PROJECT:-

CAR PRICE PREDICTION PROJECT

AUTHOR:-

ATUL DAHIMA

PROBLEM STATEMENT:-

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model. This project contains two phase

Data Collection Phase

You have to scrape at least 5000 used cars data. You can scrape more data as well, it's up to you. more the data better the model

In this section You need to scrape the data of used cars from websites (Olx, cardekho, Cars24 etc.) You need web scraping for this. You have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometers, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data. Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback.

Note – The data which you are collecting is important to us. Kindly don't share it on any public platforms.

Model Building Phase

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like.

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best modEL

DATA SCRAPPING:-

For data scrapping firs we will import the required libraries and run the driver and there after the url.

DATA COLLECTION

```
In [1]: import pandas as pd
        import selenium
        from selenium import webdriver
        import time
        from selenium.common.exceptions import NoSuchElementException     #importing exception


        import warnings
        warnings.filterwarnings('ignore')

In [2]: driver=webdriver.Chrome('chromedriver.exe')

In [3]: #scrap cars details from cardekho
        driver.get("https://www.cardekho.com/used-cars+in+new-delhi")
```

After importing the required libraries and importing the url create some empty list to store the data and start scrapping the data

```
In [ ]: #price
        for i in driver.find_elements_by_xpath('//span[@class="amnt "]'):
            price.append(i.text)
        len(price)

In [ ]: #EMI
        for i in driver.find_elements_by_xpath('//div[@class="emitextCard"]'):
            EMI.append(i.text)
        len(EMI)

In [ ]: #other details
        detail=[]
        for i in driver.find_elements_by_xpath('//div[@class="truncate dotlist"]/span'):
            detail.append(i.text)

In [ ]: #fuel
        fuel=detail[1::3]
        len(fuel)

In [ ]: #distance_covered
        distance_covered=detail[::3]
        len(distance_covered)

In [ ]: for i in driver.find_elements_by_xpath('//div[@class="gsc_col-xs-7  carsName"]/a'):
            cars.append(i.text)
        len(cars)
        #swipe up manually to load more data as there is no option for next page

In [ ]: for i in driver.find_elements_by_xpath('//div[@class="gsc_col-xs-7  carsName"]/div'):
            model.append(i.text)

        #swipe up manually to load more data as there is no option for next page
```

In above pictures we have shown how we give the xpath to scrap the data. Above is the xpath of the some details of cars like:- name, price ,fuel type, distance covered, price etc.

```
In [ ]: data=pd.DataFrame()
        data["Cars"]=cars
        data["Model"]=model
        data["Fuel"]=fuel
        data["Distance"]=distance_covered
        data["EMI"]=EMI
        data["Price"]=price
        data
```

After scrapping all the required details about the cars make the dataframe of the scrapped data as shown above and start model building over it.

MODEL BUILDING FOR SCRAPPED DATA:-

IMPORTING LIBRARIES AND LOAD DATA:-

model building

In [ ]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

Few libraries are used initially for different purpose like:-

- *Numpy*:- for mathematical operation
- *Pandas*:- for data preprocessing
- *Matplotlib and Seaborn*:- for data visualization
- *Warnings*:- for some warnings

Now check for data shape and check whether null values are present or not.

CHECK FOR NULL VALUES AND DATA DESCRIPTION :-

```
In [16]: data.shape
Out[16]: (4181, 6)

In [17]: data.isnull().sum()
Out[17]: Cars        0
         Model       0
         Fuel        0
         Distance    0
         EMI         0
         Price       0
         dtype: int64
```

```
In [50]: data.describe()
```

Out[50]:

	Cars	Model	Fuel	Distance	EMI	Price
count	4181.000000	4181.000000	4181.00000	4181.000000	4181.000000	4181.000000
mean	422.511122	530.591007	2.54652	388.257833	331.274815	427.255920
std	203.663028	288.539187	1.52163	202.643352	325.583907	225.366374
min	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000
25%	259.000000	290.000000	1.00000	218.000000	0.000000	236.000000
50%	420.000000	553.000000	4.00000	392.000000	256.000000	451.000000
75%	589.000000	792.000000	4.00000	551.000000	617.000000	612.000000
max	787.000000	1014.000000	4.00000	740.000000	986.000000	809.000000

data.isnull().sum() shows that there is no null values in the data .

While data description shows some calculation about the data like:-mean,median,mode,std and some other calculation

On the top of the data description we can see that all the columns have same value which also defines that there is no null value

As most of the columns are non numerical so we will do label encoding first

LABEL ENCODING:-

label encoding

```
In [ ]: from sklearn.preprocessing import LabelEncoder

        data = data.copy()
        label_encoder = LabelEncoder()
        for column in data.columns:
            data[column] = label_encoder.fit_transform(data[column])
        data
```

Before label encoding:-

In [15]:
```
data
```

Out[15]:

	Cars	Model	Fuel	Distance	EMI	Price
0	2020 Skoda Rapid	1.0 TSI Style	Petrol	8,826 kms	EMI @ ₹ 22,759	11,24,000
1	2021 Datsun RediGO	1.0 T Option	Petrol	1,233 kms	EMI @ ₹ 9,436	4,66,000
2	2017 Hyundai Creta	1.6 VTVT AT SX Plus	Petrol	91,147 kms	EMI @ ₹ 31,253	10,38,000
3	2019 Maruti Ignis	1.2 AMT Delta BSIV	Petrol	33,065 kms	EMI @ ₹ 11,947	5,90,000
4	2019 Honda Amaze	S	Petrol	14,689 kms	EMI @ ₹ 12,757	6,30,000
...	...	...	...	...	...	...
4176	2012 Mercedes-Benz C-Class	220 CDI AT	Diesel	37,000 kms		8,90,000
4177	2017 Maruti Ciaz	1.4 Alpha	Petrol	80,000 kms		7,40,000
4178	2019 Maruti Swift	VXI	Petrol	30,000 kms		4,70,000
4179	2008 Maruti Alto	LXi	Petrol	90,000 kms		80,000
4180	2008 Hyundai i10	Asta	Petrol	70,000 kms		1,10,000

4181 rows × 6 columns

After lable encoding:-

In [18]:
```
from sklearn.preprocessing import LabelEncoder

data = data.copy()
label_encoder = LabelEncoder()
for column in data.columns:
    data[column] = label_encoder.fit_transform(data[column])
data
```

Out[18]:

	Cars	Model	Fuel	Distance	EMI	Price
0	729	14	4	658	459	78
1	742	13	4	40	943	497
2	468	189	4	719	602	58
3	647	22	4	268	169	601
4	615	704	4	112	187	639
...	...	...	...	...	...	...
4176	150	301	1	301	0	764
4177	487	88	4	663	0	704
4178	648	853	4	240	0	500
4179	24	628	4	714	0	770
4180	23	434	4	590	0	1

4181 rows × 6 columns

Now we can clearly see that after doing label encoding all the columns have changed into numerical column.

SEPERATING LABELS AND FEATURES:-

seperating labels and features

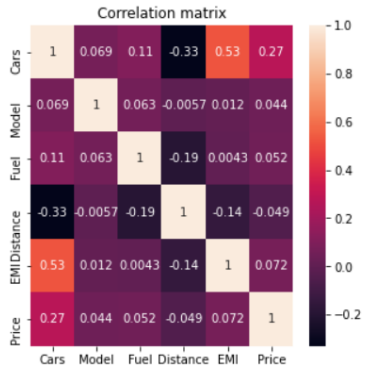
In [19]:
```
x=data.drop(["Price"],axis=1)
y=data["Price"]
```

Above we have separate the data set into label and features. As we have to predict the sales price of houses so sales price is the target vriable.

CORRELATION:-

```
In [68]: #plotting graph for correlation
         plt.figure(figsize=[5,5])

         #plot the correlation matrix
         sns.heatmap(data.corr(),annot=True)
         plt.title("Correlation matrix")
         plt.show()
```



Above diagram shows the correlation among all the columns. We can easily identify how the columns are correlated to each other using above heatmap.

DATA DISTRIBUTION:-

```
In [25]: #Let's check the data distribution among all the columns


         plt.figure(figsize=(20,25))
         plotnumber=1

         for column in data:
             if plotnumber<=9:
                 ax=plt.subplot(3,3,plotnumber)
                 sns.distplot(data[column])
                 plt.xlabel(column,fontsize=20)

             plotnumber+=1
         plt.show()
```

Above code shows the data distribution among all the columns , the data distribution is looking good , let go ahead with checking skewness in the data distribution.

SKEWNESS:-

checking skewness

In [26]:
```
x.skew().sort_values()
```

Out[26]:
```
Model      -0.160855
Fuel       -0.110294
Cars       -0.098900
Distance   -0.077934
EMI         0.516128
dtype: float64
```

data is all fine we are good to go

DATA SCALING:-

Scaled the features variables and then check the vif score for scaled data to check the multicollinearity using "variance_inflation_factor"

data scaling

In [30]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)
```

In [31]:
```
x_scaled
```

Out[31]:
```
array([[ 1.50506226, -1.79058104,  0.95532648,  1.33127698,  0.39234268],
       [ 1.56890083, -1.79404719,  0.95532648, -1.71878073,  1.87908045],
       [ 0.22338035, -1.18400509,  0.95532648,  1.63233446,  0.83160611],
       ...,
       [ 1.10729891,  1.11751738,  0.95532648, -0.73170703, -1.01760078],
       [-1.95695209,  0.33763402,  0.95532648,  1.60765762, -1.01760078],
       [-1.96186275, -0.33479875,  0.95532648,  0.99567193, -1.01760078]])
```

finding variance inflation factor in each scaled column i.e. x_scaled1

In [32]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif"]=[variance_inflation_factor(x_scaled, i) for i in range(x_scaled.shape[1])]

vif["Features"]=x.columns

#Let's check the values
vif
```

Out[32]:

	vif	Features
0	1.536770	Cars

Vif score is also good for all the columns so we are good to go

TRAINING PROCESS:-

Here we will use Linear Regression model for model training and testing

training process

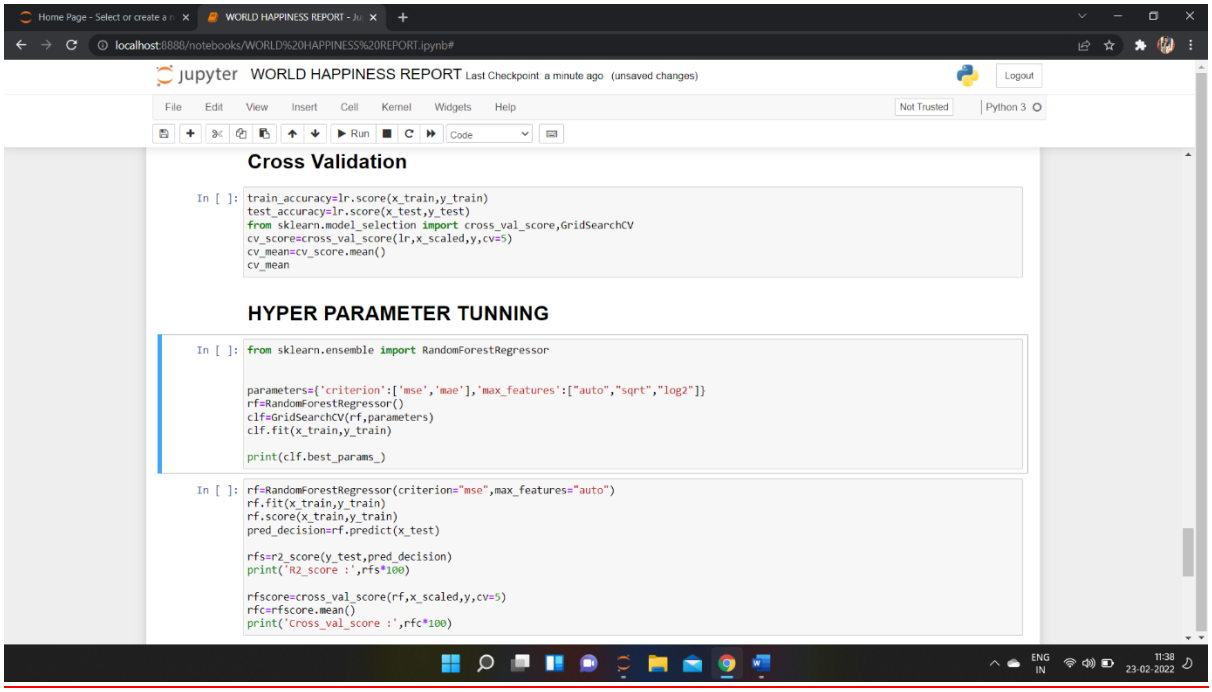
In [ ]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import r2_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
```

In [ ]:
```
#finding  best random state
maxAccu=0     #maximum accuracy
maxRS=0       #best random state value for which max accuracy is achieved

for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=i)
    lr.fit(x_train,y_train)
    pred=lr.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best Accuracy is :",maxAccu,"on Random_state",maxRS)
```

Start the training process and import some important module for model training and then split the data in training and testing data set and Check r2 score for training and testing data and then plot the graph for actual and predicted result with scatter plot

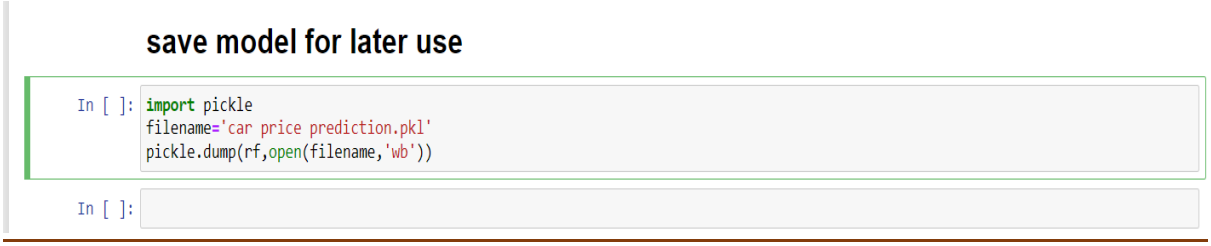
After this do cross validation and hyper parameter tunning for the model



Here we did the cross validation of the model and after do the hyper parameter tunning of the model.

We get the final accuracy of the model as

SAVE MODEL:-



Save the model for late use to prediction.

SUMMARY:-

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data pre-processing part.

Afterwards we started training model using Linear Regression and applied cross validation on it. Then we do the Hyper Parameter tunning of the model and get the final accuracy score for the model.

Later we save the model for later use for prediction.