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 t

he dataframe

car\_dataset.head()

Car\_NameYearSelling\_PricePresent\_PriceKms\_DrivenFuel\_TypeSeller\_TypeTransmissionOwner0ritz20143.355.5927000PetrolDealerManual01sx420134.759.5443000DieselDealerManual02ciaz20177.259.856900PetrolDealerManual03wagon r20112.854.155200PetrolDealerManual04swift20144.606.8742450DieselDealerManual0

Car\_NameYearSelling\_PricePresent\_PriceKms\_DrivenFuel\_TypeSeller\_TypeTransmissionOwner0ritz20143.355.5927000PetrolDealerManual01sx420134.759.5443000DieselDealerManual02ciaz20177.259.856900PetrolDealerManual03wagon r20112.854.155200PetrolDealerManual04swift20144.606.8742450DieselDealerManual0

# checking the number of rows and columns

car\_dataset.shape

(301, 9)

# getting some information about the dataset

car\_dataset.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 301 entries, 0 to 300

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Car\_Name 301 non-null object

1 Year 301 non-null int64

2 Selling\_Price 301 non-null float64

3 Present\_Price 301 non-null float64

4 Kms\_Driven 301 non-null int64

5 Fuel\_Type 301 non-null object

6 Seller\_Type 301 non-null object

7 Transmission 301 non-null object

8 Owner 301 non-null int64

dtypes: float64(2), int64(3), object(4)

memory usage: 21.3+ KB

# checking the number of missing values

car\_dataset.isnull().sum()

Car\_Name 0

Year 0

Selling\_Price 0

Present\_Price 0

Kms\_Driven 0

Fuel\_Type 0

Seller\_Type 0

Transmission 0

Owner 0

dtype: int64

# 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())

Petrol 239

Diesel 60

CNG 2

Name: Fuel\_Type, dtype: int64

Dealer 195

Individual 106

Name: Seller\_Type, dtype: int64

Manual 261

Automatic 40

Name: Transmission, dtype: int64

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()

Car\_NameYearSelling\_PricePresent\_PriceKms\_DrivenFuel\_TypeSeller\_TypeTransmissionOwner0ritz20143.355.592700000001sx420134.759.544300010002ciaz20177.259.85690000003wagon r20112.854.15520000004swift20144.606.87424501000

Splitting the data and Target

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

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

print(X)

Year Present\_Price Kms\_Driven ... Seller\_Type Transmission Owner

0 2014 5.59 27000 ... 0 0 0

1 2013 9.54 43000 ... 0 0 0

2 2017 9.85 6900 ... 0 0 0

3 2011 4.15 5200 ... 0 0 0

4 2014 6.87 42450 ... 0 0 0

.. ... ... ... ... ... ... ...

296 2016 11.60 33988 ... 0 0 0

297 2015 5.90 60000 ... 0 0 0

298 2009 11.00 87934 ... 0 0 0

299 2017 12.50 9000 ... 0 0 0

300 2016 5.90 5464 ... 0 0 0

[301 rows x 7 columns]

print(Y)

0 3.35

1 4.75

2 7.25

3 2.85

4 4.60

...

296 9.50

297 4.00

298 3.35

299 11.50

300 5.30

Name: Selling\_Price, Length: 301, dtype: float64

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)

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

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)

R squared Error : 0.8799451660493711

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)

R squared Error : 0.8365766715027051

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)

Lasso(alpha=1.0, copy\_X=True, fit\_intercept=True, max\_iter=1000,

normalize=False, positive=False, precompute=False, random\_state=None,

selection='cyclic', tol=0.0001, warm\_start=False)

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)

R squared Error : 0.8427856123435794

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)

R squared Error : 0.8709167941173195

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

plt.xlabel("Actual Price")

plt.ylabel("Predicted Price")

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

plt.show()