

# Housing Price Prediction

## 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.

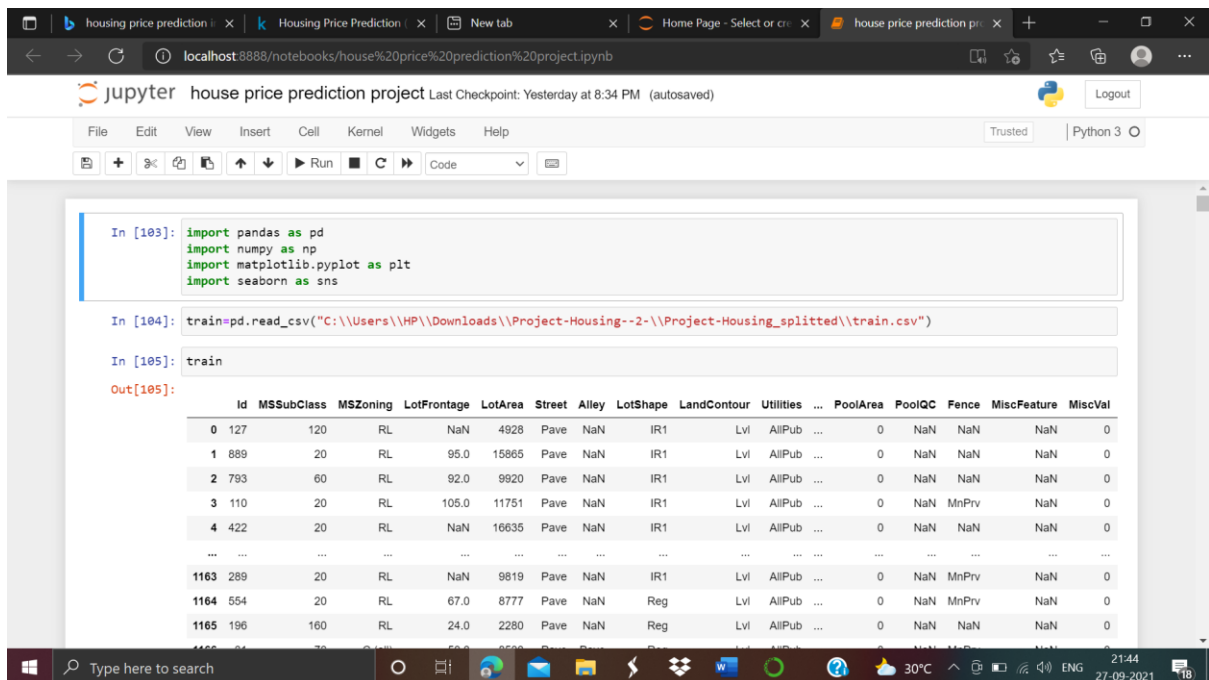
The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

- Which variables are important to predict the price of variable?
- How do these variables describe the price of the house?

## **Data**

Use housing dataset.

# Reading and Understanding the Data



The screenshot shows a Jupyter Notebook titled "house price prediction project" running on a local host. The notebook contains three input cells and one output cell. The first cell imports the necessary libraries: pandas, numpy, matplotlib.pyplot, and seaborn. The second cell loads the training data from a CSV file. The third cell displays the first few rows of the training data as a table.

```
In [103]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

In [104]: train=pd.read_csv("C:\\Users\\HP\\Downloads\\Project-Housing-2-\\Project-Housing_splitted\\train.csv")

In [105]: train
```

Out[105]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0

## Data Inspection

Let's have a look to inspect the our dataset.

1168 rows x 81 columns

we have 1168 rows and 81 columns in train data set.

```
In [106]: train.head()
```

```
Out[106]:
```

	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 x 81 columns

we have first 5 rows and 81 columns in our dataset.

```
In [107]: train.isnull().sum()
```

```
Out[107]:
```

	Id	MSSubClass
	0	0

## Analyzing the Test Variable (Sale Price):-

Let's check out the most interesting feature in this study: Sale Price. Important Note: This data is from Ames, Iowa. The location is extremely correlated with Sale Price. (I had to take a double-take at a point, since I consider myself a house-browsing enthusiast)

75% 70.000000 3.000000 80.000000 11500.000000 1.000000 3.000000 3.000000 0.0 4.000000 0.000000 ... 0.000000

max 190.000000 4.000000 313.000000 164660.000000 1.000000 3.000000 3.000000 0.0 4.000000 2.000000 ... 508.000000 48

8 rows x 76 columns

this shows the brief description of train data.

```
In [152]: train['SalePrice'].describe()
```

```
Out[152]:
```

	count	mean	std	min	25%	50%	75%	max
	1138.000000	183416.711775	79075.515414	34900.000000	132500.000000	165325.000000	216375.000000	755000.000000

Name: SalePrice, dtype: float64

Sales price is right skewed. So, we perform log transformation so that the skewness is nearly zero.

```
In [153]: test.describe()
```

```
Out[153]:
```

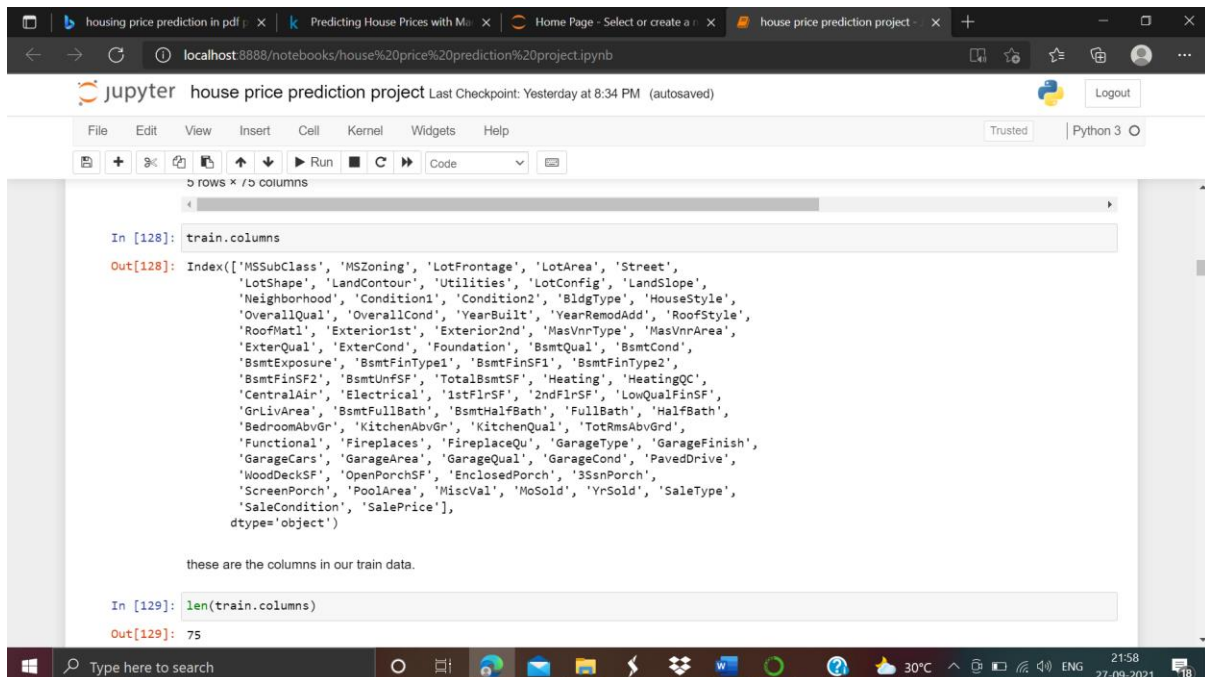
	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	RemHFinSf1	GarageArea
--	----	------------	-------------	---------	-------------	-------------	-----------	--------------	------------	------------	------------

## Multivariable Analysis:-

Let's check out all the variables! There are two types of features in housing data, categorical and numerical.

Categorical data is just like it sounds. It is in categories. It isn't necessarily linear, but it follows some kind of pattern. For example, take a feature of "Downtown". The response is either "Near", "Far", "Yes", and "No". Back then, living in downtown usually meant that you couldn't afford to live in uptown. Thus, it could be implied that downtown establishments cost less to live in. However, today, that is not the case. (Thank you, hipsters!) So we can't really establish any particular order of response to be "better" or "worse" than the other.

Numerical data is data in number form. (Who could have thought!) These features are in a linear relationship with each other. For example, a 2,000 square foot place is 2 times "bigger" than a 1,000 square foot place. Plain and simple. Simple and clean.



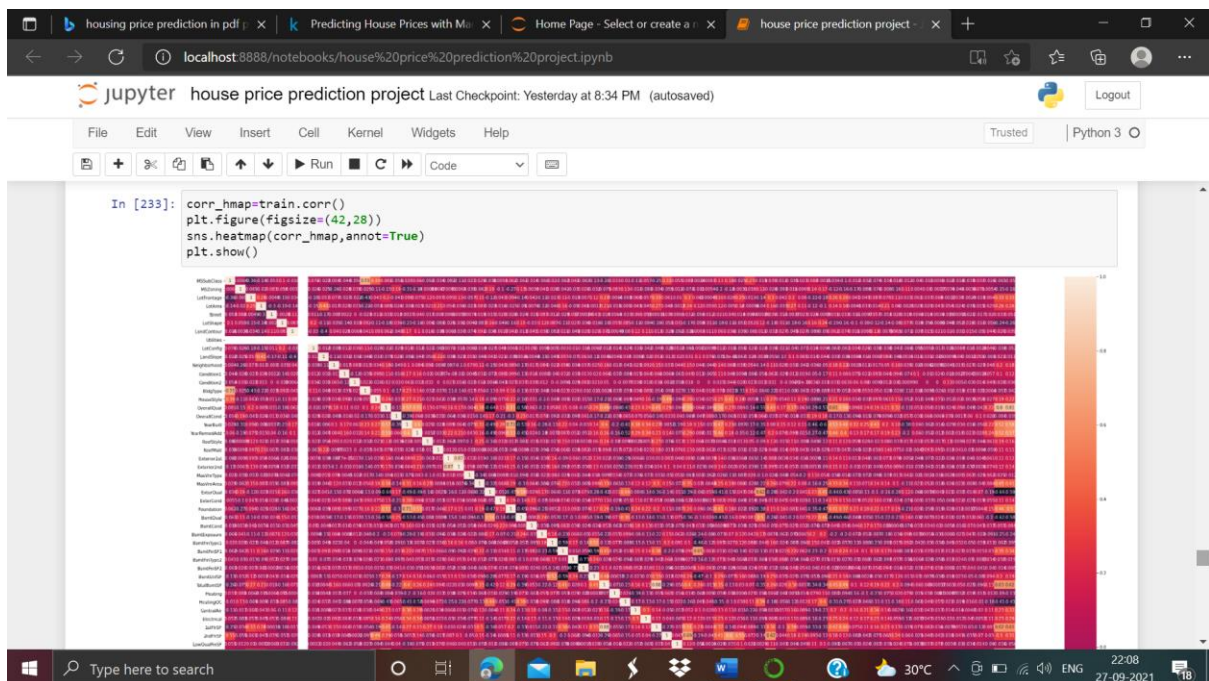
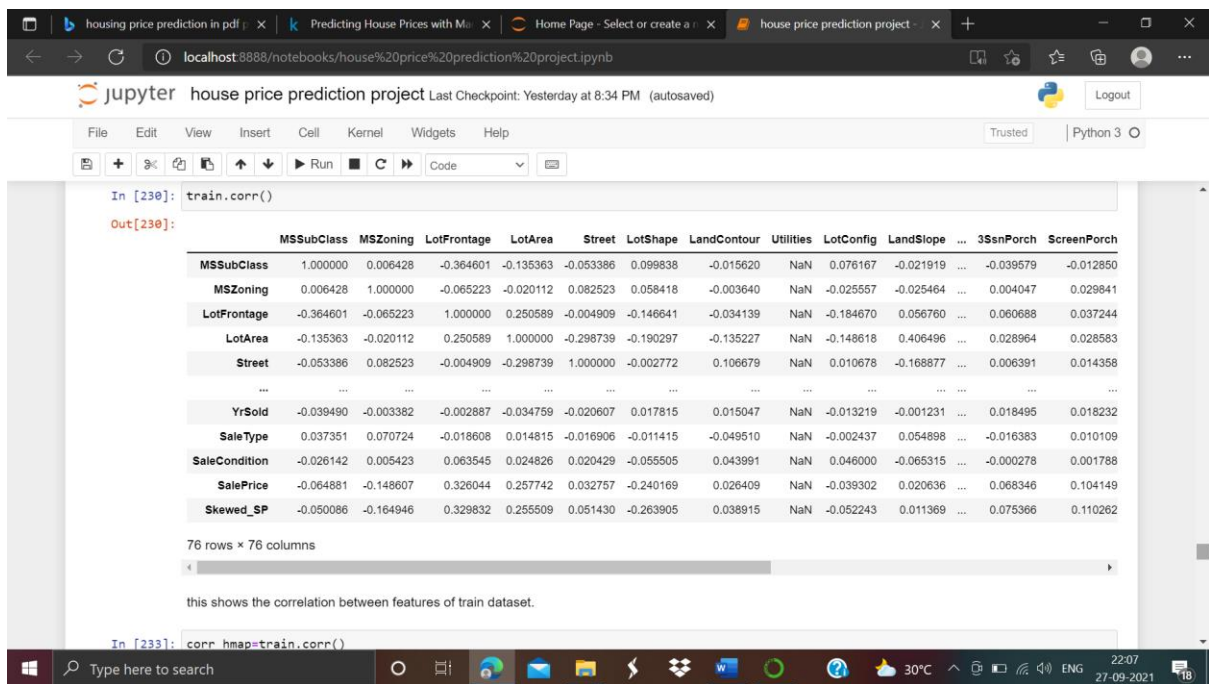
```
In [128]: train.columns
Out[128]: Index(['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
               'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
               'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
               'OverallQual', 'YearBuilt', 'YearRemodAdd', 'RoofStyle',
               'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
               'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
               'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
               'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
               'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
               'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
               'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
               'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageFinish',
               'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive',
               'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
               'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
               'SaleCondition', 'SalePrice'],
              dtype='object')

these are the columns in our train data.

In [129]: len(train.columns)
Out[129]: 75
```

## Find the correlation

It's a very important to find the correlation among the variables, but oh man is that a lot of data to look at. Let's zoom into the top 10 features most related to Sale Price.



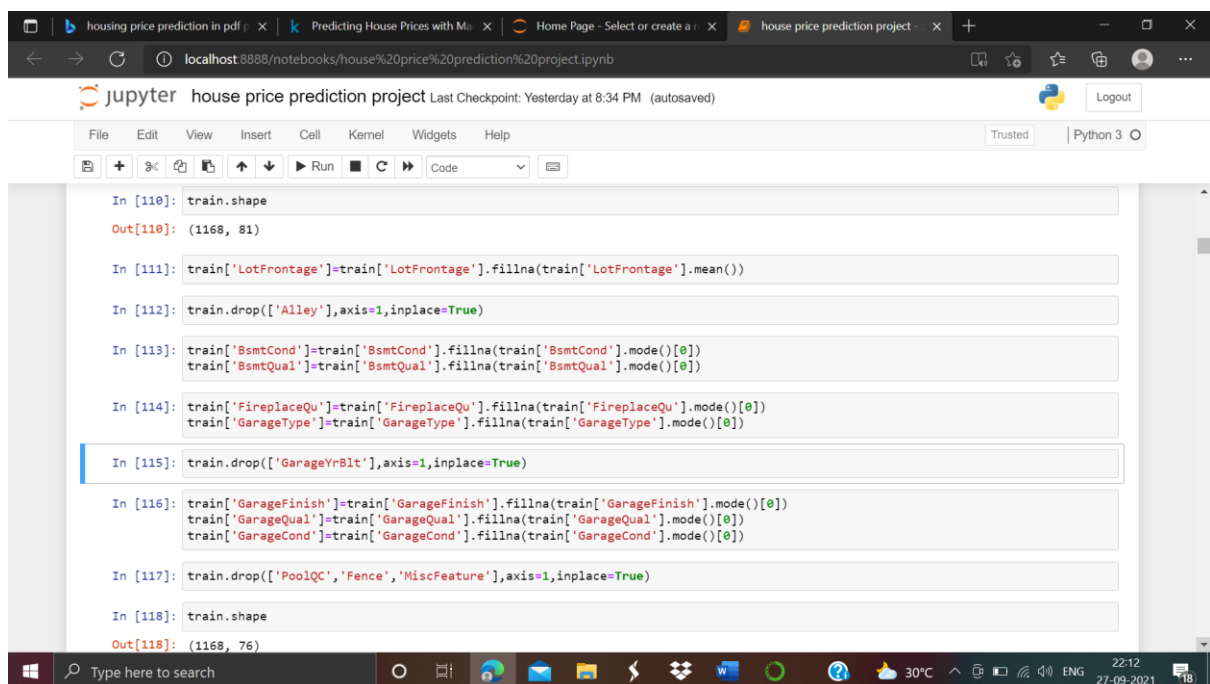
## . Impute Missing Data and Clean Data:-

Important questions when thinking about missing data:

- How prevalent is the missing data?
- Is missing data random or does it have a pattern?

The answer to these questions is important for practical reasons because missing data can imply a reduction of the sample size. This can prevent us from proceeding with the analysis. Moreover,

Let's combine both training and test data into one dataset to impute missing values and do some cleaning.





```
In [118]: train.shape
Out[118]: (1168, 76)

In [119]: train.drop(['Id'],axis=1,inplace=True)

In [120]: train.isnull().sum()
Out[120]: MSSubClass      0
         MSZoning        0
         LotFrontage     0
         LotArea         0
         Street          0
         ..
         MoSold          0
         YrSold          0
         SaleType        0
         SaleCondition    0
         SalePrice        0
         Length: 75, dtype: int64

In [121]: train['MasVnrType']=train['MasVnrType'].fillna(train['MasVnrType'].mode()[0])
         train['MasVnrArea']=train['MasVnrArea'].fillna(train['MasVnrArea'].mode()[0])

In [122]: sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='coolwarm')
Out[122]: <AxesSubplot:~>
```

```
5 rows x 80 columns

in this we have first 5 rows of test dataset.

In [194]: test.isnull().sum()
Out[194]: Id              0
         MSSubClass      0
         MSZoning        0
         LotFrontage     45
         LotArea         0
         ..
         MiscVal         0
         MoSold          0
         YrSold          0
         SaleType        0
         SaleCondition    0
         Length: 80, dtype: int64

these are the null values in our test dataset.

In [195]: test.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 80 entries, 0 to 79
Data columns (total 80 columns):
 #   Column              Non-Null Count  Dtype  
```

```
In [202]: test['LotFrontage']=test['LotFrontage'].fillna(test['LotFrontage'].mean())

In [203]: test.drop(['Alley'],axis=1,inplace=True)

In [204]: test['BsmtCond']=test['BsmtCond'].fillna(test['BsmtCond'].mode()[0])
test['BsmtQual']=test['BsmtQual'].fillna(test['BsmtQual'].mode()[0])

In [205]: test['FireplaceQu']=test['FireplaceQu'].fillna(test['FireplaceQu'].mode()[0])
test['GarageType']=test['GarageType'].fillna(test['GarageType'].mode()[0])

In [206]: test.drop(['GarageYrBlt'],axis=1,inplace=True)

In [207]: test['GarageFinish']=test['GarageFinish'].fillna(test['GarageFinish'].mode()[0])
test['GarageQual']=test['GarageQual'].fillna(test['GarageQual'].mode()[0])
test['GarageCond']=test['GarageCond'].fillna(test['GarageCond'].mode()[0])

In [208]: test.drop(['PoolQC','Fence','MiscFeature'],axis=1,inplace=True)

In [209]: test['MasVnrType']=test['MasVnrType'].fillna(test['MasVnrType'].mode()[0])
test['MasVnrArea']=test['MasVnrArea'].fillna(test['MasVnrArea'].mode()[0])

In [210]: test['BsmtExposure']=test['BsmtExposure'].fillna(test['BsmtExposure'].mode()[0])
```

as we can see there are very less no. of null values present in test dataset.

```
In [212]: test['Utilities']=test['Utilities'].fillna(test['Utilities'].mode()[0])
test['Exterior1st']=test['Exterior1st'].fillna(test['Exterior1st'].mode()[0])
test['Exterior2nd']=test['Exterior2nd'].fillna(test['Exterior2nd'].mode()[0])
test['BsmtFinType1']=test['BsmtFinType1'].fillna(test['BsmtFinType1'].mode()[0])
test['BsmtFullBath']=test['BsmtFullBath'].fillna(test['BsmtFullBath'].mode()[0])
test['BsmtHalfBath']=test['BsmtHalfBath'].fillna(test['BsmtHalfBath'].mode()[0])
test['KitchenQual']=test['KitchenQual'].fillna(test['KitchenQual'].mode()[0])
test['Functional']=test['Functional'].fillna(test['Functional'].mode()[0])
test['GarageCars']=test['GarageCars'].fillna(test['GarageCars'].mode()[0])
test['GarageArea']=test['GarageArea'].fillna(test['GarageArea'].mode()[0])
test['SaleType']=test['SaleType'].fillna(test['SaleType'].mode()[0])
test['BsmtFinSF1']=test['BsmtFinSF1'].fillna(test['BsmtFinSF1'].mean())
test['BsmtFinSF2']=test['BsmtFinSF2'].fillna(test['BsmtFinSF2'].mean())
test['BsmtUnfSF']=test['BsmtUnfSF'].fillna(test['BsmtUnfSF'].mean())
test['TotalBsmtSF']=test['TotalBsmtSF'].fillna(test['TotalBsmtSF'].mean())

In [148]: test['MSZoning']=test['MSZoning'].fillna(test['MSZoning'].mode()[0])

In [213]: test['PoolArea']=test['PoolArea'].fillna(test['PoolArea'].mode()[0])
```

## Imputing Missing Values:-

- PoolQC : data description says NA means "No Pool"
- MiscFeature : data description says NA means "no misc feature"
- Alley : data description says NA means "no alley access"
- Fence : data description says NA means "no fence"
- FireplaceQu : data description says NA means "no fireplace"



- LotFrontage : Since the area of each street connected to the house property most likely have a similar area to other houses in its neighborhood , we can fill in missing values by the median LotFrontage of the neighborhood.
- GarageType, GarageFinish, GarageQual and GarageCond : Replacing missing data with "None".
- GarageYrBlt, GarageArea and GarageCars : Replacing missing data with 0.
- BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath and BsmtHalfBath: Replacing missing data with 0.
- BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1 and BsmtFinType2 : For all these categorical basement-related features, NaN means that there isn't a basement.
- MasVnrArea and MasVnrType : NA most likely means no masonry veneer for these houses. We can fill 0 for the area and None for the type.
- MSZoning (The general zoning classification) : 'RL' is by far the most common value. So we can fill in missing values with 'RL'.
- Utilities : For this categorical feature all records are "AllPub", except for one "NoSeWa" and 2 NA . Since the house with 'NoSewa' is in the training set, this feature won't help in predictive modelling. We can then safely remove it.
- Functional : data description says NA means typical.
- Electrical : It has one NA value. Since this feature has mostly 'SBrkr', we can set that for the missing value.
- KitchenQual: Only one NA value, and same as Electrical, we set 'TA' (which is the most frequent) for the missing value in KitchenQual.
- Exterior1st and Exterior2nd : Both Exterior 1 & 2 have only one missing value. We will just substitute in the most common string
- SaleType : Fill in again with most frequent which is "WD"
- MSSubClass : Na most likely means No building class. We can replace missing values with None

## **Feature Transformation/Engineering:-**

Let's take a look at some features that may be misinterpreted to represent something it's not. MSSubClass: Identifies the type of dwelling involved in the sale. Also we apply Label encoding for both the train and test dataset.

housing price prediction in pdf | x Predicting House Prices with M... x Home Page - Select or create a... x house price prediction project - x

localhost8888/notebooks/house%20price%20prediction%20project.ipynb

jupyter house price prediction project Last Checkpoint: Yesterday at 8:34 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Normal Partial Subnorm Family Alcoa AdLand  
SaleCondition

```
In [173]: from sklearn.preprocessing import OrdinalEncoder  
enc=OrdinalEncoder()  
  
In [174]: for i in train.columns:  
    if train[i].dtypes=='object':  
        train[i]=enc.fit_transform(train[i].values.reshape(-1,1))  
  
In [175]: train  
Out[175]:
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope	...	3SsnPorch	ScreenPorch	PoolArea	M
0	120	3.0	70.98847	4928	1.0	0.0	3.0	0.0	4.0	0.0	...	0	0	0	
1	20	3.0	95.00000	15865	1.0	0.0	3.0	0.0	4.0	1.0	...	0	224	0	
2	60	3.0	92.00000	9920	1.0	0.0	3.0	0.0	1.0	0.0	...	0	0	0	
3	20	3.0	105.00000	11751	1.0	0.0	3.0	0.0	4.0	0.0	...	0	0	0	
4	20	3.0	70.98847	16635	1.0	0.0	3.0	0.0	2.0	0.0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1162	30	3.0	45.00000	8212	1.0	3.0	3.0	0.0	4.0	0.0	...	0	0	0	

Type here to search 30°C 22:24 27-09-2021

housing price prediction in pdf | x Predicting House Prices with M... x Home Page - Select or create a... x house price prediction project - x

localhost8888/notebooks/house%20price%20prediction%20project.ipynb

jupyter house price prediction project Last Checkpoint: Yesterday at 8:34 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

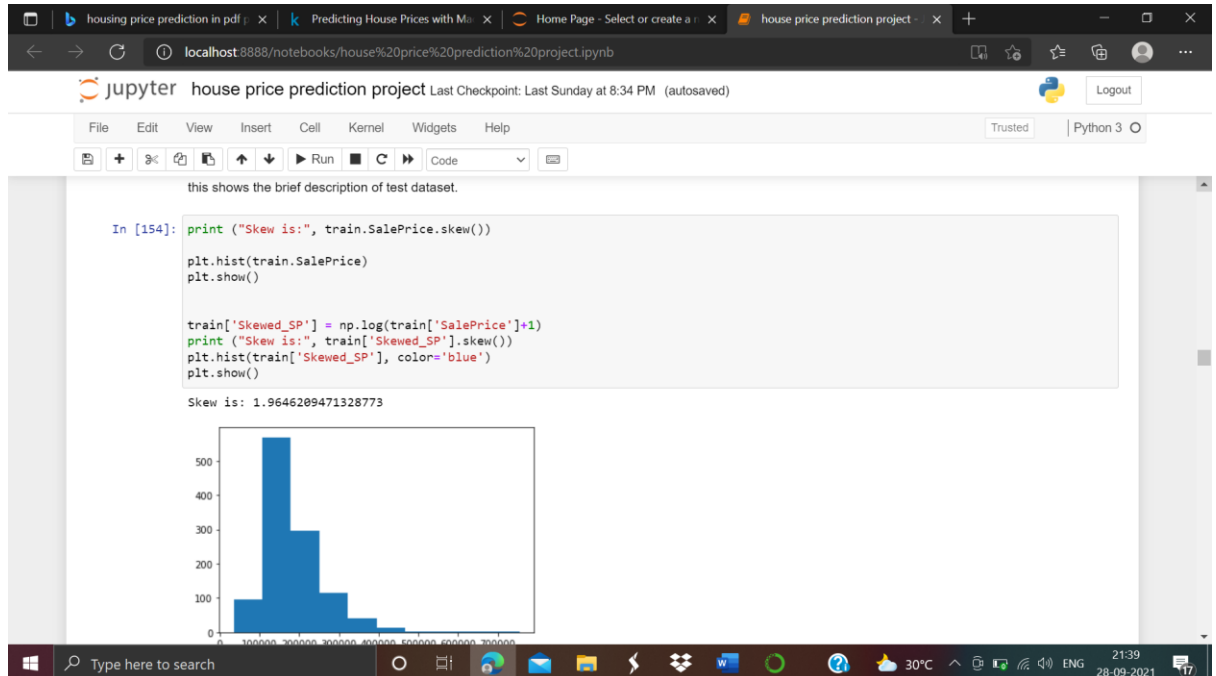
```
In [222]: test.dropna(inplace=True)  
  
In [223]: for i in test.columns:  
    if test[i].dtypes=='object':  
        test[i]=enc.fit_transform(test[i].values.reshape(-1,1))  
  
In [224]: test  
Out[224]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	WoodDeckSF	OpenPorchSF	EnclosedPorch
0	337	20	2.0	86.000000	14157	1.0	0.0	1.0	0.0	0.0	...	178	51	0
1	1018	120	2.0	66.425101	5814	1.0	0.0	3.0	0.0	1.0	...	63	0	0
2	929	20	2.0	66.425101	11838	1.0	3.0	3.0	0.0	4.0	...	202	151	0
3	1148	70	2.0	75.000000	12000	1.0	3.0	0.0	0.0	4.0	...	0	0	0
4	1227	60	2.0	86.000000	14598	1.0	0.0	3.0	0.0	1.0	...	100	18	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
287	83	20	2.0	78.000000	10206	1.0	3.0	3.0	0.0	4.0	...	144	99	0
288	1048	20	2.0	57.000000	9245	1.0	1.0	3.0	0.0	4.0	...	0	0	0
289	17	20	2.0	66.425101	11241	1.0	0.0	3.0	0.0	1.0	...	0	0	0
290	523	50	3.0	50.000000	5000	1.0	3.0	3.0	0.0	0.0	...	0	24	36
291	1379	160	3.0	21.000000	1953	1.0	3.0	3.0	0.0	4.0	...	72	0	0

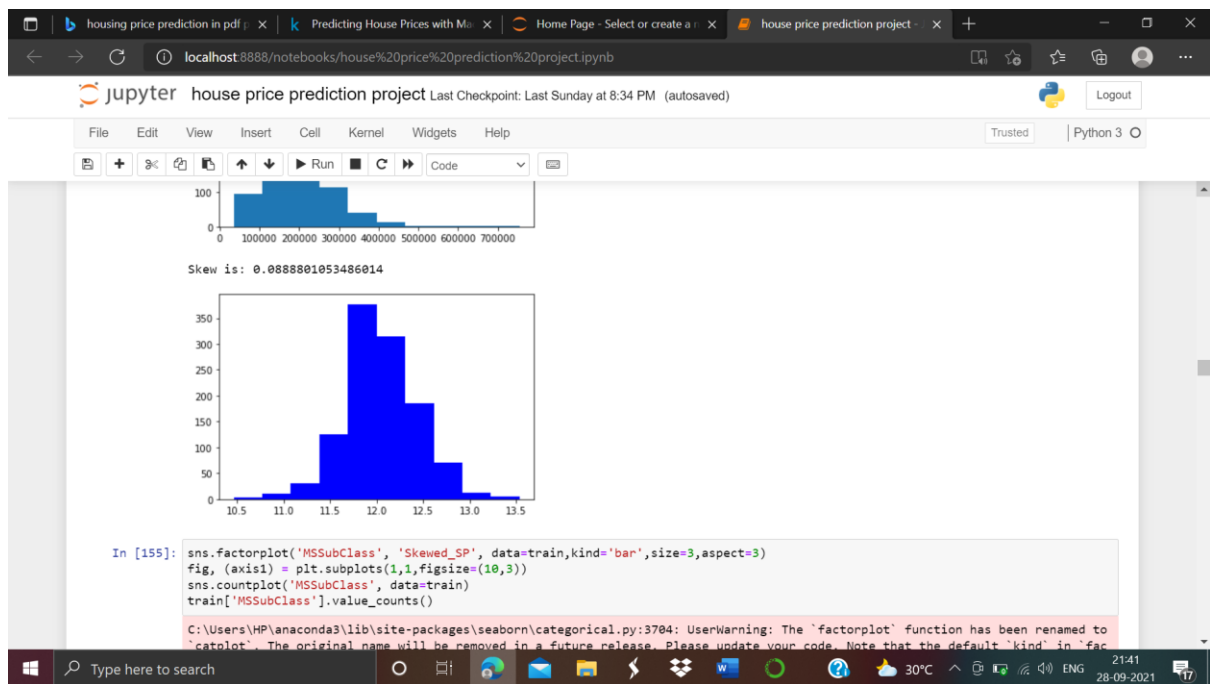
Type here to search 30°C 22:25 27-09-2021

## Fixing "skewed" features:-

Here, we fix all of the skewed data to be more normal so that our models will be more accurate when making predictions.



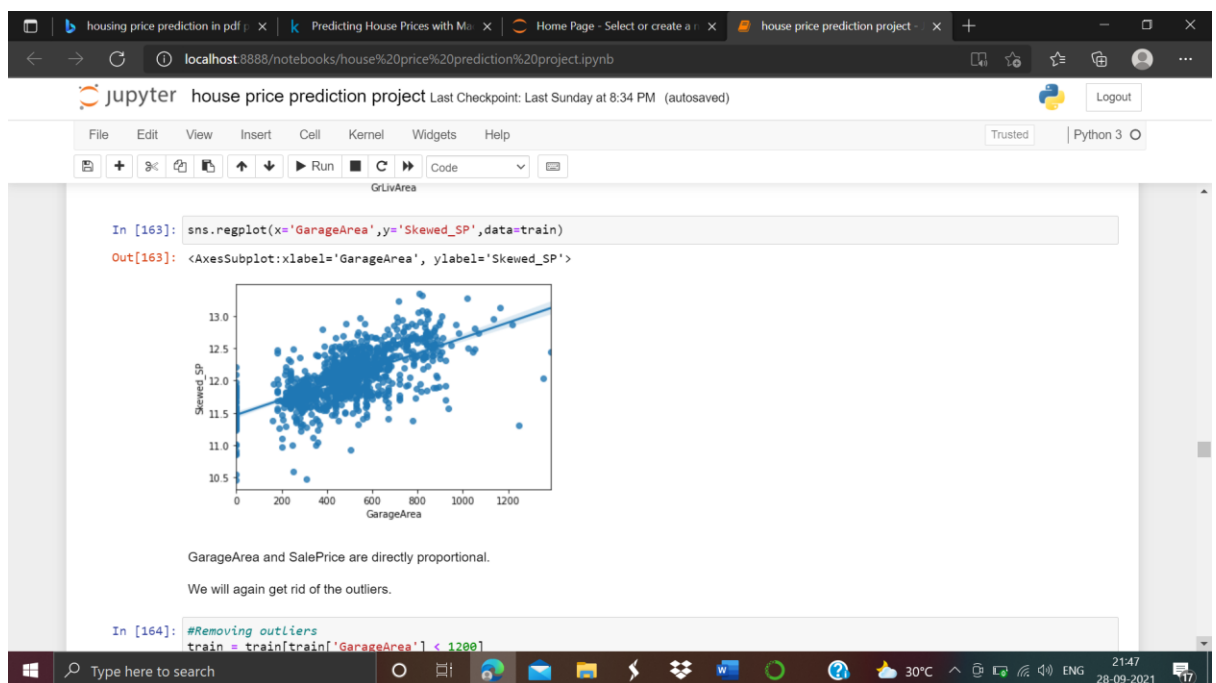
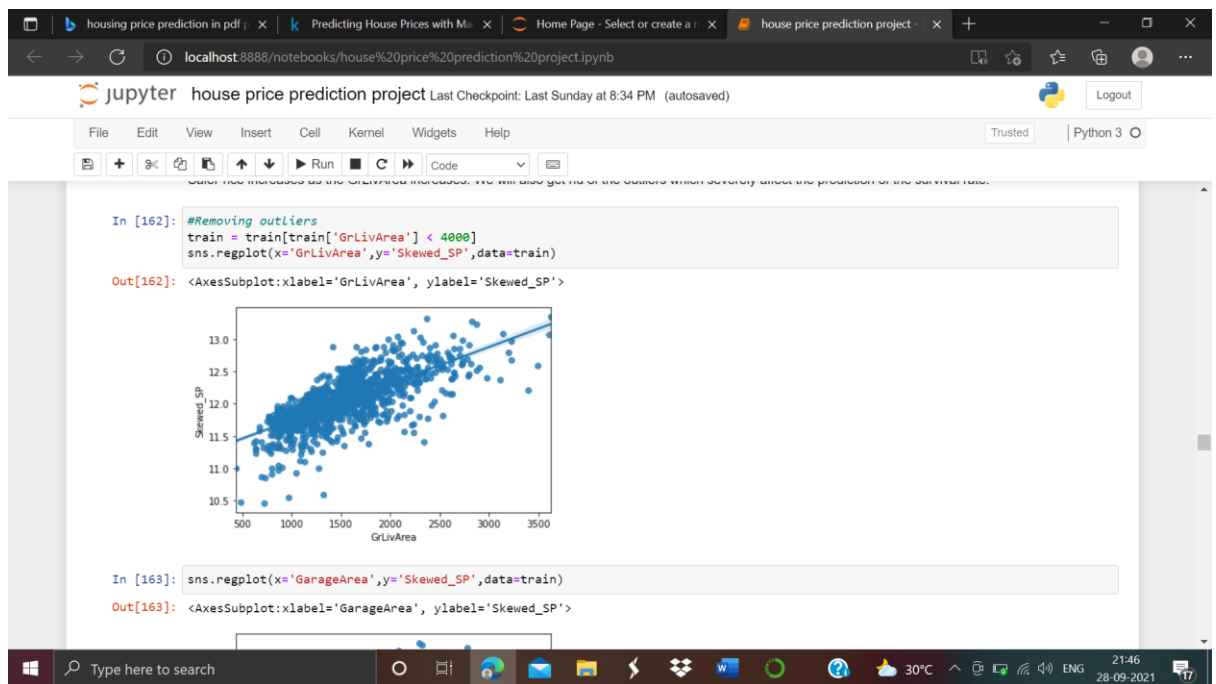
Here we can see the skewness is approx. 1.9646



Here the skewness is approx. 0.0888.

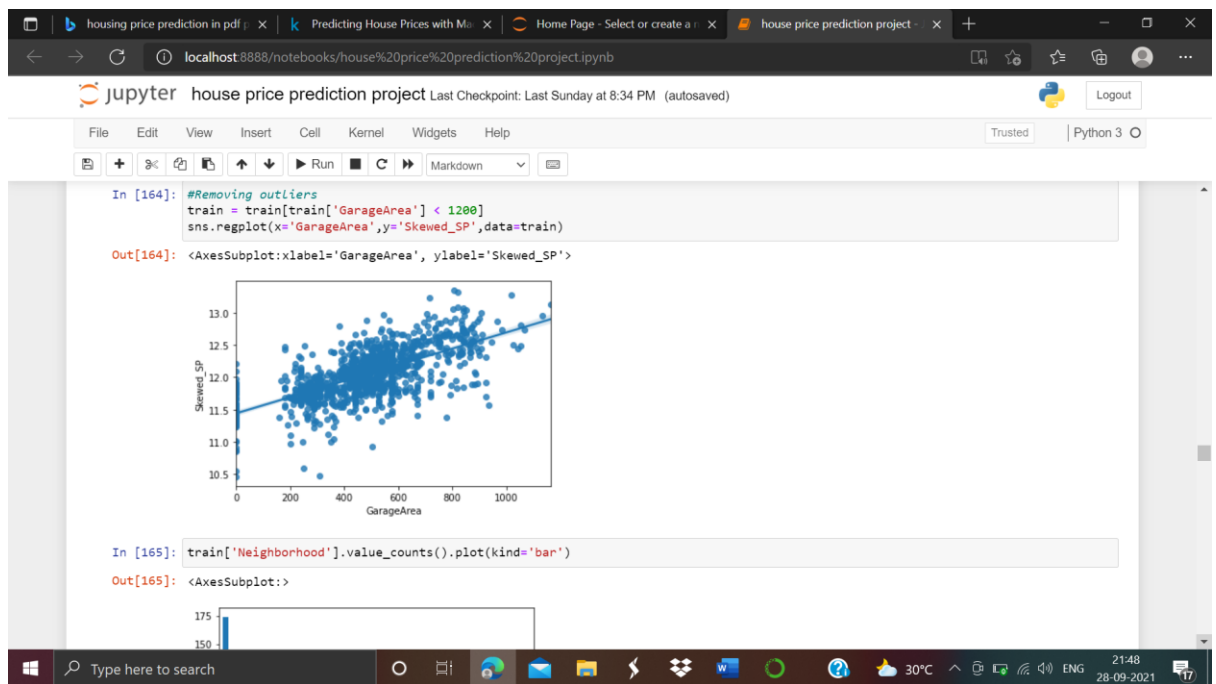
### Removing outliers:-

Now we will move to outliers. We have to remove these outliers from the dataset.



Garage Area and Sale Price are directly proportional.

We will again get rid of the outliers.



## Splitting the data and find the accuracy:-

Now we are going to split the our dataset and find the accuracy for both train and test dataset. We have used min max scaler also.

```
In [239]: y.shape
Out[239]: (1130,)

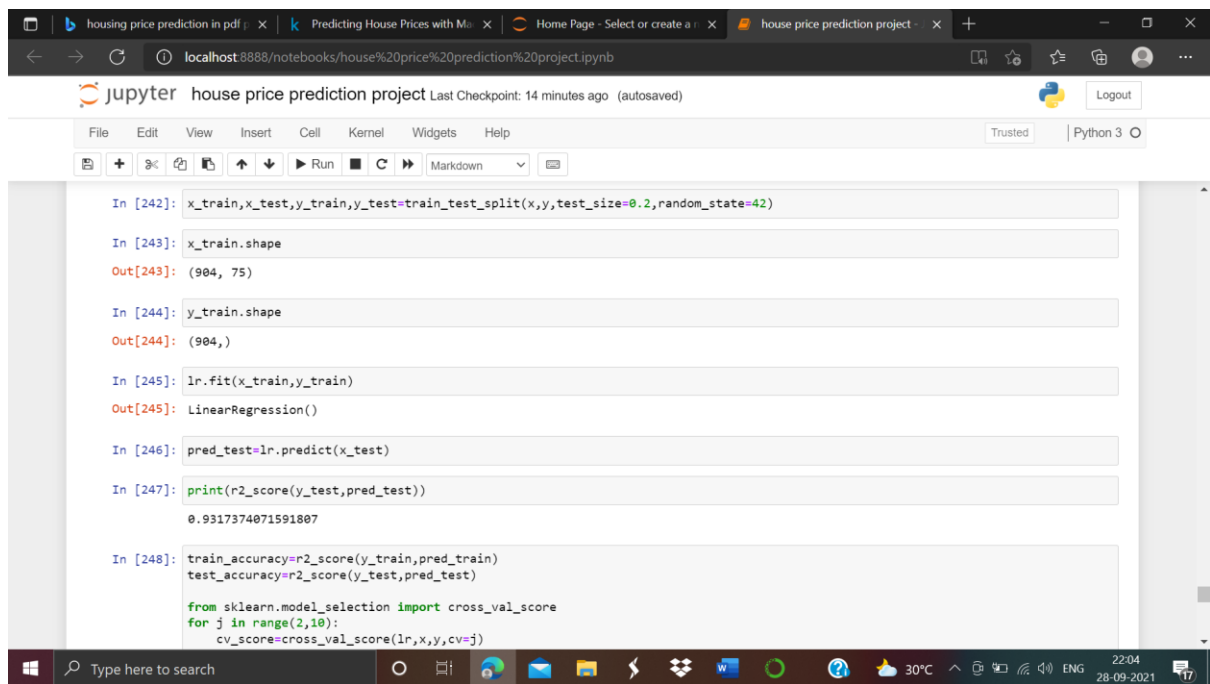
In [240]: from sklearn.preprocessing import MinMaxScaler
          from sklearn.linear_model import LinearRegression
          lr=LinearRegression()
          from sklearn.metrics import r2_score
          from sklearn.model_selection import train_test_split
          import warnings
          warnings.filterwarnings('ignore')

In [241]: for i in range(0,100):
          x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=i)
          lr.fit(x_train,y_train)
          pred_train=lr.predict(x_train)
          pred_test=lr.predict(x_test)
          print(f'At random state{i},the training accuracy is:- {r2_score(y_train,pred_train)}')
          print(f'At random state{i},the testing accuracy is:- {r2_score(y_test,pred_test)}')
          print('\n')

At random state0,the training accuracy is:- 0.9618392757916243
At random state0,the testing accuracy is:- 0.9222246131233829

At random state1,the training accuracy is:- 0.9574777409669682
At random state1,the testing accuracy is:- 0.9409671851886282
```





```
In [242]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

In [243]: x_train.shape
Out[243]: (904, 75)

In [244]: y_train.shape
Out[244]: (904,)

In [245]: lr.fit(x_train,y_train)
Out[245]: LinearRegression()

In [246]: pred_test=lr.predict(x_test)

In [247]: print(r2_score(y_test,pred_test))
0.9317374071591807

In [248]: train_accuracy=r2_score(y_train,pred_train)
test_accuracy=r2_score(y_test,pred_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(lr,x,y,cv=j)
```

## Modeling and Predictions:-

For our models, we are going to use Random forest regressor, Decision tree regressor and linear regression.

We have also calculated the r2 score and cross validation score.

The screenshot shows a Jupyter Notebook interface with the title 'house price prediction project'. The code in the cell is as follows:

```
In [248]: train_accuracy=r2_score(y_train,pred_train)
test_accuracy=r2_score(y_test,pred_test)

from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score=cross_val_score(lr,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'At cross fold {j} the cv score is {cv_mean} and accuracy score for tranning is {train_accuracy} and accuracy for the t
    print('\n')
```

The output shows the results for cross folds 2 through 6. The cv score (mean) and accuracy score for training and testing are printed for each fold. The accuracy score for training is constant at -0.9292529889412808, and the accuracy score for testing is constant at 0.9317374071591807.

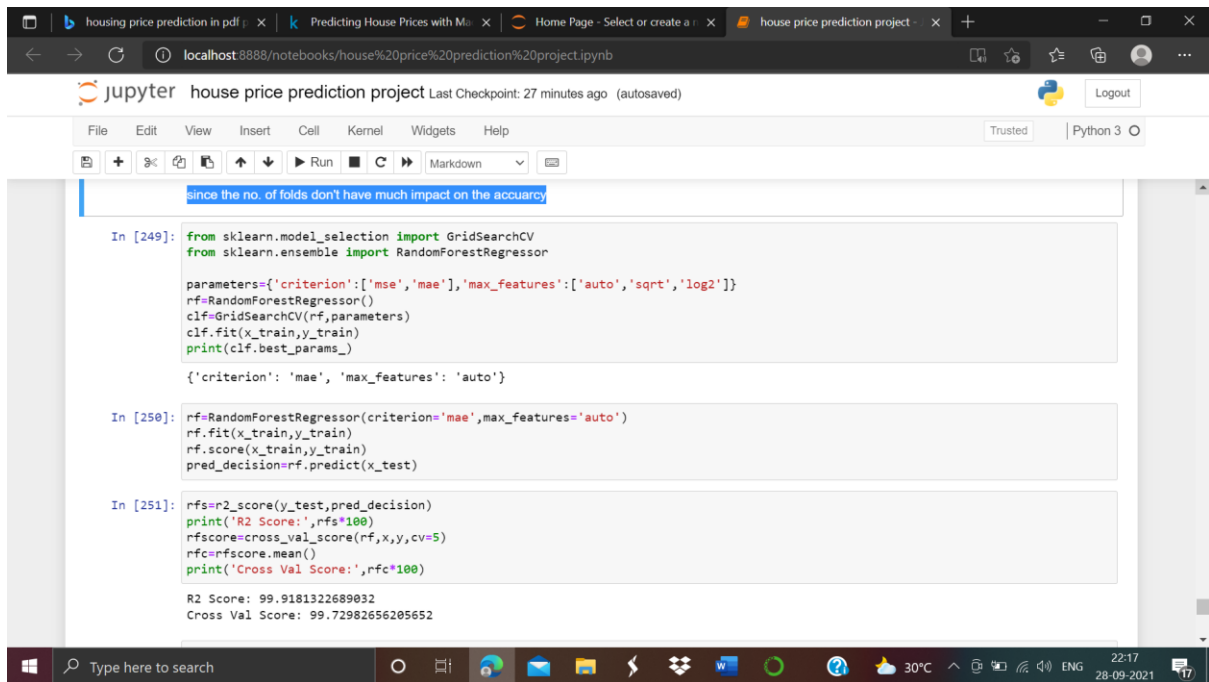
Cross Fold	CV Score (mean)	Accuracy Score for Training	Accuracy Score for Testing
2	0.9408417591646328	-0.9292529889412808	0.9317374071591807
3	0.9422227392477059	-0.9292529889412808	0.9317374071591807
4	0.941073777528771	-0.9292529889412808	0.9317374071591807
5	0.9406905022354662	-0.9292529889412808	0.9317374071591807
6	0.9439930174293281	-0.9292529889412808	0.9317374071591807

The screenshot shows the continuation of the Jupyter Notebook. The output shows the results for cross folds 3 through 9. The cv score (mean) and accuracy score for training and testing are printed for each fold. The accuracy score for training is constant at -0.9292529889412808, and the accuracy score for testing is constant at 0.9317374071591807.

Cross Fold	CV Score (mean)	Accuracy Score for Training	Accuracy Score for Testing
3	0.9422227392477059	-0.9292529889412808	0.9317374071591807
4	0.941073777528771	-0.9292529889412808	0.9317374071591807
5	0.9406905022354662	-0.9292529889412808	0.9317374071591807
6	0.9439930174293281	-0.9292529889412808	0.9317374071591807
7	0.9435478422270096	-0.9292529889412808	0.9317374071591807
8	0.943588497610495	-0.9292529889412808	0.9317374071591807
9	0.9437660301575019	-0.9292529889412808	0.9317374071591807

As above we can see that since the no. of folds don't have much impact on the accuracy.

We have to check the performance of base models by evaluating the cross-validation.



housing price prediction in pdf | Predicting House Prices with M... | Home Page - Select or create a... | house price prediction project - | +

localhost8888/notebooks/house%20price%20prediction%20project.ipynb

jupyter house price prediction project Last Checkpoint: 27 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

since the no. of folds don't have much impact on the accuracy

```
In [249]: from sklearn.model_selection import GridSearchCV
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_)

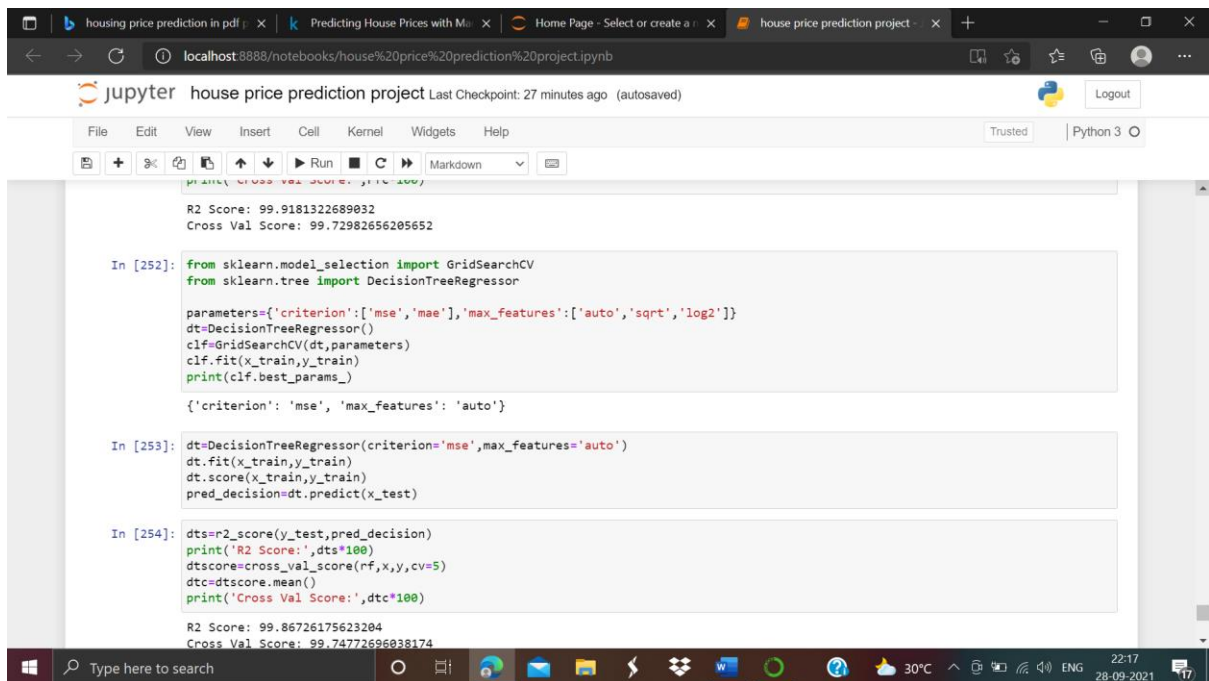
{'criterion': 'mae', 'max_features': 'auto'}

In [250]: rf=RandomForestRegressor(criterion='mae',max_features='auto')
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
pred_decision=rf.predict(x_test)

In [251]: rfs=r2_score(y_test,pred_decision)
print('R2 Score:',rfs*100)
rfscore=cross_val_score(rf,x,y,cv=5)
rfc=rfscore.mean()
print('Cross Val Score:',rfs*100)

R2 Score: 99.9181322689032
Cross Val Score: 99.72982656205652
```

Type here to search 30°C ENG 22:17 28-09-2021



housing price prediction in pdf | Predicting House Prices with M... | Home Page - Select or create a... | house price prediction project - | +

localhost8888/notebooks/house%20price%20prediction%20project.ipynb

jupyter house price prediction project Last Checkpoint: 27 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
print('Cross Val Score:', rfs*100)

R2 Score: 99.9181322689032
Cross Val Score: 99.72982656205652

In [252]: from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor

parameters={'criterion':['mse','mae'],'max_features':['auto','sqrt','log2']}
dt=DecisionTreeRegressor()
clf=GridSearchCV(dt,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)

{'criterion': 'mse', 'max_features': 'auto'}

In [253]: dt=DecisionTreeRegressor(criterion='mse',max_features='auto')
dt.fit(x_train,y_train)
dt.score(x_train,y_train)
pred_decision=dt.predict(x_test)

In [254]: dts=r2_score(y_test,pred_decision)
print('R2 Score:',dts*100)
dtscore=cross_val_score(rf,x,y,cv=5)
dte=dtscore.mean()
print('Cross Val Score:',dte*100)

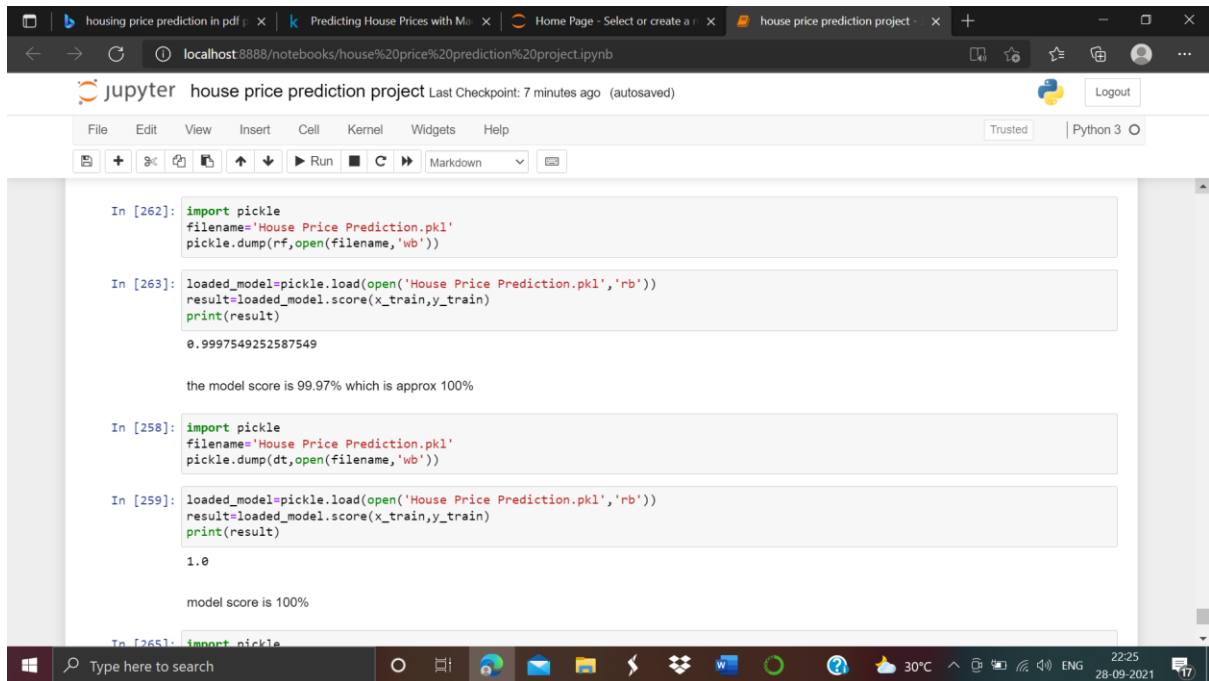
R2 Score: 99.86726175623204
Cross Val Score: 99.74772696038174
```

Type here to search 30°C ENG 22:17 28-09-2021

As we can see we have applied different -different regression techniques.

## Saving the model:-

Let's we will save the our model with file name "House price prediction" in .pkl format. As we can see below:-



The screenshot shows a Jupyter Notebook titled "house price prediction project" with the following code cells:

```
In [262]: import pickle
filename='House Price Prediction.pkl'
pickle.dump(rf,open(filename,'wb'))

In [263]: loaded_model=pickle.load(open('House Price Prediction.pkl','rb'))
result=loaded_model.score(x_train,y_train)
print(result)

0.9997549252587549

the model score is 99.97% which is approx 100%

In [258]: import pickle
filename='House Price Prediction.pkl'
pickle.dump(dt,open(filename,'wb'))

In [259]: loaded_model=pickle.load(open('House Price Prediction.pkl','rb'))
result=loaded_model.score(x_train,y_train)
print(result)

1.0

model score is 100%
```

The bottom of the image shows a Windows taskbar with the date 28-09-2021 and time 22:25.

## Conclusion:-

As we can see the our train and test data are balanced Decision tree Regressor score is 100% and Random forest regressor test score also comes with approx 100 %

linear regression model score is 96%.

Hence Decision tree and random forest regressor model's are the our best model.