FLIP ROBO

PROJECT:-

HOUSING PROJECT

AUTHOR:-

ATUL DAHIMA

PROBLEM STATEMENT:-

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company. A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below.

The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

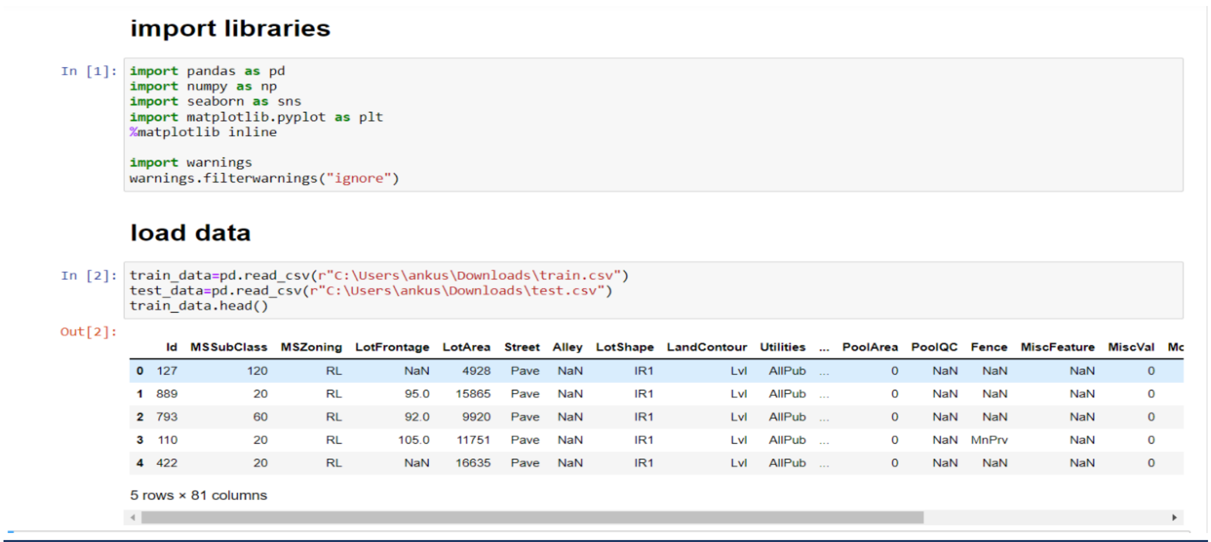
• Which variables are important to predict the price of variable?

• How do these variables describe the price of the house?

Business Goal:

You are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

IMPORTING LIBRARIES AND LOAD DATA:-



import libraries

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

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

load data

In [2]:
```python
train_data=pd.read_csv(r"C:\Users\ankus\Downloads\train.csv")
test_data=pd.read_csv(r"C:\Users\ankus\Downloads\test.csv")
train_data.head()
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mc
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows × 81 columns

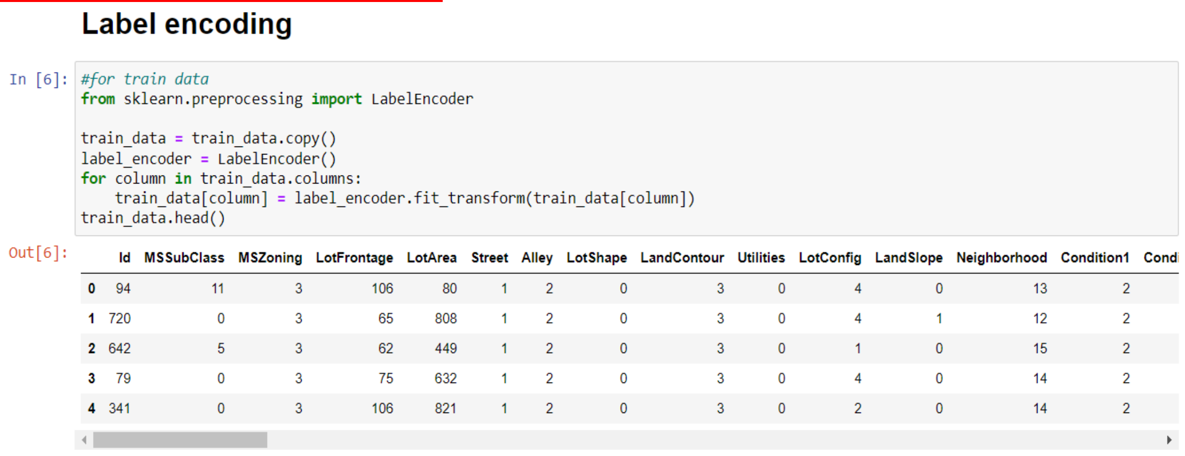
There are two data set one is train and other is test dataset . We have to make prediction on the test dataset using train dataset

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

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

LABEL ENCODING:-



Label encoding

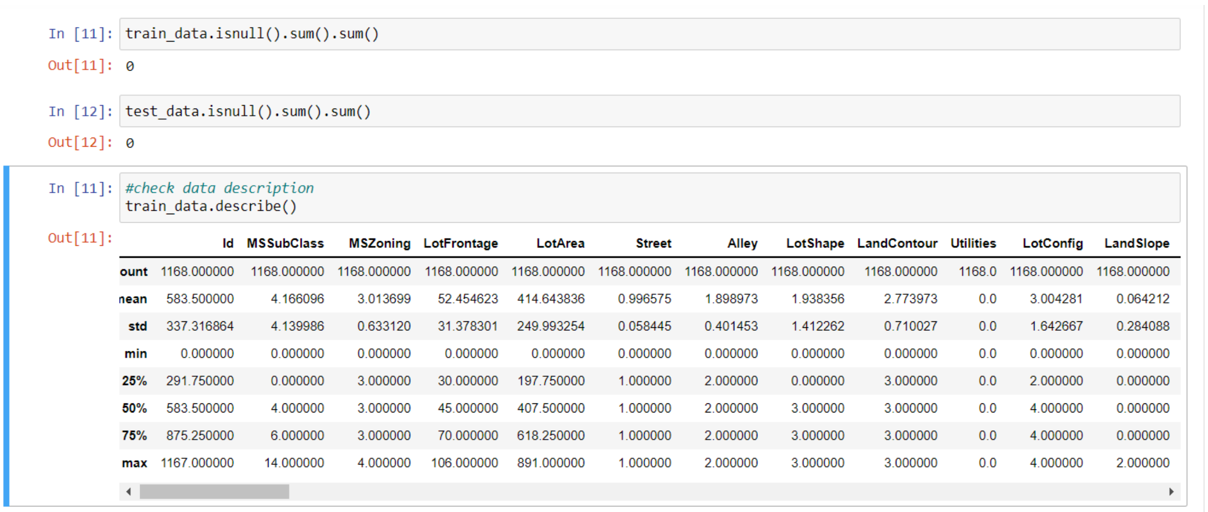
In [6]:
```python
#for train data
from sklearn.preprocessing import LabelEncoder

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

Out[6]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Cond
0	94	11	3	106	80	1	2	0	3	0	4	0	13	2	
1	720	0	3	65	808	1	2	0	3	0	4	1	12	2	
2	642	5	3	62	449	1	2	0	3	0	1	0	15	2	
3	79	0	3	75	632	1	2	0	3	0	4	0	14	2	
4	341	0	3	106	821	1	2	0	3	0	2	0	14	2	

Now we can clearly see that after doing label encoding all the columns have changed into numerical column. Same as we have to do with test data also.

CHECK FOR NULL VALUES AND DATA DESCRIPTION :-



```
In [11]: train_data.isnull().sum().sum()
Out[11]: 0

In [12]: test_data.isnull().sum().sum()
Out[12]: 0

In [11]: #check data description
         train_data.describe()
Out[11]:
```

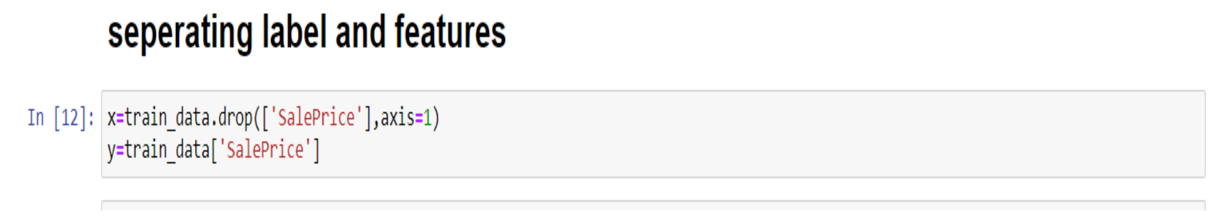
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope
ount	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1168.0	1168.000000	1168.000000
nean	583.500000	4.166096	3.013699	52.454623	414.643836	0.996575	1.898973	1.938356	2.773973	0.0	3.004281	0.064212
std	337.316864	4.139986	0.633120	31.378301	249.993254	0.058445	0.401453	1.412262	0.710027	0.0	1.642667	0.284088
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
25%	291.750000	0.000000	3.000000	30.000000	197.750000	1.000000	2.000000	0.000000	3.000000	0.0	2.000000	0.000000
50%	583.500000	4.000000	3.000000	45.000000	407.500000	1.000000	2.000000	3.000000	3.000000	0.0	4.000000	0.000000
75%	875.250000	6.000000	3.000000	70.000000	618.250000	1.000000	2.000000	3.000000	3.000000	0.0	4.000000	0.000000
max	1167.000000	14.000000	4.000000	106.000000	891.000000	1.000000	2.000000	3.000000	3.000000	0.0	4.000000	2.000000

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

SEPERATING LABELS AND FEATURES:-



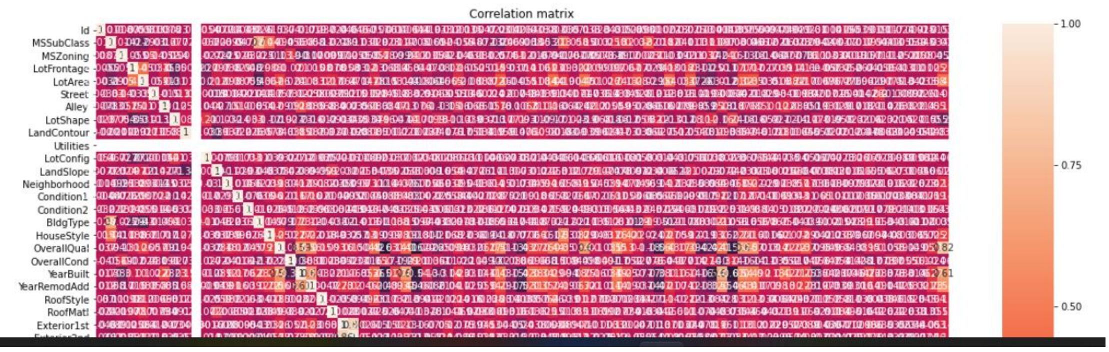
seperating label and features

```
In [12]: x=train_data.drop(['SalePrice'],axis=1)
         y=train_data['SalePrice']
```

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 [16]: #correlation using heatmap
         plt.figure(figsize=[20,20])

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



Above diagram shows the correlation among all the columns.

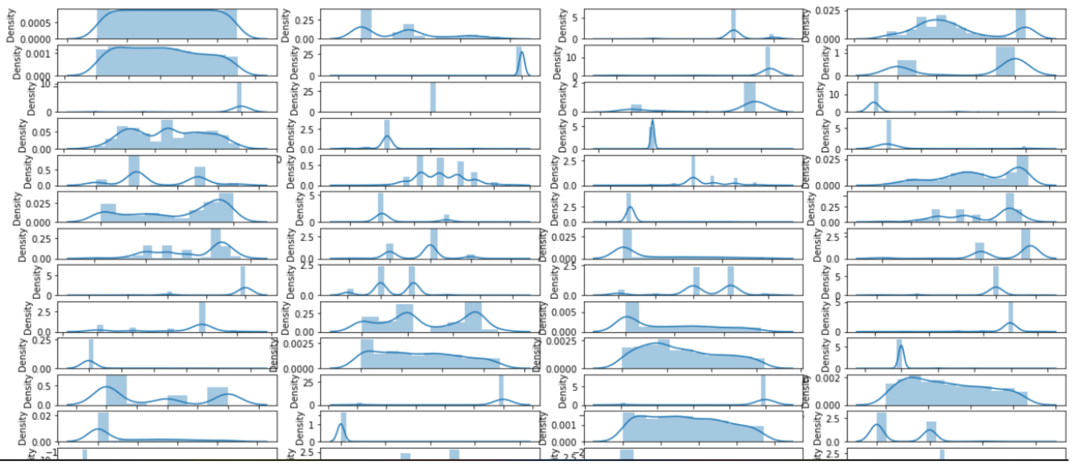
As here are so much column so we cannot easily identify the correlation among all the columns so lets print the correlation table.

```
In [15]: train_data.corr()

BsmtHalfBath  -0.042967   0.024663   0.006345   0.000975   0.026097   0.013788   0.005111  -0.017926   0.013725  NaN  -0.016040   0.074245
    FullBath  -0.015459   0.178100  -0.188837   0.103484   0.236095   0.033208   0.059465  -0.175359   0.047016  NaN   0.000177  -0.077542
    HalfBath  -0.028426   0.197825  -0.121209   0.066610   0.130289   0.045146   0.050163  -0.122586   0.030155  NaN  -0.020604  -0.000910
BedroomAbvGr   0.009781   0.031998  -0.000228   0.165149   0.277605   0.027690  -0.028218  -0.055556  -0.038239  NaN  -0.058060  -0.071538
KitchenAbvGr   0.001028   0.324594   0.026744  -0.029430  -0.028955   0.012304  -0.065582   0.082102  -0.067071  NaN  -0.002959  -0.033515
 KitchenQual  -0.000217  -0.018333   0.111689  -0.041490  -0.152446  -0.012056  -0.015901   0.122861   0.027230  NaN  -0.013554  -0.006002
TotRmsAbvGrd  -0.001837   0.116792  -0.030274   0.182701   0.397428   0.038317  -0.002743  -0.126689  -0.050646  NaN  -0.043083  -0.061747
  Functional   0.020059   0.040881  -0.091684   0.018003  -0.032568  -0.015309   0.018574  -0.012018   0.019517  NaN  -0.016756  -0.121209
  Fireplaces  -0.024571  -0.010739   0.010658   0.231717   0.366189   0.010574   0.097905  -0.183316  -0.053860  NaN  -0.056197   0.100623
 FireplaceQu   0.013160   0.031013  -0.003746  -0.106440  -0.263034  -0.018641  -0.059485   0.107408   0.038313  NaN   0.011536  -0.002928
  GarageType   0.004098   0.108879   0.135666  -0.254413  -0.304995  -0.030325  -0.253752   0.196219  -0.102398  NaN   0.037598   0.019580
 GarageYrBlt  -0.004159   0.097167  -0.239603  -0.011122  -0.011836  -0.000639   0.177902  -0.156218   0.093121  NaN   0.000228  -0.071504
```

Using abobe table we can easily identify the correlation among all the columns.

DATA DISTRIBUTION:-

```
In [17]: #let's see how data is distributed for every column
         plt.figure(figsize=(20,20))
         plotnumber=1

         for column in train_data:
             if plotnumber<=81:
                 ax=plt.subplot(27,4,plotnumber)
                 sns.distplot(train_data[column])
                 plt.xlabel(column,fontsize=20)
             plotnumber+=1
         plt.show()
```

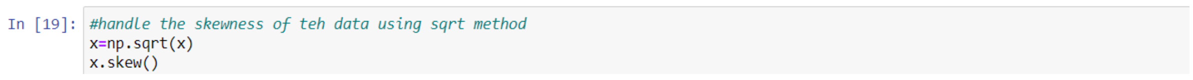


__Above code shows the__ data distribution among all the columns , the data distribution is looking not so good , let go ahead with checking skewness in the data distribution.

SKEWNESS:-

```
In [18]: #checking skewness
         x.skew().sort_values()

Out[18]: Street          -17.021969
         PoolQC          -15.903184
         MiscFeature      -5.406583
         Alley            -4.056922
         Functional       -3.999663
         GarageCond       -3.789819
         SaleType         -3.660513
         GarageQual       -3.505364
         CentralAir       -3.475188
         BsmtFinType2     -3.388419
         PavedDrive       -3.274035
         LandContour      -3.125982
         Electrical       -3.104209
         BsmtCond         -2.927336
         SaleCondition    -2.671829
         ExterCond        -2.516219
         Fence            -1.955758
         ExterQual        -1.810843
         MSZoning         -1.796785
         KitchenQual      -1.408106
         LotConfig        -1.118821
         BsmtQual         -1.107099
         BsmtExposure     -1.075098
         FireplaceQu      -0.794843
         Exterior1st      -0.612816
         LotShape         -0.603775
         GarageYrBlt      -0.602871
         Exterior2nd      -0.592349
         YearRemodAdd     -0.495864
         YearBuilt        -0.448970
         GarageCars       -0.358556
         GarageFinish     -0.129987
         BsmtFinType1     -0.019567
         Foundation       -0.002761
```

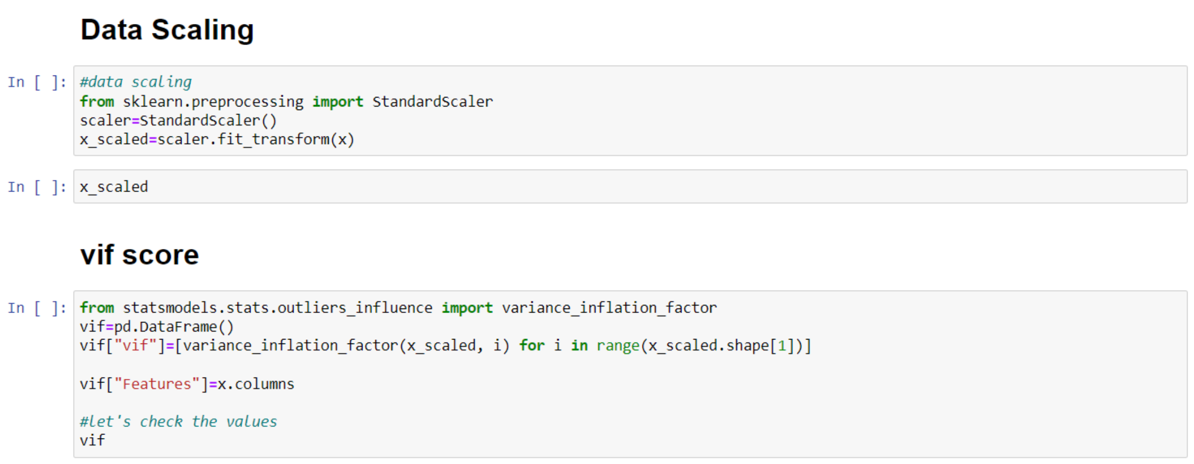
As we can see that lots of column are highly skeweed so we will try some method for handle the skewness of the data



```
In [19]: #handle the skewness of teh data using sqrt method
         x=np.sqrt(x)
         x.skew()
```

Here we use the np.sqrt() method for handling the skewness of the data.

DATA SCALING:-



Data Scaling

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

```
In [ ]: x_scaled
```

vif score

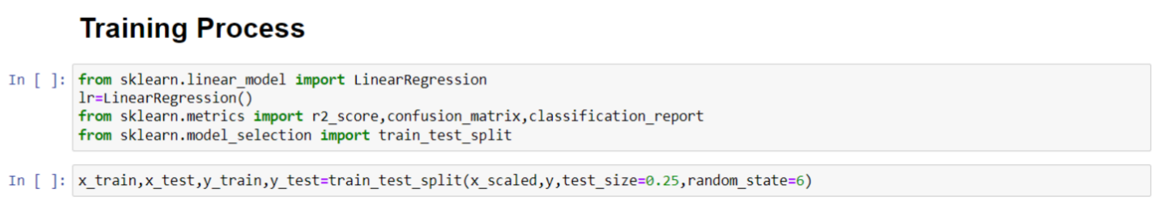
```
In [ ]: from statsmodels.stats.outliers_influence import variance_inflation_factor
        vif=pd.DataFrame()
        vif["vif"]=[variance_inflation_factor(x_scaled, i) for i in range(x_scaled.shape[1])]

        vif["Features"]=x.columns

        #let's check the values
        vif
```

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

TRAINING PROCESS:-

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

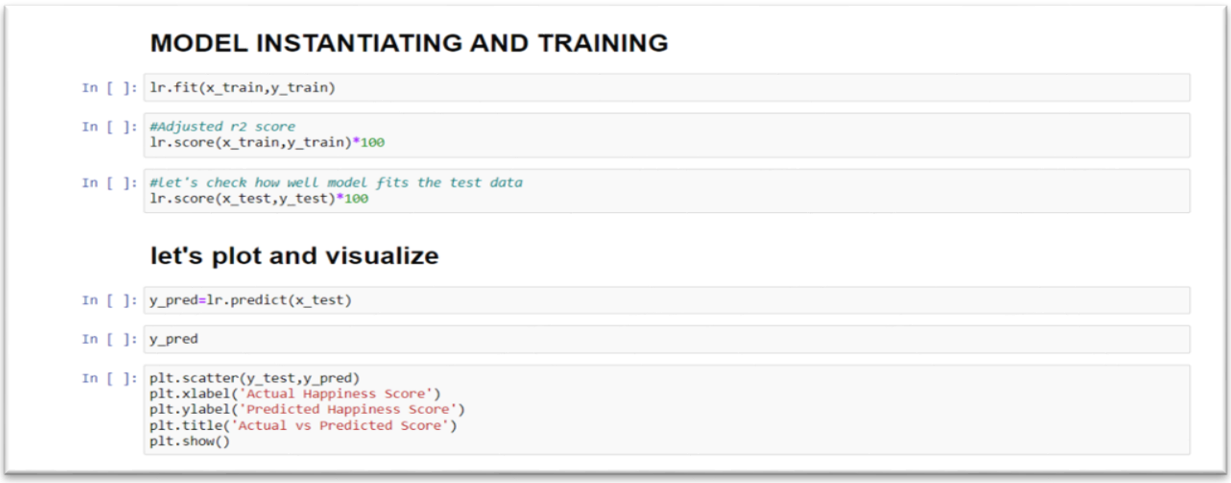
Start the training process and some important module for model training and then split the data in training and testing data set



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 [ ]: x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=6)
```



MODEL INSTANTIATING AND TRAINING

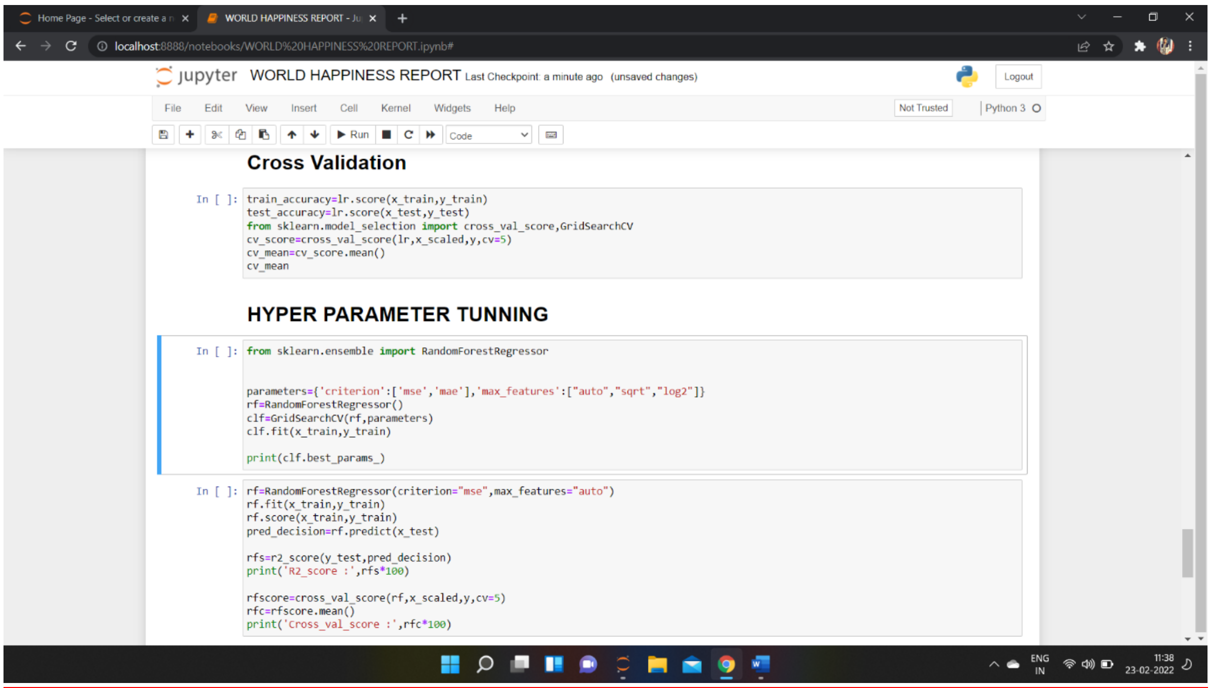
```
In [ ]: lr.fit(x_train,y_train)

In [ ]: #Adjusted r2 score
        lr.score(x_train,y_train)*100

In [ ]: #let's check how well model fits the test data
        lr.score(x_test,y_test)*100
```

let's plot and visualize

```
In [ ]: y_pred=lr.predict(x_test)

In [ ]: y_pred

In [ ]: plt.scatter(y_test,y_pred)
        plt.xlabel('Actual Happiness Score')
        plt.ylabel('Predicted Happiness Score')
        plt.title('Actual vs Predicted Score')
        plt.show()
```

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



Home Page - Select or create a n
WORLD HAPPINESS REPORT - Ju
localhost:8888/notebooks/WORLD%20HAPPINESS%20REPORT.ipynb#
jupyter
WORLD HAPPINESS REPORT Last Checkpoint: a minute ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
ENG IN
11:38 23-02-2022

Cross Validation

```
In [ ]: train_accuracy=lr.score(x_train,y_train)
        test_accuracy=lr.score(x_test,y_test)
        from sklearn.model_selection import cross_val_score,GridSearchCV
        cv_score=cross_val_score(lr,x_scaled,y,cv=5)
        cv_mean=cv_score.mean()
        cv_mean
```

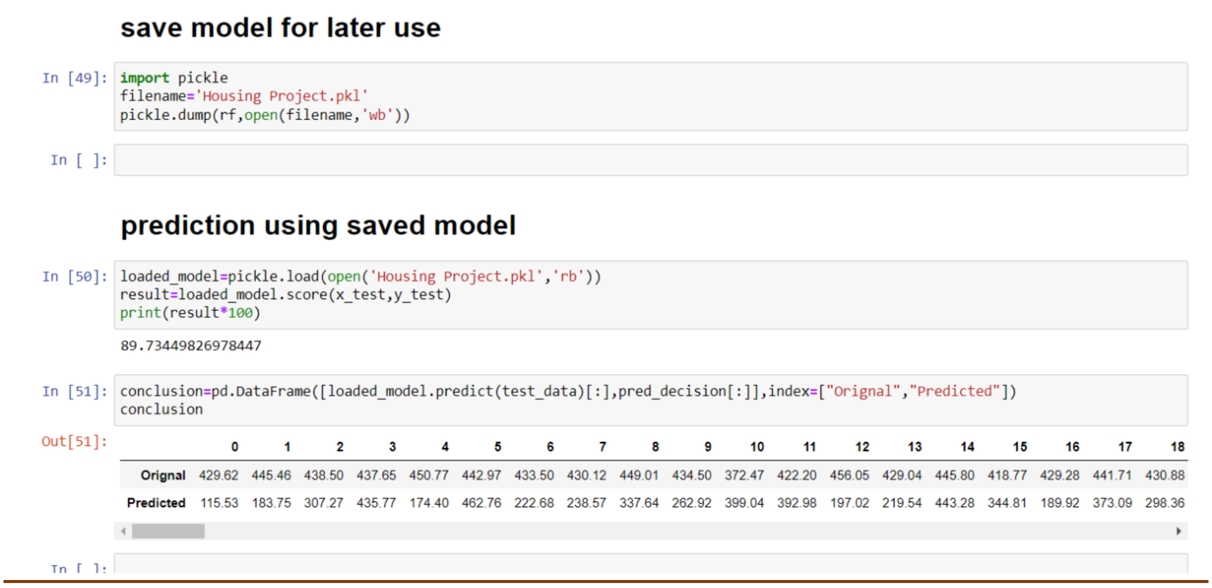
HYPER PARAMETER TUNNING

```
In [ ]: from sklearn.ensemble import RandomForestRegressor


        parameters={'criterion':['mse','mae'],'max_features':["auto","sqrt","log2"]}
        rf=RandomForestRegressor()
        clf=GridSearchCV(rf,parameters)
        clf.fit(x_train,y_train)

        print(clf.best_params_)

In [ ]: rf=RandomForestRegressor(criterion="mse",max_features="auto")
        rf.fit(x_train,y_train)
        rf.score(x_train,y_train)
        pred_decision=rf.predict(x_test)

        rfs=r2_score(y_test,pred_decision)
        print('R2_score :',rfs*100)

        rfscore=cross_val_score(rf,x_scaled,y,cv=5)
        rfc=rfscore.mean()
        print('Cross_val_score :',rfc*100)
```

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 89%.

SAVE MODEL:-



save model for later use

```
In [49]: import pickle
         filename='Housing Project.pkl'
         pickle.dump(rf,open(filename,'wb'))
```

```
In [ ]:
```

prediction using saved model

```
In [50]: loaded_model=pickle.load(open('Housing Project.pkl','rb'))
         result=loaded_model.score(x_test,y_test)
         print(result*100)

         89.73449826978447
```

```
In [51]: conclusion=pd.DataFrame([loaded_model.predict(test_data)[:],pred_decision[:]],index=["Orignal","Predicted"])
         conclusion
```

Out[51]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Orignal	429.62	445.46	438.50	437.65	450.77	442.97	433.50	430.12	449.01	434.50	372.47	422.20	456.05	429.04	445.80	418.77	429.28	441.71	430.88
Predicted	115.53	183.75	307.27	435.77	174.40	462.76	222.68	238.57	337.64	262.92	399.04	392.98	197.02	219.54	443.28	344.81	189.92	373.09	298.36

```
In [ ]:
```

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.