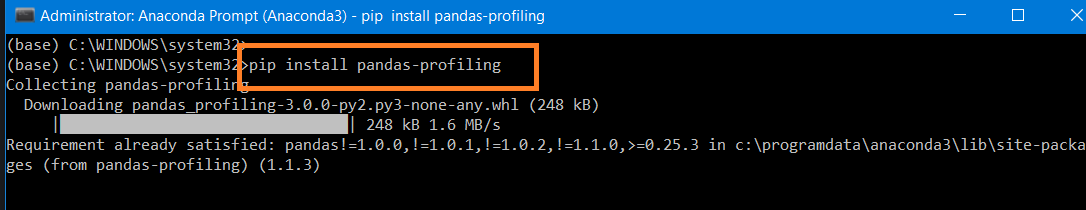
CAR PRICE PREDICTION

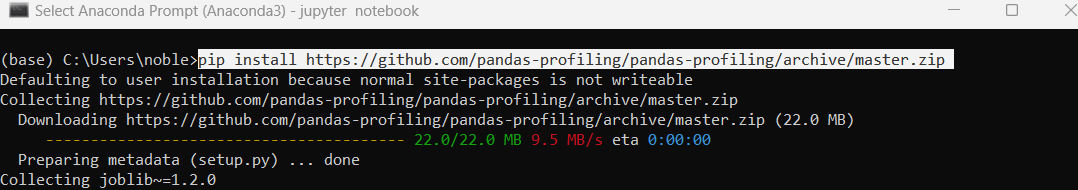
pip install pandas-profiling



If Profiling is not working and getting an error use the following statement to install pandas profiling

# TypeError: concat() got an unexpected keyword argument 'join\_axes' | Pandas Profiling

pip install <https://github.com/pandas-profiling/pandas-profiling/archive/master.zip>



Data Set: **audi.csv**

Dependent variable: **price**

**Import Library**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import warnings

warnings.filterwarnings('ignore')

**Check Current Directory**

print (os.getcwd())

**Change the directory**

os.chdir ('C:\\Noble\\Training\\Acmegrade\\Data Science\\Projects\\PRJ Car Price Prediction\\')

print (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**

print (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**

print (df.info())

**Data set details – Describe**

display (df.describe ())

**Create X**

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

display (X.shape)

display (X)

**Create Y**

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

display (Y.shape)

display (Y)

**Display Top 5 - X variable**

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

**Label Encoding Column – Model and Fuel Type**

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 (X.shape)

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)

print (X.shape, Y.shape)

print (X\_train.shape, Y\_train.shape)

print (X\_test.shape, Y\_test.shape)

**Create Random Forest Regressor**

from sklearn.ensemble import RandomForestRegressor

regression = RandomForestRegressor(random\_state=0)

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

display (regression)

**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)

print(reg)

**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)

**Create Model Extra Tree Regressor**

from sklearn.ensemble import ExtraTreesRegressor

ET\_Model=ExtraTreesRegressor(n\_estimators = 120)

ET\_Model.fit(X\_train,Y\_train)

y\_predict=ET\_Model.predict(X\_test)

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

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

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

**Display the Result**

y\_pred = reg.predict(X)

display (y\_pred)

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

display( result)

**RandomizedSearchCV**

# Hyperparameter Tuning and RandomizedSearchCV - Model used – RandomForestRegressor

from sklearn.model\_selection import RandomizedSearchCV

n\_estimators = [int(x) for x in np.linspace(start = 80, stop = 1500, num = 10)]

max\_features = ['auto', 'sqrt']

max\_depth = [int(x) for x in np.linspace(6, 45, num = 5)]

min\_samples\_split = [2, 5, 10, 15, 100]

min\_samples\_leaf = [1, 2, 5, 10]

# create random grid

rand\_grid={'n\_estimators': n\_estimators,

'max\_features': max\_features,

'max\_depth': max\_depth,

'min\_samples\_split': min\_samples\_split,

'min\_samples\_leaf': min\_samples\_leaf}

rf=RandomForestRegressor()

rCV=RandomizedSearchCV(estimator=rf,param\_distributions=rand\_grid,scoring='neg\_mean\_squared\_error',n\_iter=3,cv=3,random\_state=42, n\_jobs = 1)

**Fit Model**

display (rCV.fit(X\_train,Y\_train))

**Prediction**

rf\_pred=rCV.predict(X\_test)

display (rf\_pred)

**Mean\_absolute\_error and mean\_squared\_error**

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

print('MAE',mean\_absolute\_error(Y\_test,rf\_pred))

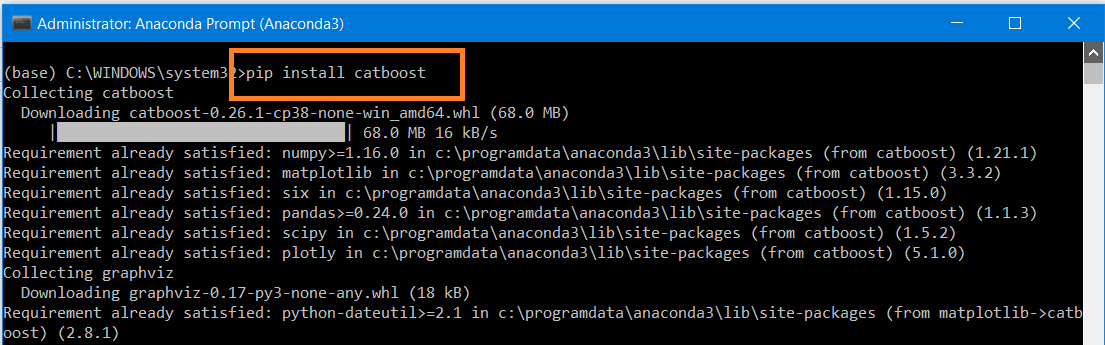
print('MSE',mean\_squared\_error(Y\_test,rf\_pred))

**Display Accuracy**

display (r2\_score(Y\_test,rf\_pred))

**Install Cat boost**

pip install catboost



**Model CatBoostRegressor**

from catboost import CatBoostRegressor

cat=CatBoostRegressor()

print (cat.fit(X\_train,Y\_train))

**Cat Boost Prediction**

cat\_pred=cat.predict(X\_test)

display (cat\_pred)

**Cat Boost Accuracy**

display (r2\_score(Y\_test,cat\_pred))

**Create Pickle File**

#Use pickle to save our model so that we can use it later

import pickle

# Saving model to disk

pickle.dump(cat, open('model.pkl','wb'))

**Load Pickle File and do Prediction**

model=pickle.load(open('model.pkl','rb'))

print (model.predict (X\_train))