|  |  |
| --- | --- |
| **Name:** | Bodhisatya Ghosh |
| **Branch:** | CSE – Data Science |
| **Batch:** | B |
| **Course:** | Soft Computing |
| **Experiment no:** | 6 |

**Aim:** To calculate standard deviation, root mean square, mean absolute error etc for measuring the fitness of a model.

**Theory:** Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a “least squares” method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).

**Program:**

# %%

import numpy as np

import pandas as pd

import seaborn as sns

# %%

data = pd.read\_csv('car price.csv')

data.drop(columns=['Unnamed: 0'],inplace=True)

data.head()

# %%

data.info()

# %%

data.describe()

# %%

data.isna().sum()

# %%

x = data

y = data['Price']

categorical = pd.DataFrame()

# %%

from sklearn.preprocessing import LabelEncoder

# %%

label = LabelEncoder()

for i in x.columns:

    if(x[i].dtype == 'object'):

        x[i] = label.fit\_transform(x[i])

        categorical[i] = x[i]

categorical

# %%

x.head()

# %%

sns.heatmap(x.corr())

# %%

from scipy import stats

for i in categorical.columns:

    stats.pointbiserialr(categorical[i], y)

    print(i +' '+ str(stats.pointbiserialr(categorical[i], y)[0]))

# %%

x.drop(columns=['Company Name','Model Name','Location','Engine Type','Color'],inplace=True)

x.head()

# %%

sns.heatmap(x.corr())

# %%

x.drop(columns=['Price'],inplace = True)

# %%

x.head()

# %%

from sklearn.preprocessing import StandardScaler

scale = StandardScaler()

linearVar = [['Mileage','Engine Capacity']]

x[['Mileage','Engine Capacity']] = scale.fit\_transform(x[['Mileage','Engine Capacity']])

# %%

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(3)

x = poly.fit\_transform(x)

# %%

x

# %%

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

xtrain,xtest,ytrain,ytest = train\_test\_split(x,y,test\_size=0.4,random\_state=100)

# %%

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(xtrain,ytrain)

result = model.fit(xtrain,ytrain)

# %%

ypred = model.predict(xtest)

# %%

ax1 = sns.kdeplot(ytest, color = 'r', label = 'Actual Value')

sns.kdeplot(ypred, color = 'b', label = 'Fitted Value', ax = ax1)

# %%

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error,r2\_score

resid = ytest - ypred

std\_dev = np.std(resid)

rmse = np.sqrt(mean\_squared\_error(ytest,ypred))

mae = mean\_absolute\_error(ytest,ypred)

r2 = r2\_score(ytest,ypred)

# %%

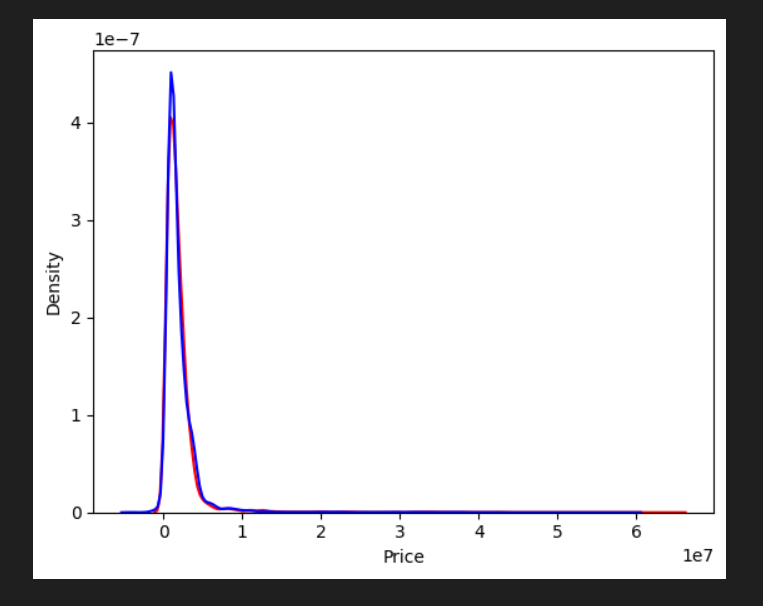
print('Standard Deviation: ' + str(std\_dev))

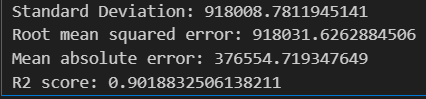
print('Root mean squared error: ' + str(rmse))

print('Mean absolute error: ' + str(mae))

print('R2 score: ' + str(r2))

**Results:**





**CONCLUSION: -** In this experiment we studied about linear regression and have measured its metrics