CAR PRICE PREDICTION

Data Set: **audi.csv**

Dependent variable: **price**

**Import Library**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

**Check Current Directory**

os.getcwd()

**Change the directory**

os.chdir ('C:\\Noble\\Training\\Top Mentor\\Training\\Presentation\\Project\\Acmegrade\\PRJ Car Price Prediction\\')

os.getcwd()

**Read Data, display records**

df=pd.read\_csv("audi.csv")

display(df)

Automated Exploratory Data Analysis (EDA)

**Pandas Profiling Report**

import pandas\_profiling as pf

display(pf.ProfileReport(df))

Manual EDA

**Number of records**

len(df)

**Number of records- Shape**

display (df.shape)

**Checking the data types**

display (df.dtypes )

**Checking null values**

display (df.isna().sum() )

**Data set details – Info**

df.info()

**Data set details – Describe**

df.describe ()

**Create X**

X = df.iloc[:,[0,1,3,4,5,6,7,8]].values

display (X)

**Create Y**

Y = df.iloc[:,[2]].values

display (Y)

**Display Top 5 - X variable**

display(pd.DataFrame(X).head(5))

**Label Encoding**

from sklearn.preprocessing import LabelEncoder

le1 = LabelEncoder()

X[:,0] = le1.fit\_transform(X[:,0])

le2 = LabelEncoder()

X[:,-4] = le2.fit\_transform(X[:,-4])

display (X)

**One hot Encoding to column – transmission**

from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

ct = ColumnTransformer(transformers = [('encoder',OneHotEncoder(),[2])],remainder='passthrough')

X = ct.fit\_transform(X)

display (pd.DataFrame(X))

**Display – X**

display (pd.DataFrame(X))

**Features Scaling – Standardization**

**f**rom sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X = sc.fit\_transform(X)

display (pd.DataFrame(X))

**Train Test Split**

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

(X\_train,X\_test,Y\_train,Y\_test) = train\_test\_split(X,Y,test\_size=0.2,random\_state=0)

**Create Random Forest Regressor**

from sklearn.ensemble import RandomForestRegressor

regression = RandomForestRegressor(random\_state=0)

regression.fit(X\_train,Y\_train)

**Prediction with Test Data**

y\_pred = regression.predict(X\_test)

display (y\_pred)

**Display actual and Predicted Values**

print(np.concatenate((y\_pred.reshape(len(y\_pred),1),Y\_test.reshape(len(Y\_test),1)),1))

**Display – Accuracy and Mean Absolute Error**

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

print ('R2 Score ', r2\_score(Y\_test, y\_pred))

print ('Mean Absolute Error', mean\_absolute\_error(Y\_test,y\_pred))

**Create a Linear Regression Model**

from sklearn.linear\_model import LinearRegression

reg = LinearRegression()

reg.fit(X\_train,Y\_train)

**Prediction with Test Data**

y\_pred = reg.predict(X\_test)

display (y\_pred)

**Display actual and Predicted Values**

print(np.concatenate((y\_pred.reshape(len(y\_pred),1),Y\_test.reshape(len(Y\_test),1)),1))

**Display – Accuracy and Mean Absolute Error**

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

print ('R2 Score ', r2\_score(Y\_test, y\_pred))

print ('Mean Absolute Error', mean\_absolute\_error(Y\_test,y\_pred))

**Prediction for complete data set**

y\_pred = reg.predict(X)

display (y\_pred)

**Display the Actual and predicted data**

result = pd.concat([df,pd.DataFrame(y\_pred)],axis=1)

display( result)