PFA Housing Project

Introduction

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

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 modeling, Market mix modeling, 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

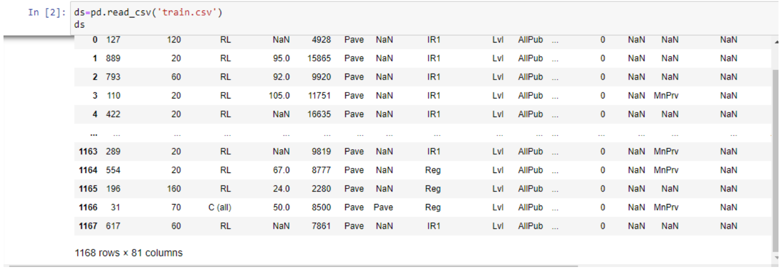
Business Goal:

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.

I hope you have understood the above problem statement about predicting the house prices. Now, I will take you through a machine learning project on House Price prediction with Python

Initially we have imported the data set and all the necessary libraries which are required

Data set consist of 1168 rows and 81 columns



```
In [2]: ds=pd.read_csv('train.csv')
        ds
```

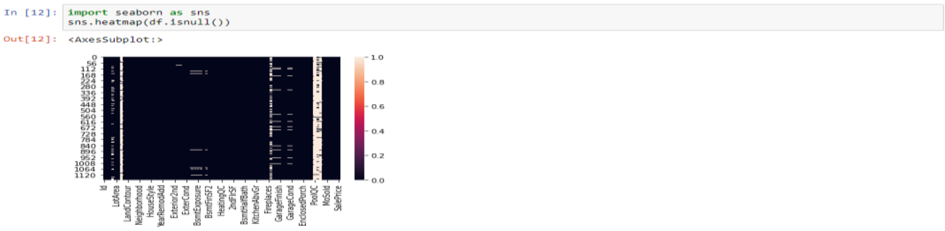
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1163	289	20	RL	NaN	9819	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN
1164	554	20	RL	67.0	8777	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN
1165	196	160	RL	24.0	2280	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN
1166	31	70	C (all)	50.0	8500	Pave	Pave	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN
1167	617	60	RL	NaN	7861	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN

1168 rows × 81 columns

Once the data set is imported we went for Exploratory Data Analysis(EDA) in which we observed the shape,types of columns,info,and also the unique values from the sales price

There we saw that sales column is our dependent variable and is continous in nature, thus it is a Linear Regression problem.

We also checked the missing values in the data set for which we plotted the heat map

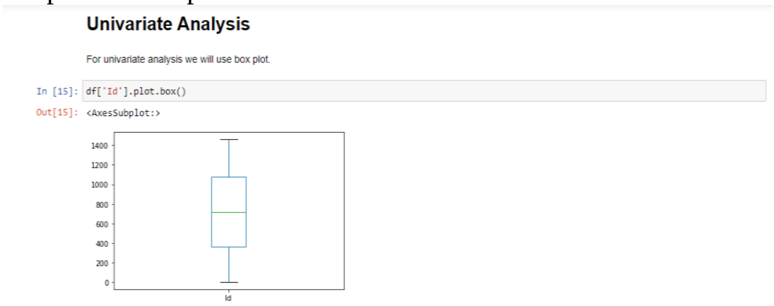


```
In [12]: import seaborn as sns
         sns.heatmap(df.isnull())
Out[12]: <AxesSubplot:>
```

After the Statistical summary is done there were some observations seen

1. We can see that in the count function many columns have different values, which clearly means many null values are present in the columns.
2. The difference between mean and median is not similar.
3. There is a small difference in 75% and max column in many columns like overallQual, OverallCond, YearBlt etc. which shows that there are few outliers present in the columns.
4. There is a large difference in 75% and max column in many columns like LotArea, MasVnArea, BsmtFinSF1, MiscVal etc. which shows that there are few outliers present in the columns.

Again importing the necessary libraries we went undergone for the **Univariate Analysis** for whichwe plotted box plots with each column



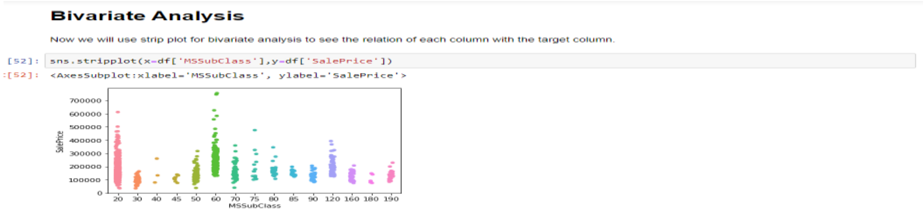
Univariate Analysis

For univariate analysis we will use box plot.

```
In [15]: df['Id'].plot.box()
Out[15]: <AxesSubplot:>
```

Bivariate Analysis

We plotted with the help of scatterplot which shows the relation of each column with the target column



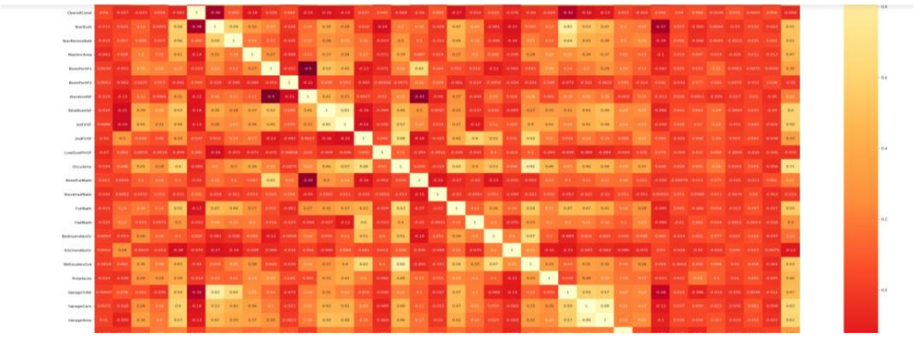
Bivariate Analysis

Now we will use strip plot for bivariate analysis to see the relation of each column with the target column.

```
[52]: sns.stripplot(x=df['MSSubClass'],y=df['SalePrice'])
:[52]: <AxesSubplot:xlabel='MSSubClass', ylabel='SalePrice'>
```

Multivariate Analysis

To check the co-relation so we plotted a heat map to check the co-relation



Key Observations

Now we can clearly indentify the correlation of independent variable with the target variable"SalePrice" . The variables which has values less than 0.01 have weak correlation with the target variable, while the variables which has value more than have strong relation with the target column.

After checking the co-relation we performed 3 main steps

1. Cheking the Skewness
2. Data Cleaning
3. Label Encoding

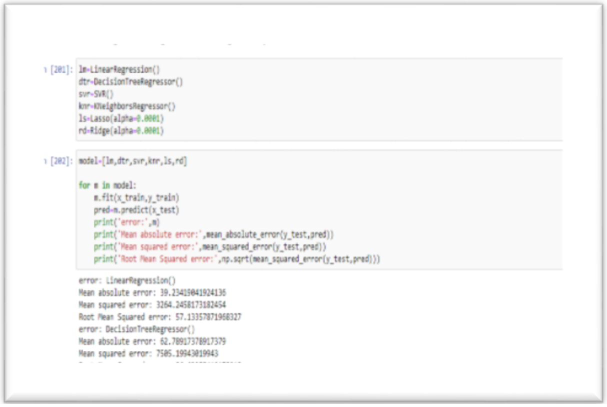
We checked for the skewness first then the data was cleaned in which we dropped the unnecessary columns which were not required and as it was observed that many columns were present in string format hence we need to convert them to neumeric format for that we performed label encoding

Removing Outliers

As we have seen from the univariate and bivariate analysis we have many outliers.But while removinf the outliers we are losing high data so we are not going to implement any method to remove outlier So now we will remove skewness than some of the outliers will also remove

Once the removing of outliers and skewness is done the data was split into train-test-split for further

Then we selected the best Algorithms using multiple Models



```
[201]: lm=LinearRegression()
       dtr=DecisionTreeRegressor()
       svr=SVR()
       knr=KNeighborsRegressor()
       ls=Lasso(alpha=0.0001)
       rd=Ridge(alpha=0.0001)
```

```
[202]: model=[lm,dtr,svr,knr,ls,rd]

       for m in model:
           m.fit(x_train,y_train)
           pred=m.predict(x_test)
           print('error:',m)
           print('Mean absolute error:',mean_absolute_error(y_test,pred))
           print('Mean squared error:',mean_squared_error(y_test,pred))
           print('Root Mean Squared error:',np.sqrt(mean_squared_error(y_test,pred)))
```

```
error: LinearRegression()
Mean absolute error: 39.23419041924136
Mean squared error: 3264.2458173182454
Root Mean Squared error: 57.13357871968327
error: DecisionTreeRegressor()
Mean absolute error: 62.78917378917379
Mean squared error: 7585.19943019943
```

We also undergone cross validation and hyperparameter tuning
Hyper parameter tuning was done with SVR using GRIDSEARCH which gave the score of 88%

Hyperparameter Tuning was done for Decision Tree Regressor which gave the score of 70%
Hyper parameter Tuning was done also for the K neighbor Regressor which gave the accuracy score of 81%

Conclusion

Looking all the observations of the accuracy score we decided to final the SVR as our best model as this was giving us the best accuracy score from all of this