**Steps performed end to end**

**Data had been in four different formats in four columns for five cities. So first step is to load** **the five city files to five dataframes with a new column “City” and merge them as a single dataframe.**

df5['City']=pd.Series(dtype='object')

df5["City"].fillna(value='Bangalore',inplace=True)

df=pd.concat([df1,df2,df3,df4,df5])

## **Seperating the columns to different fields using json normalize**

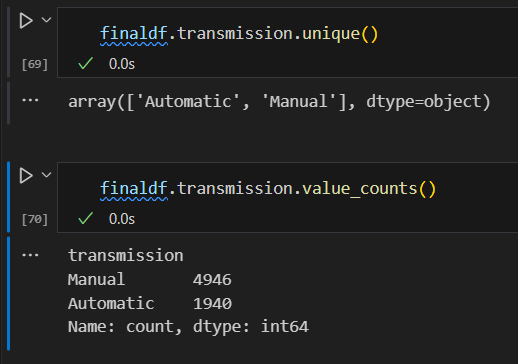
df["new\_car\_detail"]=df["new\_car\_detail"].apply(lambda i:json.loads(i.replace("'",'"').replace("None","1")))

newdf1=pd.json\_normalize(df["new\_car\_detail"]) # first column to 19 columns

## **Remove unwanted columns and add required columns from other df**

finaldf=pd.concat([finaldf,df["City"]],axis=1)

## **Exploring the values present in the columns**



## **Checking duplicates and deleting them**

finaldf.duplicated().sum()

finaldf=finaldf.drop\_duplicates()

## **Changing data types of number columns from object to int/float**

finaldf['km'] = finaldf['km'].str.replace(',', '').astype(int)

finaldf.ownerNo=finaldf.ownerNo.astype(int)

finaldf.Engine=finaldf.Engine.astype(int)

## **Renaming the columns meaningful**

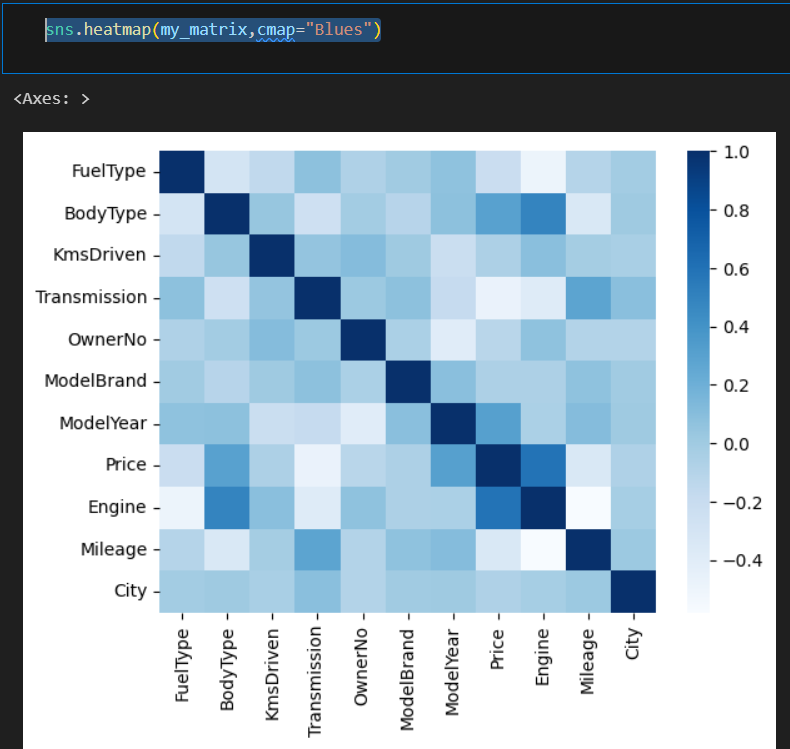
finaldf.rename(columns={'ft':'FuelType'},inplace=True)

## **Encoding the categorical column to numeric**

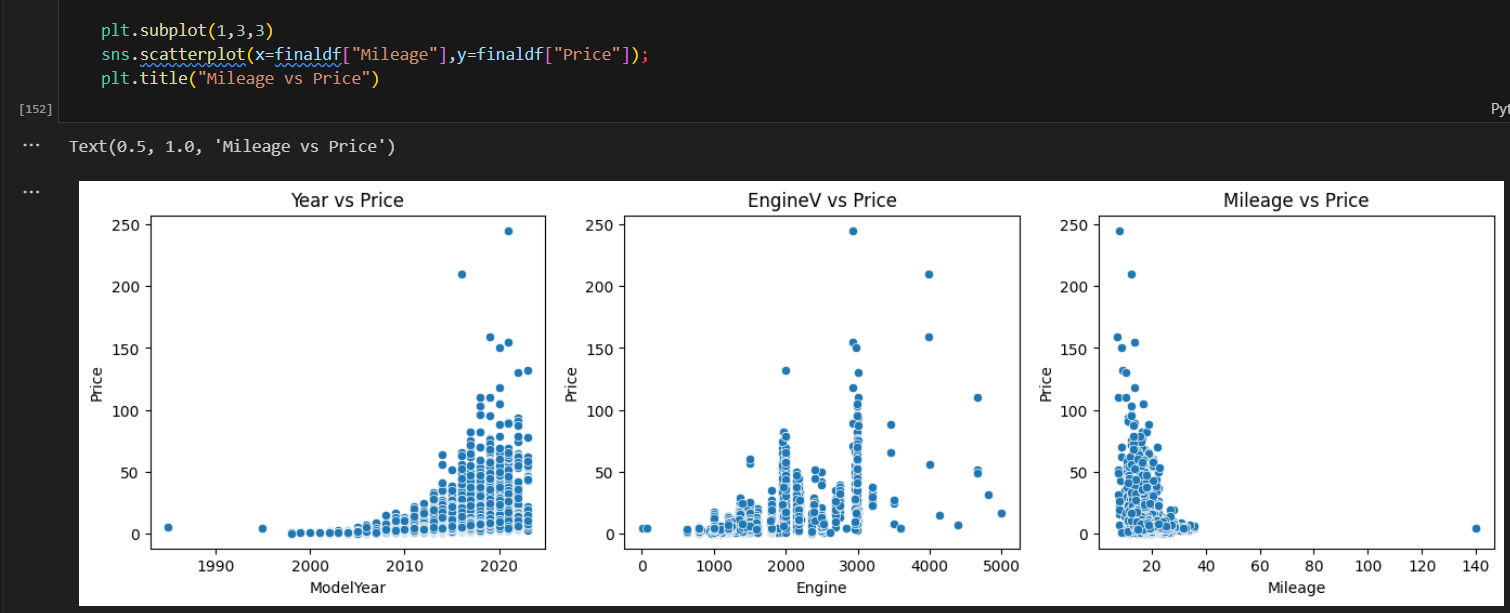
finaldf["City"]=le.fit\_transform(finaldf["City"])

## **Correlation**

my\_matrix=finaldf.corr().round(2)

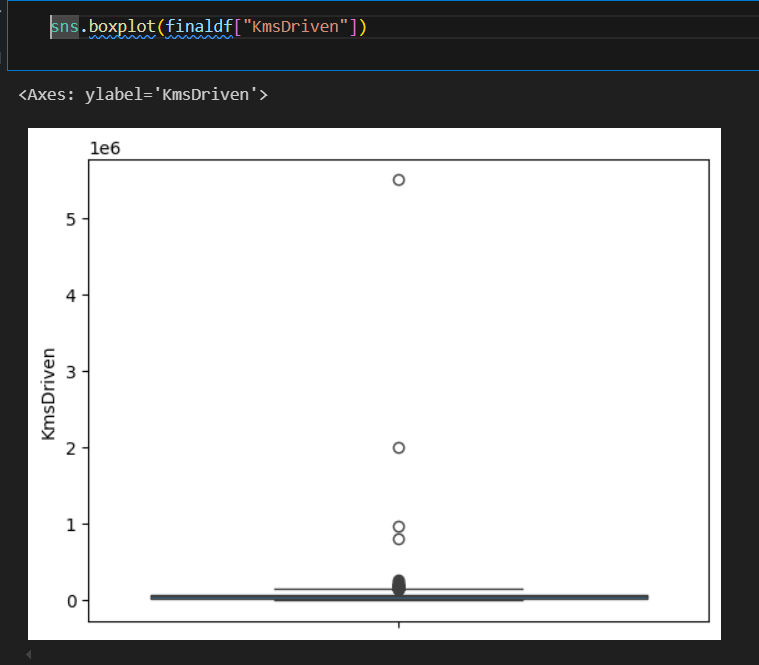


## **Exploratory data analysis**

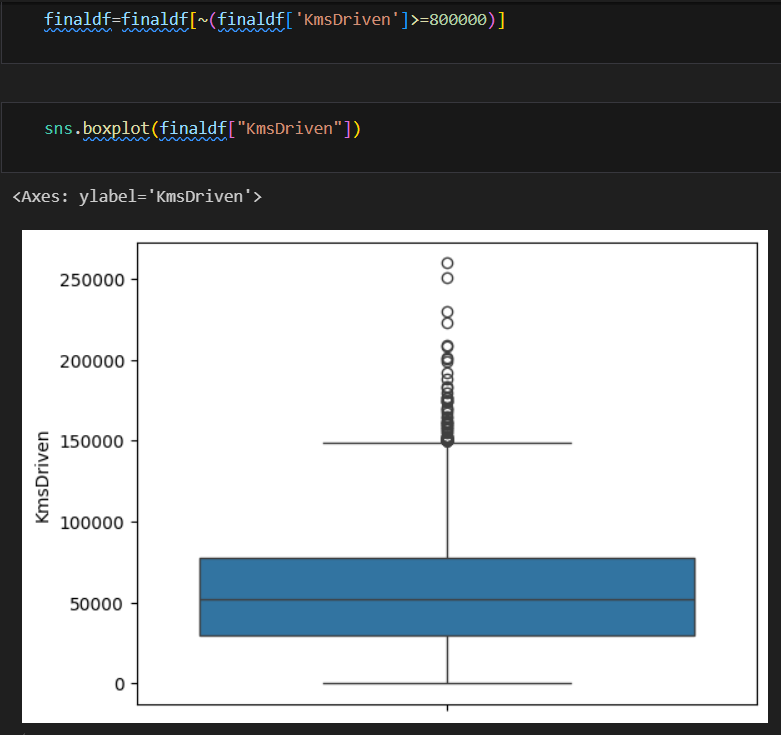


## **Outliers removal in all numerical columns**

Before outlier removal,



After outlier removal



## **Store Future matrix in X and Target in Y**

X=finaldf.drop(["Price"],axis=1)

Y=finaldf["Price"]

## **Splitting the data set in to training and testing set**

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.20,random\_state=42)

## **Import the models**

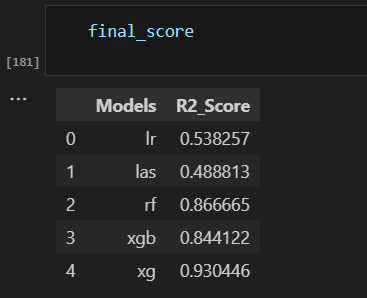
xg= XGBRegressor()

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

y\_pred5=xg.predict(X\_test)

score5=metrics.r2\_score(Y\_test,y\_pred5)

## **Evaluating the algorithms and comparing the R2 scores**



## **Min max scaling**

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler().fit(X\_train)# scalar object will understand the value in between min to max from xtrain

x\_train\_scaled=scaler.transform(X\_train)# implement and transfer the xtrain and save it to x\_train\_scaled based on xtrain

x\_test\_scaled=scaler.transform(X\_test)# implement and transfer the xtest and save it as x\_test\_scaled based on xtrain

x\_train\_scaled

## **Hyper parameter tuning**

## **Save the model**

joblib.dump(xg\_final,'car\_price\_predictor\_xg') #saving the model named car\_price\_predictor\_xg

xgmodel=joblib.load('car\_price\_predictor\_xg')

## **Prediction on new data**

newvalue=[[4,5,20000,1,1,20,2015,1196,15,1]]

result=xgmodel.predict(newvalue)

print(result)

## **Streamlit application development and deployment**