Práctica de Aprendizaje Automático

Predicción del precio del automóvil dadas sus características

Edgar Pérez Blanco
Bartomeu Perelló Comas

APA - GEI FIB
Otoño 2020

TABLA DE CONTENIDOS

Resumen del trabajo

El principal objetivo de este trabajo es conseguir predecir el precio de un automóvil dadas sus características técnicas (potencia, tamaño, características del motor, …) y algunos datos proporcionados por aseguradoras (nivel de riesgo de asegurar un vehículo, y un valor que refleja cuántas pérdidas tienen en media de dicho vehículo).

Inicialmente hemos realizado un pequeño estudio estadístico de los datos, a los cuales les hemos aplicado diferentes técnicas de preprocesado: imputando valores faltantes, eliminando tanto variables redundantes como variables vagamente representativas y reduciendo variables muy desbalanceadas a otras de una granularidad más gruesa, pero aportando capacidad de clusterización.

Para la realización del preprocesado hemos hecho uso, no solo de la información proporcionada por el estudio estadístico, sino también de nuestro conocimiento sobre el mundo del motor y las implicaciones que tienen ciertas características del motor sobre otras.

Tras el preproceso hemos hecho uso de diferentes modelos de aprendizaje automático, tanto lineales (Regresión Lineal, Ridge Regression, KNN) como no lineales (MLP, Random Forest).

Hemos obtenido modelos con puntuaciones muy parecidas (en términos de R^2). En concreto: KNN, MLP y Random Forest una vez ajustados con los datos de entrenamiento, han proporcionado en la fase de test una puntuación R^2 de entre un 0.77 y un 0.82.

Finalmente nos hemos quedado con el modelo Random Forest por su mejor generalización y por su menor coste computacional pensando en una futura expansión en la que, utilizando KNN con una gran cantidad de datos la predicción podría llegar a ser muy costosa, y por otro lado las MLP contarán con un alto coste computacional de entrenamiento.

Trabajos previos relacionados

La investigación sobre los trabajos relacionados se ha realizado una vez habíamos acabado nuestro trabajo para no estar influenciados por los resultados o las estrategias que han resuelto este problema.

Tenemos constancia de que este dataset ha sido utilizado en tres papers distintos, véase en Referencias.

En la página *Kaggle* tenemos constancia de 89 notebooks sobre este dataset y prácticamente todos han intentado hacer regresión sobre el precio de los vehículos, en muchos casos han utilizado sólo las variables más relevantes descartando la marca de los vehículos, en nuestro caso queríamos comprobar con los modelos lineales si le daría más o menos peso a las marcas solo por tener coches con precios mayores o inferiores a la media.

Exploración de los datos

Al principio hemos realizado un replace de los guiones por barras bajas para que python no se confunda con los valores negativos.

En el dataset inicialmente tenemos un total de 205 instancias, con un total de 26 características por cada elemento.

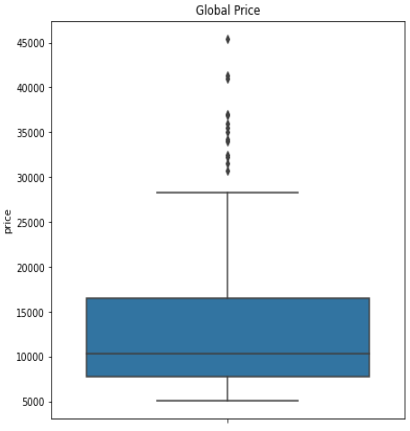
Columnas (Features)

El dataset cuenta con el siguiente conjunto de columnas (variables):

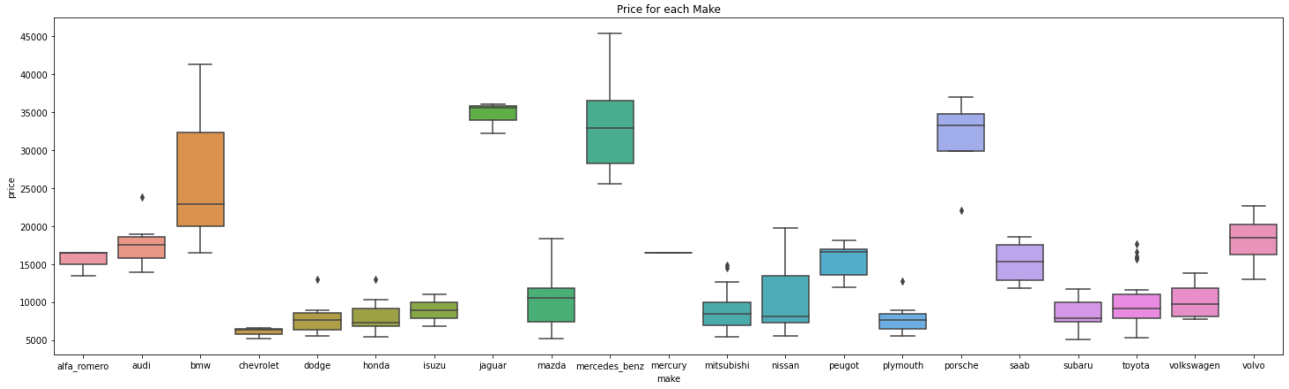
- **Symboling**: Es un número que representa desde el punto de vista de una aseguradora cuán arriesgado es asegurar ese coche respecto al precio inicial.
- **Normalized losses**: Es un valor que representa la pérdida de dinero normalizada para la aseguradora respecto a los otros coches.
- **Make:** Esta variable indica la marca del vehículo.
- **Fuel-type:** Indica si el vehículo utiliza diesel o gasolina.
- **Aspiration:** Nos indica si el vehículo tiene un turbo instalado.
- **Num of doors:** Cuántas puertas tiene el vehículo.
- **Body style:**Indica el tipo del chasis del vehículo.
- **Drive wheels:** Nos proporciona el tipo de tracción del coche.
- **Engine location:**Nos indica la posición del motor en el vehículo.
- **Wheel base:** Nos proporciona la distancia entre las dos ruedas laterales.
- **Length:** El valor de la longitud del coche.
- **Width:** La anchura del vehículo
- **Height:** La altura del vehículo
- **Curb weight:** Representa el “peso en vacío”
- **Engine type:** Nos indica el tipo de motor que utiliza el coche
- **Number of cylinders:** Nos indica el número de cilindros del motor
- **Engine size:** Indica la cilindrada del motor
- **Fuel system:** Indica el tipo de inyección de gasolina que tiene el motor.
- **Bore:** Diámetro de los cilindros.
- **Stroke:** Carrera del cilindro
- **Compression ratio:** Ratio de compresión del combustible
- **Horsepower:** Indica la potencia del motor.
- **Peak rpm:** Revoluciones donde el motor tiene la mayor fuerza.
- **City mpg:** Consumo medio en ciudad.
- **Highway mpg:** Consumo medio en carretera
- **Price:** Precio del coche

Atributos que nos dan información sobre el precio

El precio de los vehículos está principalmente localizado entre 7000 y 17000 dólares. Esto se debe a que contamos con más vehículos de marcas locales (Americanas) cuyos precios son mucho más bajos que las marcas minoritarias (Europeas). Los valores considerados outliers, no se corresponden con errores.



Como se puede ver en el gráfico inferior, el precio se localiza en zonas diferenciadas según la marca. Podemos ver grandes diferencias entre precios de marcas Americanas (baratas) y precios de marcas Europeas (caras, por ser de importación).



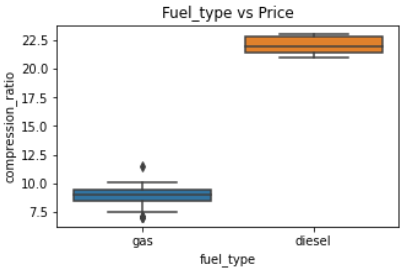
Por desgracia en algunas variables hay un alto desbalance, como por ejemplo en engine location, drive wheels, la cantidad de coches caros es bastante menor que los que tienen un precio más razonable cosa que afecta al tipo de chasis por ejemplo, al ser descapotables más caros hay menos coches de este tipo, este patrón se repite en las siguientes características: tracción del vehículo, el número de pistones y la localización del motor.

Eliminación / Transformación de variables

Además hay variables que están directamente relacionadas por lo cual hemos decidido eliminarlas, estas son:

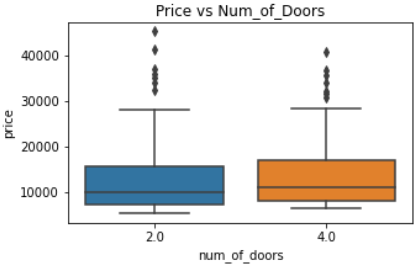
- **Fuel type:**

  Esta característica está directamente relacionada con la compression ratio ya que el diesel tiene un ratio de compresión dos veces el de gasolina. Por esta razón, manteniendo compression ratio, esta variable es redundante.



- **Número de puertas:**

  El número de puertas del vehículo, podemos ver como prácticamente no genera diferencias en el precio de los vehículos. Por esta razón, la eliminamos.



- **Peak RPM:**

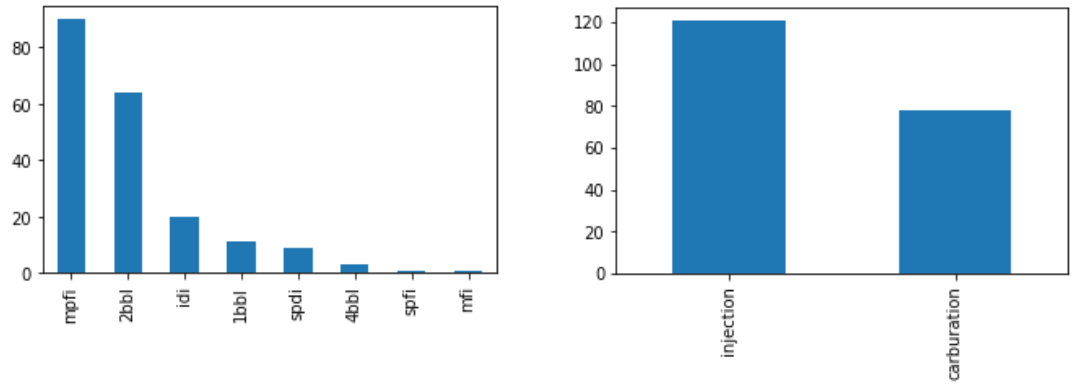
  Esta variable no nos aporta nada ya que prácticamente es igual para todos los vehículos.

- **Aspiration:**

  Este atributo es binario, o bien es estándar, o bien es turbo. Por lo tanto, hemos reformulado esta categoría doble en una sola, que se corresponde con si tiene turbo (valor = 1), o no lo tiene (valor = 0).

- **Fuel system:**

  Debido a la gran cantidad de tipos, y su alto desbalance, hemos decidido reducirlo a dos grandes grupos: Inyección y Carburación. De esta forma mantenemos una información más general, y algo más balanceada.



Por otra parte, hemos cambiado las variables categóricas a forma one hot, por ejemplo las marcas de los vehículos, el tipo de chasis y el tipo de motor; porque tanto para regresión lineal como KNN necesitamos atributos numéricos y no es conveniente darles etiquetas numéricas para no dar ni magnitud ni orden.

Valores Faltantes (Missings)

Otro aspecto que hemos tratado son los missings dentro del dataset:

- **Price:** Las tuplas que no tienen un valor asignado han sido eliminadas ya que solo eran 4 y además el objetivo de nuestra práctica es realizar regresión para encontrar el precio de los vehículos.
- **Coches marca Renault:** Nos dimos cuenta que les faltaba tres características siendo la potencia *(Horsepower)* una de ellas, esta es una de las variables más importantes para predecir el precio de un vehículo por lo tanto decidimos eliminar los dos coches que constituían esta marca.
- **Motores rotativos:** Algunos coches de la marca Mazda tienen un motor peculiar llamado Wankel que no utiliza pistones por lo tanto no tiene bore ni stroke por lo tanto los imputamos con valor 0.
- **Número de puertas:** Para los automóviles que no tiene un valor en esta característica hemos decidido utilizar KNN para imputar estos valores ya que es probable que un vehículo con las mismas características tenga un número similar de puertas.
- **Normalized losses:** En esta característica encontramos muchos missing por lo tanto hemos decidido imputar los valores de cada coche con la media aritmética de su marca, a parte, los coches de la marca “Alfa romeo” no cuentan con este valor por lo tanto hemos decidido imputar estos valores con la media aritmética de la columna.

Finalmente nos encontramos que no necesitamos normalizar las variables numéricas ya que los atributos que tienen una mayor magnitud son el peso que oscila entre 1500 y 4000 y el precio, que es el valor que queremos predecir.

Para terminar con el preprocesado mezclamos los datos para que sea más complicado encontrarnos con sesgos debido al orden de las tuplas a la hora de entrenar y testear.

Protocolo de remuestreo

Tras haber limpiado y preprocesado los datos, lo primero que hemos realizado ha sido una permutación aleatoria de los datos con el fin de eliminar sesgos de orden en los datos (los cuales sabemos que existen, porque las tuplas están ordenadas según la marca de los vehículos).

Tras permutar todo aleatoriamente, hemos realizado una partición 80%/20% para entrenamiento y test. La partición de test se ha utilizado únicamente al final, una vez decididos todos los hiperparametros de cada modelo, para evaluar la capacidad de generalización de cada uno de los modelos.

Para la selección de hiperparametros de cada uno de los modelos, no hemos utilizado partición de validación, sino que hemos utilizado K-Fold - Cross Validation sobre los datos de entrenamiento con el fin de reducir el sobreajuste de los modelos. Tras haber probado diferentes valores de K, tratando de encontrar un compromiso entre coste computacional y reducción del sobreajuste, hemos tomado K = 5, ya que era el máximo valor al que, tras bastantes pruebas, con nuestras máquinas hemos conseguido tiempos de cómputo razonables.

Resultados

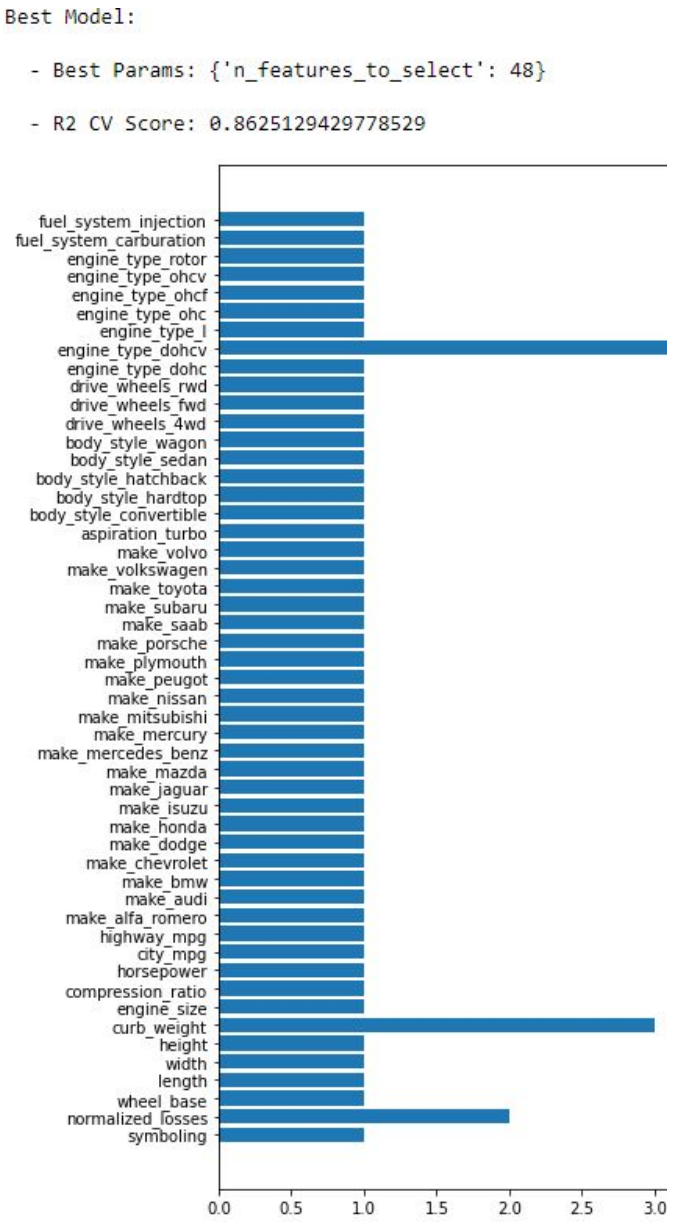
Durante la exposición de resultados de los diferentes modelos, podemos ver gráficos en los que se muestra el precio predecido (azul) contra el precio real (naranja) y el error relativo (entre 0 y 1). Estos irán a acompañados de la métrica R^2 con el fin de medir la precisión del modelo.

Métodos Lineales

Linear Regression

El primer método que hemos probado ha sido la regresión lineal. Lo hemos escogido por ser el más simple de todos y así contar una base inicial.

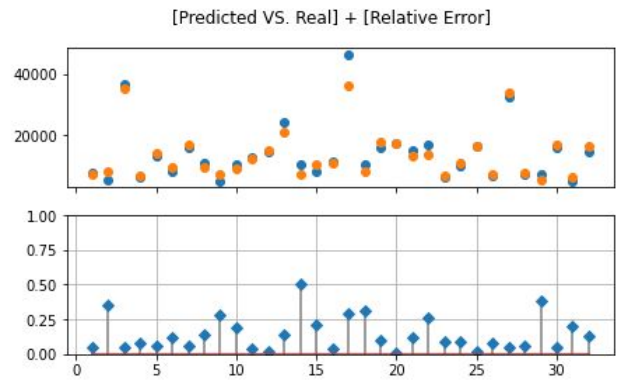
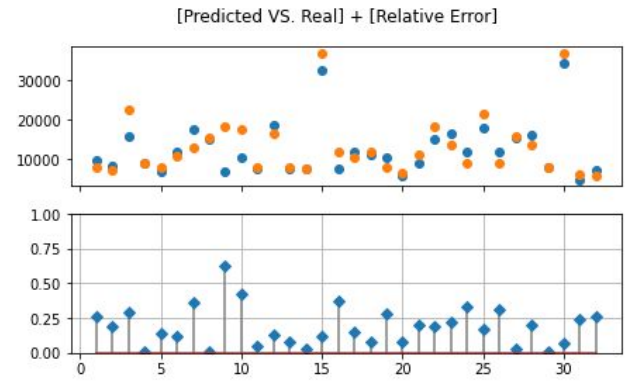
A pesar de que la regresión lineal no cuenta con hiperparametros, hemos realizado mediante RFE (Recursive Feature Selection) una evaluación mediante Cross-Validation de qué Features de las proporcionadas eran poco útiles y por lo tanto, podemos eliminarlas.



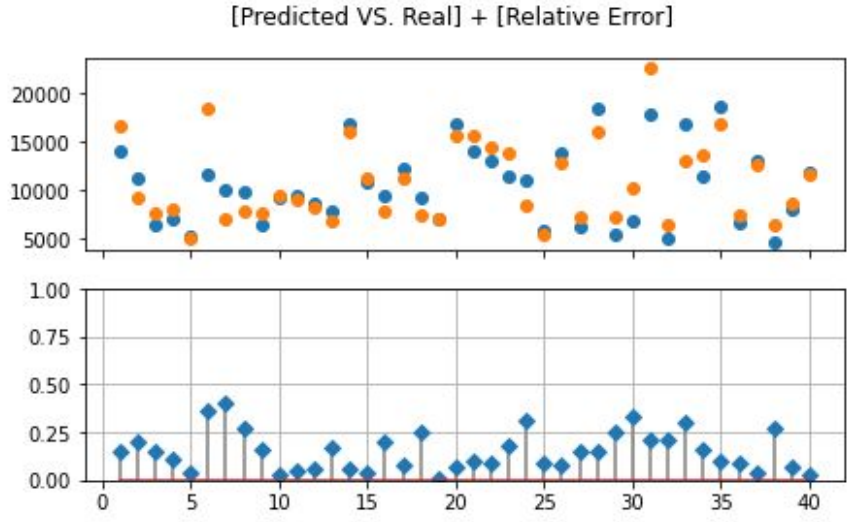
En el gráfico de la derecha podemos ver aquellas Features que han sido clasificadas como poco útiles. A menor valor (1.0), mayor prioridad hay. Por lo tanto, las features “engine_type_dohcv”, “curb_weight” y “normalized_losses” son aquellas que se van a omitir en este modelo.

Tras probar con con todas la features, y sin las prioritarias, hemos visto que evidentemente la puntuación R2 aumenta en media.

A continuación mostramos los resultados de validación:

Mejor Fold	Peor Fold
 R2 Score: 0.905683596360951	 R2 Score: 0.7909564954938614
========== CrossVal Mean R2 Score: 0.8625129429778526 ==========	

Podemos ver como en la mejor de las particiones alcanzamos hasta un 0.90 de R2, y en la peor nos quedamos cerca del 0.80 de R2. En media de todos los Fold, obtenemos una puntuación R2 (omitiendo las características previamente expuestas) de 0.86. Es una buena puntuación teniendo en cuenta la simplicidad del modelo, pero tenemos que tener en cuenta que su generalización será probablemente bastante peor, debido a que estamos trazando una línea, y para valores distantes en alguna feature concreta pueden variar mucho. A continuación mostramos los resultados con la partición de Test para comprobar dicha generalización:



A pesar de ser un método simple, consigue una puntuación R^2 de aproximadamente 0.74. En promedio contamos con aprox. 1620 dólares de error, lo cual no está nada mal, aunque el error máximo cometido, alcanzando casi un 50% de error relativo.

```
Test R2 Score: 0.7396111546623978
Test MSE: 4364820.748061767
Test MAE: 1620.037059465986
Test Max Error: 6638.940154800424
Test Explained Variance: 0.7418111179927893
```

Ridge Regression

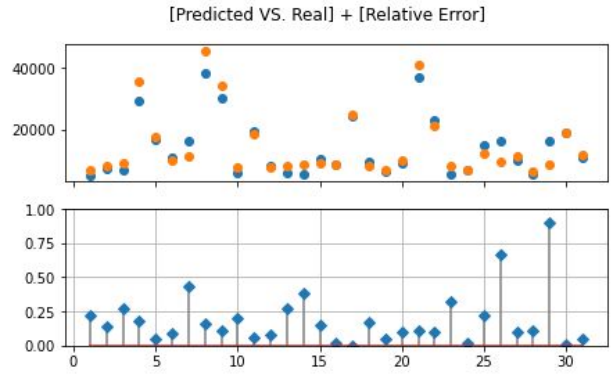
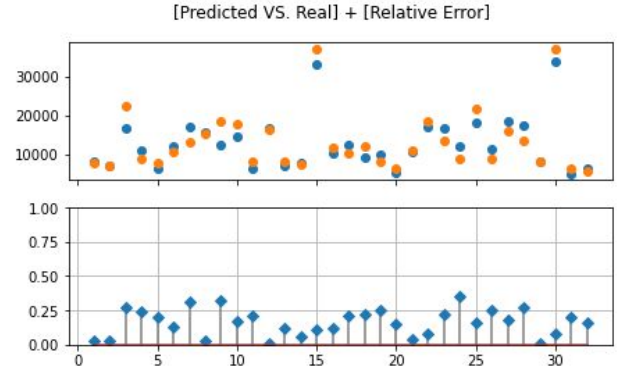
Como bien sabemos, Ridge Regression a grandes rasgos es una variante de la regresión lineal previamente utilizada pero añadiendo un parámetro de regularización con penalización L2.

Por su alta relación con el modelo anterior, hemos creído conveniente probar, tanto con el conjunto completo de features, como con el subconjunto utilizado previamente. A continuación mostramos el mejor parámetro lambda encontrado para cada uno de los modelos generados, y su puntuación R2:

```
Best Model:
  - Best lambda: 0.7575757575757576
  - R2 CV Score: 0.8937070948315584
```

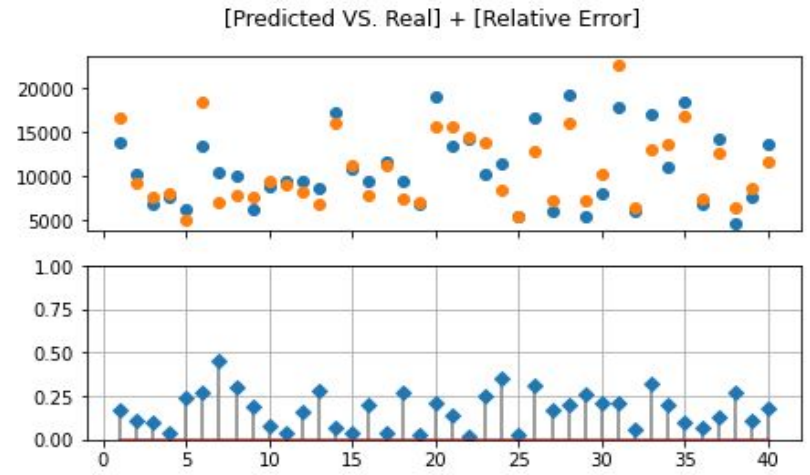
```
Best Model (without LR removed features):
  - Best lambda: 0.696969696969697
  - R2 CV Score: 0.8956747169297354
```

Podemos ver que tanto los valores de lambda son parecidos, al igual que las puntuaciones. Como solo podemos quedarnos con uno, nos decantamos con el de R2 superior. A continuación mostramos los gráficos de de este:

Mejor Fold	**Peor Fold**
 R2 Score: 0.9100554791988313	 R2 Score: 0.8846702279649781

```
============================================================
   CrossVal Mean R2 Score: 0.8956747169297353
============================================================
```

Podemos ver como en el mejor de los casos, para bastantes muestras la precisión es alta a la vez que algunas muestras se acercan y superan el 50% de error relativo. La puntuación media de R2 con K Fold-CV es bastante buena. A continuación los resultados con la partición de test:




```
Test R2 Score: 0.7004780740656745
Test MSE: 5020796.935915327
Test MAE: 1838.771306381248
Test Max Error: 4894.912584091217
Test Explained Variance: 0.7016551441024413
```

El resultado ante la muestra de Test es algo decepcionante. Vemos que la generalización, no es mala, pero está bastante lejos de de lo que prometían las particiones de en la fase de entrenamiento. Vemos que los errores relativos se mantienen cerca del 25%, aunque en algunos casos concretos este es tanto muy bajo (por debajo del 5%) como algo alto (cerca del 50%). Nos quedamos con una puntuación R2 de 0.70.

KNN: K Nearest Neighbours Regression

KNN es un modelo muy potente basado en distancias. Este cuenta con diferentes parámetros a explorar. Mediante búsqueda exhaustiva y validación cruzada (Grid Search CV) hemos barajado los siguientes parámetros:

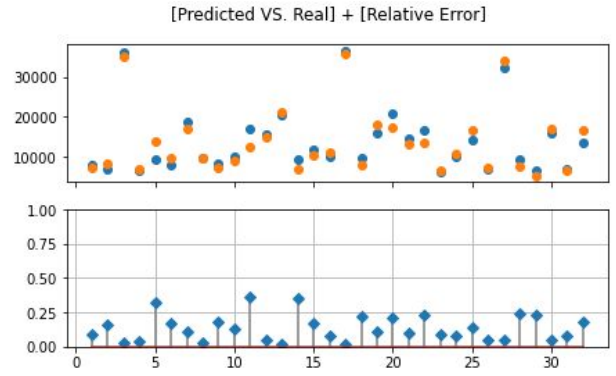
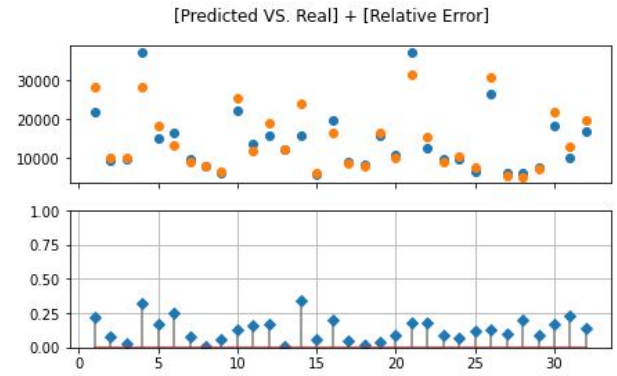
```
knn_params = {'n_neighbors' : [n for n in range(1,10)],
              'weights' : ['uniform', 'distance'],
              'metric' : ['euclidean', 'manhattan', 'chebyshev']}
```

Tras la ejecución de GridSearchCV sobre el estimador, se evaluó que los parámetros que han dado mejor resultado han sido:

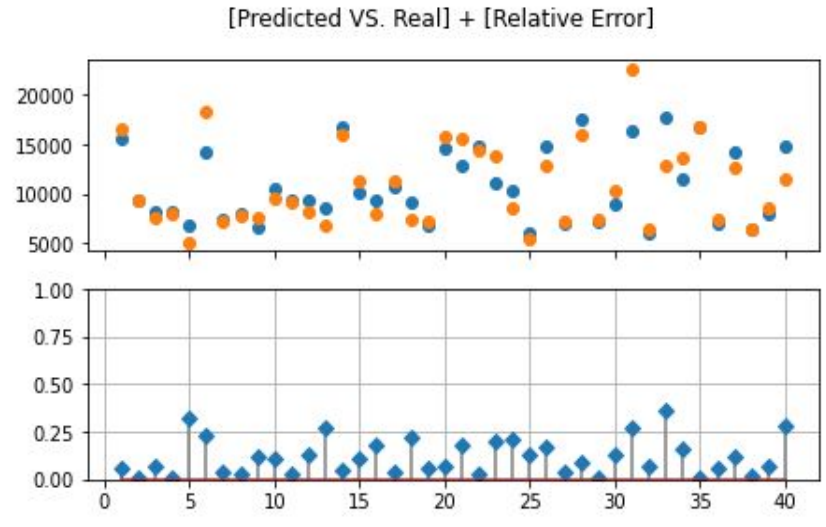
```
knn_params = {'n_neighbors' : [3],
              'weights' : ['distance'],
              'metric' : ['manhattan']}
```

Creemos que es totalmente adecuado que pondere las distancias de los vecinos para estimar el resultado, ya que la regla de “coches parecidos, precio parecidos” se manifiesta continuamente.

Hemos explorado manualmente con parámetros más extremos y otros tipos de distancias pero no nos han dado resultados mejores que los de los parámetros citados. A continuación mostramos los resultados de estos parámetros:

Mejor Fold	Peor Fold
 R2 Score: 0.94319872113837 ----------	 R2 Score: 0.8269217210018507 ----------
========== CrossVal Mean R2 Score: 0.8709180479257893 ==========	

Podemos ver que aparentemente KNN tiene un comportamiento muy bueno. Alcanzando una puntuación R2 de 0.87.




```
Test R2 Score: 0.7878095992075038
Test MSE: 3556884.5613099025
Test MAE: 1331.6601885905184
Test Max Error: 6186.7901744114315
Test Explained Variance: 0.7878132528719468
```

En cuanto a la partición de test, vemos que como pasó con Ridge perdemos cerca de un 0.10 de puntuación R2, pero en este caso, al partir de una puntuación de entrenamiento mucho más alta, nos topamos con, la hasta el momento, mejor puntuación R2 obtenida, alcanzando los 0.78 con un error medio de 1331 dólares.

Métodos No Lineales

MLP: Multiple Layer Perceptron Regression

Al tratar con modelos MLP contamos con el hándicap de que el coste computacional del entrenamiento es bastante alto. Esto nos ha penalizado bastante en la búsqueda de hiperparametros y sobretodo, en el número de particiones para hacer validación cruzada a la hora de evaluar dichos hiperparametros.

Inicialmente planteamos conjuntos de parámetros muy esparzos, con el fin de saber por que zona debíamos movernos. Vimos que las redes de 4 capas, en nuestro problema no, no solían generar resultados suficientemente mejores que las redes de 2 o 3 capas, por lo que limitamos la búsqueda exhaustiva hasta 3 capas.

En cuanto al solver de la MLP escogimos directamente Adam, por ser el que nos dio convergencias en menos iteraciones y más rápidas, gracias a poder introducir un descenso del gradiente adaptativo.

Finalmente la búsqueda exhaustiva se realizó sobre los siguientes parámetros:

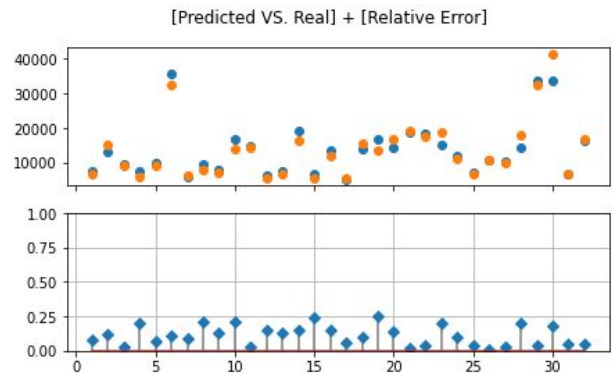
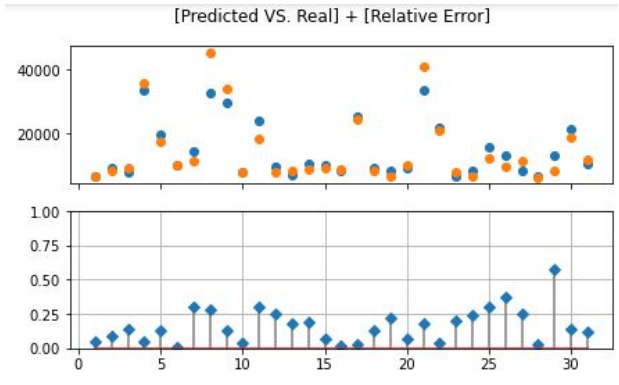
```
sizes = [[i for i in range(1,25)]]
sizes = sizes + [[i,i] for i in range(1,25)]
sizes = sizes + [[i,i,i] for i in range(1,25)]

param_grid = {'alpha' : (0.00001, 0.0001, 0.001, 0.01),
              'hidden_layer_sizes': sizes,
              'activation': ('identity', 'relu', 'logistic', 'tanh'),
              'learning_rate_init':[0.005,0.05,0.001,0.1]
             }
```

Y los parámetros obtenidos por validación cruzada fueron los siguientes:

```
param_grid = {'alpha' : [0.001],
              'hidden_layer_sizes': [[24, 24]],
              'activation': ['relu'],
              'learning_rate_init' : [0.001]
             }
```

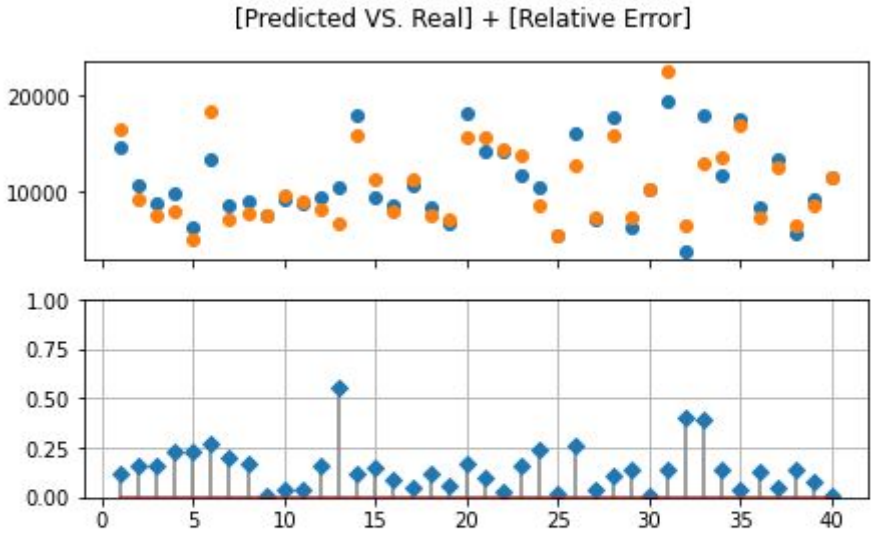
```
MLPRegressor(max_iter=100000,
             solver='adam',
             learning_rate='adaptive',
             random_state=42),
```

A continuación mostramos los resultados de estos parámetros:

Mejor Fold	Peor Fold
 R2 Score: 0.9336764109447286	 R2 Score: 0.8921830233282817
CrossVal Mean R2 Score: 0.9171952077885446	

Los resultados son buenos. Obtenemos un 0.91 de R2 en la partición de entrenamiento.

En cuanto los resultados en la partición de Test, nos llevamos un pequeño fiasco, ya que vemos que la red no termina de generalizar como nos gustaría. Esta parece que está sobre ajustada a los datos de entrenamiento, ya que llegamos a perder cerca de un 0.14 de puntuación R2.




```
Test R2 Score: 0.7786831478903247
Test MSE: 3709868.549597693
Test MAE: 1483.5024510411888
Test Max Error: 5091.686306828378
Test Explained Variance: 0.7830540128578598
```

Random Forest Regression

El último modelo que hemos utilizado ha sido el RandomForest Regressor. Este, a diferencia de las MLP se entrena bastante rápido. Por esta razón hemos podido realizar bastante más exploración de hiperparametros.

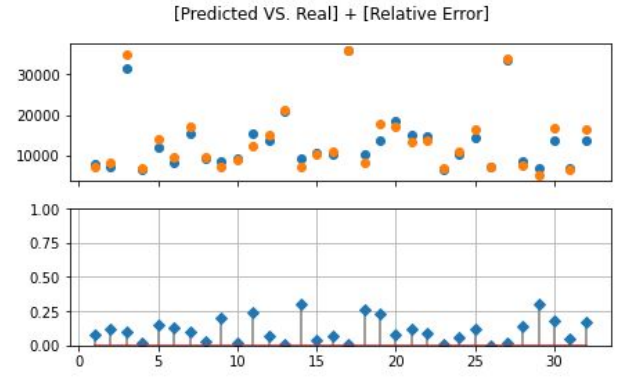
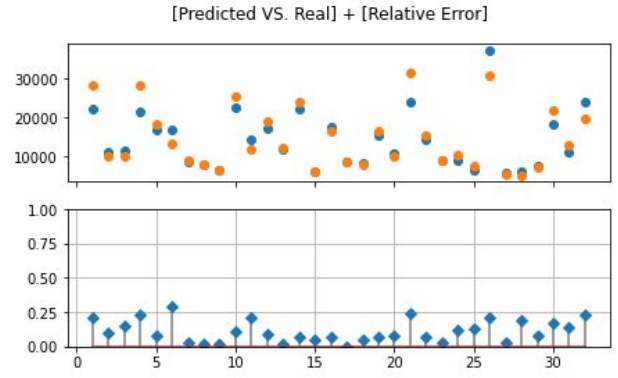
Estos han sido los hiperparametros explorados en la última iteración:

```
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]
```

Y estos han sido los parámetros escogidos por validación cruzada:

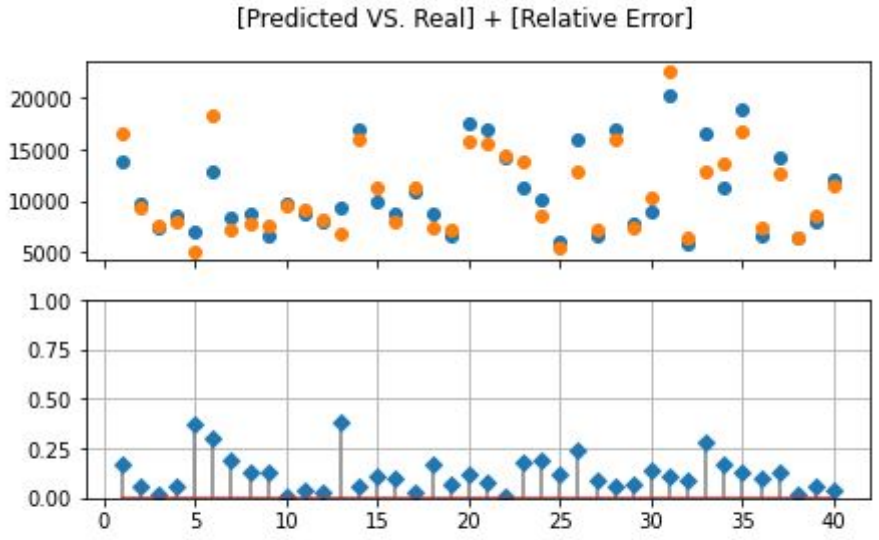
```
param_grid = {'n_estimators': [200],
              'min_samples_split': [5],
              'min_samples_leaf': [1],
              'max_features': ['sqrt'],
              'max_depth': [40],
              'bootstrap': [False]}
```

Los resultados con estos hiperparametros son los siguientes:

Mejor Fold	Peor Fold
 R2 Score: 0.9557265476319248	 R2 Score: 0.8621433163640351
========== CrossVal Mean R2 Score: 0.9130278204531435 ==========	

Vemos que el ajuste es muy bueno. La única preocupación en este punto es si, al igual que pasó con las MLP, estamos teniendo bastante sobreajuste, ya que el R2 es bastante alto, alcanzando los 0.91.

En cuanto a la partición de test, vemos que la generalización, como es normal nos hace perder precisión, pero nos mantenemos de igual forma en un valor muy bueno, 0.82. El error medio absoluto continúa estando, respecto a los otros modelos por encima de 1300 dólares.




```
Test R2 Score: 0.8217729964163878
Test MSE: 2987566.2380929566
Test MAE: 1309.9842812500003
Test Max Error: 5516.981250000003
Test Explained Variance: 0.8226962780136972
```

Modelo Final Escogido

Los dos modelos de regresión creemos que no han generalizado tan bien porque se han ajustado principalmente a donde hay mayor número de coches, en el precio medio-bajo. Por otro lado la MLP nos ha decepcionado porque se ha preajustado demasiado, quedándose a la altura de KNN, el cual nos ha sorprendido gratamente con un R2 final de 0.78.

Finalmente el que mejor resultados nos ha dado, el Random Forest. Por ser superior en R2, por no perder tanta puntuación a la hora de generalizar y porque creemos que se expandirá bien a nuevos datos, hemos decidido quedarnos con este como modelo final.

Nos habría gustado ser más precisos, pero creemos que hay bastantes más datos/variables que influyen en el precio de un vehículo, y desde nuestro punto de vista, contar con un error medio de 1300 dólares esta muy bien.

Está claro que si intentamos predecir coches de otras marcas no presentes en el dataset (sin tener en cuenta que a priori esa entrada no estaría aceptada por que las variables están codificadas en OneHot), este fallará estrepitosamente, dado que estamos seguros (por lo que hemos visto en la regresión lineal) que esto afecta mucho al precio y el resultado sería probablemente incoherente.

En resumen, para datos de coches parecidos, creemos que el modelo generaliza bien, pero si tratamos de introducir valores ligeramente desplazados de lo convencional, como podría ser un coche Chrysler deportivo, el modelo fallará.

Conclusiones y reflexiones

Creemos que el trabajo realizado es adecuado y estamos contentos con los resultados obtenidos, a pesar de que en relación a algunos modelos, algo más de conocimiento sobre los mismos nos habría venido bien. El estudio estadístico podría haber sido mucho mejor y por lo tanto, se habrían tomado decisiones más sólidas sobre cómo tratar los datos en fase de pre-proceso.

Nos ha quedado en el tintero, cómo podríamos mejorar la MLP, ya que creemos que deberían generalizar mucho mejor. Por otro lado nos sorprendió muchísimo como la regresión lineal es capaz de proporcionar resultados tan buenos, para lo simple que es.

En cuanto al modelo nos da pena no haber contado con un dataset mas grande y actual para poder probar con nuestros vehículos. Ya que el dataset está altamente sesgado por región y época es difícilmente expandible, no solo por eso, sino porque se utilizan datos de aseguradoras de las cuales ni sabemos su origen ni cómo extraerlo.

Referencias

Librerías/Métodos de Python utilizadas:

- Pandas
- Numpy
- Seaborn
- Sklearn

```
# Imports
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, KFold, cross_val_score
from sklearn.feature_selection import RFE
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.linear_model import LinearRegression, LassoCV, RidgeCV
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
```

Información

- **Ameisen, E**. (2018, 20 junio). Always start with a stupid model, no exceptions. - Insight. [Recuperado enero de 2021, de Enlace]
- API reference — pandas 1.2.0 documentation. (2020). [Recuperado enero de 2021, de Enlace]
- API Reference — scikit-learn 0.24.0 documentation. (2020). [Recuperado enero de 2021, de Enlace]
- **Brownlee, J.** (2020, 18 agosto). How to Perform Feature Selection for Regression Data. [Recuperado enero de 2021, de Enlace]
- Choice of K in K-fold cross-validation. (2012, 4 mayo). [Recuperado enero de 2021, de Enlace]
- **Finance Train.** (2020, 7 enero). Cross Validation to Avoid Overfitting in Machine Learning. [Recuperado enero de 2021, de Enlace]
- **Koehrsen, W.** (2019, 10 diciembre). Hyperparameter Tuning the Random Forest in Python - Towards Data Science. [Recuperado enero de 2021, de Enlace]