

NSA DATA

Andrei Constantin, Sergio Arranz Moya, Nazariy Gunko

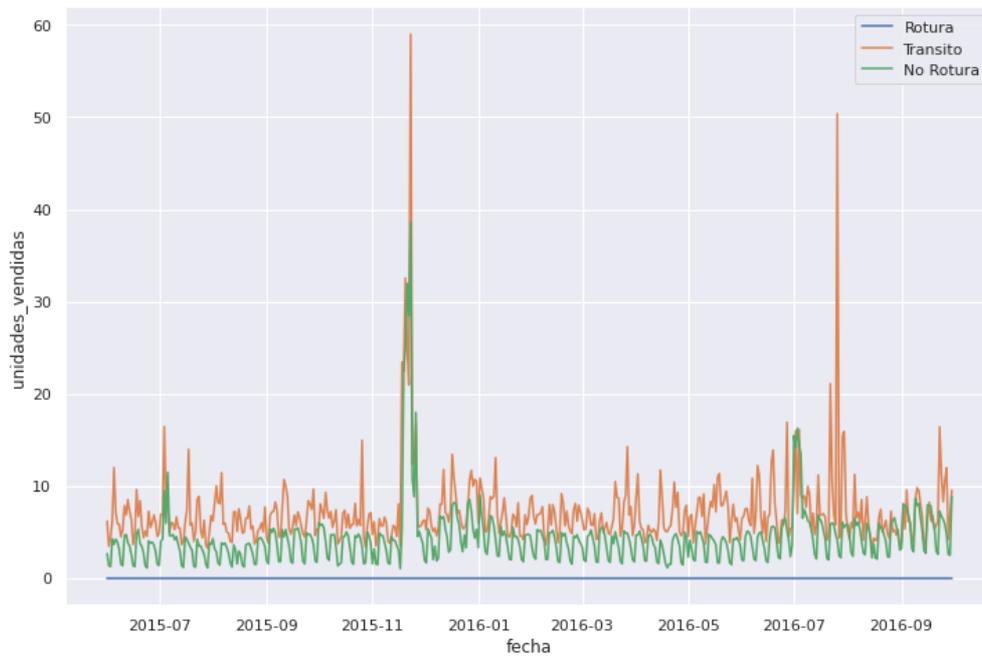
1. Análisis inicial y limpieza

- Conversión a tipos de datos correctos: el atributo “fecha” se convierte a datetime y el atributo “precio” a float.
- Filling de los NaNs: se detectaron gran número de valores perdidos en los campos “categoria_dos”, “precio” y “antigüedad”. Para cada artículo, se completa su precio con valores anteriores disponibles. Para el atributo “categoria_dos”, al haber relativamente pocos valores perdidos, se genera una nueva categoría con valor 0 para estos casos.
- Análisis duplicados: se detectaron numerosos datos duplicados, es decir, que tienen todos sus campos exactamente iguales. Dichos duplicados fueron eliminados del conjunto de datos para optimizar los resultados de los modelos utilizados. También se eliminaron aquellos duplicados cuya única diferencia es el atributo campaña, manteniendo aquellos que son campaña.

2. Visualización atributos e interpretaciones

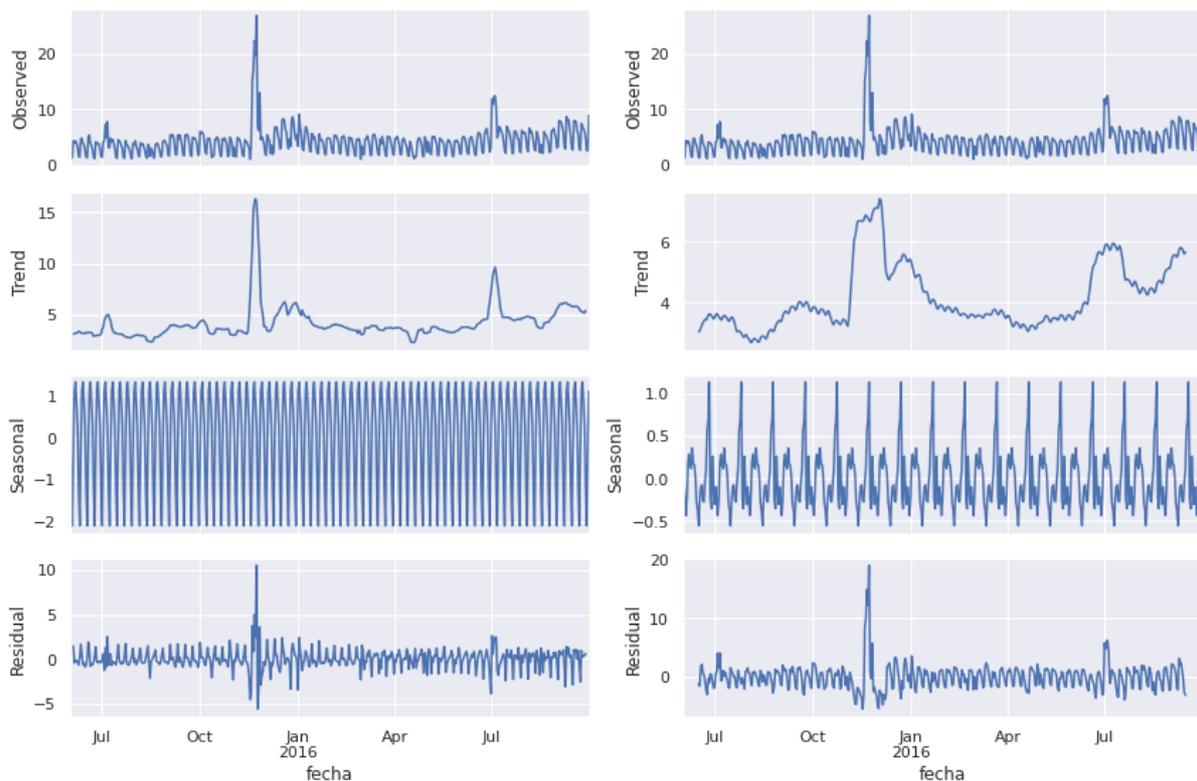
Se realizó un estudio visual de los atributos del conjunto de datos para entender la naturaleza de los mismos.

- Se representan los atributos numéricos “visitas”, “precio” y “unidades_vendidas” con un diagrama de cajas para ver cuales su distribución , tanto en el conjunto de Modelar como en Estimar, y así poder valores extremos.
- Unidades vendidas por cada estado.
- Se obtiene la matriz de correlación de los atributos, en la que podemos observar los atributos más correlacionados con el campo objetivo “unidades_vendidas” que serán a priori los más importantes de nuestro modelo. También se puede detectar atributos fuertemente correlacionados entre sí, pudiendo así escoger cuales eliminar de nuestro modelo.
- Se han representado los atributos “visitas” y “precio” en función del tiempo para ver la evolución temporal, concatenando los datos de Modelar con los de Estimar, para ver si las tendencias concuerdan.
- Se realizó un análisis de frecuencias de los campos de categorías para ver las más y menos frecuentes en nuestro conjunto de datos.
- Se observaron datos en estado “Rotura” los cuales siempre tienen cero unidades vendidas. Dichos casos fueron eliminados debido a que no aportan información relevante al modelo, puesto que queremos predecir la demanda.



3. Análisis de temporalidad

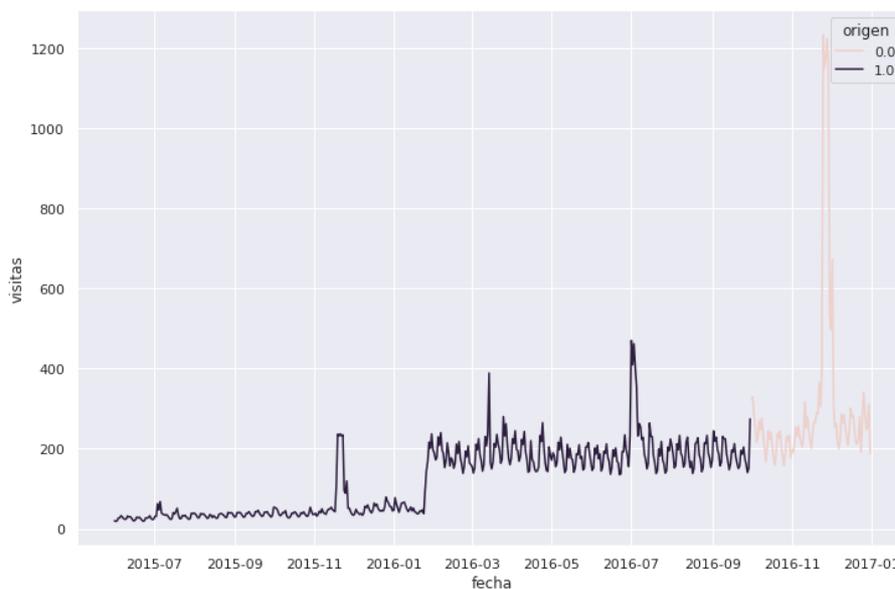
Ya que estamos tratando con series temporales, hemos realizado el análisis de la estacionalidad y la tendencia de forma semanal y mensual tanto de la media de las visitas como de la media de unidades vendidas de cada día.



En la gráfica del análisis de unidades vendidas que se muestra se pueden observar pequeñas estacionalidades o patrones en ambos análisis, siendo el semanal el de la izquierda y el mensual el de la derecha. Estos patrones indican un incremento de ventas en ciertos días de la semana y secciones concretas de cada mes.

Con respecto a la temporalidad también es destacable que encontramos una disparidad muy grande de visitas de una fecha a otra (aproximadamente el 26 de Enero). Esta disparidad no se ve respaldada por nada concreto, y mirando en las tendencias de google no hemos visto ningún incremento de relevancia que lo explique.

https://trends.google.es/trends/explore?date=2015-01-01%202016-12-31&geo=ES&q=%2Fq%2F11h_9y0zpx&hl=es&tz=-120



Teniendo esto en cuenta se realizaron distintas pruebas de reescalado de la primera sección, comparando la forma de la gráfica obtenida y la correlación entre visitas y unidades vendidas. Se probaron StandardScaler, MinMaxScaler y RobustScaler, además de realizar escalado a mano, y simplemente multiplicar por un valor fijo, pues la estacionalidad es igual tanto antes como después de la fecha en cuestión.

4. Feature engineering

Se ha probado:

- Ver si existe relación entre el día de la semana y las ventas. Podemos observar que el atributo añadido del día de la semana es interesante ya que las ventas varían considerablemente en función del día de la semana.
- Ya que el día de la semana, el día del mes y el mes son atributos cíclicos (es decir, tras el último valor se encuentra el primero), hemos probado a representarlas como seno y coseno para mostrar correctamente la distancia entre ellos.
- Medias, mínimos y máximos móviles de los últimos 7 y 30 días, tanto para las visitas como para las unidades vendidas, estos son la media de los valores del producto en esos periodos, su máximo y su mínimo. Hemos probado a incorporarlas de tres

maneras distintas:

- Obteniendo al principio de todo dichos atributos de todo el dataset. Aunque eficaz, esto no es representativo para estimar ya que no conocemos las unidades vendidas de dichos datos.
- Predicción recursiva, donde predecimos dato a dato y utilizamos dicha predicción para obtener los atributos móviles del siguiente dato.
- Utilizar como unidades vendidas la predicción obtenida de otro algoritmo, y basarnos en ellas para calcular los atributos móviles.

4. Modelos, funciones de coste, series temporales

- Funciones de coste personalizadas: Para la red neuronal se generó una función asimétrica y desplazada que favorece la sobreestimación para evitar roturas. Para el Gradient Boosting se observó que con una función de coste básica como puede ser "rmse" o "regression" el modelo no era capaz de predecir ceros, lo cual es un problema ya que en nuestro conjunto de datos existe gran cantidad de ceros. Por ello, para rectificar este comportamiento se ha utilizado la función de pérdida "tweedie" la cual está especialmente diseñada para este tipo de casos.
- Gradient Boosting: se entrenaron dos modelos, uno con función de coste normal y otro con "tweedie", obteniendo mejores resultados la función "tweedie". Se realizó una búsqueda aleatoria para obtener los parámetros óptimos, siendo finalmente 750 estimadores, 0.01 de tasa de aprendizaje, función objetivo "tweedie".
- Random Forest: se realizó hiper parametrización para seleccionar los parámetros. Es sobre todo relevante el número de features utilizado por cada árbol y la profundidad de los mismos.
- Perceptrón multicapa: se ha construido una pequeña red neuronal con 2 capas ocultas de 100 y 50 neuronas respectivamente y una capa de salida de una neurona función de activación ReLu para evitar predecir valores negativos.
- Para el tratamiento de series temporales se ha utilizado la herramienta de forecasting de series temporales Facebook Prophet, el cual es capaz de detectar patrones y tendencias estacionales semanales, mensuales, anuales, etc. Dicho modelo recibe variables regresoras para poder con ellas explicar fenómenos atípicos que no siguen la tendencia lógica de la serie objetivo, como por ejemplo cero unidades vendidas de un artículo cuando para ese momento la predicción sería que sí que hay demanda, pero se sabe que el producto se va a encontrar en rotura. Para nuestro modelo hemos probado diferentes combinaciones de las siguientes variables regresoras: visitas, precio, campaña, día atípico y estado.

5. Resultados, comparativa, métricas

- Partición Train/Test: para el entrenamiento de los modelos se utilizó principalmente una división dejando el 20% de los datos para test. La división se realizó sin mezclar los datos, es decir, los datos de entrenamiento son anteriores temporalmente a los de test, evitando de esta forma predecir valores del pasado utilizando conocimiento futuro. Esta decisión surge de que realizar validación en el dataset de modelar resulta muy enrevesado, debido que los datos corresponden a un año y con ello eventos como puede ser el Black Friday o la campaña de navidad donde las ventas aumentan considerablemente figuran únicamente una vez. En el Script exploratorio

se describe más en profundidad este problema y se plantean algunas posibles soluciones.

- Los objetivos principales son minimizar las desviaciones con respecto a los datos reales y evitar las roturas de stock. Por ello, la principal métrica para evaluar los modelos se basa en rRMSE y CF (% de casos favorables) respectivamente para cada objetivo.
- rRMSE representa la desviación de las predicciones en cuanto a los valores reales, y penaliza las desviaciones más grandes en mayor medida proporcionalmente.
- Un caso favorable se define como aquel en el que la demanda ha sido igual o inferior a la previsión (No existe rotura de stock)
- La métrica final es por tanto: $(0.7 * rRMSE) + (0.3 * (1-CF))$
- También mostramos el MAE (el error absoluto medio) por ser una métrica más interpretable para nosotros.
- Teniendo en cuenta la diferencia del valor de las predicciones entre los algoritmos, probamos a realizar un voto ponderado tanto por parejas como utilizando los 3 para él, obteniendo de nuevo el score y el MAE para poder comparar.

	LightGBM		Random Forest		Red neuronal		Stacking 3 modelos		Peso de cada modelo % (GBM, RF, NN)
	% CF	Score	% CF	Score	% CF	Score	% CF	Score	
Modelo entregado para local (Baseline)	77	2,550	72	2,830	44	2,360	62	2,400	0 30 70
Modelo con los 3 y visitas x 4	74	2,280	77	2,240	63	2,318	68	2,210	53 22 25
Modelo con los 3 con variables circulares	78	2,320	78	2,647	69	2,423	67	2,210	8 42 50
Modelo con los 3 con variables circulares y visitas x4	74	2,271	80	2,260	48	2,369	65	2,224	41 36 23
Modelo entregado para local optimizado	72	2,348	74	2,629	62	2,397	63	2,249	6 50 44
Modelo autoregresivo	74	2,874	84	3,060	66	3,218	76	2,811	17 59 24
	% CF	Score							
Prophet	78	2,982							

En vista a los resultados obtenidos y al análisis realizado, las predicciones entregadas para estimar se han generado con un modelo entrenado con las visitas anteriores al día 26 de enero multiplicadas por 4, ya que como hemos visto existe una anomalía dentro de las visitas, y siendo esta variable muy correlacionada con las unidades vendidas, el multiplicar las visitas por 4 maximiza la correlación de esta variable con la variable objetivo. En cuanto al modelo en sí, se ha utilizado el stacking de los tres modelos con los pesos que figuran en la tabla (53% GB, 22% RF, 25% NN). En cuanto a los atributos extra creados sólo se ha añadido el día de la semana, ni los atributos móviles ni las fechas cíclicas se han utilizado.