HOUSE PRICE PREDICTION

Submitted by:

SAYAN KUMAR BHUNIA

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion of the project.

- https://towardsdatascience.com/
- https://anshikaaxena.medium.com/
- https://medium.com/https://medium.com/

INTRODUCTION

➢ Business Problem Framing

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 company is looking at prospective properties to buy houses to enter the market. I have 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?

➢ Conceptual Background of the Domain Problem

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. My problem is related to one such housing company.

➢ Review of Literature

I am 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.

➢ Motivation for the Problem Undertaken

Dataset has many variables so that it will be challenging to do all operations for building a best model which can give best accuracy and I can learn some new techniques.

Analytical Problem Framing

➢ Mathematical/ Analytical Modelling of the Problem

We shall build a supervised regression model to predict the risk of loan default.

Now, when we talk about building a supervised regression catering to certain use, following three things come into our minds:

- **Data** appropriate to the business requirement or use-case we are trying to solve

- A **regression model** which we think (or, rather assess) to be the best for our solution.
- **Optimize** the chosen model to ensure best performance.

➢ Data Sources and their formats

I am using CSV (comma-separated values) format file for training the model which is having 1168 rows ×81 columns.

8]:

	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
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

1168 rows × 81 columns

And I am using another dataset for predicting house price.

```
test_df
```

581]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFea
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
287	83	20	RL	78.0	10206	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
288	1048	20	RL	57.0	9245	Pave	NaN	IR2	Lvl	AllPub	...	0	0	NaN	NaN	
289	17	20	RL	NaN	11241	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	
290	523	50	RM	50.0	5000	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	
291	1379	160	RM	21.0	1953	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	

292 rows × 80 columns

➢ Data Pre-processing

Following steps have been performed on the data.

- **checking missing values**-
  - If there is any missing value present in your data set then for a better and correct accuracy you have to impute it.
  - If missing data present in object type column, then you have to take most frequent value for your missing data.
  - If missing data present in int or float type column then use mean/median for missing value.

In the following case missing value present:

```
77]: import sklearn
     import numpy as np
     from sklearn.impute import SimpleImputer
     imp=SimpleImputer(missing_values=np.nan,strategy='most_frequent')
     df['GarageCond']=imp.fit_transform(df['GarageCond'].values.reshape(-1,1))
     df['GarageType']=imp.fit_transform(df['GarageType'].values.reshape(-1,1))
     df['LotFrontage']=imp.fit_transform(df['LotFrontage'].values.reshape(-1,1))
     df['BsmtExposure']=imp.fit_transform(df['BsmtExposure'].values.reshape(-1,1))
     df['GarageFinish']=imp.fit_transform(df['GarageFinish'].values.reshape(-1,1))
     df['BsmtCond']=imp.fit_transform(df['BsmtCond'].values.reshape(-1,1))
     df['GarageFinish']=imp.fit_transform(df['GarageFinish'].values.reshape(-1,1))
     df['GarageYrBlt']=imp.fit_transform(df['GarageYrBlt'].values.reshape(-1,1))

     df
```

```
In [562]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
 14  Condition2     1168 non-null   object
 15  BldgType       1168 non-null   object
 16  HouseStyle     1168 non-null   object
 17  OverallQual    1168 non-null   int64
 18  OverallCond    1168 non-null   int64
 19  YearBuilt      1168 non-null   int64
 20  YearRemodAdd   1168 non-null   int64
 21  RoofStyle      1168 non-null   object
 22  RoofMatl       1168 non-null   object
 23  Exterior1st    1168 non-null   object
 24  Exterior2nd    1168 non-null   object
 25  MasVnrType     1161 non-null   object
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object
 28  ExterCond      1168 non-null   object
 29  Foundation     1168 non-null   object
 30  BsmtQual       1138 non-null   object
 31  BsmtCond       1138 non-null   object
 32  BsmtExposure   1137 non-null   object
```

- **Encoding categorical variables** -as we can see there are 3 object data type columns present so we will encode it into (int) format.
    - Apply **Label Encoding**, if number of categories in a categorical variable is equal to 2.
    - Apply **One-Hot Encoding**, if number of categories in a categorical variable is greater than 2.

In the following case Label Encoding is used.

```
[51]: import sklearn
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
le=LabelEncoder()
list1=['SaleCondition','SaleType','PavedDrive','GarageCond','GarageFinish','GarageType','Functional','KitchenQual','CentralAir',
       'BsmtExposure','RoofStyle','RoofMatl','Exterior1st','Exterior2nd','MasVnrType','MSSubClass', 'MSZoning', 'LotFrontage', '
for i in list1:
    df[i]=le.fit_transform(df[i].astype(str))
df
```

t[51]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	OpenPorchSF	EnclosedPorch	3SsnPorch
0	127	0	3	66	506	1	0	3	0	4	...	205	0	0
1	889	4	3	101	354	1	0	3	0	4	...	207	0	0
2	793	9	3	98	878	1	0	3	0	1	...	130	0	0

- **Feature scaling-** Feature Scaling ensures that all features will get equal importance in supervised classifier models. Standard scaler was used to scale all features in the data.

```
]: from sklearn.preprocessing import StandardScaler
```

```
]: sc=StandardScaler()
x=sc.fit_transform(x)
x
```

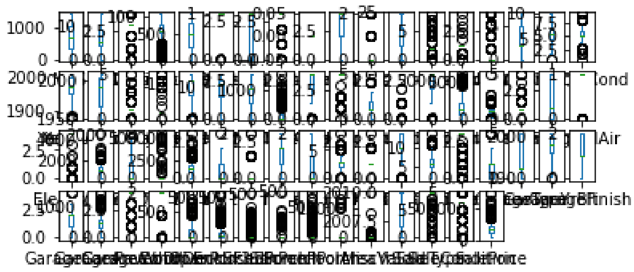
```
]: array([[-1.47236395, -1.67168039, -0.30715915, ..., -0.6485719 ,
          0.32800058, -0.25284016],
        [ 0.12130708,  0.94719524, -0.30715915, ..., -0.6485719 ,
          0.32800058, -0.25284016],
        [ 1.08803845,  0.94719524, -0.30715915, ..., -1.38179951,
         -2.36110343,  2.83295907],
        ...,
        [-1.08471424, -0.50773567, -0.30715915, ...,  1.55111091,
          0.32800058, -0.25284016],
        [-1.30725389, -1.38069421, -0.30715915, ...,  0.81788331,
          0.32800058, -0.25284016],
        [-0.29984322,  0.94719524, -0.30715915, ..., -1.38179951,
          0.32800058, -0.25284016]])
```

- **Reducing dimension of the data-** Sklearn's pca can be used to apply principal component analysis on the data. This helped in finding the vectors of maximal variance in the data.

- **Outliers detection-** In simple words, an outlier is an observation that diverges from an overall pattern on a sample.

  In the following case box plot is used to detect outliers.

```
:3]: df.plot(kind='box',subplots=True,layout=(6,18))

:3]: Id                AxesSubplot(0.125,0.772143;0.036215x0.107857)
     MSSubClass     AxesSubplot(0.168458,0.772143;0.036215x0.107857)
     MSZoning       AxesSubplot(0.211916,0.772143;0.036215x0.107857)
     LotFrontage    AxesSubplot(0.255374,0.772143;0.036215x0.107857)
     LotArea        AxesSubplot(0.298832,0.772143;0.036215x0.107857)
                                          ...
     MiscVal        AxesSubplot(0.559579,0.383857;0.036215x0.107857)
     YrSold         AxesSubplot(0.603037,0.383857;0.036215x0.107857)
     SaleType       AxesSubplot(0.646495,0.383857;0.036215x0.107857)
     SaleCondition  AxesSubplot(0.689953,0.383857;0.036215x0.107857)
     SalePrice      AxesSubplot(0.733411,0.383857;0.036215x0.107857)
     Length: 69, dtype: object
```



  There are many types of outlier detection techniques such as Z-Score or Extreme Value Analysis, Probabilistic and Statistical Modelling, Information Theory Models, Standard Deviation etc.

➢ Outliers Removal

In our dataset, we observed variations in the relation between values of some attributes.

So that these types of rows are dropped from the dataset using z-scaler method.

```
9]: from scipy.stats import zscore
z=np.abs(zscore(df))
z
```

9]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	...	F
0	1.432988	1.770863	0.030250	0.053067	0.162455	0.058798	1.371997	0.319529	0.606320	0.226842	...	
1	0.397806	0.641053	0.030250	1.643537	0.420384	0.058798	1.371997	0.319529	0.606320	3.284671	...	
2	0.167155	0.771210	0.030250	1.498113	1.588877	0.058798	1.371997	0.319529	1.220507	0.226842	...	
3	1.473832	0.641053	0.030250	3.010005	1.110589	0.058798	1.371997	0.319529	0.606320	0.226842	...	
4	0.724216	0.641053	0.030250	0.053067	0.362867	0.058798	1.371997	0.319529	0.611565	0.226842	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1162	1.043764	0.641053	0.030250	0.053067	1.546697	0.058798	1.371997	0.319529	0.606320	0.226842		

```
: threshold=3
print(np.where(z>3))
```

```
(array([   1,    1,    3, ..., 1159, 1159, 1159], dtype=int64), array([ 9, 20,  3, ..., 34, 55, 56],
dtype=int64))
```

```
: df_new=df[(z<3).all(axis=1)]
df_new
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConfig	LandSlope	...	PavedDr
0	127	0	3	66	506	1	0	3	4	0	...	
2	793	9	3	98	878	1	0	3	1	0	...	
5	1197	9	3	64	309	1	0	3	4	0	...	
6	561	4	3	66	142	1	0	3	4	0	...	
11	833	9	3	50	840	1	0	3	1	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1158	673	4	3	66	134	1	0	3	4	0	...	
1161	1301	9	3	66	85	1	0	3	1	0	...	
1163	289	4	3	66	867	1	0	3	4	0	...	
1165	196	1	3	36	421	1	3	3	2	0	...	

➢ Software Requirements and library Used

```
[1]: import pandas
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
10]: import sklearn
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

- **NumPy**
  NumPy is a popular Python library for multi-dimensional array and matrix processing because it can be used to perform a great variety of mathematical operations. Its capability to handle linear algebra, Fourier transform, and more, makes NumPy ideal for machine learning and artificial intelligence (AI) projects, allowing users to manipulate the matrix to easily improve machine learning performance. NumPy is faster and easier to use than most other Python libraries.

- **Scikit-learn**
  Scikit-learn is a very popular machine learning library that is built on NumPy and SciPy. It supports most of the classic supervised and unsupervised learning algorithms, and it can also be used for data mining, modelling, and analysis.

- **Seaborn**
  Seaborn is another open-source Python library, one that is based on Matplotlib (which focuses on plotting and data visualization) but features Pandas' data structures. Seaborn is often used in ML projects because it can generate plots of learning data. Of all the Python libraries, it produces the most aesthetically pleasing graphs and plots, making it an effective choice if you'll also use it for marketing and data analysis.

- **Pandas**
  Pandas is another Python library that is built on top of NumPy, responsible for preparing high-level data sets for machine learning and training. It relies on two types of data structures, one-dimensional (series) and two-dimensional (Data Frame). This allows Pandas to be applicable in a variety of industries including finance, engineering, and statistics. Unlike the slow-moving animals themselves, the Pandas library is quick, compliant, and flexible.

- Class imbalance problem

The first challenge we hit upon exploring the data, is class imbalanced problem. Imbalance data will lead to a bad accuracy of a model. To achieve better accuracy, we'll balance the data by using Smote Over Sampling Method.
But in this dataset I found data is balanced so that I will not be using this.

Model/s Development and Evaluation

- Run and evaluate selected models

Let's select our regression model for this project:

- LinearRegression
- RandomForestRegressor
- KNeighborsRegressor
- SVR
- DecisionTreeRegressor

- Testing of Identified Approaches (Algorithms)

Using for loop I will be using all above mentioned model at one go.

```
In [648]: from sklearn.metrics import r2_score
          from sklearn.linear_model import LinearRegression
          from sklearn.ensemble import RandomForestRegressor
          from sklearn.neighbors import KNeighborsRegressor
          from sklearn.svm import SVR
          from sklearn.tree import DecisionTreeRegressor
          from sklearn.metrics import mean_squared_error,mean_absolute_error
          from sklearn.model_selection import train_test_split
```

```
In [649]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=45)
```

```
In [650]: lr=LinearRegression()
          knr=KNeighborsRegressor()
          dtr=DecisionTreeRegressor()
          svr=SVR()
          rf=RandomForestRegressor(n_estimators=100,random_state=30)
          model=[lr,knr,dtr,svr,rf]
          for m in model:
              m.fit(x_train,y_train)
              predm=m.predict(x_test)
              print("predicted price:",m,predm)
              print("actual price:",m,y_test)
              print('r2_score:', r2_score(y_test,predm))
              print('error:')
              print('mean absolute error:',m,mean_absolute_error(y_test,predm))
              print('mean squared error:',m,mean_squared_error(y_test,predm))
              print('root mean squarred error:',m,np.sqrt(mean_squared_error(y_test,predm)))
```

```
 126757.65506455  77714.81280898 165437.73643487 168052.52334462
 254814.26437899 255088.70892169 275059.9777994  207597.22374773
 182694.4112342  205629.62899106 137004.97938572 184865.45242038
 201685.75717325 234290.20537343 157907.69500478 189036.97097215
 209624.49909182 223268.69409609]
actual price: LinearRegression() 208     173000
1052     140000
```

➢ Key Metrics for success in solving problem under consideration

Selection of a model requires evaluation and evaluation requires a good metric. This is indeed important. If we optimize a model based on incorrect metric, then, our model might not be suitable for the business goals.

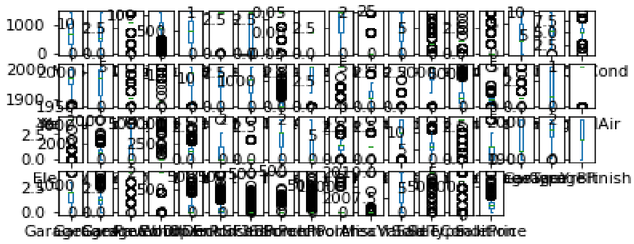
We have a number of metrics, for example mean absolute error ,mean squared error, root mean squared error etc.

```
print('error:')
print('mean absolute error:',m,mean_absolute_error(y_test,predm))
print('mean squared error:',m,mean_squared_error(y_test,predm))
print('root mean squarred error:',m,np.sqrt(mean_squared_error(y_test,predm)))
```

➢ Visualizations

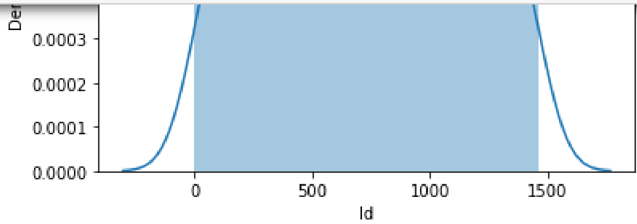
For better understanding of outliers, I have used boxplot.

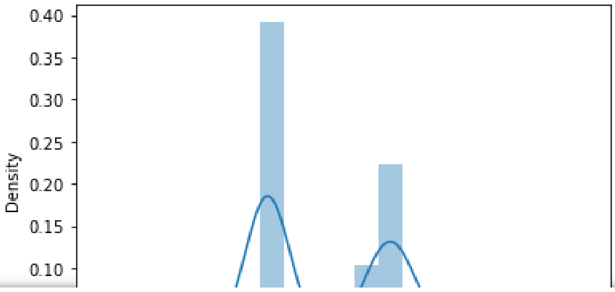
```
583]: df.plot(kind='box',subplots=True,layout=(6,18))

583]: Id                  AxesSubplot(0.125,0.772143;0.036215x0.107857)
      MSSubClass       AxesSubplot(0.168458,0.772143;0.036215x0.107857)
      MSZoning         AxesSubplot(0.211916,0.772143;0.036215x0.107857)
      LotFrontage      AxesSubplot(0.255374,0.772143;0.036215x0.107857)
      LotArea          AxesSubplot(0.298832,0.772143;0.036215x0.107857)
                                             ...
      MiscVal          AxesSubplot(0.559579,0.383857;0.036215x0.107857)
      YrSold           AxesSubplot(0.603037,0.383857;0.036215x0.107857)
      SaleType         AxesSubplot(0.646495,0.383857;0.036215x0.107857)
      SaleCondition    AxesSubplot(0.689953,0.383857;0.036215x0.107857)
      SalePrice        AxesSubplot(0.733411,0.383857;0.036215x0.107857)
      Length: 69, dtype: object
```



For better understanding of skewness, I have used distribution plot.

In [584]:
```
for i in df.columns:
    plt.figure()
    sns.distplot(df[i])
```





CONCLUSION

Key aspects of building successful classifier are:

- Selecting correct data according to the purpose or problem statement.
- Proper processing and understanding of the data
- Selecting the model and optimizing the model.

In this project I have dealt with outliers and using z score I removed those outliers for better accuracy.

I have used simple imputer for filling up missing values.

I have used label encoder for encoding object data type into int datatype as machine doesn’t understand object type data.

I have used lasso to be the best fit. It is giving 91% accuracy.