Sports Car Project

By: Tilak Muley

Background Of The Data Chosen

Two datasets were used. One of the data used was: "Sports Car Prices dataset" which was name SportCar for this project. The dataset was found on kaggle.com(https://www.kaggle.com/datasets/rkiattisak/sports-car-prices-dataset/code). This contains data regarding Sport cars so the dataset has many variables: Car Make, Car Model, Year, Engine Size (L), Horsepower, Torque (lb-ft), 0-60 MPH Time (seconds) and Price (in USD). Car Make is the make/company of the car. Car Model is the model/version/variant of the sport car. Year of the sport car is the year the sport car was produced. Engine Size (L) is the size of the cars engine in liters and in some sport cars cases it could be a electric motor. Horsepower is the power output of the sport car engines. Torque (lb-ft) is the force produced by the sport cars engine and determines the acceleration of the car. 0-60 MPH Time (seconds) is the time it takes the sport car to accelerate from 0 MPH to 60 MPH. Price (in USD) is the purchasing price of the sport car in USD.

The second dataset used was Auto Groups which was named AutoGroups for this project this was a dataset created by me. The dataset includes Car Make, Country and Car Group. Car Make being the company, Country being the origin of the Car Make and Car group being what group the Car Make is apart of.

Goal

Do some data analysis to better understand both the datasets, with some EDA, data cleaning, data visualization, data wranging, data imputations, and data aggregation. As well as find some answers to questions such as: What Country makes the most powerful car determined by horsepower in each price bracket? As well as what is each Car Groups fastest car based on 0-60 MPH Time? Also how much correlation is between horsepower, torque and 0-60 MPH Time?

Loading Dataset and Libraries

In [ ]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import datetime
import statsmodels.api as stat
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot as plt
```

Load Dataset

In [ ]:
```
 #dataset assigned Sc
Sc= pd.read_csv('SportCar.csv',encoding= 'latin1')
```

In [ ]:
```
AG= pd.read_csv('AutoGroups.csv',encoding= 'latin1')
```

Exploratory Data Analysis on AutoGroups Data

In [ ]:
```
#First five of data
AG.head(5)
```

In [ ]:
```
#First five of data
AG.tail(5)
```

In [ ]:
```
#Shape of data
AG.shape
```

In [ ]:
```
#Information on data
AG.info()
```

In [ ]:
```
#check for missing values and print each column and the sum of each column
AG.isnull().sum()
```

Exploratory Data Analysis on SportCar Data

In [ ]:
```
#First five of data
Sc.head(5)
```

In [ ]:
```
#Last five of data
Sc.tail(5)
```

In [ ]:
```
#Shape of data
Sc.shape
```

In [ ]:
```
Sc.columns.tolist()
```

All of the column names were printed in a list to get an idea of what the original data set has.

In [ ]:
```
#Describing the data
Sc.describe()
```

With only year coming up when using .describe() that means the other columns/variable are not being recognized as numerical data. Which will be checked below to see what data type each column/variable is assigned.

In [ ]:
```
#Information on data
Sc.info()
```

Above all of the variables in the original data set have a data type of object except for year which is a int64. This is not ideal as columns like 'Engine Size (L)', 'Horsepower', 'Torque (lb-ft)', '0-60 MPH Time (seconds)', and 'Price (in USD)' should be represented as numeric types such as int or float. The data types will be modified later on.

In [ ]:
```
#check for missing values and print each column and the sum of each column
Sc.isnull().sum()
```

First it is checked for missing values and then returns the sum of null values for each column. This gives us a feel for the data set and how many missing values we may have to deal with.

In [ ]:
```
#checking duplicate values
Sc.nunique()
```

.unique() gives us the amount of unique values in each column of this dataset. Which helps us get an idea of exactly how many unique values exists and the variety.

In [ ]:
```
#Filter rows with missing values
rows_missing_values = Sc[Sc.isnull().any(axis=1)]

#Print rows with missing values
rows_missing_values
```

In [ ]:
```
#Sum of the number of rows with missing values
rows_missing_values_count = Sc.isnull().any(axis=1).sum()

print("Number of rows with missing values:", rows_missing_values_count)
```

Number of rows with missing values and the rows that are missing values are displayed. This will help decide if the rows with missing values will be completly removed or the missing values will be replaced with a value to help keep the row.

No plots can be made as of right now until data types are changed

Data Imputations

In [ ]:
```
#New data set is made by copying original data set
Sc2=Sc.copy()
```

In [ ]:
```
#Replace missing values in 'Engine Size (L)' column with 'Electric'
Sc2['Engine Size (L)'].fillna('Electric', inplace=True)
```

After futher review it was found that all rows missing a value in the 'Engine Size (L)' column were all cars with 'Electric' engines so the missing values in the column were replaced with 'Electric'

In [ ]:
```
#Filter rows with missing values
rows_missing_values = Sc2[Sc2.isnull().any(axis=1)]

#Print rows with missing values
rows_missing_values
```

In [ ]:
```
#Rows with missing values dropped
Sc2.dropna()
```

In [ ]:
```
Sc2.dropna(inplace=True)
```

Rows with missing values in the 'Torque (lb-ft)' were removed as averaging all the values in this column would not make sense as each car has a different torque value and a solution without looking up values couldn't be found.

In [ ]:
```
#Replace Electric/Electric Motor with 0 in the 'Engine Size (L)' column
Sc2['Engine Size (L)'] = Sc2['Engine Size (L)'].replace(['Electric', 'Electric Motor']
```

To help clean up the dataset and get the 'Engine Size (L)' column ready for a data type change any Electric/ Electric motor was changed to a 0 as Electric vehicles don't have a engine size.

Data Cleaning

In [ ]:
```
Sc2.drop_duplicates(inplace= True)
Sc2
```

To start cleaning the data any duplicates in the dataset was removed to help get the data ready and only have a single row for each car model so no data is repeating and skewing any results. So data drops from 1004 to 717 observations.

In [ ]:
```
Sc2['Price (in USD)']= Sc2['Price (in USD)'].str.replace(',','',regex= True)
```

To get some of the columns ready to convert to a numerical value for the data type any non-numeric characters need to be removed so the comma in the column 'Price (in USD)' needs to be removed first

In [ ]:
```
#All relevant columns were converted to numeric data types
numeric_columns = ['Engine Size (L)', 'Horsepower', 'Torque (lb-ft)', '0-60 MPH Time (
Sc2[numeric_columns] = Sc2[numeric_columns].apply(pd.to_numeric, errors='coerce')
```

Exploratory Data Analysis for New SportCar Data

In [ ]:
```
# Display updated data types
print(Sc2.info())
```

Columns: Engine Size (L),Horsepower,Torque (lb-ft),0-60 MPH Time (seconds) were all changed to float64 data types while column: Price (in USD) was changed to int64 data type. Car Make, Car Model can remain objects because they are just letters or words. As with Year it can remain int64 data type.

Data Wrangling

In [ ]:
```
#Merging the two datasets together
Scag= Sc2.merge(AG,on= 'Car Make')
Scag
```

AutoGroups Data and SportCar data were merged together on the column Car Make

Exploratory Data Analysis for Merged Data

Data Visualization

In [ ]:
```
#Grouped
grouped= Scag.groupby("Car Group")["Car Make"].nunique()

#bar plot
colors = ['blue', 'red','green','purple','yellow'] * (len(grouped) // 2 + 1)

# Create a bar plot with alternating colors
grouped.plot(kind='bar', color=colors[:len(grouped)])

#Tittle and labels
plt.title('Bar Plot of Number of Car Makes in each Car Group')
plt.xlabel('Car Group')
plt.ylabel('Number of Car Makes')

#plot
plt.show()
```

Bar graph above helps visualize how many Car Makes are in each Car Group. The .groupby() was used to make this. With Volkswagen Group leading the way with five.

In [ ]:
```
#Grouped
grouped= Scag.groupby("Country")["Car Make"].nunique()


#Bar plot with alternating colors
colors = ['orange', 'red','black','purple','yellow'] * (len(grouped) // 2 + 1)

grouped.plot(kind='bar', color=colors[:len(grouped)])
```

```
#Title and labels
plt.title('Bar Plot of Number of Car Makes From Each Country')
plt.xlabel('Countries')
plt.ylabel('Number of Car Makes')

#plot
plt.show()
```

In [ ]:
```
fig, ax = plt.subplots()

ax.pie(grouped, labels = grouped.index, autopct=lambda p: '{:.0f}'.format(p * sum(grou

plt.title('Pie Chart of Number of Car Makes From Each Country')
plt.show()
```

Bar plot and the pie chart above help visualize how many Car Makes from each Country. The .groupby() was used to make this. With England leading the way with 8 Car Makes from its country.

Data Aggregation

Most Powerful Car in Each Price Bracket by Horsepower

In [ ]:
```
#price brackets
bins = [0, 25000, 50000, 100000, 150000, 200000, float('inf')]
labels = ['<25K', '25-50K','50-100K', '100-150K', '150-200K', '>200K']
Scag['Price Bracket'] = pd.cut(Scag['Price (in USD)'], bins=bins, labels=labels)

#Powerful car in each price bracket
most_powerful_price = Scag.loc[Scag.groupby('Price Bracket')['Horsepower'].idxmax()]

most_powerful_price#
```

In [ ]:
```
#bar plot
plt.figure(figsize=(10,6))
c= 'red','blue','purple','orange','green','yellow'
plt.bar(most_powerful_price['Price Bracket'], most_powerful_price['Horsepower'], color
plt.xlabel('Price Bracket')
plt.ylabel('Horsepower')
plt.title('Bar Plot of Most Powerful Car in Each Price Bracket by HP')

#plot
plt.show()
```

6 Price brackets were created to see which country makes the most powerful car determined by horsepower in each price bracket. The bracket labels ended up being '<25K', '25-50K','50-100K', '100-150K', '150-200K', '>200K'. The results ended up being Car Makes made in America holding 5 of the first 6 spots ('<25K', '25-50K','50-100K', '100-150K', '150-200K',) and England getting the last spot (>200k). Then the bar plot made of Price Bracket and Horsepower was a bit interesting to see as rice increases so does horsepower but 100-150k price bracket actually has a car with more horsepower then the one in the 150-200k bracket.

Fastest Car in Each Car Group

In [ ]:
```
fastest_cars = Scag.loc[Scag.groupby('Car Group')['0-60 MPH Time (seconds)'].idxmin()]
fastest_cars = fastest_cars.sort_values('0-60 MPH Time (seconds)')
fastest_cars
```

In [ ]:
```
#Line plot
plt.figure(figsize=(10,6))
plt.plot(fastest_cars['Car Group'], fastest_cars['0-60 MPH Time (seconds)'], marker='c

plt.xlabel('Car Group')
plt.ylabel('0-60 MPH Time (seconds)')
plt.title('Line Plot of Fastest Car in Each Car Group')
#Rotate x-axis labels
plt.xticks(rotation=90)
#Plot
plt.show()
```

Fastest car for each Car Group based on '0-60 MPH Time (seconds)' was found above. Then it was sorted from fastest time to slowest time. Fastest 0-60 MPH Time turned out to be 1.8 seconds by the Bugatti Rimac joint company with the Rimac C_two. While the slowest fastest 0-60 MPH Time was 6.5 seconds by theMazda Motor Corporation with the Mazda MX-5 Miata. Creating a line plot to help visualize the fastest 0-60 MPH Time for each Car Group was very interesting as the gap isn't to big as you go from fastest to fastest slowest 0-60 from the next Group until you get to Hyundai Motor Group the gap between them and Mazda Motor Coperation is quite big when visualization over a 1 second difference.

Correlation Matrix

In [ ]:
```
corr = Scag[["Horsepower", "Torque (lb-ft)", "0-60 MPH Time (seconds)"]].corr()

# Generate a heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm')

# Show the plot
plt.show()
```

A Correlation matrix was created to show the correlation between Horsepower, Torque and 0-60MPH Time. 1 indicates a strong positive relationship, -1 indicates a strong negative relationship, and 0 indicates no relationship. So for example 0-60 MPH time and horsepower are connected to the box -0.73 which means as horsepower increases 0-60 MPH Time decreases which is expected as faster car equals faster 0-60 MPH Time. while another example Torque and Horsepower sharing a box with 0.95 that mean as horsepower increases torque increase as well. Overall these results are consistent with what we know. More horsepower equals faster 0-60 MPH time and increase in torque while more torque equals faster 0-60MPH time and increase in horsepower. So overall yes there is a correlation between 0-60 MPH Time and Horsepower and Torque.