustos estacionales. Corporación Favorita, una gran cadena minorista de comestibles de Ecuador lo sabe muy bien ya que operan en cientos de supermercados con mas de 200.000 diferentes productos en sus estantes.

Corporación Favorita ha lanzado una competición para la comunidad Kaggle para la construcción de un modelo de predicción de las ventas de sus productos. Actualmente la planificación de ventas y compras no está respaldada en datos sino que en pronósticos subjetivos y ello supone poca automatización para ejecutar los planes.

Como patrocinadores de la competición, en Corporación Favorita están muy interesados en comprobar como mediante aprendizaje automático es posible mejorar la aceptación de los clientes simplemente teniendo los productos adecuados en el momento preciso. (Kaggle, 2017)

1.2 Objetivos del trabajo

Los principales objetivos que se pretende alcanzar son los siguientes:

- Adquirir una metodología profesional en la organización de un proyecto de análisis predictivo, incluyendo la utilización de plataformas standard de discusión y desarrollo como por ejemplo:
    - Kaggle
    - Github
    - Google drive
    - Pycharm
    - Etc.
- Profundizar en el aprendizaje del lenguaje Python, ya que durante el curso del máster he podido conocer más en profundidad el lenguaje R y encuentro en éste trabajo de final de máster la oportunidad ideal para realizar una inmersión en una herramienta con indudable interés laboral.
- Adquirir competencias en la comprensión y la ejecución de análisis predictivos, haciendo uso de las técnicas “state of the art” como los algoritmos lgbm, xgboost, etc.
- Aprender técnicas de ingeniería de datos a efectos de afinar los resultados.

- Obtener de un modelo predictivo de las ventas de Corporación Favorita lo mas preciso posible y con ello obtener una buena calificación en la competición.

1.3 Enfoque y método seguido

Esta competición trata de obtener un pronóstico del número de ítem/tienda que se venderán durante 2 semanas de agosto de 2017, basado en los datos de ventas del año 2016 y de enero a julio de 2017.

Durante el mismo período de tiempo, se dispone además del numero de transacciones por tienda, clasificación de ítems, precio de petróleo, localización de tiendas y lista de festivos.

A fin de obtener un modelo predictivo con un nivel mínimo de precisión implementando el algoritmo LGBM, he procedido con los siguientes pasos:

- Carga y limpieza de los datos: La carga de los datos se realiza mediante 6 archivos en formato de valores separados por comas (csv). El dataset de esta competición esta suficientemente depurado, no se han detectado valores atípicos y no requiere modificación destacable de los datos.
- Ingeniería de los atributos: A efectos de conseguir una buena precisión, se realizan cálculos de media de ventas durante diferentes lapsos, como se verá mas adelante y se completará con la información de festivos, precio del petróleo, clase de producto y promociones.
- Análisis exploratorio de datos: Tiene como objeto reflejar las características principales de los datos que se utilizarán en las predicciones.
- Estudio predictivo y envío de resultados: Es la esencia de este trabajo de final de máster. Se aplica el algoritmo LGBM y se intenta un ajuste de parámetros tendente a conseguir el menor error posible de la predicción aplicada al test set.

1.4 Planificación del trabajo

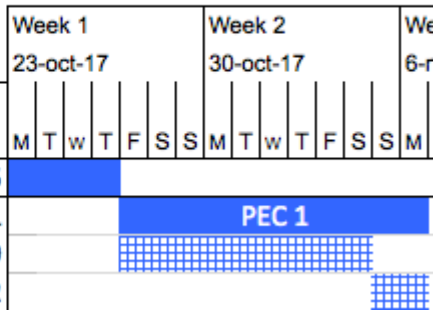
La planificación del trabajo se divide en la propuesta, análisis y diseño preliminar, estructura de la memoria, tuneado de la predicción y evaluación. El detalle de la planificación se puede visualizar en la Tabla 1. Planificación.

B2.342 Trabajo Final de Master - AD

Corporación Favorita Grocery Sales Forecasting - Kaggle

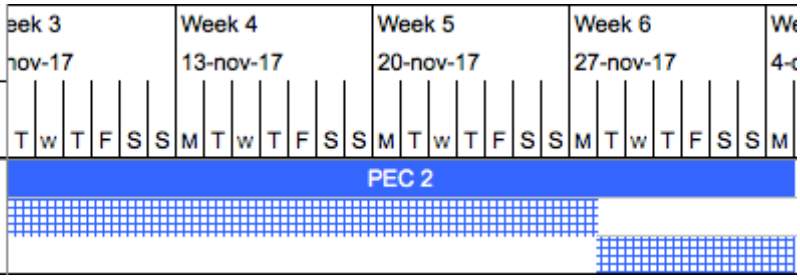
Author: Gabriel Kreplak

WBS	Task	Start	End	Cal. Days
1	Theme Selection	22/10/17	26/10/17	5
2	Initial Proposal	27/10/17	6/11/17	11
2.1	Preliminary Analysis	27/10/17	4/11/17	9
2.2	Preliminary design	5/11/17	6/11/17	2



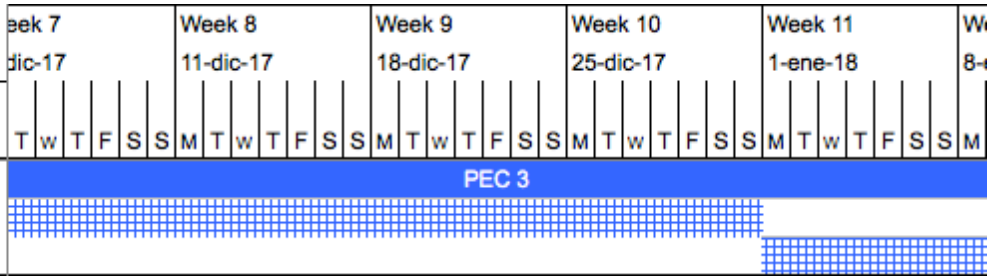
Author: Gabriel Kreplak

WBS	Task
3	Memory Structure
3.1	EDA design
3.2	Preliminary Predict



Author: Gabriel Kreplak

WBS	Task
4	First Version
4.1	Prediction Tuning
4.2	Report Issuing



Author: Gabriel Kreplak

WBS	Task
5	Delivery
5.1	Last review
	Milestone: Submittal
5.2	Report Correction
	Milestone: Delivery
6	Evaluation

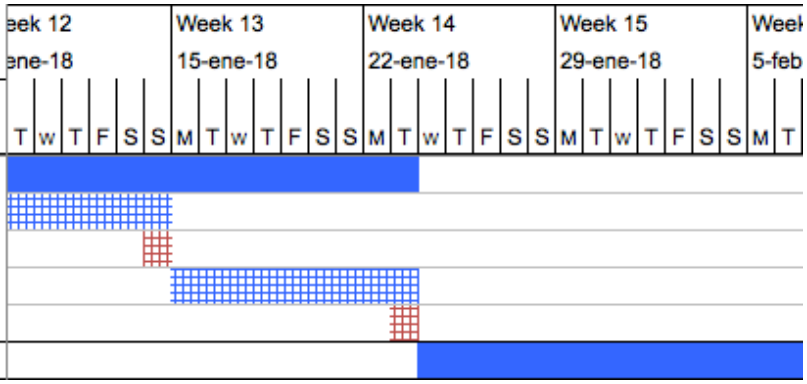


Tabla 1. Planificación

1.5 Descripción de los demás capítulos de la memoria

A continuación se realiza una descripción muy somera del contenido del resto de los capítulos de esta memoria:

- Estado del Arte: esbozo de las tecnologías actuales y referencias a estudios realizados relacionados con el desarrollo de este trabajo final de máster.
- Competición Corporación Favorita: exposición de las reglas de la competición, características de los datos de entrada, evaluación y demás detalles de las condiciones iniciales del trabajo.
- Análisis Exploratorio de Datos: Detalle y características de los distintos archivos de datos de la competición.
- Estudio Predictivo: Despliegue del proceso de carga de datos, depuración, ingeniería de atributos, técnica predictiva LGBM y generación de los datos de salida.
- Conclusiones: Lecciones aprendidas, reflexión sobre la consecución de los objetivos planteados inicialmente, análisis crítico del seguimiento de la planificación y líneas de trabajo futuras.
- Glosario.
- Bibliografía.

2 Estado del Arte

La Corporación Favorita es uno de los tres supermercados líderes de ventas en Ecuador. Actualmente hay estudios que explican el desarrollo organizacional para su crecimiento (Serrano, 2016) y analizan el comportamiento del cliente (Baquerizo Ramon, 2017). También hay estudios estadísticos utilizando análisis multivariante de los factores que intervienen en la compra de marcas propias en el Ecuador, y éstas utilizan el programa R (Santamaria, 2014).

Actualmente hay diferentes técnicas para problemas de regresión. Una de las más usadas actualmente son los gradient boosting decision trees (GBDT) (Friedman, 2002). En Compact Multi-Class Boosted Trees (Natalia Ponomareva, 2017), se puede ver una comparativa de las librerías *gradient boosting*. Actualmente, han aparecido algunas técnicas que mejoran su rapidez, como por ejemplo LightGBM .

GBDT (Si Si Huan Zhang) es un popular algoritmo y de momento tiene algunas implementaciones efectivas como XGBoost y pGBRT en las que la eficiencia y escalabilidad pueden verse comprometidas en datasets grandes y/o alta dimensionalidad. Ello se debe a que por cada atributo o columna se requiere examinar todas las instancias de datos para estimar la *ganancia de información* para cada posible nueva rama.

Para mitigar estos problemas, proponen en "*LightGBM: A Highly Efficient Gradient Boosting Decision Tree*" (Guolin Ke, 2017) dos técnicas novedosas llamadas:

a) *Gradient-based One-Side Sampling* (GOSS), en la que haciendo uso de grandes gradientes, excluyen proporciones significativas de líneas de datos para calcular la *ganancia de información*, mencionado en el párrafo anterior y

b) *Exclusive Feature Bundling* (EFB), en la que se reducen atributos mutuamente exclusivos (que raramente toman valores distinto de cero simultáneamente)

Algunos estudios ya implementan la librería LightGBM, pero en el ámbito de la espectroscopia de suelo (Lanfa Liu, 2017)], la música (Eduardo Fonseca, 2017) y experimentos de física de partículas [Belavin2017]. Esta librería es utilizada en algunos concursos de problemas de regresión. Por ejemplo el ganador de un concurso de e-commerce de 2017 utilizaba la librería LightGBM [Wen2017].

3 Competición Corporación Favorita

El objetivo de este trabajo es generar un modelo que permita predecir las ventas de la Corporación Favorita (http://www.corporacionfavorita.com), con el mínimo de error posible.

Para la predicción se ha generado un código disponible en 9 Anexos. Como el código y los datos son abiertos, el resultado es completamente reproducible.

Para la evaluación de la eficacia de la predicción se ha testeado el algoritmo con el 31% de los datos de test. Hay que tener en cuenta no obstante, que para la clasificación final se usa el 69% de los datos de test así que la clasificación puede ser diferente.

3.1 Reglas más relevantes

- URL: https://www.kaggle.com/c/favorita-grocery-sales-forecasting/rules
- Fecha de Inicio: 19 de octubre de 2017
- Fecha final de entrega: 15 de enero de 2017
- Título de la Competición: Corporación Favorita Grocery Sales Forecasting
- Patrocinador: Corporación Favorita C.A.
- Primer premio: $15.000
- Segundo premio: $10.000
- Tercer premio: $5.000
- Datos Externos: Permitidos si no pertenecen a Corporación Favorita ni otra entidad en la misma línea de negocio. La fuente de datos externos se ha de validar previamente con los organizadores de la competencia.
- Reglas generales: https://www.kaggle.com/c/favorita-grocery-sales-forecasting/rules

3.2 Resultados obtenidos en la competición

El objetivo de este proyecto fin de máster es predecir las ventas de la Corporación Favorita propuesto por Kaggle https://www.kaggle.com/c/favorita-grocery-sales-forecasting/leaderboard.

Para participar en la competición el usuario utilizado ha sido gkreplak https://www.kaggle.com/gkreplak, que ha quedado en la posición 269 del ranking privado, utilizando el 69% de los datos de test, de un total de 1675 participantes, lo que supone haber quedado entre el 17% superior.

El valor de la métrica de evaluación NWRMSLE, cuyo cálculo se describe en el apartado 3.5 Evaluación fue de 0.522, habiendo procesado 29 predicciones durante 10 días.

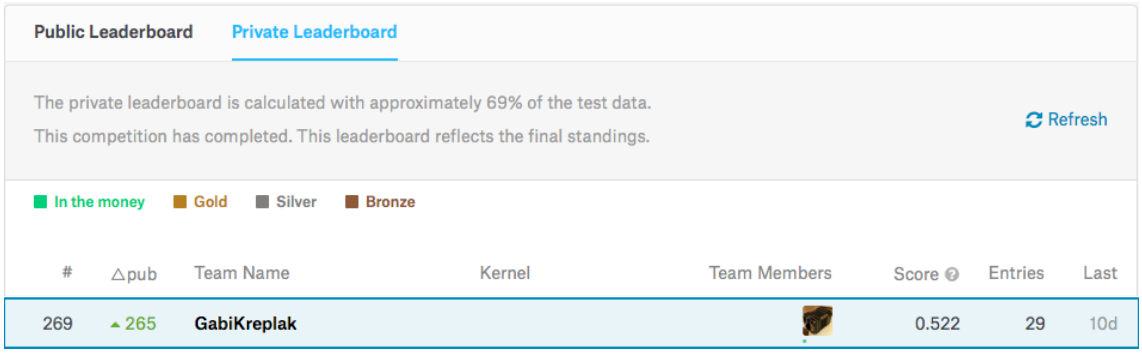


Figura 1: Score Privado

Respecto al ranking público, es decir utilizando el 31% de los datos de test, que no es el utilizado en la puntuación final, la posición fue 265/1675, con un valor de evaluación de 0.513.

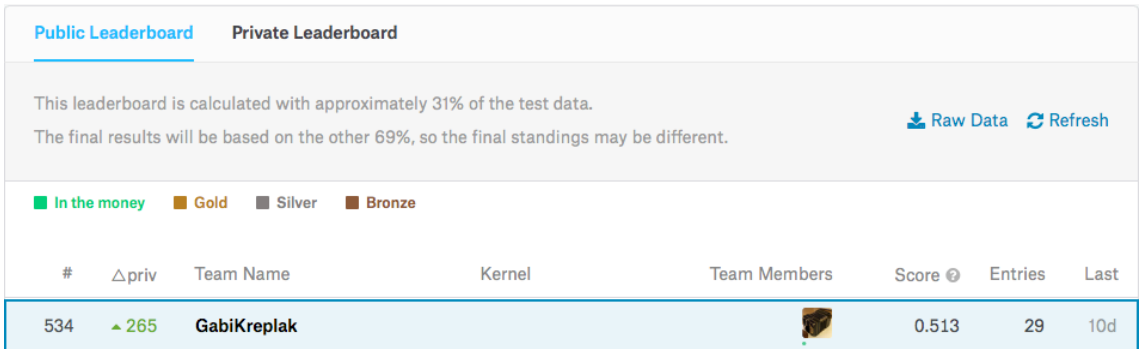


Figura 2: Score Público

3.3 Descripción de los Datos de Entrada

En esta competición, hay que predecir las ventas de unidades por miles de artículos vendidos en diferentes tiendas Favorita ubicadas en Ecuador. Los datos de capacitación incluyen fechas, información de la tienda y del artículo, ya sea que se promocione ese artículo, así como las ventas de la unidad. Los archivos adicionales incluyen información complementaria que puede ser útil para construir sus modelos.

Atendiendo a que se obtienen unos resultados bastante optimizados y debido a restricciones materiales y de tiempo, este trabajo solo utilizará el archivo train.csv para el aprendizaje automático del algoritmo de predicción, descartándose el uso del resto de información complementaria que, no obstante, se describe en el capítulo 4 Análisis Exploratorio de Datos.

Descripciones de archivos e información:

train.csv

- Datos de training, que incluyen *unit_sales by date, store_nbr*, y *item_nbr* y un identificador *id*.
- El objetivo *unit_sales* que puede ser un entero (e.g., bolsa de patatas) o número real (e.g., 1.5 kg de queso).
- Números negativos de *unit_sales* representan las devoluciones de un producto.

- La columna *onpromotion* column dice si un *item_nbr* estaba en promoción en una fecha específica *date* y *store_nbr*.
- Aproximadamente 16% de los valores *onpromotion* de este fichero son *NaN*.
- NOTA: Los datos de entrenamiento no incluyen filas para los artículos que tenían cero ventas por unidad para una combinación de tienda / fecha. No hay información sobre si el artículo estaba o no disponible para la tienda en la fecha, y los equipos deberán decidir la mejor manera de manejar esa situación. Además, hay una pequeña cantidad de elementos vistos en los datos de capacitación que no se ven en los datos de prueba.

test.csv

- Datos de Test, con combinaciones *date, store_nbr, item_nbr* que tienen que ser predichas, con la información *onpromotion*.
- NOTA: Los datos de prueba tienen una pequeña cantidad de elementos que no están contenidos en los datos de entrenamiento. Parte del ejercicio será predecir un nuevo artículo de ventas basado en productos similares.
- La división de tabla de clasificación pública / privada se basa en el tiempo. Todos los artículos en la división pública también se incluyen en la división privada.

sample_submission.csv

- Un ejemplo de fichero de envío.

stores.csv

- Metadato de la tienda, incluye city, state, type, y cluster.
- cluster es un conjunto de tiendas similares.

items.csv

- Metadatos del ítem, incluye family, class y perishable.
- NOTA: los artículos marcados como perecederos tienen un peso de 1,25; de lo contrario, el peso es 1.0.

transactions.csv

- El numero de transacciones de ventas por cada combinación *date, store_nbr*. Sólo incluido para el período de tiempo de los datos de entrenamiento.

oil.csv

- Precio diario del petróleo Incluye valores durante el período de tiempo del tren y de la prueba. (Ecuador es un país dependiente del petróleo y su salud económica es altamente vulnerable a los choques en los precios del petróleo).

holidays_events.csv

- Holidays y Events, nacionales, provinciales y locales

- NOTA: prestar especial atención a la columna transferida. Las vacaciones que se transfieren oficialmente caen en ese día calendario, pero el gobierno las cambió a otra fecha. Un día transferido es más como un día normal que un día festivo.
- Los días festivos adicionales son días que se agregan a las vacaciones regulares del calendario, por ejemplo, como suele ocurrir en Navidad (lo que hace que la Navidad sea un feriado).

Notas adicionales

- Los salarios en el sector público se pagan cada dos semanas el día 15 y el último día del mes. Las ventas de supermercados podrían verse afectadas por esto.
- Un terremoto de magnitud 7.8 sacudió a Ecuador el 16 de abril de 2016. La gente se unió en esfuerzos de ayuda para donar agua y otros productos de primera necesidad que afectaron en gran medida las ventas de los supermercados durante varias semanas después del terremoto.

3.4 Descripción de los Datos de Salida

Para cada identificación en el conjunto de prueba, se ha predicho la unidad de ventas. Dado que la métrica utiliza ln (y + 1), las predicciones se validan para garantizar que no haya predicciones negativas. El archivo enviado debe contener un encabezado y tener el siguiente formato:

id, unidad de ventas
125497040,2.5
125497041,0.0
125497042,27.9
etc.

3.5 Evaluación

Las predicciones se evalúan utilizando el Normalized Weighted Root Mean Squared Logarithmic Error (NWRMSLE), calculado de la siguiente manera:

$$NWRMSLE = \sqrt{\frac{\sum_{i=1}^{n} w_i \left(\ln\left(\hat{y}_i + 1 \right) - \ln\left(y_i + 1 \right) \right)^2}{\sum_{i=1}^{n} w_i}}$$

Ecuación 1: NWRMSLE

donde para la fila i, $\hat{y}$ es la venta de unidad predicha de un artículo e y_i es la unidad de ventas real; n es el número total de filas en el conjunto de test.

Los pesos, w_i, se pueden encontrar en el archivo items.csv (consulte la página de datos). Los artículos perecederos tienen un peso de 1.25 donde todos los demás artículos tienen un peso de 1.00.

Esta métrica es adecuada para predecir valores en un amplio rango de órdenes de magnitud. Evita penalizar las grandes diferencias en la predicción cuando tanto el número predicho como el verdadero son grandes: predecir 5 cuando el

valor verdadero es 50 se penaliza más que la predicción de 500 cuando el valor verdadero es 545.

4 Análisis Exploratorio de Datos

4.1 Estructura de los archivos de datos de entrada

Los datos de entrenamiento constan de 6 archivos cuyos atributos se relacionan como se indica en la Figura 3: Estructura de los Datos de Entrada.

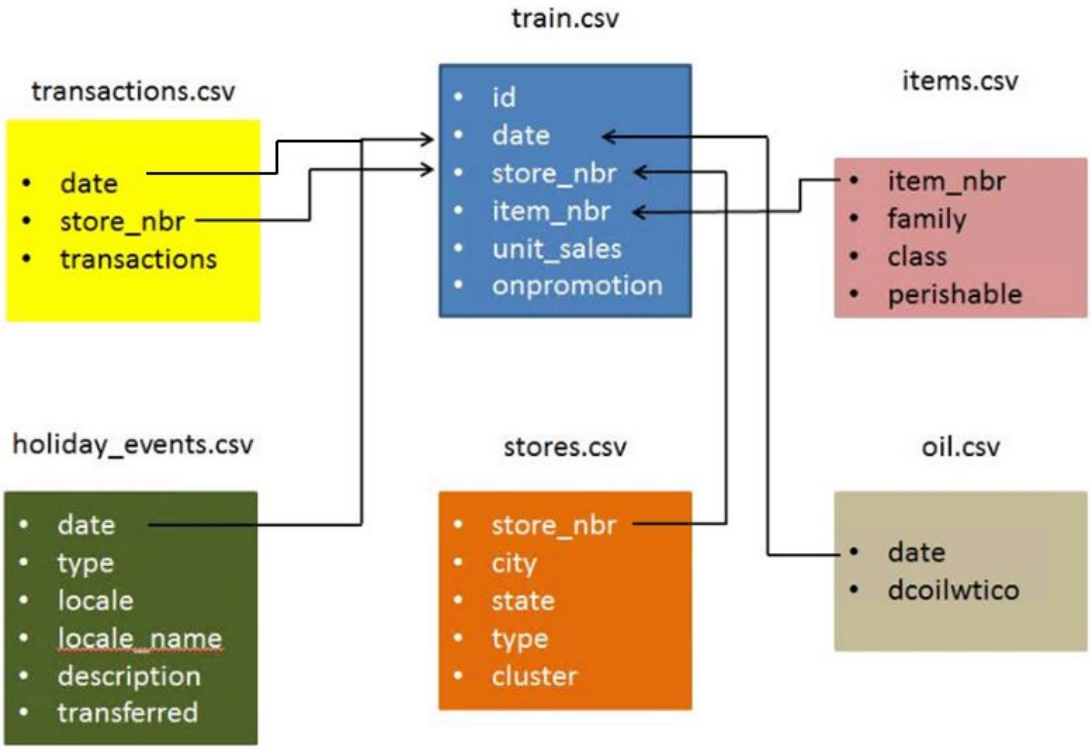


Figura 3: Estructura de los Datos de Entrada.

Fuente: https://www.kaggle.com/jeru666/all-csv-files-a-glance

Recuento:

```
There are 4100 rows and 4 columns in the items data
There are 350 rows and 6 columns in the holiday_events data
There are 54 rows and 5 columns in the stores data
There are 1218 rows and 2 columns in the oil data
There are 83488 rows and 3 columns in the transactions data
There are 6000000 rows and 6 columns in the train data
```

4.2 Archivo train.csv

Cada línea supone la venta de un único producto. Como ejemplo 5 líneas:

	id	date	store_nbr	item_nbr	unit_sales	onpromotion
0	0	2013-01-01	25	103665	7.0	NaN
1	1	2013-01-01	25	105574	1.0	NaN
2	2	2013-01-01	25	105575	2.0	NaN
3	3	2013-01-01	25	108079	1.0	NaN
4	4	2013-01-01	25	108701	1.0	NaN

Tabla 2: train.csv, 125 millones de líneas x 6 columnas

4.3 Archivo transactions.csv

Número de transacciones por tienda y día desde el año 2013. 5 líneas de ejemplo:

	date	store_nbr	transactions
0	2013-01-01	25	770
1	2013-01-02	1	2111
2	2013-01-02	2	2358
3	2013-01-02	3	3487
4	2013-01-02	4	1922
5	2013-01-02	5	1903

Tabla 3: transactions.csv, 83.500 líneas x 3 columnas

4.4 Archivo holiday_events.csv

Días festivos desde el año 2012 hasta la actualidad.

	date	type	locale	locale_name	description	transferred
0	2012-03-02	Holiday	Local	Manta	Fundacion de Manta	False
1	2012-04-01	Holiday	Regional	Cotopaxi	Provincializacion de Cotopaxi	False
2	2012-04-12	Holiday	Local	Cuenca	Fundacion de Cuenca	False
3	2012-04-14	Holiday	Local	Libertad	Cantonizacion de Libertad	False
4	2012-04-21	Holiday	Local	Riobamba	Cantonizacion de Riobamba	False
5	2012-05-12	Holiday	Local	Puyo	Cantonizacion del Puyo	False

Tabla 4: holiday_events.csv, 350 líneas x 6 columnas

4.5 Archivo oil.csv

Cotización del petróleo desde el año 2013

	date	dcoilwtico
0	2013-01-01	NaN
1	2013-01-02	93.14
2	2013-01-03	92.97
3	2013-01-04	93.12
4	2013-01-07	93.20
5	2013-01-08	93.21

Tabla 5: oil.csv, 1.218 x 2 col

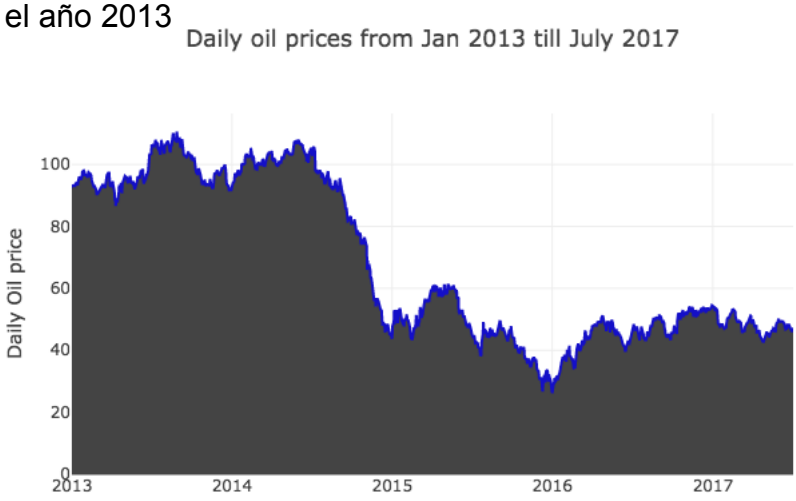


Figura 4: Precios de petróleo

4.6 Archivo items.csv

Catálogo y clasificación de productos a la venta.

La información de los productos se limita a asignar familia a cada producto y como se verá abajo, la inmensa mayoría con comestibles. Indica también si son precederos

	item_nbr	family	class	perishable
0	96995	GROCERY I	1093	0
1	99197	GROCERY I	1067	0
2	103501	CLEANING	3008	0
3	103520	GROCERY I	1028	0
4	103665	BREAD/BAKERY	2712	1
5	105574	GROCERY I	1045	0

Tabla 6: items.csv, 4.100 productos x 4 attr

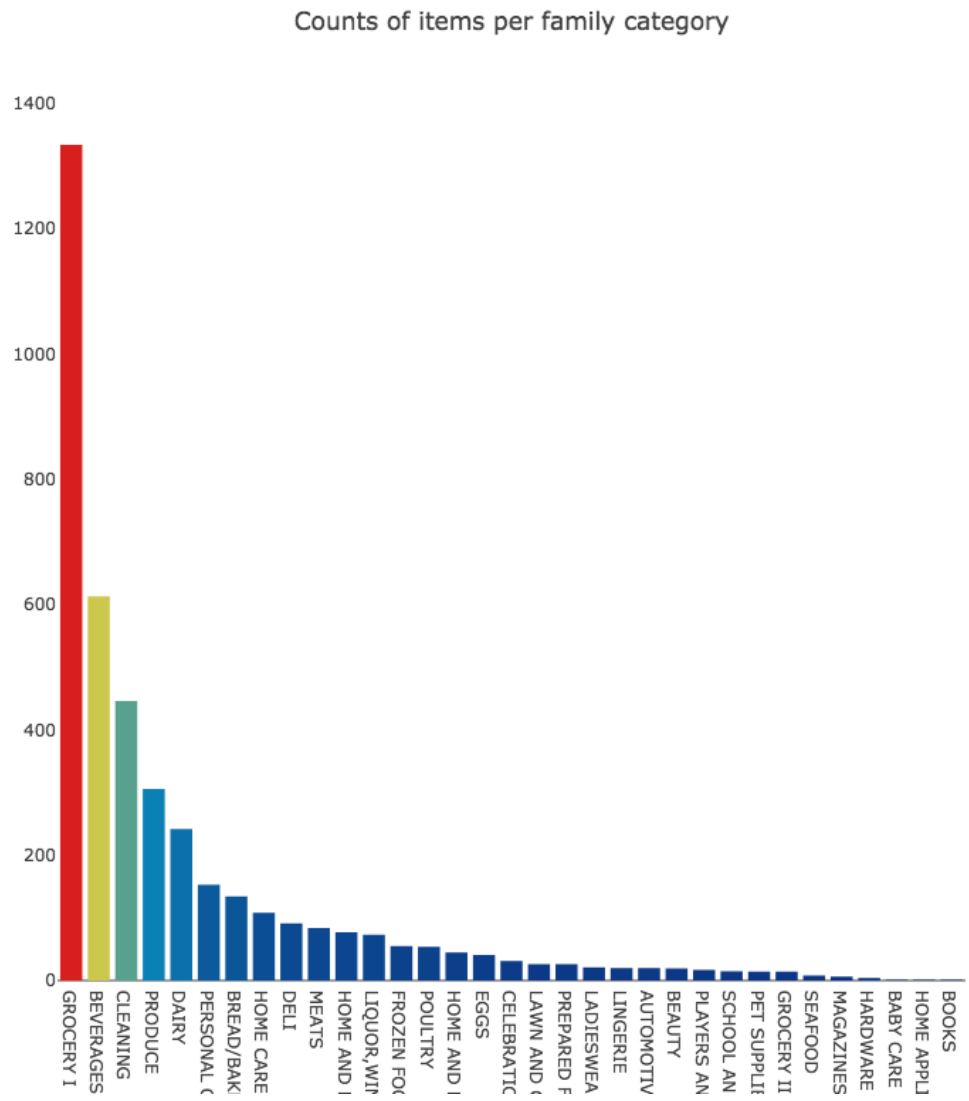


Figura 5: Número de productos por cada categoría

Este gráfico interactivo destaca la proporción logarítmica de numero de productos, siendo los mas numerosos los comestibles, bebidas, limpieza y lácteos.

4.7 Archivo stores.csv

Se trata de 54 tiendas con indicación de la ciudad y el estado. También se incluye una agrupación de tiendas en 17 grupos (clusters) y una clasificación de tienda entre 8 tipos.

18 tiendas están en Quito y 8 en Guayaquil, las 28 restantes están distribuidas entre 20 ciudades distintas.

	store_nbr	city	state	type	cluster
0	1	Quito	Pichincha	D	13
1	2	Quito	Pichincha	D	13
2	3	Quito	Pichincha	D	8
3	4	Quito	Pichincha	D	9
4	5	Santo Domingo	Santo Domingo de los Tsachilas	D	4
5	6	Quito	Pichincha	D	13

Tabla 7: stores.csv, 54 tiendas x 5 atributos

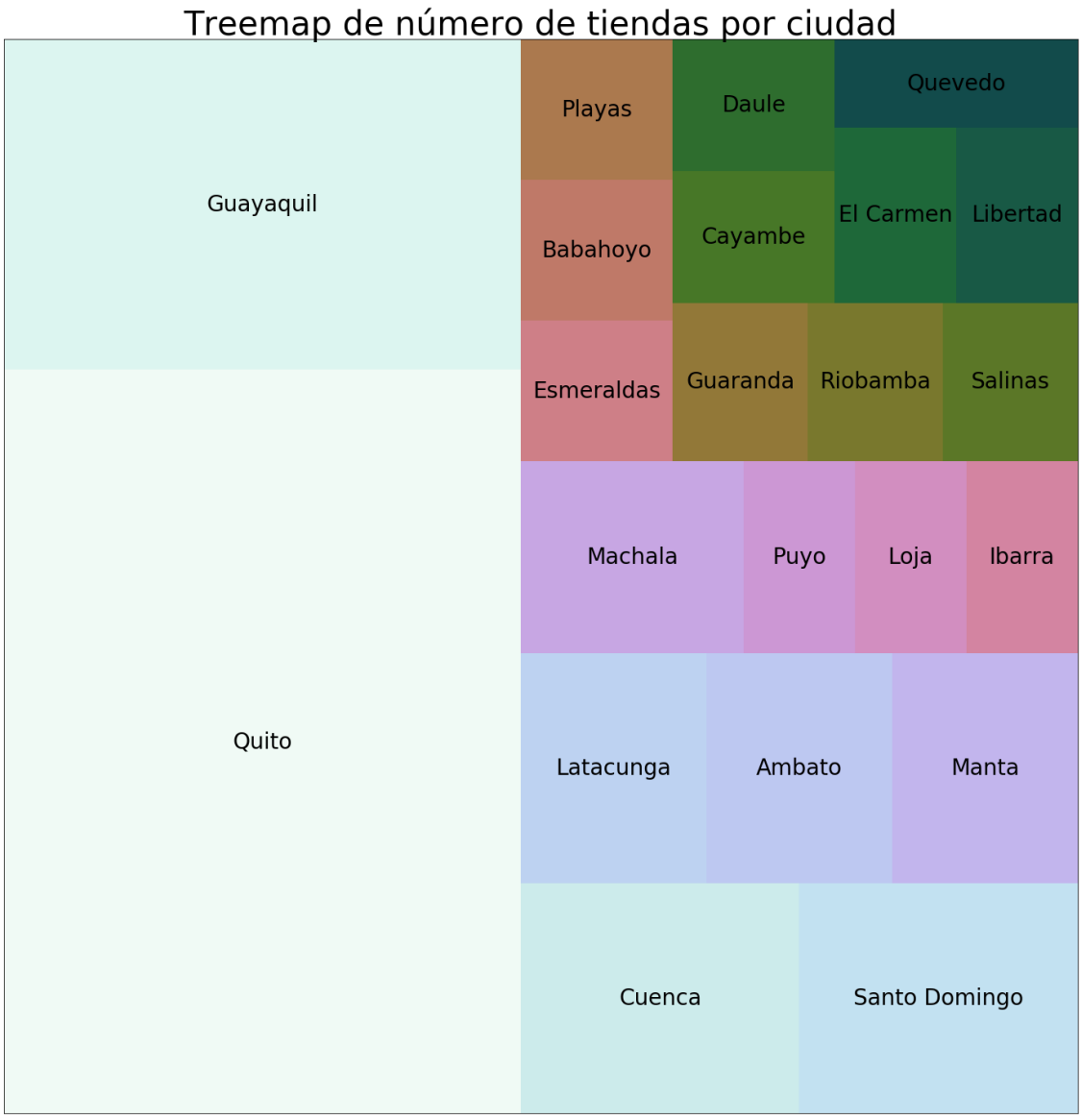


Figura 6 : Treemap de tiendas/ciudad

A continuación se incluye un diagrama indicativo de la distribución del número de tienda en cada uno de los 16 grupos (clusters) predefinidos.

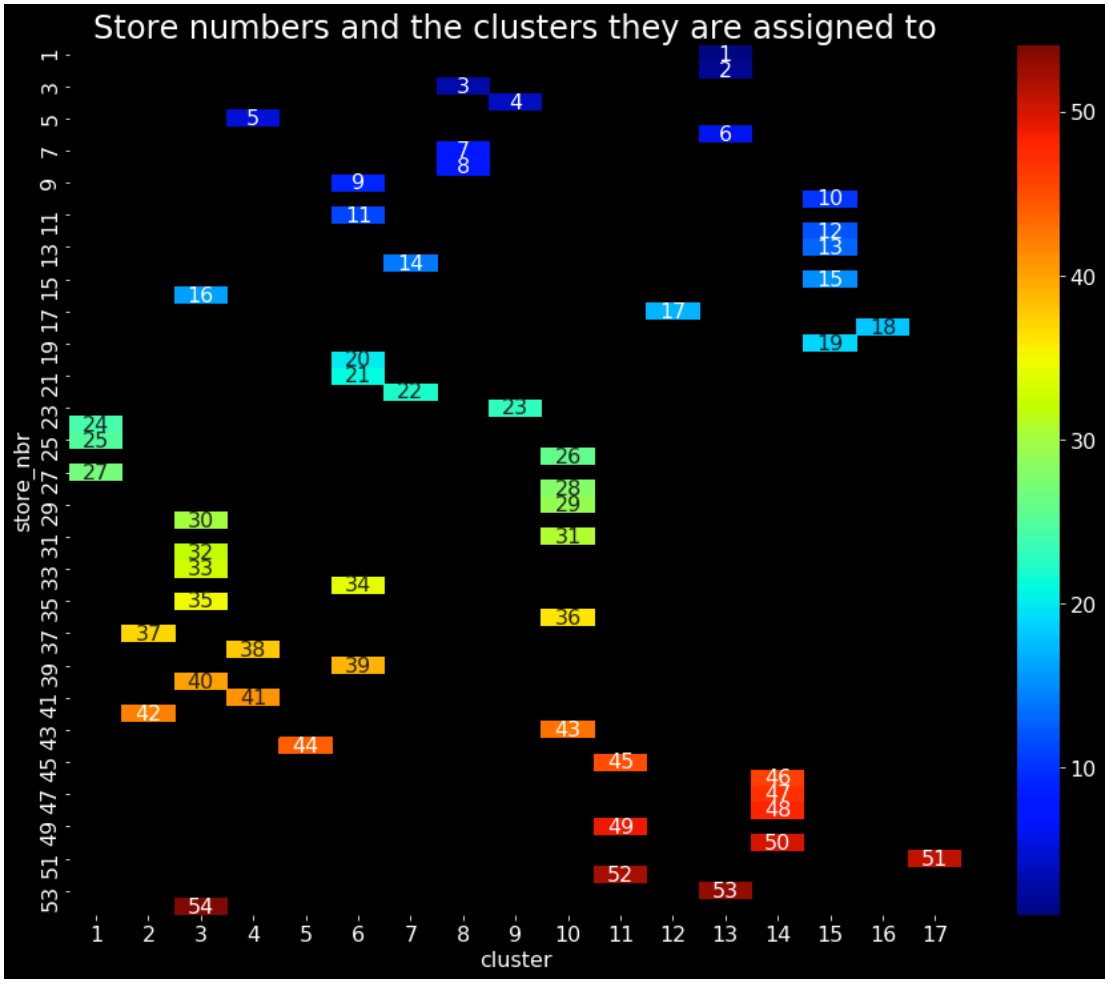


Figura 7: Segmentación de las tiendas

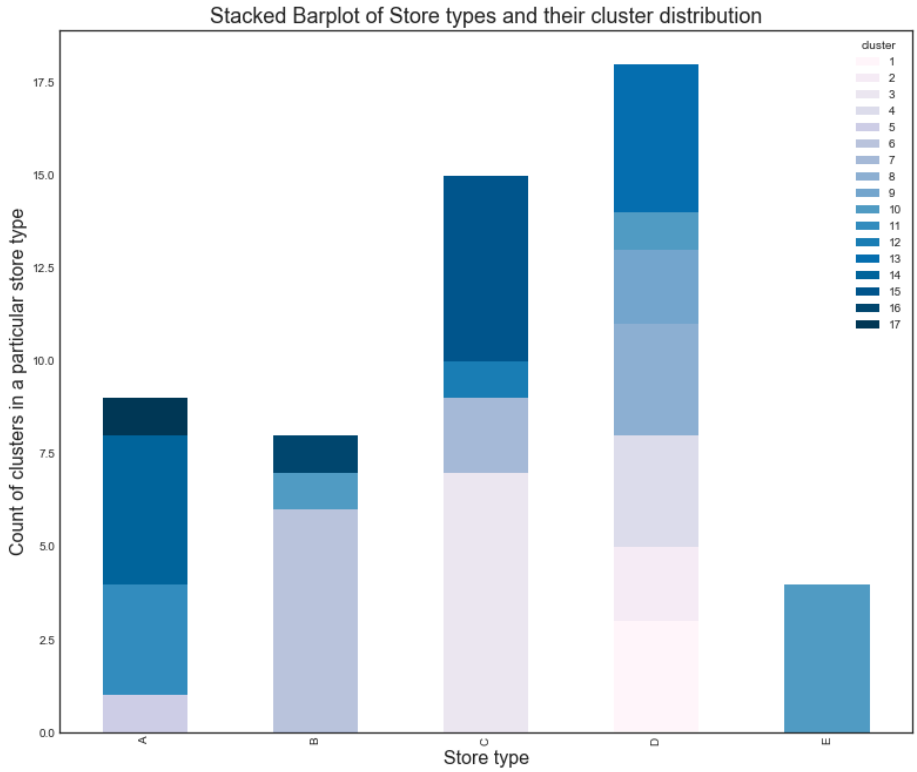


Figura 8: Clasificación de tiendas

4.8 Archivo holiday_events.csv

Este archivo contiene una línea por cada día festivo en Ecuador, que puede ser nacional, estatal o local. Las ventas son muy sensibles a los días festivos, produciéndose variaciones significativas.

	date	type	locale	locale_name	description	transferred
0	2012-03-02	Holiday	Local	Manta	Fundacion de Manta	False
1	2012-04-01	Holiday	Regional	Cotopaxi	Provincializacion de Cotopaxi	False
2	2012-04-12	Holiday	Local	Cuenca	Fundacion de Cuenca	False
3	2012-04-14	Holiday	Local	Libertad	Cantonizacion de Libertad	False
4	2012-04-21	Holiday	Local	Riobamba	Cantonizacion de Riobamba	False
5	2012-05-12	Holiday	Local	Puyo	Cantonizacion del Puyo	False

Tabla 8: holiday_events.csv, 350 líneas x 6 columnas

4.9 Archivo transactions.csv

Un dato importante a considerar es que la ventana de tiempo de las transacciones incluidas en este apartado corresponde al tiempo de los datos de training.

	date	store_nbr	transactions
0	2013-01-01	25	770
1	2013-01-02	1	2111
2	2013-01-02	2	2358
3	2013-01-02	3	3487
4	2013-01-02	4	1922

Tabla 9: transactions.csv, 83.500 líneas x 3 columnas

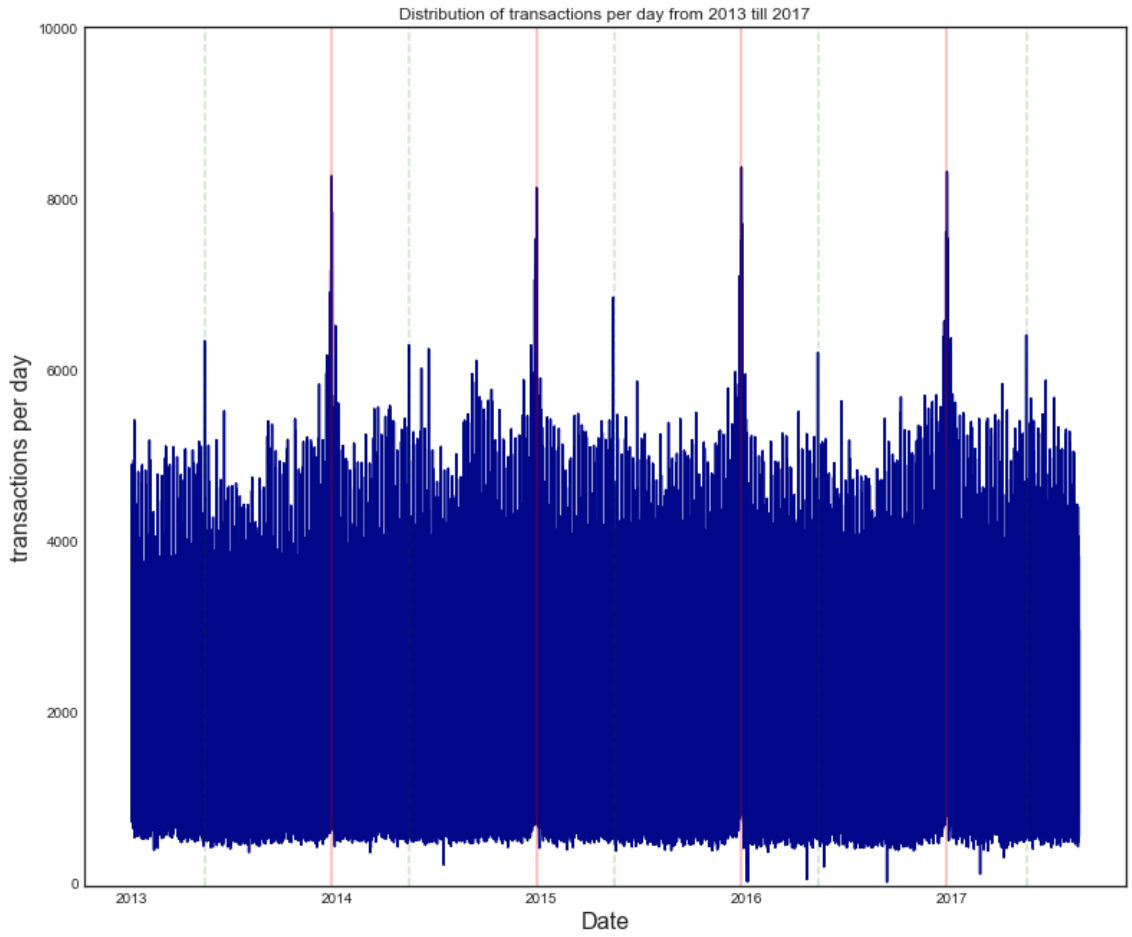


Figura 9: Distribución de transacciones por día

4.10 Archivo test.csv

Este archivo posee la misma estructura que train.csv, excepto la columna unit_sales, que no existe en este fichero.

Básicamente contiene los datos de numero de ítem, numero de tienda, fecha y en promoción. Con estos identificadores, se procederá a la predicción de las unidades a ser vendidas en esa fecha para cada producto en cada tienda que lo ponga a la venta.

El archivo contiene 3.370.464 de líneas que corresponden a cada uno de los productos existentes en cada tienda durante los 16 días desde 18/8/2017 hasta 31/8/2017.

test_ids

	id	date	store_nbr	item_nbr	onpromotion
0	125497040	2017-08-16 00:00:00	1	96995	False
1	125497041	2017-08-16 00:00:00	1	99197	False
2	125497042	2017-08-16 00:00:00	1	103501	False
3	125497043	2017-08-16 00:00:00	1	103520	False
4	125497044	2017-08-16 00:00:00	1	103665	False
5	125497045	2017-08-16 00:00:00	1	105574	False
6	125497046	2017-08-16 00:00:00	1	105575	False
7	125497047	2017-08-16 00:00:00	1	105576	False
8	125497048	2017-08-16 00:00:00	1	105577	False
9	125497049	2017-08-16 00:00:00	1	105693	False
10	125497050	2017-08-16 00:00:00	1	105737	False
11	125497051	2017-08-16 00:00:00	1	105857	False
12	125497052	2017-08-16 00:00:00	1	106716	False
13	125497053	2017-08-16 00:00:00	1	108079	False
14	125497054	2017-08-16 00:00:00	1	108634	False
15	125497055	2017-08-16 00:00:00	1	108696	False
16	125497056	2017-08-16 00:00:00	1	108698	False
17	125497057	2017-08-16 00:00:00	1	108701	True
18	125497058	2017-08-16 00:00:00	1	108786	False
19	125497059	2017-08-16 00:00:00	1	108797	True
20	125497060	2017-08-16 00:00:00	1	108831	False
21	125497061	2017-08-16 00:00:00	1	108833	False
22	125497062	2017-08-16 00:00:00	1	108862	False
23	125497063	2017-08-16 00:00:00	1	108952	False
24	125497064	2017-08-16 00:00:00	1	111223	False

Tabla 10: test.csv 3.370.464 líneas x 5 columnas

5 Estudio Predictivo

5.1 Algoritmo Light Gradient Boosting Machine

Como se dice más arriba, el algoritmo de predicción escogido para el cálculo de las predicciones es: Light Gradient Boosting Machine (LGBM).

Es un algoritmo de tipo Gradient Boosting basado en árboles de decisión desarrollado en el marco del proyecto DMTK de Microsoft.

http://github.com/microsoft/dmtk

Las ventajas más importantes de ésta herramienta son:

- Rapidez en el aprendizaje con alta eficiencia
- Precisión mejorada respecto a otros GBM
- Soporta proceso paralelo y GPU
- Adecuado para datasets de gran tamaño.

La mayoría de algoritmos de árboles crecen por nivel, tal como se indica a continuación:

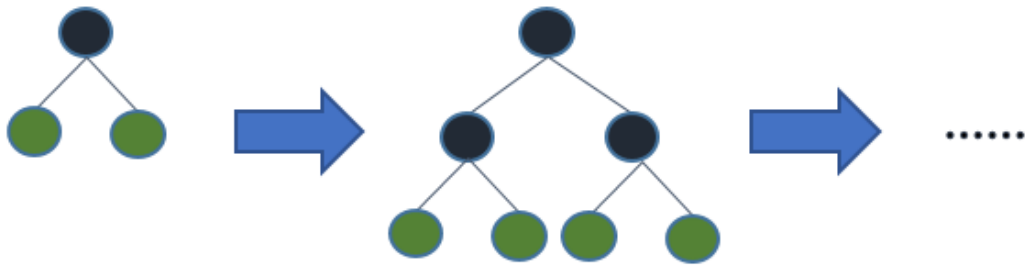


Figura 10: Crecimiento del árbol por niveles.

Fuente: https://github.com/Microsoft/LightGBM/blob/máster/docs/Features.rst (Microsoft, 2018)

En cambio, los árboles LightGBM crecen por hojas, según la Figura 11: Crecimiento LGBM por hojas

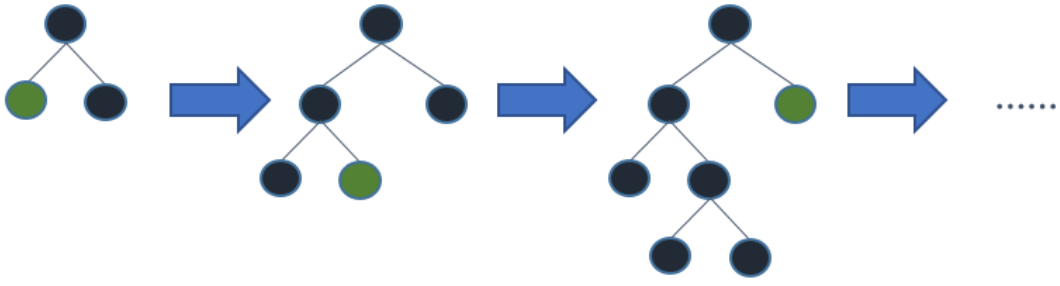


Figura 11: Crecimiento LGBM por hojas

LGBM escogerá para crecer la hoja con mayor delta loss, reduciendo el mismo respecto a un modelo de árbol de crecimiento por niveles.

LGBM no es adecuado cuando el dataset es pequeño porque el crecimiento por hojas conduce al sobre-entrenamiento. En ese caso conviene limitar la profundidad del árbol mediante el parámetro max_depth.

Una ventaja importante del LGBM es la mayor rapidez de convergencia respecto a árboles de crecimiento por nivel.

5.2 Ajuste de parámetros LGBM

Algunos parámetros importantes:

- **num_leaves**: Es el principal parámetro para controlar la complejidad del árbol. Tiene que ser menor de $2^{(max\_depth)}$ para evitar sobre-entrenamiento.
- **min_data_in_leaf**: Importante para controlar el sobre-entrenamiento y depende de num_leaves. Un número alto evita el crecimiento excesivo del árbol, aunque puede producir sub-entrenamiento.
- **max_depth**: Límite explícito de la profundidad del árbol.

Para aumentar la velocidad:

- Utilizar bagging activando de **bagging_fraction** y **bagging_freq**
- Utilizar feature sub-sampling activando **feature_fraction**
- Utilizar **max_bin** pequeño
- Guardar datos binarios usando **save_binary**

Para aumentar la precisión:

- Utilizar max_bin grande (puede ralentizar)
- Usar bajo learning_rate con alto num_iterations
- Usar alto **num_leaves** (puede causar sobre-entrenamiento)
- Probar **dart**

Para gestionar el sobre-entrenamiento

- Reducir **max_bin**
- Reducir num_leaves
- Usar min_data_in_leaf y min_sum_hessian_in_leaf
- Utilizar bagging activando de **bagging_fraction** y **bagging_freq**

- Utilizar feature sub-sampling activando **feature_fraction**
- Probar lambda_l1, lambda_l2 y min_gain_to_split para regularización
- Probar **max_depth** para evitar crecimiento de la profundidad del árbol

5.3 Descripción del programa

En el anexo, se adjunta copia del script escrito en python y basado originalmente en una publicación en la plataforma Kaggle de Ceshine Lee, aunque transformado a efectos de mejorar la exactitud de las predicciones. (Lee, 2017)

Es un programa sencillo que realiza movimientos de datos a veces voluminosos y complejos. Los pasos en los que se puede dividir son los siguientes:

1. **Lectura**

    a. Lectura de datasets de entrada: Los archivos train.csv, test.csv y ítems.csv son leídos y almacenados en los dataframes *df_train, df_test* e *items*.

    b. De *df_train* se descartan las operaciones que no corresponden al año 2017 y quedando los datos útiles en el dataframe *df_2017*.

    c. Se crea el dataset *promo_2017* que contiene 167515 líneas que corresponden a las distintas combinaciones tienda/producto que existen en los dataframes *df_train* y *df_test*. *promo_2017* contiene 243 columnas correspondiente a cada uno de los días desde el 1/1/2017 hasta el 31/8/2017. El contenido de *promo_2017* es binario. Informa sobre los productos/tienda en promoción cada uno de los días.

2. **Preparación Dataset**

    A continuación se detalla la preparación de los datasets necesarios para la ejecución del entrenamiento y predicción del algoritmo LGBM.

    Se utiliza respectivamente como predictores y etiquetas X_train y Y_train para aprendizaje y X_val y Y_val para validación.

    Para el cálculo de las predicciones finales para enviar a Kaggle, se dispondrá de X_test.

    a. Predictores Entrenamiento X_train

    El formato del dataset de entrenamiento del algoritmo LGBM sufre una transformación radical respecto a la estructura de atributos original que se puede ver en la Figura 3: Estructura de los Datos de Entrada.

Esta estructura original constaba de unas 125 millones de líneas correspondientes a cada referencia de ítem vendida en cada tienda durante los años 2016 y hasta agosto de 2017.

A efectos de esta predicción de ventas se descarta la información del año 2016 y se trabaja únicamente con los 59 millones de registros correspondientes al año 2017. Las columnas originales de entrenamiento son: *date, store_nbr y onpromotion.* La etiqueta o target es el atributo: *unit_sales.*

La transformación aplicada en este dataset original produce un nuevo dataset tal como se describe en la Tabla 11: Estructura de X_train.

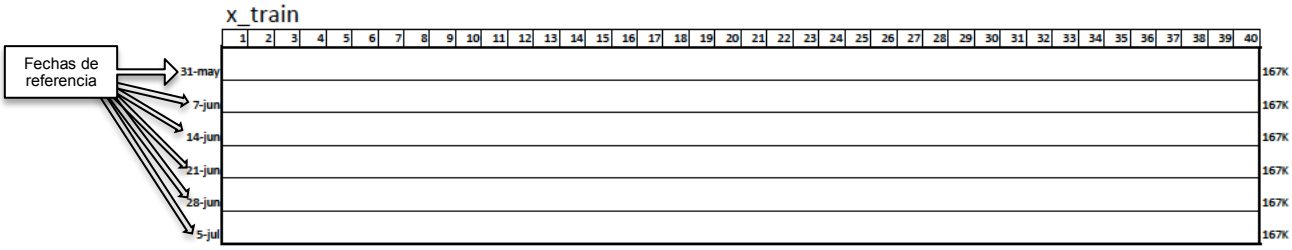


Tabla 11: Estructura de X_train

Este nuevo dataset consta, para cada una de las fechas de referencia y cada uno de los 167.515 items/tienda de 40 columnas según se detalla a continuación:

Nombre atributo	Atributo original	Agregación	Días antes de fecha referencia
day_1_2017	*unit_sales*		0
mean_3_2017	*unit_sales*	Promedio	3
mean_7_2017	*unit_sales*	Promedio	7
mean_14_2017	*unit_sales*	Promedio	14
mean_30_2017	*unit_sales*	Promedio	30
mean_60_2017	*unit_sales*	Promedio	60
mean_140_2017	*unit_sales*	Promedio	140
promo_14_2017	*promo*	Suma	14
promo_60_2017	*promo*	Suma	60
promo_140_2017	*promo*	Suma	140
mean_4_dow0_2017	*unit_sales*	Promedio ventas de domingos	

		durante 4 sem. Antes fecha ref.
mean_4_dow1_2017	*unit_sales*	Promedio ventas de lunes durante 4 sem. Antes fecha ref.
mean_4_dow2_2017	*unit_sales*	Promedio ventas de martes durante 4 sem. Antes fecha ref.
mean_4_dow3_2017	*unit_sales*	Promedio ventas de miércoles durante 4 sem. Antes fecha ref.
mean_4_dow4_2017	*unit_sales*	Promedio ventas de jueves durante 4 sem. Antes fecha ref.
mean_4_dow5_2017	*unit_sales*	Promedio ventas de viernes durante 4 sem. Antes fecha ref.
mean_4_dow6_2017	*unit_sales*	Promedio ventas de sábados durante 4 sem. Antes fecha ref.
mean_20_dow0_2017	*unit_sales*	Promedio ventas de domingos durante 20 sem. Antes fecha ref.
mean_20_dow1_2017	*unit_sales*	Promedio ventas de lunes durante 20 sem. Antes fecha ref.
mean_20_dow2_2017	*unit_sales*	Promedio ventas de martes durante 20 sem. Antes fecha ref.
mean_20_dow3_2017	*unit_sales*	Promedio ventas de miércoles durante 20 sem. Antes fecha ref.
mean_20_dow4_2017	*unit_sales*	Promedio ventas de jueves durante 20 sem. Antes fecha ref.
mean_20_dow5_2017	*unit_sales*	Promedio ventas de viernes durante 20 sem. Antes fecha ref.
mean_20_dow6_2017	*unit_sales*	Promedio ventas de sábados durante 20 sem. Antes fecha ref.
promo_1	*promo*	Estado promo 1 día antes f.r.
promo_2	*promo*	Estado promo 2 días antes f.r.
promo_3	*promo*	Estado promo 3 días antes f.r.
promo_4	*promo*	Estado promo 4 días antes f.r.
promo_5	*promo*	Estado promo 5 días antes f.r.
promo_6	*promo*	Estado promo 6 días antes f.r.

promo_7	*promo*	Estado promo 7 días antes f.r.
promo_8	*promo*	Estado promo 8 días antes f.r.
promo_9	*promo*	Estado promo 9 días antes f.r.
promo_10	*promo*	Estado promo 10 días antes f.r.
promo_11	*promo*	Estado promo 11 días antes f.r.
promo_12	*promo*	Estado promo 12 días antes f.r.
promo_13	*promo*	Estado promo 13 días antes f.r.
promo_14	*promo*	Estado promo 14 días antes f.r.
promo_15	*promo*	Estado promo 15 días antes f.r.
promo_16	*promo*	Estado promo 16 días antes f.r.

Tabla 12: Atributos dataset entrenamiento

b. Etiquetas Entrenamiento Y_train

Para las etiquetas o target, se prepara un dataframe correspondiente a las ventas durante los 16 días anteriores a la fecha de referencia, según consta en la Tabla 13: Estructura dataframe Y_train.

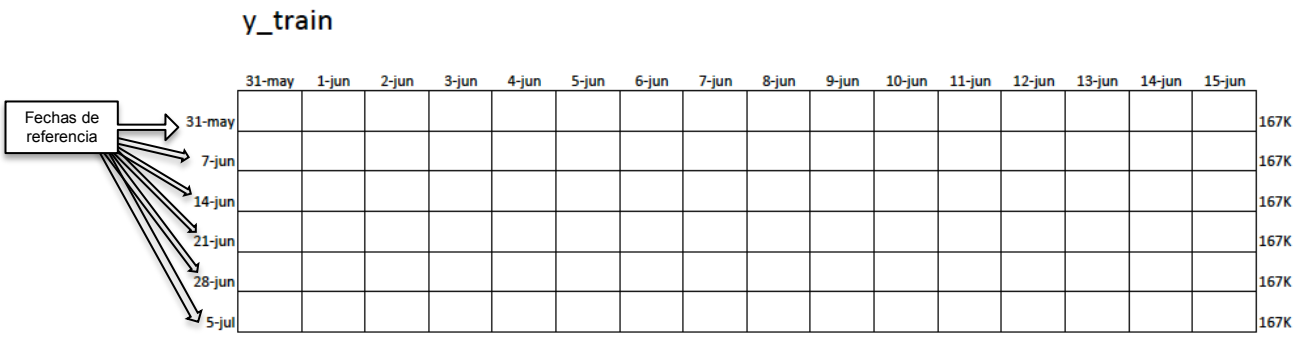


Tabla 13: Estructura dataframe Y_train

Como se verá mas adelante, para calcular las predicciones de unidades vendidas de cada producto/tienda (atributo: *unit_sales*) durante el período correspondiente a los datos de test, se utilizan 16 etiquetas por cada producto/tienda correspondientes a los 15 días posteriores a la fecha referencia.

Finalmente, los predictores y etiquetas de entrenamiento son el conjunto de 6 grupos formados cada uno por las 167.515 líneas generadas según se describe en este apartado. Cada uno de estos grupos se genera con fechas de referencia semanal de 6 semanas posteriores a la fecha de referencia inicial.

c. Predictores y Etiquetas Validación

Estos datasets tienen un uso interno para el cálculo de las predicciones del algoritmo LGBM, como se verá en el apartado correspondiente.

En cuanto a su estructura de columnas, son idénticos a los datasets de predictores y etiquetas de entrenamiento, descritos en: Predictores Entrenamiento X_train y Etiquetas Entrenamiento Y_train.

Estos datasets de validación tienen un tamaño de 167.515 líneas correspondientes a las combinaciones producto/tienda de fecha referencia: 26/7/2017, tal como se refleja en la Tabla 14 y la Tabla 15.

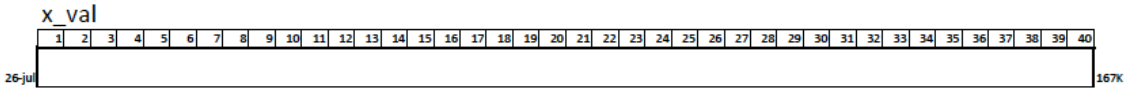


Tabla 14: Estructura de Predictores de Validación

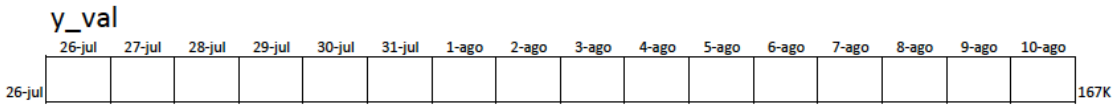


Tabla 15: Estructura de Etiquetas de Validación

d. Predictores de Test

El dataframe de predictores de test se genera de forma análoga a lo descrito en los apartados descriptivos de los dataframes de entrenamiento y validación.

La fecha de referencia utilizada es 16/8/2017

En el caso de los datos de test, no existen etiquetas y la operación de medición de precisión de la predicción la realiza Kaggle utilizando según la Ecuación 1: NWRMSLE

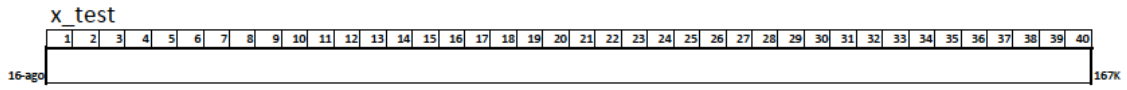


Tabla 16: Estructura de predictores de test

3. Entrenamiento de los modelos

Previo al lanzamiento del aprendizaje del algoritmo LGBM, se definen los parámetros principales tal como se detallan en Ajuste de parámetros LGBM. Estos serán elementos susceptibles de ajuste posterior para obtener una predicción ajustada.

El proceso de entrenamiento de los modelos se ejecuta en un bucle de 16 iteraciones en la que en cada una se utiliza como etiqueta, una de las 16 columnas generadas como se describe en Etiquetas Entrenamiento Y_train.

Para el entrenamiento se utilizan los predictores y las 16 columnas de etiquetas de entrenamiento así como los predictores y la columna de etiquetas de validación.

La estrategia de ajuste seguida para la optimización de los resultados del algoritmo ha estado basada en la modificación intuitiva de los parámetros que se resumen en la Tabla 17: Resumen de Ajustes.

Básicamente, después de reiteradas pruebas con los parámetros más relevantes, que se listan a continuación, escojo los valores que se indican:

a) **Parámetros Básicos:**

- **num_leaves**: para control de complejidad del modelo. Valor: 2^6-1
- **objective**: para determinar si la aplicación es clasificación binaria, clasificación multiclase o regresión. Valor: 'regression'
- **learning_rate**: para determinar el impacto de cada árbol sobre el resultado final. Valor: ver Tabla 17: Resumen de Ajustes
- **num_threads**: Número de hilos para LightGBM. Valor: 4

b) **Parámetros de Control de Aprendizaje:**

- **min_data_in_leaf**: para limitar la profundidad del árbol de forma indirecta. Valor: 300
- **max_depth**: de forma explícita. Valor: 6
- **feature_fraction**: Sólo aplica cuando el parámetro boosting_type es ‘rf’ (random forests). En nuestro caso utilizamos el valor por defecto de boosting_type: ‘gbdt’ (gradient boosting decision tree). Valor: 0.8
- **bagging_fraction**: Porcentaje de datos seleccionados para cada iteración. Valor: 0.8
- **bagging_freq**: Número de iteraciones para ejecutar cada bagging. Valor: 2

c) **Parámetros de Métrica:**

- **metric**: Métrica utilizada para la evaluación interna de resultados. Valor: 'l2' (square loss)

d) **Parámetros de Entrada/Salida:**

- **max_bin**: Número máximo de conjuntos de atributos a ser utilizados. Valor: 500
- **num_boost_round**: Número de iteraciones de boosting. Valor: 100 por defecto que según se puede apreciar en la figura Evolución de la Métrica l2.

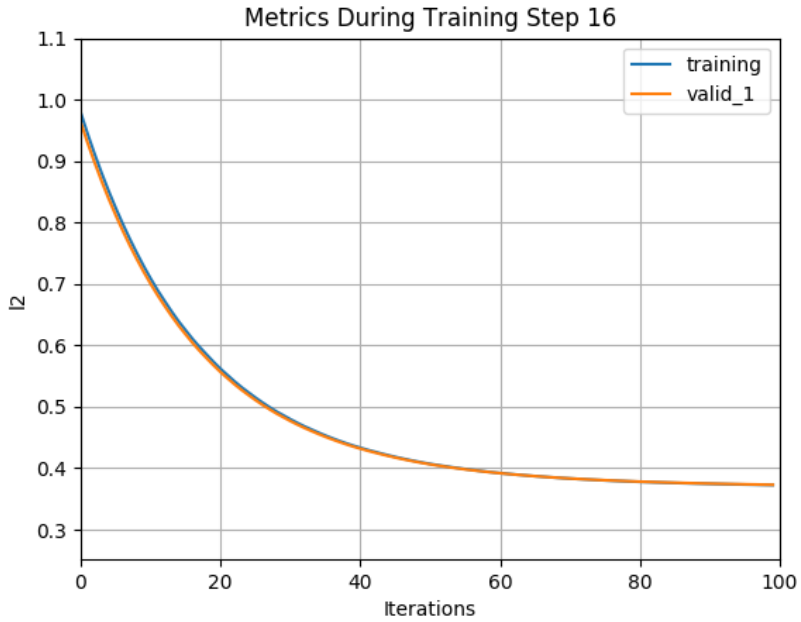


Figura 12: Evolución de la métrica l2

Ajuste del Modelo

Después de diversas pruebas de modificación de los parámetros indicados en el párrafo anterior, resumiendo el proceso, fui cambiando diversas fechas de referencias, rangos de fecha de etiquetas (según se describe en 5.3 Descripción del programa) y learning_rate, obteniendo los resultados siguientes:.

Learning rate	**Fecha Referencia**	**Score**
0.03	21/6/17	**0.513**
0.03	31/5/17	**0.515**
0.025	31/5/17	**0.516**
0.03	20/7/17	
0.01	31/5/17	**0.517**
0.03	26/6/17	**0.518**
0.03	19/6/17	**0.519**
0.03	22/6/17	**0.520**
0.03	18/6/17	**0.523**
0.03	24/6/17	**0.524**

0.1	31/5/17	**0.529**

Tabla 17: Resumen de Ajustes

Importancia de los atributos

El modelo generado por el algoritmo LightGBM contiene un valor indicativo de la importancia de cada atributo en el parámetro .feature_importance.

Recogiendo la importancia de cada atributo en cada una de las 16 iteraciones que corresponden a los datos de los 16 días previos a la fecha ref., se obtiene la siguiente tabla Importancia de los Atributos, de la que se deduce la fuerte influencia de los atributos:

1) mean_14_2017: media de las ventas de cada producto/tienda en los 14 días anteriores a la fecha ref.

2) mean_30_2017: media de las ventas de cada producto/tienda en los 30 días anteriores a la fecha ref.

3) mean_7_2017: media de las ventas de cada producto/tienda en los 7 días anteriores a la fecha ref.

Promedio de Importancia	Etiquetas de columna																
Etiquetas de fila	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Total general
day_1_2017	160.622	56.138	28.217	21.764	13.825	14.335	28.777	26.231	26.877	19.636	16.489	12.356	19.198	44.515	26.142	24.261	33.711
mean_14_2017	4.128.828	4.428.409	4.910.235	6.156.865	5.212.872	5.450.219	4.279.540	3.929.089	2.279.225	2.951.064	1.959.727	1.471.708	1.972.244	1.563.001	2.408.127	1.796.801	3.431.122
mean_140_2017	16.564	20.683	8.100	21.256	16.691	21.636	29.281	32.117	32.160	12.896	29.481	30.213	39.859	65.898	43.291	39.583	28.732
mean_20_dow0_2017	215.805	9.243	7.881	9.387	11.231	9.116	13.678	342.235	12.838	10.461	12.529	11.944	14.363	17.490	584.267	14.812	81.080
mean_20_dow1_2017	2.167	184.470	14.544	5.497	13.766	6.194	14.978	7.558	265.245	24.721	7.685	13.847	9.950	24.163	12.442	342.243	59.342
mean_20_dow2_2017	10.869	21.562	571.342	10.685	10.621	9.581	7.089	22.485	31.042	1.009.301	15.111	12.310	10.961	8.315	34.056	62.810	115.509
mean_20_dow3_2017	3.246	4.108	5.396	407.964	13.069	10.996	8.614	7.335	6.042	7.986	523.440	17.210	9.843	8.341	7.241	6.663	65.468
mean_20_dow4_2017	12.245	15.935	22.210	8.137	615.720	6.359	6.613	12.835	13.593	19.435	10.320	763.844	7.344	7.161	12.795	14.345	96.806
mean_20_dow5_2017	1.796	5.556	11.554	13.674	7.794	196.527	28.732	7.688	6.498	11.128	13.433	8.470	250.839	30.409	8.566	8.249	38.182
mean_20_dow6_2017	3.314	6.947	8.003	10.018	8.468	15.643	267.464	8.697	8.780	9.291	12.595	11.024	17.442	468.941	11.246	7.640	54.720
mean_3_2017	143.005	60.012	55.104	122.137	581.129	493.830	269.890	54.458	39.319	45.448	96.674	395.306	354.048	206.931	59.183	37.836	188.394
mean_30_2017	332.451	557.462	576.342	1.381.768	937.647	1.905.109	2.743.973	3.504.018	4.480.375	3.892.873	6.286.383	3.510.288	6.286.895	6.389.828	5.647.747	5.333.865	3.360.439
mean_4_dow0_2017	147.833	7.065	9.108	7.595	8.079	8.779	8.704	257.829	9.899	12.311	11.147	9.091	11.236	10.817	204.577	9.538	45.851
mean_4_dow1_2017	1.660	60.974	11.113	5.938	5.141	5.878	8.595	6.054	75.202	10.289	5.968	5.294	6.063	8.127	6.164	47.692	16.885
mean_4_dow2_2017	11.807	11.136	670.486	19.624	6.168	7.075	6.560	9.357	11.270	865.967	23.744	5.993	6.692	6.544	13.890	14.362	105.667
mean_4_dow3_2017	1.108	4.796	15.656	597.026	43.326	5.797	5.927	5.223	5.481	12.055	790.104	53.158	7.322	6.276	5.465	6.841	97.848
mean_4_dow4_2017	2.425	7.920	5.994	44.206	4.620.635	8.459	5.793	5.395	6.912	6.346	74.936	5.610.876	7.794	6.416	5.639	7.135	651.680
mean_4_dow5_2017	11.682	5.546	4.732	5.477	6.366	230.780	36.863	33.431	6.423	6.033	4.997	6.477	183.586	20.167	14.381	7.664	36.538
mean_4_dow6_2017	6.039	12.693	5.024	5.686	5.976	19.498	498.967	10.677	15.375	6.182	5.986	6.816	10.313	308.341	9.115	8.591	58.455
mean_60_2017	90.368	109.683	60.996	224.590	100.251	332.175	129.127	186.000	563.591	171.772	548.298	371.389	653.152	303.833	397.852	708.687	309.485
mean_7_2017	5.832.902	3.865.253	3.958.437	3.567.055	1.396.162	2.234.859	1.922.907	2.188.047	1.200.394	1.416.603	1.637.820	693.781	747.839	490.626	806.020	367.355	2.020.379
promo_0	339.622	15.187	15.127	13.442	10.083	9.420	11.486	74.583	8.010	5.922	4.754	5.033	6.588	9.490	47.543	6.831	36.445
promo_1	1.760	243.618	11.676	8.210	10.083	9.001	7.966	4.137	4.836	4.306	3.204	3.057	2.409	3.231	2.631	2.785	20.182
promo_10	672	549	4.391	2.535	2.868	2.121	1.866	2.047	34.707	37.733	357.641	29.495	33.876	17.668	14.174	9.499	34.490
promo_11	1.802	871	4.224	639	6.291	2.446	2.133	2.500	8.044	8.160	14.911	315.001	19.559	7.028	3.259	3.113	24.999
promo_12	762	425	1.196	826	1.584	2.413	1.802	2.115	12.612	9.905	16.960	52.605	323.828	15.339	20.418	6.334	29.320
promo_13	2.253	995	3.346	1.313	2.173	3.037	15.275	2.964	7.518	6.719	10.415	18.862	46.035	500.251	32.007	9.772	41.433
promo_14	11.518	1.494	4.192	2.842	4.486	3.071	6.769	41.059	6.556	11.262	16.746	29.200	28.861	31.011	565.191	25.110	49.336
promo_14_2017	88.930	85.962	82.152	69.508	66.804	68.761	74.273	53.538	55.723	70.714	55.888	62.767	53.622	54.255	65.137	44.318	65.772
promo_140_2017	14.537	11.722	27.895	17.974	16.792	10.768	13.903	22.255	18.255	35.145	26.840	25.174	16.921	20.889	33.258	27.399	21.233
promo_15	7.711	924	2.846	1.553	1.645	1.351	4.371	6.545	3.154	5.244	5.135	6.901	8.581	8.780	19.092	424.986	31.801
promo_2	2.334	4.803	313.192	11.957	10.356	4.726	3.740	6.056	3.386	37.101	1.994	2.415	2.477	2.389	5.364	2.443	25.921
promo_3	5.616	26.343	21.204	239.874	19.332	31.687	25.336	14.576	9.030	1.044	7.762	1.972	1.597	1.467	1.373	1.284	25.594
promo_4	2.677	10.476	10.050	17.076	247.783	9.286	6.556	4.062	3.969	2.477	2.260	17.872	2.416	2.488	3.123	2.248	21.551
promo_5	904	10.721	10.136	19.349	28.947	247.311	16.555	10.600	3.008	754	2.166	1.741	6.411	1.080	1.184	951	22.614
promo_6	663	3.211	3.060	6.474	10.655	22.760	376.698	14.440	3.597	2.739	3.685	3.486	4.166	18.668	4.234	1.884	30.026
promo_60_2017	19.364	16.490	28.005	20.773	19.681	22.026	23.230	35.719	27.005	37.250	27.295	23.535	25.699	30.162	39.923	32.155	26.770
promo_7	37.416	2.875	7.903	10.385	16.724	19.234	26.639	517.580	21.863	21.680	22.185	16.767	11.462	9.383	48.436	6.702	49.827
promo_8	289	2.027	2.042	2.969	3.571	4.976	5.001	8.490	370.763	18.885	14.795	11.394	8.407	6.022	5.583	7.780	29.562
promo_9	4.313	1.100	23.375	2.668	3.724	4.457	6.305	10.028	10.825	429.927	19.432	18.415	8.401	6.844	11.020	3.735	35.286
Total general	291.997	247.385	288.412	327.418	352.963	286.792	273.799	287.251	242.735	281.719	317.523	341.677	280.957	268.565	281.029	237.157	287.961

Tabla 18: Importancia de los Atributos

4. Generación de las Predicciones

Una vez creado el modelo *bst* basado en los predictores *X_train*, se utiliza generando 2 predicciones:

a. *val*_pred: predicción generada con los predictores *X_val*, descritos en Predictores y Etiquetas Validación. Esta predicción se utilizará para el cálculo interno de las métricas: mse (mean squared error) para:

- el train dataset sin y con los pesos de productos perecederos (1,25)

- el train dataset considerando las primeras 5 líneas y las líneas siguientes

b. *test_pred*: predicción generada con los predictores *X_test,* descritos en Predictores de Test.

5. Preparación del envío de archivos

En esta última fase, se unen las predicciones generadas basadas en X_test con el fichero inicial test.csv, obteniéndose un archivo para el envío a Kaggle de 2.680.240 líneas con las predicciones de los 167.515 producto/tienda para cada uno de los 16 días de los datos del test dataset.

6 Conclusiones

Las conclusiones de este trabajo en general creo que son positivas y las resumiré desde tres perspectivas diferentes.

Un primer punto de vista son las lecciones aprendidas del ajuste de un modelo predictivo en la plataforma Kaggle:

Lección 1: Para obtener una calificación profesional es indispensable acometer el problema con conocimientos del negocio.

Lección 2: Para el desarrollo de un modelo predictivo como el que se aplica en este trabajo es fundamental contar con tiempo suficiente para:

- Experimentar con todos los parámetros disponibles.
- Procesar la información y comentarios de otros usuarios que se enfrentan al mismo problema predictivo.
- Contar con Hardware o plataforma en la nube suficientemente potente y versátil para el tratamiento de datasets de tamaño medio a una velocidad asumible.

La segunda perspectiva a considerar en estas conclusiones es la consecución de los objetivos planteados.

Objetivo 1: Conseguir score de Kaggle profesional.

He superado mis expectativas iniciales al quedar entre el 17% de los concursantes más acertados. Sin embargo, para consolidar lo aprendido es indispensable tener, en el futuro, una participación intensiva en competiciones y realizar un curso avanzado de Python para data science.

Objetivo 2: Hacer uso de las plataformas GitHub, PyCharm, etc.

No considero haber experimentado suficientemente con la gestión de proyectos predictivos con base en repositorios y herramientas profesionales. Mis próximos pasos en el desarrollo de algoritmos de Machine Learning tendrán en el conocimiento derivado del uso de estas plataformas un objetivo principal.

Objetivo 3: Profundizar conocimientos del lenguaje Python.

He tenido una cierta mejora en mis habilidades con Python. Aunque la manera mas rápida de adquirir unas competencias profesionales es acometiendo un entrenamiento específico.

Objetivo 4: Aprendizaje técnicas de afinación de predicciones.

Si bien objetivo no lo considero completamente conseguido, la motivación para experimentar en este campo ha crecido, viendo soluciones que podrían mejorar mis resultados que por falta de tiempo no he podido experimentar.

La tercera perspectiva de estas conclusiones es considerar que el seguimiento de la planificación considero que ha sido correcto.

Quiero finalmente agradecer a la tutora de este TFM, Dra. Laia Subirats Maté su asesoramiento y sus ánimos, ayudándome a hacer posible este trabajo.

7 Glosario

CSV: comma separated values

DART: Dropouts meet multiple additive regression trees

EFB: Exclusive feature bundling

GBDT: Gradient boosting decision tree

GOSS: Gradient-based one-side sampling

GPU: Graphics processor unit

LGBM: Light gradient boosting machine

NWRMSLE: Normalized weighted root mean squared logarithmic error

pGBRT: Parallel boosted regression tree

XBOOST: Extreme gradient boosting

8 Bibliografía

Baquerizo Ramon, R. Y. (2017). Análisis del comportamiento del cliente y de la demanda de productos en los principales supermercados de la ciudad de Machala. (M. E. Universidad Técnica de Machala, Ed.)

Eduardo Fonseca, R. G. (2017). Acoustic Scene Classification by Fusing LightGBM and VGG-net Multichannel Predictions. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events* .

Friedman, J. H. (2002). Stochastic gradient boosting. *Journal Computational Statistics & Data Analysis - Nonlinear methods and data mining , 34* (4), 367-378.

Guolin Ke, Q. M.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.*

Kaggle. (19 de Octubre de 2017). *Corporación Favorita Grocery Sales Forecasting*. Obtenido de https://www.kaggle.com/c/favorita-grocery-sales-forecasting

Lanfa Liu, M. J. (12 de Diciembre de 2017). Combining Partial Least Squares and the Gradient-Boosting Method for Soil Property Retrieval Using Visible Near-Infrared Shortwave Infrared Spectra. *Remote Sensing* .

Lee, C. (2017). *LGBM Starter | Kaggle*. Obtenido de Kaggle: https://www.kaggle.com/ceshine/lgbm-starter

Microsoft. (Enero de 2018). *Microsoft/LightGBM*. Obtenido de GitHub: https://github.com/Microsoft/LightGBM/blob/master/docs/Features.rst

Natalia Ponomareva, T. C. (31 de Octubre de 2017). *Compact Multi-Class Boosted Trees.* Obtenido de https://arxiv.org/abs/1710.11547v1

Santamaria, E. J. (2014). Influencia de los Factores Culturales y Demográficos en el Perfil del Consumidor de Marcas Propias en Ecuador. *Revista Politécnica , 34* (2).

Serrano, D. (2016 de Julio de 2016). *Desarrollo Organizacional (modelos de D.O.)*. Obtenido de Prezi: https://prezi.com/z8fw2x6zjauu/desarrollo-organizacional-modelos-de-do/

Si Si Huan Zhang, S. S.-J. (s.f.). Output, Gradient Boosted Decision Trees for High Dimensional Sparse.

9 Anexos

Se incluye a continuación el código de carga y preparación de datos.

a) lgbm_arrange.py

```
"""
This is an upgraded version of Ceshine's LGBM starter script.

"""


from datetime import date, timedelta
import time
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
import lightgbm as lgb

camino = './input/' #google drive
camino = '/Users/gabrielkreplak/Documents/Gabi/UOC/Materiales/E4 Big Data y
Sistemas NoSQL/B2.342 Trabajo Final de Máster MIB-AD/Favorita/input/'

start_time = time.time()
df_train = pd.read_csv(
    camino + 'train.csv', usecols=[1, 2, 3, 4, 5],
    dtype={'onpromotion': bool},
    converters={'unit_sales': lambda u: np.log1p(float(u)) if float(u) > 0
else 0},
    parse_dates=["date"],
    skiprows=range(1, 66458909)  # 2016-01-01
)

items = pd.read_csv(
    camino + "items.csv",
).set_index("item_nbr")

df_2017 = df_train.loc[df_train.date>=pd.datetime(2017,1,1)]
df_2016 = df_train.loc[df_train.date < pd.datetime(2017,1,1)]  # GK
del df_train

df_test = pd.read_csv(
    camino + "test.csv", usecols=[0, 1, 2, 3, 4],
    dtype={'onpromotion': bool},
    parse_dates=["date"]  # , date_parser=parser
)

df_test = df_test.set_index(['store_nbr', 'item_nbr', 'date'])

promo_2017_train = df_2017.set_index(["store_nbr", "item_nbr",
"date"])[["onpromotion"]].unstack(level=-1).fillna(False)
promo_2017_train.columns = promo_2017_train.columns.get_level_values(1)

promo_2017_test = df_test[["onpromotion"]].unstack(level=-1).fillna(False)
promo_2017_test.columns = promo_2017_test.columns.get_level_values(1)
promo_2017_test =
promo_2017_test.reindex(promo_2017_train.index).fillna(False)
```

```
promo_2017 = pd.concat([promo_2017_train, promo_2017_test], axis=1) #table
depicting whether train & test store/item are on promotion: 167515 store/item
x 243 day

del promo_2017_test, promo_2017_train

df_2017 = df_2017.set_index(
    ["store_nbr", "item_nbr", "date"])[["unit_sales"]].unstack(
        level=-1).fillna(0)
df_2017.columns = df_2017.columns.get_level_values(1) #table depicting train
unit_sales for 167515 store/item x 227 day

items = items.reindex(df_2017.index.get_level_values(1)) #table depicting
family, class & perishable of df_2017 existing store/items




def get_timespan(df, dt, minus, periods, freq='D'):
    # get unit_sales of all 167515 store/item during certain time span

    return df[pd.date_range(dt - timedelta(days=minus), periods=periods,
freq=freq)]

def prepare_dataset(t2017, is_train=True):
    X = pd.DataFrame({

        "day_1_2017": get_timespan(df_2017, t2017, 1, 1).values.ravel(),
# unit_sales         during                  31/5/2017 for each of all 167515
store/item
        "mean_3_2017": get_timespan(df_2017, t2017, 3,
3).mean(axis=1).values,         # unit_sales average during   3 days before
31/5/2017 for each of all 167515 store/item
        "mean_7_2017": get_timespan(df_2017, t2017, 7,
7).mean(axis=1).values,         # unit_sales average during   7 days before
31/5/2017 for each of all 167515 store/item
        "mean_14_2017": get_timespan(df_2017, t2017, 14,
14).mean(axis=1).values,      # unit_sales average during  14 days before
31/5/2017 for each of all 167515 store/item
        "mean_30_2017": get_timespan(df_2017, t2017, 30,
30).mean(axis=1).values,      # unit_sales average during  30 days before
31/5/2017 for each of all 167515 store/item
        "mean_60_2017": get_timespan(df_2017, t2017, 60,
60).mean(axis=1).values,      # unit_sales average during  60 days before
31/5/2017 for each of all 167515 store/item
        "mean_140_2017": get_timespan(df_2017, t2017, 140,
140).mean(axis=1).values,   # unit_sales average during 140 days before
31/5/2017 for each of all 167515 store/item
        "promo_14_2017": get_timespan(promo_2017, t2017, 14,
14).sum(axis=1).values,   # promo status sum   during  14 days before
31/5/2017 for each of all 167515 store/item
        "promo_60_2017": get_timespan(promo_2017, t2017, 60,
60).sum(axis=1).values,   # promo status sum   during  60 days before
31/5/2017 for each of all 167515 store/item
        "promo_140_2017": get_timespan(promo_2017, t2017, 140,
140).sum(axis=1).values # promo status sum   during 140 days before 31/5/2017
for each of all 167515 store/item
    })
    for i in range(7):
```

```
        X['mean_4_dow{}_2017'.format(i)]  = get_timespan(df_2017, t2017, 28-
i, 4, freq='7D').mean(axis=1).values   # unit_sales average for each weekday
during  4 weeks before 30/5/2017 for each of all 167515 store/item
        X['mean_20_dow{}_2017'.format(i)] = get_timespan(df_2017, t2017, 140-
i, 20, freq='7D').mean(axis=1).values # unit_sales average for each weekday
during 20 weeks before 30/5/2017 for each of all 167515 store/item
    for i in range(16):
        X["promo_{}".format(i)] = promo_2017[t2017 +
timedelta(days=i)].values.astype(np.uint8) # promo status of all 167515
store/item for each of the 16 days after 30-5-17

    if is_train:
        y = df_2017[
            pd.date_range(t2017, periods=16) # unit_sales for 16 days after
31-5-17 for each of all 167515 store/item
        ].values
        return X, y
    return X

t = (time.time() - start_time)/60
print ("Time %s min" % t)

print("Preparing dataset...")
t2017 = date(2017,6,21) #(2017,6,22)0.520, (2017,6,26)0.518,
(2017,6,21)0.513, (2017, 5, 31)0.515
X_l, y_l = [], []

for i in range(6): # Build training dataset 40 columns x (167 store/item x 6
weeks)
# NUEVO: range(4)
    delta = timedelta(days=7 * i)
    X_tmp, y_tmp = prepare_dataset(t2017 + delta)
    X_l.append(X_tmp)
    y_l.append(y_tmp)
X_train = pd.concat(X_l, axis=0)
y_train = np.concatenate(y_l, axis=0)
del X_l, y_l

X_val, y_val = prepare_dataset(date(2017, 7, 26)) # build validation dataset
X_test = prepare_dataset(date(2017, 8, 16), is_train=False) # build test
dataset


X_train                          .to_csv(camino + 'temp/X_train.csv',
float_format='%.6f', index=None)
X_val                            .to_csv(camino + 'temp/X_val.csv',
float_format='%.6f', index=None)
X_test                           .to_csv(camino + 'temp/X_test.csv',
float_format='%.6f', index=None)

pd.DataFrame(y_train)            .to_csv(camino + 'temp/y_train.csv',
float_format='%.6f', index=None)
pd.DataFrame(y_val)              .to_csv(camino + 'temp/y_val.csv',
float_format='%.6f', index=None)
pd.DataFrame(index=df_2017.index).to_csv(camino + 'temp/stores_items.csv')


t = (time.time() - start_time)/60
print ("Total processing time %s min" % t)
```

b) lgbm_predict.py

Se reproduce a continuación el código de aprendizaje del algoritmo LGBM, la producción de las predicciones y la generación del archivo de salida.

```
"""
This is an upgraded version of Ceshine's LGBM starter script, simply adding
more
average features and weekly average features on it.
"""


from datetime import date, timedelta
import time
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
import lightgbm as lgb
camino = './input/' #google drive
camino = '/Users/gabrielkreplak/Documents/Gabi/UOC/Materiales/E4 Big Data y
Sistemas NoSQL/B2.342 Trabajo Final de Máster MIB-AD/Favorita/input/'

start_time = time.time()


X_train = pd.read_csv(camino + 'temp/X_train.csv')
y_train = np.array(pd.read_csv(camino + 'temp/y_train.csv'))
X_val = pd.read_csv(camino + 'temp/X_val.csv')
y_val = np.array(pd.read_csv(camino + 'temp/y_val.csv'))
X_test = pd.read_csv(camino + 'temp/X_test.csv')
stores_items = pd.read_csv(camino + 'temp/stores_items.csv',
index_col=['store_nbr', 'item_nbr'])

test_ids = pd.read_csv(camino + "test.csv", usecols=[0, 1, 2, 3, 4],
dtype={'onpromotion': bool}, parse_dates=["date"]  # , date_parser=parser
).set_index(['store_nbr', 'item_nbr', 'date'])


items = pd.read_csv( camino + 'items.csv' ).set_index("item_nbr")
items = items.reindex( stores_items.index.get_level_values(1) )

print("Training and predicting models...")
params = {
    'num_leaves': 63,#
    'objective': 'regression',#
    'min_data_in_leaf': 300,#
#    'learning_rate': 0.1, #score 0.529, 'min_data_in_leaf': 300
#    'learning_rate': 0.06, #score 0.515, 'min_data_in_leaf': 300
#    'learning_rate': 0.05, #score 0.515, 'min_data_in_leaf': 300
#    'learning_rate': 0.025, #score 0.516, 'min_data_in_leaf': 300
#    'learning_rate': 0.01, #score 0.517, 'min_data_in_leaf': 300
   'learning_rate': 0.03 ,#
    'feature_fraction': 0.8,#
    'bagging_fraction': 0.8,#
    'bagging_freq': 2,
    'metric': 'l2',#
```

```
    'num_threads': 4,
   'max_bin': 500,#
}
evals_result = {}  # to record eval results for plotting
MAX_ROUNDS = 100
val_pred = []
test_pred = []
cate_vars = []#
for i in range(16):
    print("=" * 50)
    print("Step %d" % (i+1))
    print("=" * 50)
    dtrain = lgb.Dataset(
        X_train, label=y_train[:, i],
        categorical_feature=cate_vars,
        weight=pd.concat([items["perishable"]] * 6) * 0.25 + 1)

    dval = lgb.Dataset(
        X_val, label=y_val[:, i], reference=dtrain,
        weight=items["perishable"] * 0.25 + 1,
        categorical_feature=cate_vars)

    bst = lgb.train(
        params, dtrain, num_boost_round=MAX_ROUNDS,
        valid_sets=[dtrain, dval], early_stopping_rounds=50,
verbose_eval=100, evals_result=evals_result)

    print("\n".join(("%s: %.2f" % x) for x in sorted(
        zip(X_train.columns, bst.feature_importance("gain")),
        key=lambda x: x[1], reverse=True)))

    val_pred. append(bst.predict(X_val,  num_iteration=bst.best_iteration or
MAX_ROUNDS))
    test_pred.append(bst.predict(X_test, num_iteration=bst.best_iteration or
MAX_ROUNDS))


    t = (time.time ( ) - start_time) / 60
    print ( "Time %s min" % t )

ax = lgb.plot_metric ( evals_result , metric='l2', title='Metrics During
Training Step %d' % (i+1) )
ax = lgb.plot_importance ( bst , max_num_features=20, title='Feature
Importance Step %d' % (i+1) )

n_public = 5  # Number of days in public test set
weights = pd.concat ( [ items[ "perishable" ] ] ) * 0.25 + 1
print ( "Unweighted validation mse: " , mean_squared_error (y_val
, np.array ( val_pred ).transpose ( ) ) )
print ( "Full validation mse:       " , mean_squared_error (y_val
, np.array ( val_pred ).transpose ( ) , sample_weight=weights ) )
print ( "'Public' validation mse:   " , mean_squared_error (y_val[ : ,
:n_public ] , np.array ( val_pred ).transpose ( )[ : , :n_public ] ,
sample_weight=weights ) )
print ( "'Private' validation mse:  " , mean_squared_error (y_val[ : ,
n_public: ] , np.array ( val_pred ).transpose ( )[ : , n_public: ] ,
sample_weight=weights ) )
```

```
print("Making submission...")
y_test = np.array(test_pred).transpose()
df_preds = pd.DataFrame( y_test, index=stores_items.index,
columns=pd.date_range("2017-08-16",
periods=16)).stack().to_frame("unit_sales")

df_preds.index.set_names(["store_nbr", "item_nbr", "date"], inplace=True)

submission = test_ids[['id']].join(df_preds, how="left").fillna(0)
submission["unit_sales"] = np.clip(np.expm1(submission["unit_sales"]), 0,
1000)
submission.to_csv(camino + 'out/lgb_ceshine.csv', float_format='%.4f',
index=None)


t = (time.time() - start_time)/60
print ("Total processing time %s min" % t)
```