Importing the Dependencies

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn import metrics

Data Collection and Processing

# loading the data from csv file to pandas dataframe

car\_dataset = pd.read\_csv('/content/car data.csv')

# inspecting the first 5 rows of the dataframe

car\_dataset.head()

# checking the number of rows and columns

car\_dataset.shape

# getting some information about the dataset

car\_dataset.info()

# checking the number of missing values

car\_dataset.isnull().sum()

# checking the distribution of categorical data

print(car\_dataset.Fuel\_Type.value\_counts())

print(car\_dataset.Seller\_Type.value\_counts())

print(car\_dataset.Transmission.value\_counts())

Encoding the Categorical Data

# encoding "Fuel\_Type" Column

car\_dataset.replace({'Fuel\_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)

# encoding "Seller\_Type" Column

car\_dataset.replace({'Seller\_Type':{'Dealer':0,'Individual':1}},inplace=True)

# encoding "Transmission" Column

car\_dataset.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True)

car\_dataset.head()

Splitting the data and Target

X = car\_dataset.drop(['Car\_Name','Selling\_Price'],axis=1)

Y = car\_dataset['Selling\_Price']

print(X)

print(Y)

Splitting Training and Test data

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.1, random\_state=2)

Model Training

1. Linear Regression

# loading the linear regression model

lin\_reg\_model = LinearRegression()

lin\_reg\_model.fit(X\_train,Y\_train)

Model Evaluation

# prediction on Training data

training\_data\_prediction = lin\_reg\_model.predict(X\_train)

# R squared Error

error\_score = metrics.r2\_score(Y\_train, training\_data\_prediction)

print("R squared Error : ", error\_score)

Visualize the actual prices and Predicted prices

plt.scatter(Y\_train, training\_data\_prediction)

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title(" Actual Prices vs Predicted Prices")

plt.show()

# prediction on Training data

test\_data\_prediction = lin\_reg\_model.predict(X\_test)

# R squared Error

error\_score = metrics.r2\_score(Y\_test, test\_data\_prediction)

print("R squared Error : ", error\_score)

plt.scatter(Y\_test, test\_data\_prediction)

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title(" Actual Prices vs Predicted Prices")

plt.show()

1. Lasso Regression

# loading the linear regression model

lass\_reg\_model = Lasso()

lass\_reg\_model.fit(X\_train,Y\_train)

Model Evaluation

# prediction on Training data

training\_data\_prediction = lass\_reg\_model.predict(X\_train)

# R squared Error

error\_score = metrics.r2\_score(Y\_train, training\_data\_prediction)

print("R squared Error : ", error\_score)

Visualize the actual prices and Predicted prices

plt.scatter(Y\_train, training\_data\_prediction)

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title(" Actual Prices vs Predicted Prices")

plt.show()

# prediction on Training data

test\_data\_prediction = lass\_reg\_model.predict(X\_test)

# R squared Error

error\_score = metrics.r2\_score(Y\_test, test\_data\_prediction)

print("R squared Error : ", error\_score)

plt.scatter(Y\_test, test\_data\_prediction)

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

plt.title(" Actual Prices vs Predicted Prices")

plt.show()