Universidad Europea de Madrid

Escuela de Arquitectura, Ingeniería y Diseño

Trabajo final: Used Cars Dataset

Proyecto de Open Data II

Profesor: Rafael Muñoz Gil

Presentado por:

Elena Delgado del Rey

Javier García García

2 DE JUNIO DEL 2020

Universidad Europea

Índice

Recordatorio

Used Cars Dataset reúne todas las entradas de vehículos usados dentro de los Estados Unidos. Tras hacer el estudio del dataset con su respectiva limpieza y rellenado de datos, nos quedamos con las columnas necesarias para entrenar el modelo.

Las columnas con las que decidimos quedarnos fueron las siguientes:

- **City**: reúne los estados donde se compraron los coches de segunda mano.
- **Price**: agrupa los precios de los coches en dólares.
- **Year**: almacena los años donde se comparan los coches desde 1990 hasta 2020.
- **Manufacturer**: contiene las marcas de los coches.
- **Make**: reúne los modelos de los coches.
- **Condition**: agrupa la condición en la que se vendieron los coches.
- **Cylinders**: almacena la cantidad de cilindradas.
- **Fuel**: contiene el tipo gasolina.
- **Odometer**: reúne los kilómetros recorridos por el coche.
- **Transmission**: agrupa el tipo transmisión (automático, manual u otro)
- **Drive**: es el tipo de tracción (rwd, fwd y 4wd)
- **Type**: es el tipo de coche
- **Paint_color**: contiene los colores de cada coche vendido.

Inicios

Para comenzar este trabajo, importamos dos tablas. La primera que tenía gran parte de las columnas numéricas (coches_df) y la segunda que venían los valores originales (coches1_df). Esto lo hicimos para poder visualizar nuestros datos numéricos y saber que significaban en caso de que lo necesitáramos para la predicción.

Para poder empezar a entrenar nuestro modelo, tuvimos que crear un esquema de nuestro dataset numérico y descriptivos, ya que es el que vamos a utilizar para realizar las predicciones. A continuación mostramos el esquema:

```
root
 |-- _c0: integer (nullable = true)
 |-- city: integer (nullable = true)
 |-- price: integer (nullable = true)
 |-- year: double (nullable = true)
 |-- manufacturer: integer (nullable = true)
 |-- make: string (nullable = true)
 |-- condition: string (nullable = true)
 |-- cylinders: integer (nullable = true)
 |-- fuel: integer (nullable = true)
 |-- odometer: double (nullable = true)
 |-- transmission: double (nullable = true)
 |-- drive: integer (nullable = true)
 |-- type: integer (nullable = true)
 |-- paint_color: integer (nullable = true)
```

Como podemos ver arriba gran parte de las columnas de nuestro dataset son de tipo Integer exceptuando year, odometer y transmisión que eran de tipo Double, y make y condition que eran de tipo String.

Viendo las columnas que teníamos nos dimos cuenta que algunas no eran necesarias por lo que creamos un nuevo Data Frame llamado cochesnuevo con las siguientes columnas: price, city, year, manufacturer, fuel, odometer, transmission, drive, type y paint_color.

Tras evaluar nuestros datos y preparar el Dataset pensamos para qué queríamos entrenar a nuestro modelo. Teníamos dos opciones o que nos predijera la marca del coche según los valores y las características o el precio. Dado a que nuestro Dataset es de ventas de coches quisimos hacer la predicción de los precios. Lo cual significaba que nuestro estudio iba a ser de regression.

SQL

Para poder visualizar nuestros datos creamos tablas SQL, revisamos cuantas filas teníamos para asegurarnos de que no habíamos perdido ninguna por el camino y procedimos a visualizar los datos.

Como no queríamos centrar gran parte de nuestro trabajo en esta parte, lo que hicimos fue visualizar los distintos tipos de color que tenían los coches, y las primeras ciudades en aparecer en nuestra lista. Estos datos los sacamos de la tabla con Strings.

Por otro lado, imprimimos dos tablas para tener una visualización más general de los datos con los que vamos a trabajar.

Una vez tuvimos nuestros datos preparados y sabiamos que queriamos predecir, toca entrenar al modelo y elegir qué método nos iba mejor.

ENTRENAMIENTO

Como nuestro dataset no pertenecía a ningún Challenge de Kaggle, decidimos que la variable más interesante para iba a ser price, por lo que tuvimos que utilizar un algoritmo de regresión. La regresión es un algoritmo de aprendizaje supervisado, es una aproximación para modelar la relación entre una variable escalar dependiente y una o más variables explicativas nombradas con.

Teniendo en cuenta que nuestro tenemos que usar un algoritmo de regresión, vamos a ver todos las posibilidades que tenemos. Los algoritmos que podíamos usar son los siguientes:

- Decision Tree Regressor
- Gradient Boosted Tree Regressor
- Linear Regression
- Random Forest Regressor

Pasos de los Algoritmos

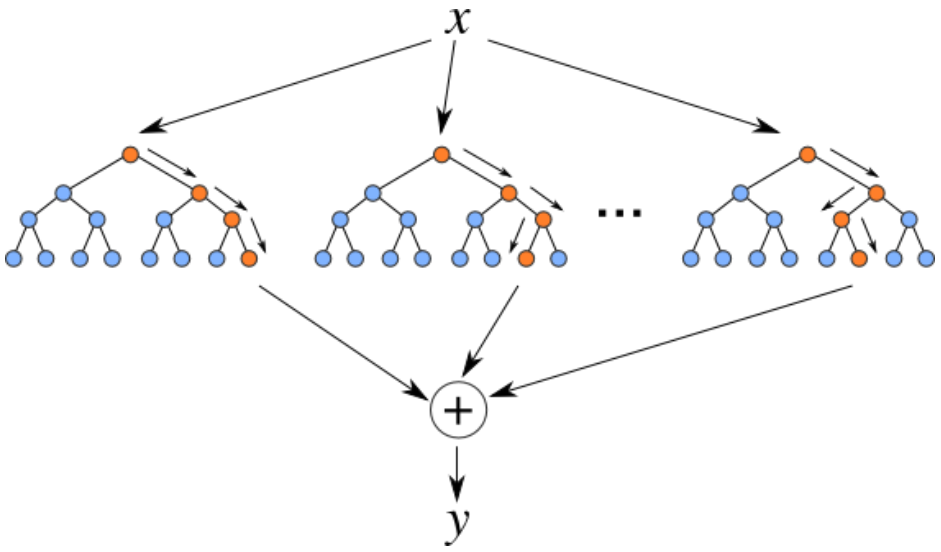
Los pasos que hemos seguido para realizar los 3 algoritmos han sido:

1. Para realizar este algoritmo vamos a empezar creando una columna que reúna todas las variables utilizando el transformador VectorAssembler.
2. Ahora vamos a preparar nuestro modelo con sus respectivos valores en caso de tener.
3. Una vez tenemos todo lo anterior preparado, hacemos un pipeline. El Pipeline nos va a ayudar a meter nuestro VectorAssembler y el modelo para ejecutarlo uno detrás de otro.
4. Modelamos nuestros datos de training el cual tiene el 80% de nuestros datos.
5. A continuación entrenamos el modelo y visualizamos las predicciones que nos ha hecho.
6. Finalmente, evaluamos el modelo y calculamos el RMSE (raíz del error cuadrático medio) y el MSE (error cuadrático medio) de nuestro modelo.

Antes de comenzar a predecir la variable price, decidimos convertirla en DoubleType para poder utilizarla en los modelos. Por otro lado separamos los datos en test y train, el

80 % va a ser de los datos de entrenamiento y el otro 20 % es para los datos de test. Estos datos los usaremos para entrenar el modelo.

Random Forest

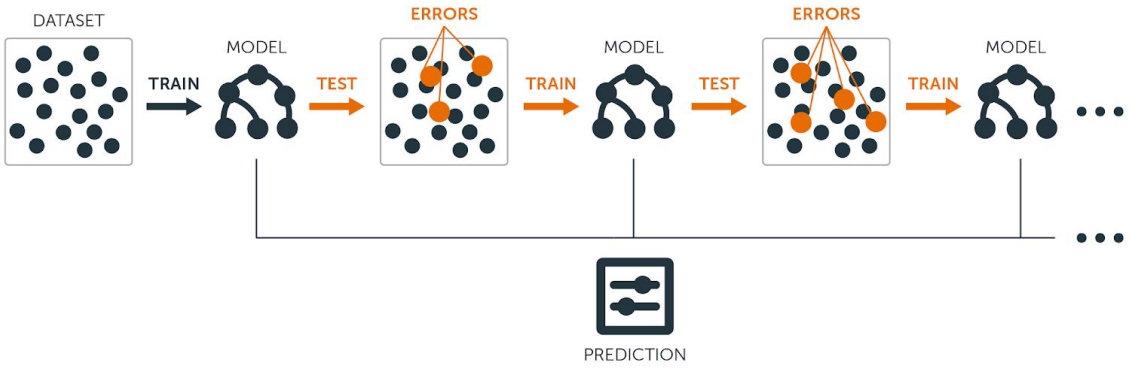


El primer algoritmo que decidimos utilizar fue Random Forest ya que este algoritmo de regresión consiste en un conjunto de múltiples árboles de decisión independientes, al cual se le asigna un conjunto de datos aleatorios con una misma distribución. La salida es el promedio de los resultados finales de cada árbol.

Para realizar este método hemos utilizado lo siguiente:

- 5 árboles para entrenar nuestros datos
- 5 niveles de profundidad de los árboles
- 'price', la columna a predecir

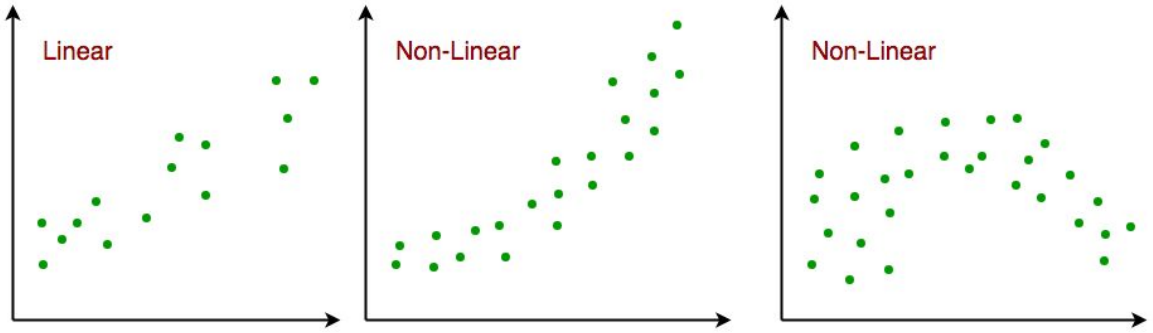
Gradient Boosted Tree Regressor



El segundo algoritmo que utilizamos fue Gradient Boosted Tree Regressor es una técnica de aprendizaje automático que produce un modelo de predicción en forma de un conjunto de modelos de predicción débiles, típicamente árboles de decisión. En estos casos, construye cada árbol de regresión de forma gradual, utilizando una función de pérdida predefinida para medir el error en cada paso y corregirlo en el siguiente.

Habiendo visto que Random Forest no es un modelo adecuado para nuestros datos, hemos decidido comprobar si este modelo nos proporciona mejores predicciones.

Linear Regression



El tercer algoritmo que utilizamos fue Linear Regression es un modelo que aproxima la dependencia entre una variable dependiente y las independientes.

Comparación de Algoritmos

	Error Cuadrático Medio
Random Forest	1.15632e+08
Gradient Boosted Tree Regressor	4.48182e+07
Linear Regression	1.51051e-19

Como podemos ver el mejor algoritmo es **Linear Regression** ya que tiene un error de predicción casi nulo. El segundo mejor algoritmo es Gradient Boosted Tree Regressor y finalmente estaría Random Forest.

Grid Search

El Grid Search es el proceso de escanear los datos para configurar parámetros óptimos para un modelo dado. Grid-Search creará un modelo en cada combinación de

parámetros posible. Recorre cada combinación de parámetros y almacena un modelo para cada combinación.

Hemos realizado el método de grid search en nuestro algoritmo de random forest para ver qué parámetros son los más óptimos para nuestra predicción. Para poder realizar esto se le mete al método varios parámetros para su comparación. Los resultados que hemos obtenido han sido los siguientes:

- Número de árboles: 2
- Profundidad: 2
- Error: 1.02457e+08

Cross Validation con k folders

K-means es uno de los algoritmos de agrupamiento más utilizados que agrupa los puntos de datos en un número predefinido de grupos. Cross validation es cualquiera de varias técnicas de validación de modelos similares para evaluar cómo los resultados de un análisis estadístico se generalizaron a un conjunto de datos independiente.

A continuación hemos utilizado Cross Validation. El estimador utilizado ha sido Random Forest Regressor, un evaluador de regresión y número de folders igual a 2.

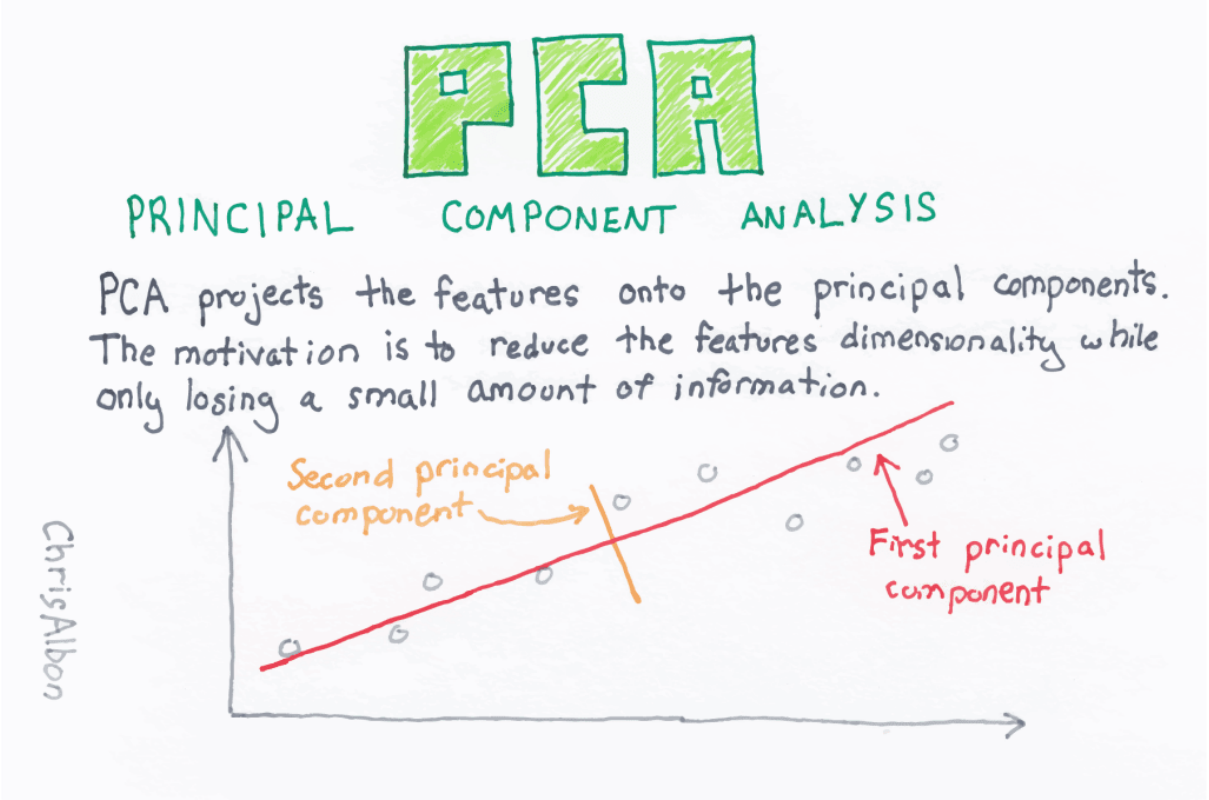
Discretizar variables continuas

Discretizar datos quiere decir convertir variables que son continuas en variables agrupadas por intervalos. Esta operación simplifica la información agrupando los objetos geográficos que presentan las mismas características en distintas clases. Hemos discretizado la columna de Price, ya que esta al ser una medición es una variable contínua.
Tras haber discretizado la columna, hemos realizado el algoritmo de regresión lineal con estos nuevos datos discretizados.

Estandarizar variables continuas

Por otro lado, también hemos estandarizado la columna Price debido a que es una variable continua, para ello hemos utilizado un Normalizer. La normalización de índices significa ajustar los valores medidos en diferentes escalas respecto a una escala común, a menudo previo a un proceso de realizar promedios.
Al igual que en el caso anterior, hemos realizado regresión lineal para predecir la columna price con estos nuevos valores

PCA



PCA es una técnica para resaltar patrones fuertes de un conjunto de datos. Este método se utiliza a menudo para facilitar la visualización y exploración de los datos.

Extracción de características

- ChiSqSelector: ChiSqSelector significa selección de funciones Chi-Squared. Opera en datos etiquetados con características categóricas. ChiSqSelector utiliza la prueba de independencia Chi-Squared para decidir qué características elegir. Para variables categóricas, te permite seleccionar un número de características/variables (definido en el parámetro 'numTopFeatures', que mejor explica la varianza. ->PCA

Tabla de datos completa

	Error Cuadrático Medio
Random Forest	1.15632e+08
Gradient Boosted Tree Regressor	4.48182e+07
Linear Regression	1.51051e-19
Random Forest (Gris Search)	1.02457e+08
Datos discretizados	4.52862e-19
Datos estandarizados	2.7131e-23

Conclusiones

Antes de comenzar con las conclusiones vamos a recordar de qué iban nuestros datos. Nuestro Dataset eran coches vendidos de segunda mano en Estados Unidos, por lo tanto eso nos colocaba en un modelo de regresión.

Tras evaluar los datos que teníamos, elegimos preparar nuestros datos para predecir el precio de los coches según los parámetros de entrada que consideramos más relevantes, por lo tanto tuvimos que borrar algunas columnas. Para esto decidimos entrenar nuestro modelo con tres algoritmos distintos, los cuales consideramos más adecuados para nuestro Dataset. Tras entrenar los datos con Random Forest, Gradient Boosted Tree Regressor y finalmente con Linear Regression, hemos podido apreciar que Linear Regression es el algoritmo que mejor predicciones nos da, con el error menor.

Como vimos que el que más error nos daba era Random Forest, decidimos hacer un Grid Search para visualizar cómo podríamos mejorar el modelo. Haciendo esto nos disminuye el error, indicándonos que el modelo se entrena mejor con menos árboles.

Como hemos indicado el modelo que mejor nos predice es Linear Regression vamos a hacer el entrenamiento con los datos discretizados y estandarizados. Después de hacer esto podemos ver que nuestro entrenamiento mejora.

Aprendizaje

En este curso hemos aprendido a implementar SQL en python, además de los diferentes tipos de aprendizaje que puede tener un modelo. Estos tipos de aprendizaje pueden ser de clasificación y de regresión, y al ser nuestros datos de tipo regresión nos hemos adentrado más en este campo. Hemos aprendido a implementar los diferentes tipos de algoritmos como Linear Regression o Random Forest, con sus respectivos entrenamientos y test.

También hemos aprendido cómo mejorar un aprendizaje, y estudiar su mejor comportamiento ya sea optimizando los parámetros y discretizando o estandarizando los datos.