```python
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

import warnings
warnings.filterwarnings('ignore')


# In[2]:


data=pd.read_csv("auto_mpg_dataset.csv",sep=",")


# In[3]:


data.head()


# In[4]:


data.tail()


# In[5]:


data.shape


# In[6]:


data.info()


# In[7]:


data.isna().sum()


# In[8]:


data.describe()


# In[9]:


sns.heatmap(data.isnull(),yticklabels=False,cmap="viridis")


# # columns of data

# In[10]:


data.columns


# # checking for duplicated values

# In[11]:


data.duplicated().sum()


# # Checking for value counts in categorical columns

# In[12]:


data['cylinders'].value_counts()


# In[13]:


data['model_year'].value_counts()


# In[14]:


data['origin'].value_counts()


# In[15]:


data["car_name"].nunique()
```

```python
# In[16]:


data["car_name"].value_counts().head(20)


# In[17]:


data["car_name"].values[:10]


# # Let's extract Brand information from car_name column

# In[18]:


data['brand'] = data["car_name"].str.extract('(^.*?)\s')
#brands=data["brand"]
#brands=data["brand"].unique().astype('str')
#brands
data["brand"].value_counts()[:10]


# # There are few brand names which are repeated but in different letters

# # replace those brand names as basic brand name

# In[19]:


data['brand'] = data['brand'].replace(['volkswagen','vokswagen','vw'],'VW')
data['brand'] = data['brand'].replace(['chevrolet','chevy','chevroelt'],'chevrolet')
data['brand'] = data['brand'].replace('maxda','mazda')
data['brand'] = data['brand'].replace('toyouta','toyota')
data['brand'] = data['brand'].replace('mercedes','mercedes-benz')
data['brand'] = data['brand'].replace('nissan','datsun')
data['brand'] = data['brand'].replace('capri','ford')
data['brand'] = data['brand'].replace('nissan','datsun')


# # Checking for any null values in brand

# In[20]:


data[data['brand'].isnull()]


# # We can fill those values with their car name "subaru"

# In[21]:


data['brand'].fillna(value = 'subaru',inplace=True)


# In[22]:


data["brand"]=data["brand"].str.capitalize()


# In[23]:


data.head()


# In[24]:


def country(x):
    if x==1:
        return "USA"
    elif x==2:
        return "Europe"
    elif x==3:
        return "Japan"


# In[25]:


data["origin"]=data["origin"].apply(country)


# In[26]:


data.head()


# In[27]:


sns.pairplot(data,hue="origin")


# # There is some issue with "horsepower" column
#

# # These are the outliers in our data

# # Distribution of horsepower for all cars
```

```python
# In[28]:


data["horsepower"].hist()


# # There are few data points which have horsepower as -10000

# In[29]:


data[data["horsepower"]<=0]


# In[30]:


data[data["car_name"]=="ford pinto"]


# In[31]:


data[data["car_name"]=="ford maverick"]


# In[32]:


data[data["car_name"]=="renault 18i"]


# # We noticed that car_name as renault 18i has only one data point

# In[33]:


data[data["brand"]=="Renault"]


# # We can fill the median values of each car horsepower for outliers

# # If there are any single outlier for a particular car, Let's go with the similar Brand's median value

# In[34]:


cars=data[data["horsepower"]<=0]["car_name"].unique()
cars


# In[35]:


for car in cars:
    med=0
    brand=data[data["car_name"]==cars[3]]["brand"].values
    med=data.loc[(data["car_name"]==car) & (data["horsepower"]>0),"horsepower"].median()

    data.loc[(data["car_name"]==car) & (data["horsepower"]<=0),"horsepower"]=np.nan
    data.fillna(med,inplace=True)
    med_brand=data.loc[(data["brand"]==brand[0]) & (data["horsepower"]>0),"horsepower"].median()
    data.fillna(med_brand,inplace=True)


# In[36]:


data[data["car_name"]=="ford maverick"]


# In[37]:


data[data["horsepower"]<=0]


# # or if the car_name has only single row then it has been filled with the their Brand's me

# In[38]:


data[data["car_name"]=="renault 18i"]


# In[39]:


data[data["brand"]=="Renault"]


# # Visualizing the distribution of horsepower

# In[40]:


data["horsepower"].hist(bins=20)


# # Number of brands from each origin

# In[41]:


plt.figure(figsize=(20,8))
brands_USA=data[data["origin"]=="USA"]["brand"]
```

```
brands_Europe=data[data["origin"]=="Europe"]["brand"]
brands_Japan=data[data["origin"]=="Japan"]["brand"]
brand_origin=pd.DataFrame([["USA",brands_USA.nunique()],["Europe",brands_Europe.nunique()],["Japan",brands_Japan.nunique()]],
                          columns=["Origin Country","Total no. of brands"])
brand_origin


# # Number of unique car_names

# In[42]:


plt.figure(figsize=(12,8))
data["car_name"].value_counts().hist()


# # Most of the car_names are unique. So there is no useful information in that column. Let's drop that column

# In[43]:


data.drop("car_name",axis=1,inplace=True)


# # Correlation of data

# In[44]:


#numeric_data = data.select_dtypes(include=[np.number])
#correlation_matrix = numeric_data.corr()


# In[45]:


#data = data.apply(pd.to_numeric, errors='coerce')
#correlation_matrix = data.corr()


# In[46]:


#data.corr()


# # 4. Data visualization

# # Setting palette

# In[47]:


sns.set_palette("bright")


# In[48]:


data.head()


# # 4.1 Plots for categorical features

# # 1. Number of cars belong to each Origin(country)

# In[49]:


import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
plt.title("Number of cars according to Origin", fontsize=30)
plt.xticks(fontsize=20)
sns.countplot(x=data["origin"])
plt.show()

USA has the most number of cars

Japan and Europe have almost same number of cars
# # 2. Number of cars belong to Total number of Cylinders present

# In[50]:


import seaborn as sns
import matplotlib.pyplot as plt

# Creating a single plot
plt.figure(figsize=(12, 8))
plt.title("Number of cars according to Total no. of Cylinders present", fontsize=25)
plt.xticks(fontsize=20)
sns.countplot(x=data["cylinders"])
plt.show()  # This line ensures the plot is displayed


# Cars with 4 cylinders have the most number of cars
#
# Cars with 3 and 5 cyliners have the least number of cars

# # 3. Number of cars belong to Total number of Cylinders present in each Origin(country)
#

# In[51]:
```

```
plt.figure(figsize=(20,8))
plt.title("Total number of cars according to Number of Cylinders in each Origin",fontsize=25)
plt.xticks(fontsize=20)
sns.countplot(x="cylinders",data=data,hue="origin")

Only USA cars have 8 cylinders

Similarly Ony Japan cars have 3 cylinders and Europe cars have 5 cylinders
# Most common number of cylinders is 4

# # 4. Number of cars belong to each Model year

# In[52]:


plt.figure(figsize=(20,8))
plt.title("Total number of cars according to Model year",fontsize=25)
plt.xticks(fontsize=20)

sns.countplot(x=data["model_year"])

Cars of model_year 73 has the highest number of cars

Other model_years are almost distributed similarly
# In[53]:


import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
plt.title("Total number of cars according to Brand", fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right', fontweight='light', fontsize='x-large')
sns.countplot(x=data["brand"])  # Specify 'x' to indicate that 'brand' is the categorical variable
plt.show()


#  Ford has the most number of cars, followed by Chevrolet
#
# Hi and Triumph have the least number of cars

# # 4. Number of brands from each origin

# In[54]:


plt.figure(figsize=(20,8))
plt.title("Total number of brands from each Origin",fontsize=25)
plt.xticks(fontsize=20)

sns.barplot(data=brand_origin ,x="Origin Country",y="Total no. of brands")


# USA and Europe have almost equal number of brands
#
# Japan has ony 5 brands

# # 5. Number of cars according to brand in Top 20 cars with highest mpg

# In[55]:


import seaborn as sns
import matplotlib.pyplot as plt

# Sort by 'mpg' and reset the index
sorted_data = data.sort_values(by="mpg", ascending=False).reset_index(drop=True)

# Create the plot using the top 20 rows
plt.figure(figsize=(12, 8))
plt.title("Number of cars according to brand in Top 20 cars with highest mpg", fontsize=25)

# Adjusting the x-axis labels
plt.xticks(rotation=45, ha='right', fontsize=20)

# Adjust the plot to ensure the x-axis labels are fully visible
plt.subplots_adjust(bottom=0.3)

# Create the count plot
sns.countplot(x=sorted_data["brand"][:20])

# Display the plot
plt.show()


#
# Volkswagen(VW) has 5 cars in top 20 highest mpg followed by Datsun with 4 cars

# # 6.Average mpg values of cars in each cylinders from each origin

# In[56]:


plt.figure(figsize=(12,8))
plt.title("MPG values according to Number of Cylinders",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(x="cylinders",y="mpg",data=data,hue="origin")


# Cars with 8 cylinders has the least Average mpg value
#
# Overall USA has low Average mpg value

# # 6.Average mpg values of cars in each Model year from each origin
```

```python
# In[57]:


plt.figure(figsize=(20,8))
plt.title("MPG values according to Model year",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(y="mpg",x="model_year",data=data)


# Cars of model year 80 has the highest average MPG value
#
#

# In[58]:


plt.figure(figsize=(20,8))
plt.title("MPG values according to Model year from each Origin",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(x="model_year",y="mpg",data=data,hue="origin")


# As we can see Average mpg values have been clearly improved as years passed by
#
# Japan cars have the most number of highest averages of each year
#
# Europe cars have improved average mpg value much better in 82

# # 7. MPG values of each brand

# In[59]:


plt.figure(figsize=(20,8))
plt.title("Average mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large')


sns.barplot(x="brand",y="mpg",data=data)


# # Maximum mpg value of each Brand

# In[60]:


plt.figure(figsize=(20,8))
plt.title("Maximum mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large')


sns.barplot(x="brand",y="mpg",data=data,estimator=max)


# As we can see Mazda has Highest mpg value and Hi has the lowest mgp value
# Renault cars have the highest average mpg value

# # 8. Number of cars according to brand in Top 20 cars with highest Acceleration

# In[61]:


import seaborn as sns
import matplotlib.pyplot as plt

# Sort by 'acceleration' and reset the index
sorted_data = data.sort_values(by="acceleration", ascending=False).reset_index(drop=True)

# Create the plot using the top 20 rows
plt.figure(figsize=(12, 8))
plt.title("Number of cars according to brand in Top 10 cars with highest Acceleration", fontsize=25)

# Adjusting the x-axis labels
plt.xticks(rotation=45, ha='right', fontsize=25)

# Adjust the plot to ensure the x-axis labels are fully visible
plt.subplots_adjust(bottom=0.3)

# Create the count plot
sns.countplot(x=sorted_data["brand"].iloc[:20])

# Display the plot
plt.show()


#  Volkswagen has the most number of cars in Top 20 Cars with Highest Acceleration

# # 8. Average of Mpg values according to origin

# In[62]:


plt.figure(figsize=(20,8))
plt.title("Average MPG values according to Origin",fontsize=25)
plt.xticks(fontsize=20)
sns.barplot(y="mpg",x="origin",data=data)


#
# Japan has highest average mpg value and USA has leas

# # 4.2 Plots for Numerical features
1. Distribution plots

Distribution of mpg values
# In[63]:
```

```
sns.set_style("whitegrid")

plt.figure(figsize=(12,8))
plt.title("Distribution of MPG",fontsize=25)

sns.distplot(data["mpg"])


# # Distribution of weight of cars

# In[64]:


plt.figure(figsize=(12,8))
plt.title("Distribution of weight",fontsize=25)

sns.distplot(data["weight"])


# # Distribution of Acceleration of cars

# In[65]:


plt.figure(figsize=(12,8))
plt.title("Distribution of Acceleration",fontsize=25)

sns.distplot(data["acceleration"])


# # Distribution of Horsepower of cars

# In[66]:


plt.figure(figsize=(12,8))
plt.title("Distribution of HorsePower",fontsize=25)

sns.distplot(data["horsepower"])


# # Distribution of Displacement of cars

# In[67]:


plt.figure(figsize=(12,8))
plt.title("Distribution of Displacement",fontsize=25)

sns.distplot(data["displacement"])


# # 2. Joint plots

#
# Mpg vs Displacement

# In[68]:


import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
sns.jointplot(x="displacement", y="mpg", data=data)
plt.show()


# # Mpg vs Horsepower

# # joint plot for horsepower and weight of car

# In[ ]:





# In[69]:


sns.jointplot(x="weight",y="mpg",data=data)


# In[70]:


sns.jointplot(x="horsepower",y="mpg",data=data)


# As we see MPG value decreases as we increase weight or displacement or horsepower of car
#
# Mpg value only increases slightly when we increase Acceleration of car

# # 3. Violin and box plots

# In[71]:


plt.figure(figsize=(12,8))
plt.title("MPG values according to Origin",fontsize=25)
plt.xticks(fontsize=20)

sns.violinplot(x="origin",y="mpg",data=data)
```

```python
# In[72]:


plt.figure(figsize=(12,8))
sns.boxplot(x="origin",y="mpg",data=data)


# # MPG vs Model Year

# In[73]:


plt.figure(figsize=(20,8))
plt.title("MPG values according to Model Year",fontsize=25)
plt.xticks(fontsize=20)

sns.violinplot(x="model_year",y="mpg",data=data)


# In[74]:


plt.figure(figsize=(20,8))
plt.title("MPG values according to Model Year",fontsize=25)
plt.xticks(fontsize=20)

sns.boxplot(x="model_year",y="mpg",data=data)


# # MPG vs Brand

# In[75]:


plt.figure(figsize=(25,8))
plt.title("Mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large')


sns.violinplot(x="brand",y="mpg",data=data)


# In[76]:


plt.figure(figsize=(22,8))
plt.title("Mpg values of each brand",fontsize=25)
plt.xticks(rotation=45, horizontalalignment='right',fontweight='light',fontsize='x-large')


sns.boxplot(x="brand",y="mpg",data=data)


# # MPG vs Cylinders

# In[77]:


plt.figure(figsize=(20,8))
plt.title("MPG values according to Number of Cylinders",fontsize=25)
plt.xticks(fontsize=20)

sns.violinplot(x="cylinders",y="mpg",data=data)


# In[78]:


plt.figure(figsize=(20,8))
plt.title("MPG values according to Number of Cylinders",fontsize=25)
plt.xticks(fontsize=20)

sns.boxplot(x="cylinders",y="mpg",data=data)


# # 4.3 Heatmaps

# Heat map of correlation of data

# In[79]:


import seaborn as sns
import matplotlib.pyplot as plt

# Option 1: Automatically filter out non-numeric columns
numeric_data = data.select_dtypes(include=[float, int])

# Option 2: Manually drop non-numeric columns
# numeric_data = data.drop(columns=["column_name"])

plt.figure(figsize=(12, 8))
sns.heatmap(numeric_data.corr(), annot=True, cmap="rainbow")
plt.show()


# # Cluster Map

# In[80]:


import seaborn as sns
import matplotlib.pyplot as plt
```

```python
# Select only numeric columns
numeric_data = data.select_dtypes(include=[float, int])

plt.figure(figsize=(12, 8))
sns.clustermap(numeric_data.corr(), cmap="viridis", annot=True)
plt.show()


# # 4.4 Pair plot

# In[81]:


sns.pairplot(data,hue="origin")


# # 4.5 Pie charts

# In[82]:


import matplotlib.pyplot as plt

plt.figure(figsize=(20, 20))
ax = data["brand"].value_counts()
labels = data["brand"].value_counts().index

plt.pie(ax, labels=labels, autopct='%.2f')
plt.title("Number of Cars According to Brand", fontsize=25, color='purple')

# Increase the legend font size
plt.legend(prop={'size': 25})

plt.show()


# # Ford has the most number of cars

# In[83]:


plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["brand"][:50].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["brand"][:50].value_counts().index
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg values according to Brand ",fontsize=25,color='purple')
plt.legend()
plt.show()


# In[84]:


plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["model_year"][:20].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["model_year"][:20].value_counts().index
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg according to Model year ",fontsize=25,color='purple')
plt.legend()
plt.show()


# # Cars of Model year 80 has the most number of cars in Top 50 cars with highest mpg values

# In[85]:


plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["origin"][:50].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["origin"][:50].value_counts().index

plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg values according to Origin ",fontsize=25,color='purple')
plt.legend()
plt.show()


# # Japan cars have the most number of cars in Top 50 cars with highest Mpg values

# In[86]:


plt.figure(figsize=(20,20))
ax =data.sort_values(by="mpg",ascending=False)["cylinders"][:50].value_counts()
labels=data.sort_values(by="mpg",ascending=False)["cylinders"][:50].value_counts().index

plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 50 mpg according to Number of Cylinders ",fontsize=25,color='purple')
plt.legend()
plt.show()


# # Cars with 4 cylinders have the most number of cars(48 cars out of 50) in Top 50 cars with highest Mpg values

# In[87]:


plt.figure(figsize=(20,20))
ax =data.sort_values(by="horsepower",ascending=False)["brand"][:30].value_counts()
labels=data.sort_values(by="horsepower",ascending=False)["brand"][:30].value_counts().index
plt.pie(ax,labels=labels,autopct='%.2f')
plt.title("Number of cars in Top 30 Horse Powers according to Brand ",fontsize=25,color='purple')
plt.legend()
plt.show()
```

```python
# # Pontiac cars have the most number of cars in Top 30 cars with Highest Horsepower

# # 5. Data Preprocessing

# data.head()

# # 5.1 X and Y values

# mpg column is target variable

# # Dependent variable

# In[88]:


y=data.iloc[:,7].values


# # Independent variable

# In[89]:


x=data.drop("mpg",axis=1).values


# In[90]:


x[:10]


# In[91]:


y[:10]


# In[92]:


x.shape


# # 5.2 Encoding Categorical Data

# OneHotEncoding

# In[93]:


from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encode',OneHotEncoder(),[0,6])],
                    remainder="passthrough")
x_s=np.array(ct.fit_transform(x))
x_s[29]


# In[94]:


x_s[:,8:]


# # LabelEncoding

# In[95]:


from sklearn.preprocessing import LabelEncoder
le_brand=LabelEncoder()
le_year=LabelEncoder()
x_s[:,13]=le_brand.fit_transform(x_s[:,13].astype(str))
x_s[:,12]=le_year.fit_transform(x_s[:,12])
x_s[:2]


# In[96]:


newdata=pd.DataFrame(x_s,columns=["3","4","5","6","8","Europe","Japan","USA","displacement","horsepower","weight","acceleration","model_year","brand"])
newdata.head()


#
# The first eight columns have binary values
#
# The last two columns have labelled values

# # 5.3 Train_test_split

# In[97]:


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_s,y,test_size=0.3,random_state=101)


# In[98]:


x_train


# In[99]:
```

```
x_test.shape


# # 5.4 Feature Scaling

# In[100]:


x_train[:2]


# # numerical columns starts from 8 and ends at 11

# column 8 is displacement
#
# column 9 is horsepower
#
# column 10 is weight
#
# column 11 is acceleration

# In[101]:


x_train_scaled=x_train
x_test_scaled=x_test


# In[102]:


from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_scaled[:,8:12]=sc.fit_transform(x_train_scaled[:,8:12])
x_test_scaled[:,8:12]=sc.transform(x_test_scaled[:,8:12])
x_train_scaled


# In[103]:


x_train


# # 6. Machine Learning Models

# # 6.1 Linear Regression
#

# Training the model

# In[104]:


from sklearn.linear_model import LinearRegression
Linreg=LinearRegression()
Linreg_fs=LinearRegression()
Linreg.fit(x_train,y_train)


# Model Prediction

# In[105]:


P_linreg=Linreg.predict(x_test)


# Evaluting the model

# In[106]:


from sklearn.metrics import mean_squared_error,r2_score
mse_linreg=mean_squared_error(y_test,P_linreg)

print(np.sqrt(mse_linreg))


# In[107]:


lin_score=r2_score(y_test,P_linreg)*100
print(lin_score)


# # 6.2 Support Vector Regression

# Scaling y values

# In[108]:


sc_y=StandardScaler()
y_train_svm=y_train.reshape(len(y_train),1)
ys_train=sc_y.fit_transform(y_train_svm)
y_test_svm=y_test.reshape(len(y_test),1)
ys_test=sc_y.transform(y_test_svm)


# # Training the model

# In[109]:
```

```
from sklearn.svm import SVR
svr=SVR(kernel='rbf')
svr.fit(x_train_scaled,ys_train)


# # Model Prediction

# In[110]:


import numpy as np
from sklearn.preprocessing import StandardScaler

# Assuming sc_y is your StandardScaler for the target variable
# and Ps_svr is your predicted values from SVR

# Reshape Ps_svr to be a 2D array with one column
Ps_svr_reshaped = Ps_svr.reshape(-1, 1)

# Apply inverse transform
P_svr = sc_y.inverse_transform(Ps_svr_reshaped)

# If you want to flatten it back to 1D
P_svr = P_svr.flatten()


# # Evaluting the model

# In[ ]:


mse_svr=mean_squared_error(y_test,P_svr)
print(np.sqrt(mse_svr))


# In[ ]:


svr_score=r2_score(y_test,P_svr)*100
print(svr_score)


# # Train test split once again

# In[ ]:


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_s,y,test_size=0.3,random_state=101)


# # 6.3 Random Forest Regression

# Training the model

# In[ ]:


from sklearn.ensemble import RandomForestRegressor
randomforest=RandomForestRegressor(n_estimators=100,random_state=101)
randomforest.fit(x_train,y_train)


# # Model Prediction

# In[ ]:


P_forest=randomforest.predict(x_test)


# # Evaluting the model

# In[ ]:


forest_score=r2_score(y_test,P_forest)*100
print(forest_score)


# # 6.4 Lasso

#
# Training the model

# In[ ]:


from sklearn.linear_model import Lasso
lass=Lasso()
lass.fit(x_train,y_train)


# # Model Prediction

# In[ ]:


P_lasso=lass.predict(x_test)


# # Evaluting the model

# In[ ]:
```

```python
lasso_score=r2_score(y_test,P_lasso)*100
print(lasso_score)


# # 6.5 Ridge Regression

#
# Training the model

# In[ ]:


from sklearn.linear_model import Ridge
ridge=Ridge()
ridge.fit(x_train,y_train)


# # Model Prediction

# In[ ]:


P_ridge=ridge.predict(x_test)


# # Evaluting the model

# In[ ]:


ridge_score=r2_score(y_test,P_ridge)*100
print(ridge_score)


# # 6.6 Elastic net Regression

# Training the model

# In[ ]:


from sklearn.linear_model import ElasticNet
elastic=ElasticNet()
elastic.fit(x_train,y_train)


# # Model Prediction

# In[ ]:


P_elastic=elastic.predict(x_test)


# # Evaluting the model

# In[ ]:


elastic_score=r2_score(y_test,P_elastic)*100
print(elastic_score)


# # 7. Model Selection

# Score comparison

# In[ ]:


Score=pd.DataFrame({"Model_name":["Linear Regression","Support Vector Regression", "Random Forest Regression","Lasso Regression",
                                 "Ridge Regression","Elastic Net Regression"],
                   "Accuracy_score":[lin_score,svr_score,forest_score,lasso_score,ridge_score,elastic_score]})


# In[ ]:


Score


# As you can see from above data "Random Forest Regressor" Given the highest Accuracy score

# # Feature Importance

# In[ ]:


Feature_importance=pd.DataFrame(randomforest.feature_importances_,index=["3","4","5","6","8","Europe","Japan","USA","displacement","horsepower","weight","accel
Feature_importance[8:]


#
# We found that Fuel consumption of a car is mostly affected by Displacement and Weight of the car

# In[ ]:


sample=pd.DataFrame({"Actual mpg":y_test,
             "Predicted mpg":np.round(P_forest,2)})


# In[ ]:
```

```python
sample

# AS you can see Our model is working good its predicted Mpg is Nearly to the Actual Mpg

#
```