16 de mayo del 2021

REPORTE VEHICLE PRICE PREDICTION MODEL

MLM FINAL PROJECT

por Hans Walter

Introduction

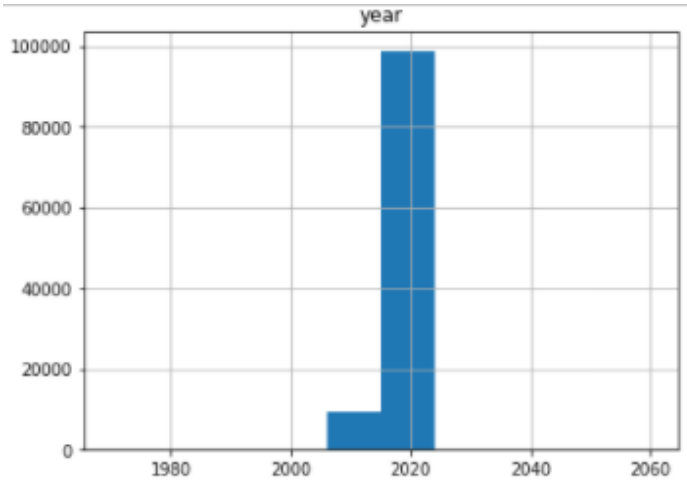
En este proyecto final, estuvimos trabajando con un dataset sobre precios históricos de carros en el Reino Unido. El punto del proyecto es poder desarrollar un modelo que sea capaz de predecir el precio del carro, a base del set de features que trae el dataset y poder minimizar el MAE(Mean Absolute Error) lo más posible. De esta manera podemos asegurar que nuestro modelo es adecuado para poder predecir el precio de un carro si nos proveen data foránea. En el proyecto utilicé 5 modelos los cuales son: Linear Regression, Decision Tree, Adaboost Regression, GradientBoosting y KNN Neighbours. Mi modelo resultó con un MAE relativamente alto, el cual puede mejorar si más adelante podría probar más parámetros con más poder computacional como también tiempo.

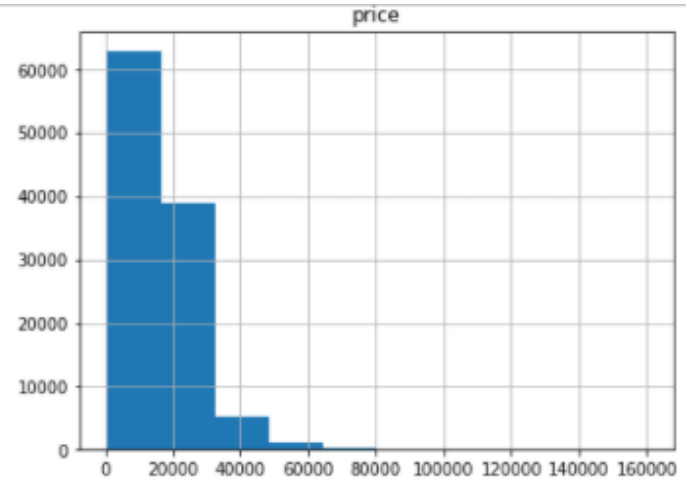
Data

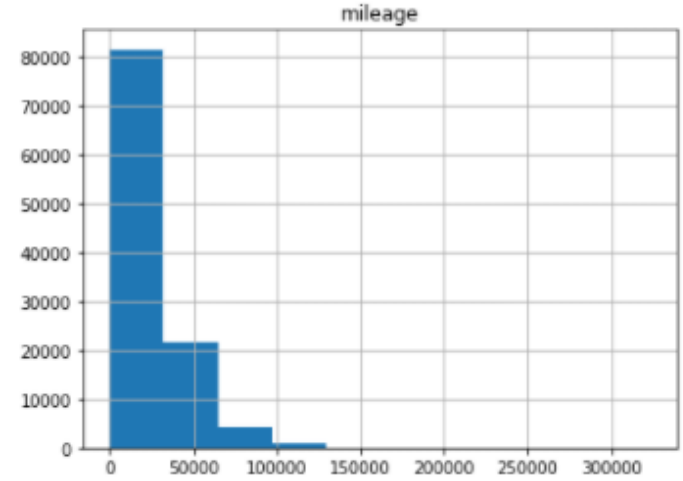
El dataset cuenta con 108,540 observaciones, y con 10 columnas las cuales 9 son variables independientes y solo 1 (price) es la variable objetivo. De las variables X's, 3 son categóricas (año lo tomamos como número y model la eliminamos) y 4 son números. Durante el análisis de NA's, noté que las columnas **Tax** y **MPG** tenían la misma cantidad de datos faltantes(9,353). Es interesante ver la relación de los datos faltantes ya que por sí, las variables no tienen nada que ver entre ellas. Durante el análisis de la distribución de los datos, noté que la mayoría de columnas tenían una distribución no normal. Adicionalmente, todas contaban con outliers.

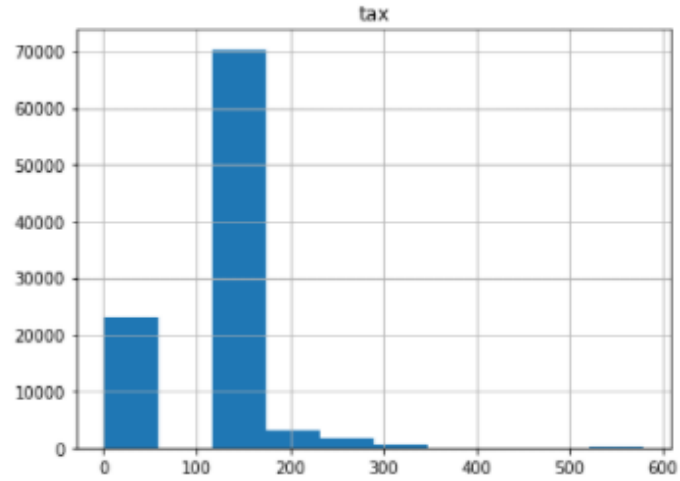
Dataset Columns

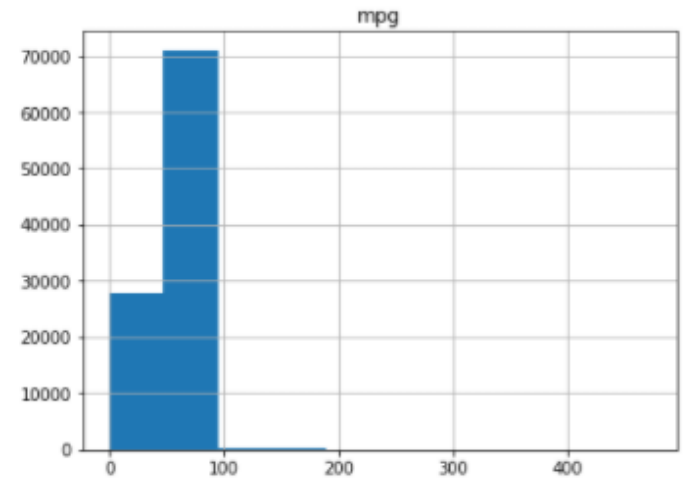
`model:`	Model of the car.
`year:`	Year the car was built.
`price:`	Price of sale (**Target variable**).
`transmission:`	Type of transmission of the vehicle.
`mileage:`	Current mileage of the vehicle.
`fuelType:`	Fuel type of the vehicle.
`tax:`	Current tax value of the vehicle.
`mpg:`	Motor efficiency in miles per gallon.
`engineSize:`	Size of the engine.
`make:`	Vehicle manufacturer.

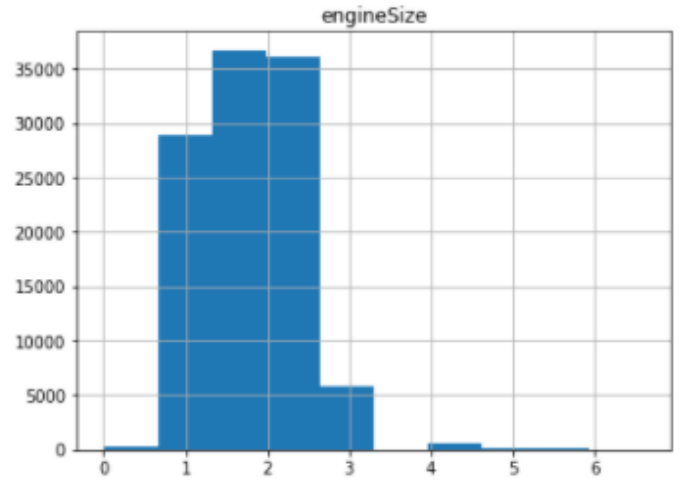

year
100000
80000
60000
40000
20000
0
1980
2000
2020
2040
2060


price
60000
50000
40000
30000
20000
10000
0
0
20000
40000
60000
80000
100000
120000
140000
160000


mileage
80000
70000
60000
50000
40000
30000
20000
10000
0
0
50000
100000
150000
200000
250000
300000


tax
70000
60000
50000
40000
30000
20000
10000
0
0
100
200
300
400
500
600


mpg
70000
60000
50000
40000
30000
20000
10000
0
0
100
200
300
400


engineSize
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
6

Methods

NA's en Tax y MPG: Para los valores nulos en ambas columnas, usé KNN imputer dado a que la data faltante era MNAR (Missing Not At Random).

Processing Categorical Columns: Utilicé one hot encoding para poder hacer dummify las variables, es decir, volverlas númericas para que el modelo de machine learning pudiera procesarlas.

Processing Numerical Columns: Apliqué el método de Normalizer, ya que la distribución de los datos era no normal. Normalmente usaría Standard Scaler pero este solo es apropiado cuando los datos tienen un comportamiento de la campana de Gauss.

Pipeline: El processing de numerical y categorical columns fueron combinados en un pipeline para poder facilitar el proceso de aplicar las transformaciones necesarias.

Methods

Linear Regression: El modelo es una regresión lineal simple. Calcula los thetas respectivos para cada variable independiente con la data de entrenamiento. Al pasarle los nuevos datos, los pasa multiplicando dentro de su theta correspondiente para finalmente sumar los resultados y tener una "y". El parámetro de *fit_intercept=True* es para utilizar el intercepto en los cálculos, y ponemos *n_jobs=-1* para que el modelo utilice todos los núcleos de la computadora al procesar la información.

Methods

Decision Tree Regressor: Funciona como un Decision Tree Classifier, donde en cada nodo establece un umbral de los features X's. La observación va pasando de nodo en nodo, hasta finalmente logra el modelo hacer una predicción "y". Aquí también se trabaja con los coeficientes Gini, donde el árblo termina de expanderse cuando el coeficiente no logra reducirse más. El root node es la variable que tiene el gini coefficient más bajo. El parámetro de splitter sirve para escoger de qué manera se divide el árbol. Normalmente se hace con el método best que sería por medio del coeficiente gini que se establecen las divisiones. Sin embargo, existe la opción de hacerlo random puesto que a veces es mejor un método totalmente aleatorio para poder utilizar en la regresión del decision tree. La opción de max_depth significa qué tan profundo puede ser el árbol / cuántos niveles puede tener. Dejamos un intervalo de 5, 10, 15 porque si incrementamos la profundidad corremos riesgo a que el decision tree haga overfit. Finalmente el min_samples_leaf se refiere a la cantidad de observaciones que deben de estar presentes en el leaf node para poder considerarse un leaf node. El random_state se le puede pasar cualquier número, simplemente es para objetivos de reproducibilidad del modelo. El criterion lo dejamos como mae que es Mean Absolute Error. Nuestra métrica de decisión de modelo es Mean Absolute Error por lo que es adecuado utilizar el mismo. El CV se refiere a la cantidad de cross-validations o k-folds que deseamos realizar, lo mantenemos en 3 por temas de tiempo y poder computacional. Verbose con valor de 1, como lo dice en la documentación de sklearn despliega: "the computation time for each fold and parameter candidate is displayed". n_jobs con valor -1 es para que el modelo utilice todos los procesadores de la computadora al momento de procesar el modelo. return_train_score lo dejamos como True, sin embargo esto solo se refleja cuando uno saca cv_results para que incluya los training scores.

Adaboost Regressor: El método de boosting, se refiere a la forma de entrenar modelos de una manera secuencial. Es decir, que va aprende a base de las etapas anteriores. Adaboost crea numerosos stumps secuencialmente. Corrigiendo los errores de los procesos anteriores para tener un modelo más óptimo. Boosting incluye varias etapas de training, y al final se utiliza un group weighted voting para clasificar/hacer regressions. El learning_rate en términos simples, detalla qué tan rápido o lento aprende el modelo. Un valor más cercano a 1 aprende más rápido, y si es más alejado de 1 aprende más lento. Los n_estimators se refiere al número de sequential trees que se realizarán en el modelo. (Se refiere a árboles ya que el default estimator del modelo es decision tree. El parámetro de loss simplemente detalla cómo va a penalizar los errores del modelo, linear siendo el más permisivo y exponential el que más penaliza. El random_state se le puede pasar cualquier número, simplemente es para objetivos de reproducibilidad del modelo. El CV se refiere a la cantidad de cross-validations o k-folds que deseamos realizar, lo mantenemos en 3 por temas de tiempo y poder computacional. Verbose con valor de 1, como lo dice en la documentación de sklearn despliega: "the computation time for each fold and parameter candidate is displayed". n_jobs con valor -1 es para que el modelo utilice todos los procesadores de la computadora al momento de procesar el modelo. return_train_score lo dejamos como True, sin embargo esto solo se refleja cuando uno saca cv_results para que incluya los training scores.

GradientBoosting: A diferencia del Adaboost, lo que hace GradientBoosting es que empieza con un leaf node que adivina el peso de todos los samples. Después, este crea un árbol que normalmente es más grande que los stumps creados por el Adaboost. El algoritmo crea árboles que van influenciados por los árboles anteriores, y los hace "scale". Esto se repite hasta que alcanza el límite que le ponemos nosotros o cuando ya no puede mejorar el rendimiento de los árboles. El modelo trabaja con los residuos de la diferencia entre las predicciones y los valores reales. El random_state se le puede pasar cualquier número, simplemente es para objetivos de reproducibilidad del modelo. El parámetro sub_sample lo dejamos en 0.8, si el valor es menor a cero hace que el modelo reduzca variance/overfitting. De igual manera, el learning_rate se puede ajustar reduciendolo para que tenga menos variance o aumentándolo para que tenga menos bias. Max_features lo establecemos como 7, puesto que nuestro dataset simplemente tiene 8 features. Dejamos el número más bajo que el total de features para poder reducir los features más relevantes al modelo.

KNN Neighbours: El modelo trabaja con la metodología de KNN neighbours, donde las predicciones son realizadas con los vecinos a los que se encuentra más cercano el valor de predicción. El parámetro de weights puede tomar forma como uniform, donde toma en cuenta los pesos de todos los neighbors de igual manera, o distance que mientras más cercano sea el neighbor va a tener más influencia que los que se encuentren más lejos. El algorithm lo dejamos en auto, el modelo escoge entre kd_tree, ball_tree o brute. Son varios algoritmos que utiliza el modelo y dependiendo de los valores que se le pasen en la función de fit, escoge el más apropiado. El CV se refiere a la cantidad de cross-validations o k-folds que deseamos realizar, lo mantenemos en 3 por temas de tiempo y poder computacional. Verbose con valor de 1, como lo dice en la documentación de sklearn despliega: "the computation time for each fold and parameter candidate is displayed". n_jobs con valor -1 es para que el modelo utilice todos los procesadores de la computadora al momento de procesar el modelo. return_train_score lo dejamos como True, sin embargo esto solo se refleja cuando uno saca cv_results para que incluya los training scores.

Results

Decision Tree MAE Score: 2,501

Fue el modelo con el mejor rendimiento de todos. Recordemos que el MAE es la diferencia promedio entre el valor de la predicción con el valor real "y". El modelo óptimo tiene parámetros de min_leaf de 10 observaciones, max_depth de 15 y su estrategia de splitting como best. El depth es adecuado ya que teníamos la opción de ponerle 20 pero el GridSearchCV escogió 15, evitando un overfitting.

Gradient Boost MAE Score: 2,894

El modelo quedó en segundo lugar de la lista. Tiene potencial en tener un MAE más bajo que el Decision Tree, pero sería cuestión de meter más parámetros al GridSearchCV. Adicionalmente, se requiere de bastante poder computacional y tiempo porque el algoritmo tardó aproximadamente 10 minutos en procesar.

KNN Neighbours MAE Score: 3,109

El modelo no trabajó con tantos parámetros a diferencia de Gradient Boost, por lo cual estaba limitado a las diferentes combinaciones de modelo que podía utilizar. Resultando así en un MAE score subóptimo.

Linear Regression MAE Score: 3,438

Dado a que el modelo no tiene el nivel de complejidad y parámetros para poder jugar con, es posible que por lo mismo sacó un MAE relativamente alto.

Adaboost Regression MAE Score: 4,601

Este fue el peor modelo de todos, y es una combinación de su learning rate como también de la función de loss que utilizó. El base estimator es un Decision Tree y el modelo es una colección de los mismos. El GridSearchCV determinó que el mejor modelo contaba con una función de loss exponencial la cual penaliza severamente los datos alejados, como también regresaba un learning rate de 0.1 el cual puede ser que tuvo problemas de overfitting. Sin embargo, el depth de los trees era de 3 (que es el default value), por lo cual lo más probable es que el modelo tenga bias.

Conclusions

El propósito del proyecto era lograr crear un modelo capaz de predecir el precio de los carros basado en un set de features. Nuestro mejor modelo terminó siendo DecisionTree, pero con las herramientas necesarias el Adaboost Regressor puede superar el score de MAE. La data que nos proporcionaron era de buena calidad, pues no tenía un número alto de data faltante y tampoco errores de ingreso de datos. Como se ha resaltado a lo largo del proyecto, con suficiente tiempo como también recursos computacionales es posible crear un modelo más robusto y con un score MAE sumamente más bajo. Por cuestiones de tiempo y alcance, decidí quedarme con los modelos y parámetros que enseñé anteriormente. El modelo tiene posibilidad de mejorar si se pudiera hacer un subsample de los modelos para poder tomarlo en cuenta como un feature. Hay que resaltar que el uso de la columna de años como número y no como variable categórica pudo haber afectado el rendimiento de los modelos. Se puede concluir que es posible predecir el precio de los carros basados en los features del dataset.

References

- Geller, S. (2019, April 5). Normalization vs Standardization - Quantitative analysis. Medium. https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf#:~:text=Normalization%20typically%20means%20rescales%20the,of%201%20(unit%20variance)

- joshstarmer. (2019, March 25). Gradient Boost Part 1 (of 4): Regression Main Ideas. YouTube. https://www.youtube.com/watch?v=3CC4N4z3GJc.

- Obadia, Y. (2018, February 21). The use of KNN for missing values. Medium. https://towardsdatascience.com/the-use-of-knn-for-missing-values-cf33d935c637.

- Rocca, J. (2021, March 21). Ensemble methods: bagging, boosting and stacking. Medium. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.

- sklearn.ensemble.AdaBoostRegressor¶. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html.

- sklearn.ensemble.GradientBoostingRegressor¶. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html.

- sklearn.linear_model.LinearRegression¶. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

- sklearn.neighbors.KNeighborsRegressor¶. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html.

- sklearn.tree.DecisionTreeRegressor¶. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html.

- Team, T. A. I. (2020, May 29). How, When, and Why Should You Normalize / Standardize / Rescale Your Data? Towards AI - The Best of Tech, Science, and Engineering. https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff.

- YouTube. (2021, February 4). Decision Tree Regression Clearly Explained! YouTube. https://www.youtube.com/watch?v=UhY5vPfQIrA.