FLIP ROBO

HOUSING: PRICE PREDICTION

Submitted by:

PURNIMA PATI

ACKNOWLEDGMENT

A house is usually the single largest purchase an individual will make in their lifetime. Such significant purchase warrants being well-informed about what a house's selling price should be; for the buyer, as well as the seller or real estate broker involved. The power of machine learning provides us with the tools we need to look at a large data set and spit out a predicted value, which was our main goal in this project. Using a dataset containing information on houses in Ames, Iowa, our team leveraged different machine learning techniques to predict sale prices based on both practical intuition and those observed through our exploratory data analysis and model fitting processes.

To complete this project I have gone through various case studies and project reports , how house price prediction is useful for the public & real-estate market. I have gone through some articles available in google out of which one is Author(s): Zhou, YichenAdvisor(s): Wu, Yingnian, and the other is below link which I followed to complete this project.
https://escholarship.org/uc/item/3ft2m7z5.I wish to express our sincere thanks to the above people, without whom I would not have been able to complete this project.

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 on 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 to enter the market. You are required to build a regression model using regularisation in order to predict the actual value of the prospective properties and decide whether to invest in them or not.

We 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.

- ## Conceptual Background of the Domain Problem

  In this example we will build a predictive model to predict house price (price is a number from some defined range, so it will be regression task). For example, you want to sell a house and you don't know the price which you can take — it can't be too low or too high. To find house price you usually try to find similar properties in your neighborhood and based on gathered data you will try to assess your house price. We will do something similar, but with Machine Learning methods

  The objective of the project is to perform data visualization techniques to understand the insight of the data. Machine learning often required to getting the understanding of the data and its insights. This project aims apply various python tools to get a visual understanding of the data and clean it to make it ready to apply machine learning and deep learning models on it.

- ## Review of Literature

  This study proposes a performance comparison between machine learning regression algorithms and Artificial Neural Network (ANN). The regression algorithms used in this study are Multiple linear, Least Absolute Selection Operator (Lasso), Ridge, Random Forest. Moreover, this study attempts to analyse the correlation between variables to determine the most important factors that affect house prices There are two datasets used in this study which called public and local. They contain house prices. The accuracy of the prediction is evaluated by checking the root square and root mean square error scores of the training model. The test is performed after applying the required pre-processing methods and splitting the data into two parts. However, one part will be used in the training and the other in the test phase.I have also presented a binning strategy that improved the accuracy of the models. This thesis attempts to show that GradientBoosting Regressor when using the public dataset in training. The correlation graphs show the variables' level of dependency.

Motivation for the Problem Undertaken

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. In this project, house prices will be predicted given explanatory variables that cover many aspects of residential houses. The goal of this project is to create a regression model that are able to accurately estimate the price of the house given the features.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  A statistical model is a mathematical representation (or mathematical model) of observed data.

  When data analysts apply various statistical models to the data they are investigating, they are able to understand and interpret the information more strategically. Rather than sifting through the raw data, this practice allows them to identify relationships between variables, make predictions about future sets of data, and visualize that data so that non-analysts and stakeholders can consume and leverage it.

  Multiple Linear Regression (MLR) is a supervised technique used to estimate the relationship between one dependent variable and more than one independent variables. Identifying the correlation and its cause-effect helps to make predictions by using these relations [4]. To estimate these relationships, the prediction accuracy of the model is essential; the complexity of the model is of more interest. However, Multiple Linear Regression is prone to many problems such as multicollinearity, noises, and overfitting, which effect on the prediction accuracy.Here I have used Linear Regression, RandomForestRegression & GradientBoostingRegressor.

- ## Data Sources and their formats

  Our data comes from "House Prices: Prediction". It contains *1168* training data points and 81 features that might help us predict the selling price of a house.

We will load the dataset into a Pandas data frame

We're going to predict the sale price column ($ USD), let's start with it, Most of the density lies between 100k and 250k, but there appears to be a lot of outliers on the pricier side.Next, let's have a look at the greater living area (square feet) against the sale price. You might've expected that larger living area should mean a higher price. This chart shows you're generally correct. But what are those 2–3 "cheap" houses offering huge living area?One column you might not think about exploring is the "TotalBsmtSF" — Total square feet of the basement area. Intriguing, isn't it? The basement area seems like it might have a lot of predictive power for our model.Ok, last one. Let's look at "OverallQual" — overall material and finish quality. Of course, this one seems like a much more subjective feature, so it might provide a bit different perspective on the sale price. Everything seems fine for this one, except that when you look to the right things start getting much more nuanced. Will that "confuse" our model?We have a more general view on the *top 8* correlated features with the sale price

- Data Preprocessing Done

  Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, greater is the reliance on the produced results. **Incomplete, noisy, and inconsistent data** are the properties of large real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise and resolving inconsistencies.

- **Incomplete data** can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions.
- There are many possible reasons for **noisy data** (having incorrect attribute values). The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry. Errors in data transmission can also occur. Incorrect data may also result from inconsistencies in naming conventions or data codes used, or inconsistent formats for input fields, such as date.

  There are a number of data preprocessing techniques available such as,

1. **Data Cleaning**
2. **Data Integration**
3. **Data Transformation**
4. **Data Reduction**

Data cleaning can be applied to filling in missing values, remove noise, resolving inconsistencies, identifying and removing outliers in the data.

Data integration merges data from multiple sources into a coherent data store, such as a data warehouse.

Data transformations, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.

Data reduction can reduce the data size by eliminating redundant features, or clustering, for instance.

- Data Inputs- Logic- Output Relationships

  Data Science is the process of making some assumptions and hypothesis on the data, and testing them by performing some tasks. Initially we could make the following intuitive assumptions for each feature.We will start by creating a scatterplot matrix that will allow us to visualize the pair-wise relationships and correlations between the different features.It is also quite useful to have a quick overview of how the data is distributed and wheter it cointains or not outliers.
  We are going to create now a correlation matrix to quantify and summarize the relationships between the variables.

- State the set of assumptions (if any) related to the problem under consideration

  This study will not cover all regression algorithms; instead, it is focused on the chosen algorithm, starting from the basic regression techniques to the advanced ones. Likewise, the artificial neural network that has many techniques and a wide area and several training methods that do not fit in this study.

- Hardware and Software Requirements and Tools Used

  Hardware requirements
  Operating system- Windows 10
  Processor- dual core 2.4 GHz (i5 series Intel processor
  or equivalent AMD)
  RAM-4GB

  Software Requirements
  Python
  PIP 2.7
  Jupyter Notebook
  Chrome

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  This study has been organised through theoretical research and practical implementation of regression algorithms. The theoretical part relies on peer-reviewed articles to answer the research questions, which is going to be detailed . The practical part will be performed according to the design described below and detailed furthermore .

  The experiment is done to pre-process the data and evaluate the prediction accuracy of the models. The experiment has multiple stages that are required to get the prediction results. These stages can be defined as: - Pre-processing: both datasets will be checked and pre-processed using different methods.These methods have various ways of handling data. Thus, the pre-processing is done on multiple iterations where each time the accuracy will be evaluated with the used combination. - Data splitting: dividing the dataset into two parts is essential to train the model with one and use the other in the evaluation. The dataset will be split 75% for training and 25% for testing. - Evaluation: the accuracy of both datasets will be evaluated. Performance: alongside the evaluation metrics, the required time to train the model will be measured to show the algorithm vary in terms of time. - Correlation: correlation between the available features and house price will be evaluated using the Pearson Coefficient Correlation to identify whether the features have a negative, positive or zero correlation with the house price.

- Testing of Identified Approaches (Algorithms)

  The algorithms used in this study have different properties that will be used during the implementation. The experiment is done with jupyter notebook Python as a programming language. However, in all

algorithms, the data is split into four variables, namely, X_train, X_test, y_train, and y_test, by using train_test_split class from the library sklearn.model_selection. In addition, in all algorithms, the train_test_split class takes as parameters the independent variables, which is the data, the dependent variable, which is the SalePrice.
The properties and design of each algorithm are as below:

Regression Model

Linear Regression is a machine learning algorithm based on supervised learning.
It performs a regression task. Regression models a target prediction value based on
independent variables.
It is mostly used for finding out the relationship between variables and forecasting.

Random Forest Regression Model

A Random Forest is an ensemble technique capable of performing both regression and
classification tasks with the use of multiple decision trees and a technique
called Bootstrap Aggregation, commonly known as bagging.
Bagging, in the Random Forest method, involves training each decision tree on a
different data sample where sampling is done with replacement.
The basic idea behind this is to combine multiple decision trees in determining the
final output rather than relying on individual decision trees.

Gradient boosting

Gradient boosting is **a machine learning technique used in regression and classification tasks**, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.

- Run and Evaluate selected models

Linear Regression

```
#Train the model
from sklearn import linear_model
model = linear_model.LinearRegression()
```

```
#Fit the model
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
#Prediction
print("Predict value " + str(model.predict([X_test[142]])))
print("Real value " + str(y_test[142]))

```

```
Predict value [11.76773873]
Real value 11.824087219587643
```

```
#Score/Accuracy
print("Accuracy --> ", model.score(X_test, y_test)*100)

```

```
Accuracy -->  87.23405014711032
```

RandomForestRegression

```
#Train the model
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=1000)

```

```
#Fit
model.fit(X_train, y_train)

```

```
RandomForestRegressor(n_estimators=1000)
```

```
#Score/Accuracy
print("Accuracy --> ", model.score(X_test, y_test)*100)

```

```
Accuracy -->  86.42341872256421
```

GradientBoostingRegressor

```
#Train the model
from sklearn.ensemble import GradientBoostingRegressor
GBR = GradientBoostingRegressor(n_estimators=100, max_depth=4)

```

```
#Fit
GBR.fit(X_train, y_train)

```

```
GradientBoostingRegressor(max_depth=4)
```

```
print("Accuracy --> ", GBR.score(X_test, y_test)*100)
```

- Key Metrics for success in solving problem under consideration

Will start by creating a scatterplot matrix that will allow us to visualize the pair-wise relationships and correlations between the different features.

Corralation between train attributes

```
#Separate variable into new dataframe from original dataframe which has only numerical values
#there is 38 numerical attribute from 81 attributes
train_corr = train.select_dtypes(include=[np.number])
```

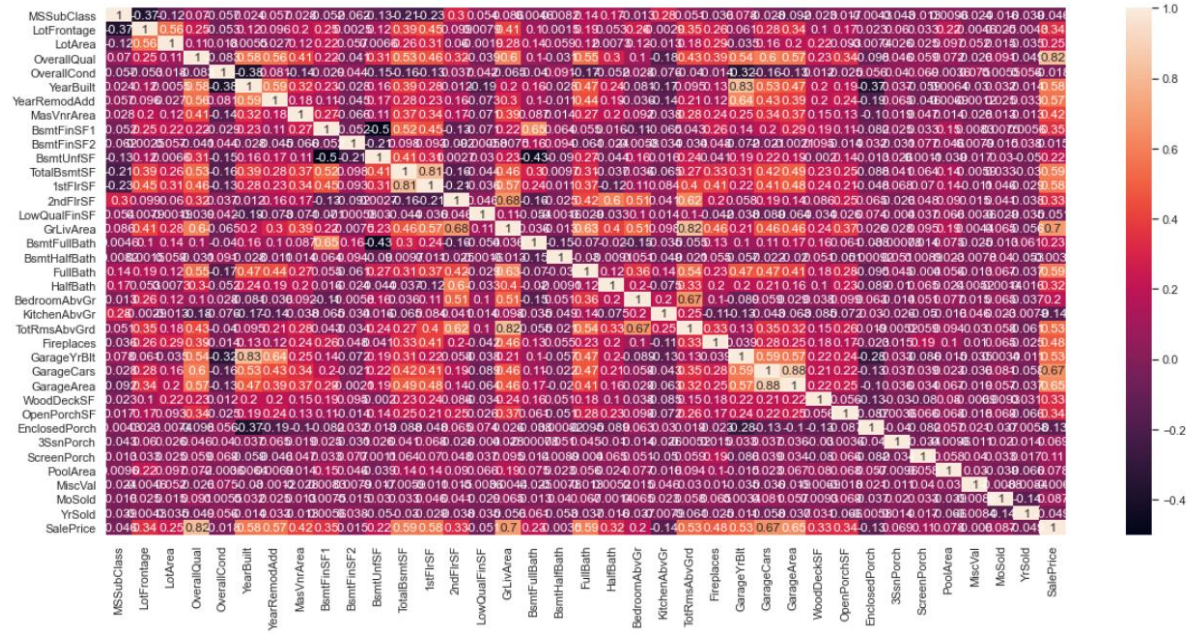
```
train_corr.shape
```

```
(1168, 38)
```

```
#Delete Id because that is not need for corralation plot
del train_corr['Id']
```

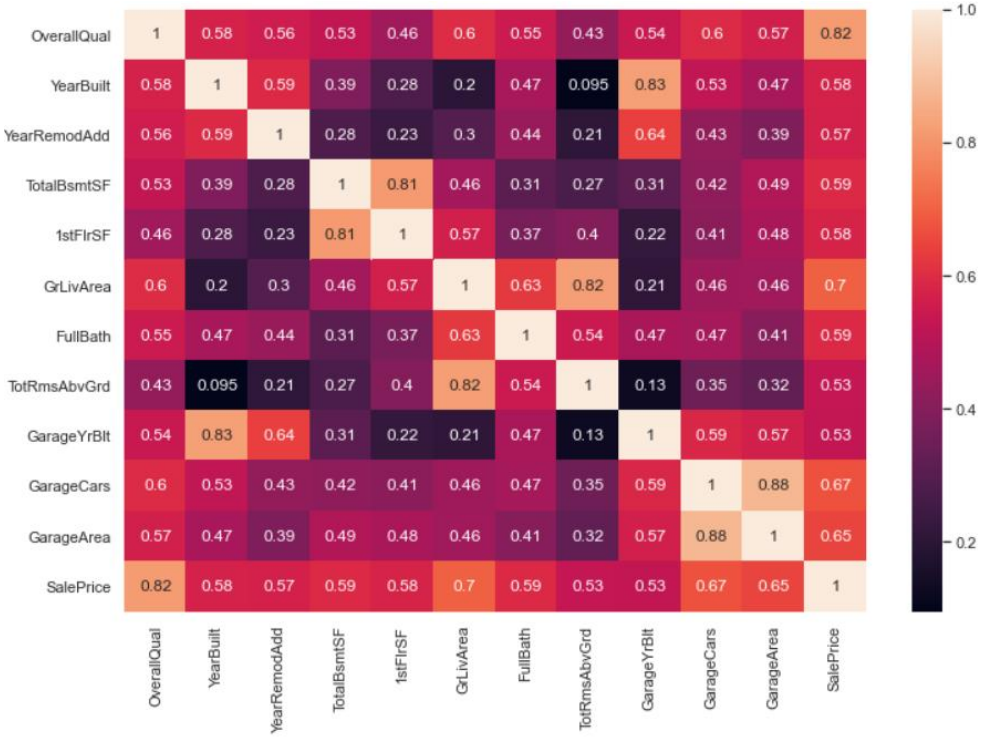
```
#Coralation plot
corr = train_corr.corr()
plt.subplots(figsize=(20,9))
sns.heatmap(corr, annot=True)
```

<AxesSubplot:>



Top 50% Corralation train attributes with sale-price

```
top_feature = corr.index[abs(corr['SalePrice']>0.5)]
plt.subplots(figsize=(12, 8))
top_corr = train[top_feature].corr()
sns.heatmap(top_corr, annot=True)
plt.show()
```



Here OverallQual is highly correlated with target feature of saleprice by 82%

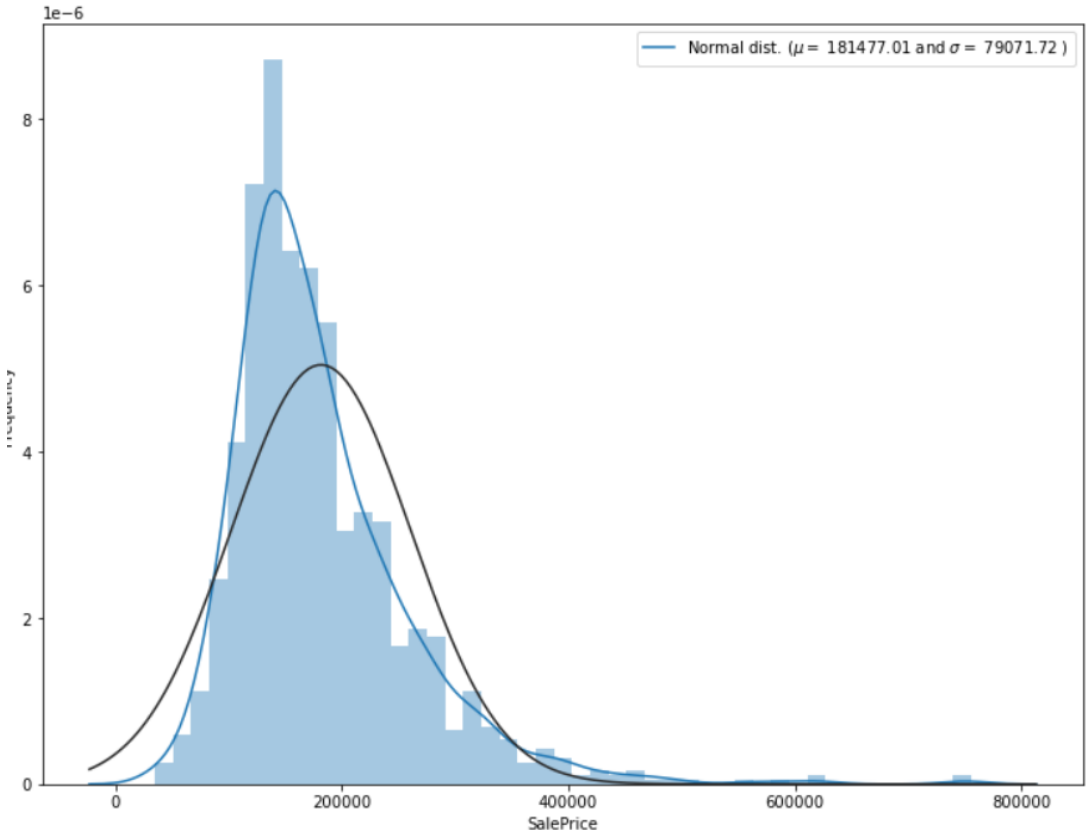
- Visualizations

Basic Visualization

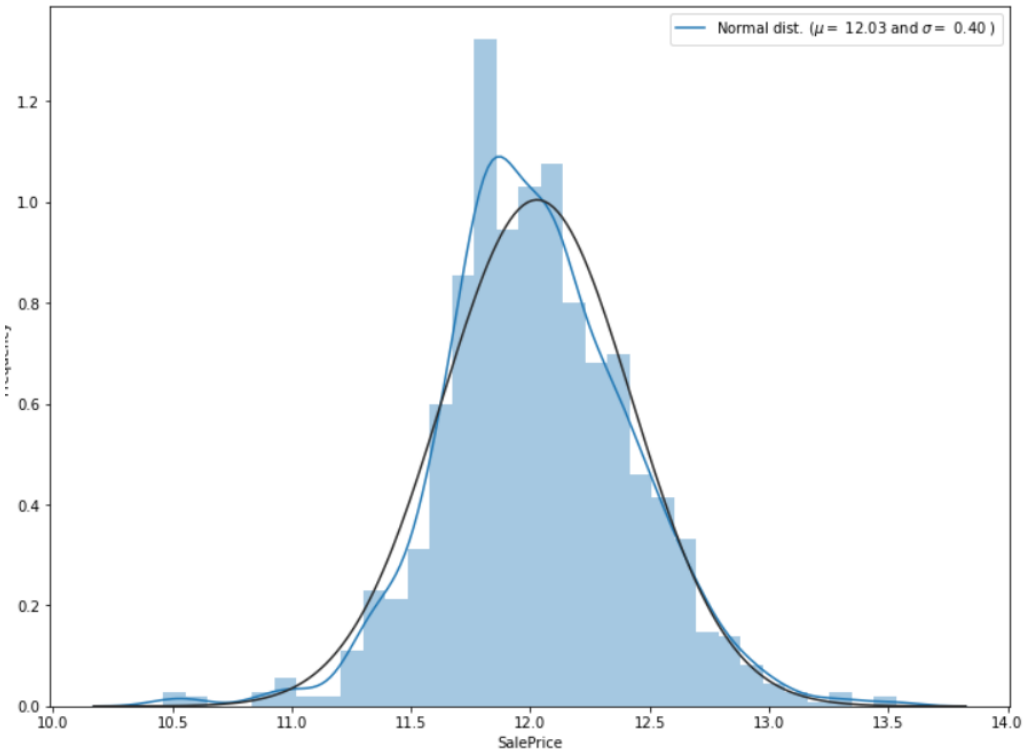
- ***Correlation among variables***
- ***Various datatypes columns***
- ***Heatmap***
- ***SalePrice: the variable we're trying to prediction***
- ***Visualising missing values***

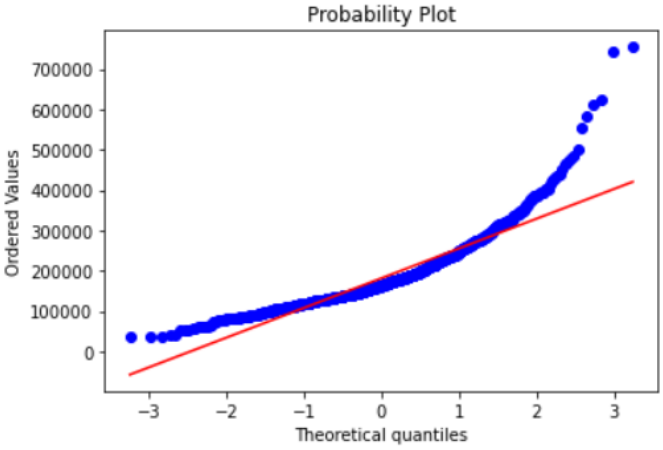
Target variable Some analysis on target variable

```
plt.subplots(figsize=(12,9))
sns.distplot(train['SalePrice'], fit=stats.norm)

# Get the fitted parameters used by the function

(mu, sigma) = stats.norm.fit(train['SalePrice'])

# plot with the distribution

plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)], loc='best')
plt.ylabel('Frequency')

#Probablity plot

fig = plt.figure()
stats.probplot(train['SalePrice'], plot=plt)
plt.show()
```




```
#we use log function which is in numpy
train['SalePrice'] = np.log1p(train['SalePrice'])

#Check again for more normal distribution

plt.subplots(figsize=(12,9))
sns.distplot(train['SalePrice'], fit=stats.norm)

# Get the fitted parameters used by the function

(mu, sigma) = stats.norm.fit(train['SalePrice'])

# plot with the distribution

plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f} )'.format(mu, sigma)], loc='best')
plt.ylabel('Frequency')

#Probablity plot

fig = plt.figure()
stats.probplot(train['SalePrice'], plot=plt)
plt.show()
```





This target varibale is right skewed. Now, we need to tranform this variable and make it normal distribution.

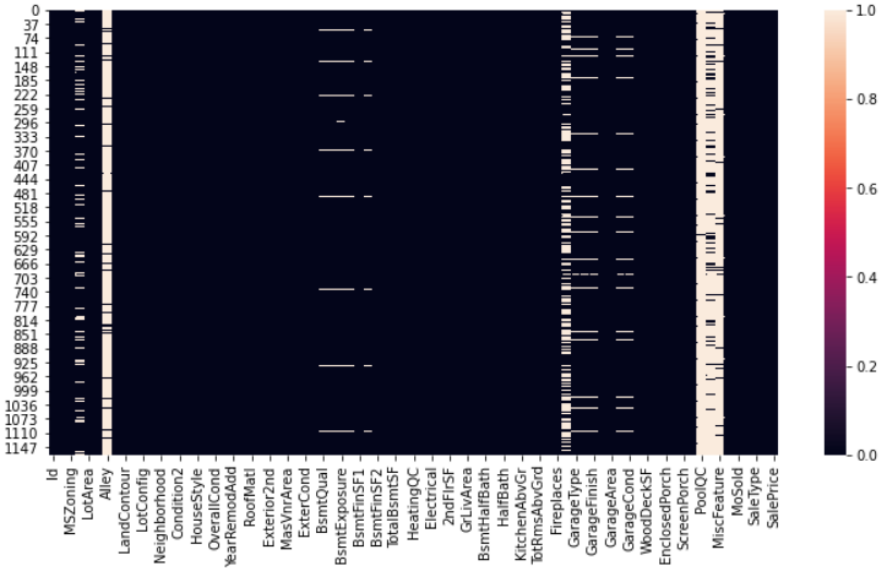
Here we use log for target variable to make more normal distribution

Check the missing values

```
:
#Let's check if the data set has any missing values.
train.columns[train.isnull().any()]
```

```
:
Index(['LotFrontage', 'Alley', 'MasVnrType', 'MasVnrArea', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2',
       'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish',
       'GarageQual', 'GarageCond', 'PoolQC', 'Fence', 'MiscFeature'],
      dtype='object')
```

```
:
#plot of missing value attributes
plt.figure(figsize=(12, 6))
sns.heatmap(train.isnull())
plt.show()
```




```
#missing value counts in each of these columns
Isnull = train.isnull().sum()/len(train)*100
Isnull = Isnull[Isnull>0]
Isnull.sort_values(inplace=True, ascending=False)
Isnull
```

```
PoolQC          99.400685
MiscFeature     96.232877
Alley           93.407534
Fence           79.708904
FireplaceQu     47.174658
LotFrontage     18.321918
GarageType       5.479452
GarageYrBlt      5.479452
GarageFinish     5.479452
GarageQual       5.479452
GarageCond       5.479452
BsmtExposure     2.654110
BsmtFinType2     2.654110
BsmtCond         2.568493
BsmtFinType1     2.568493
BsmtQual         2.568493
MasVnrArea       0.599315
MasVnrType       0.599315
dtype: float64
```

Visualising missing values

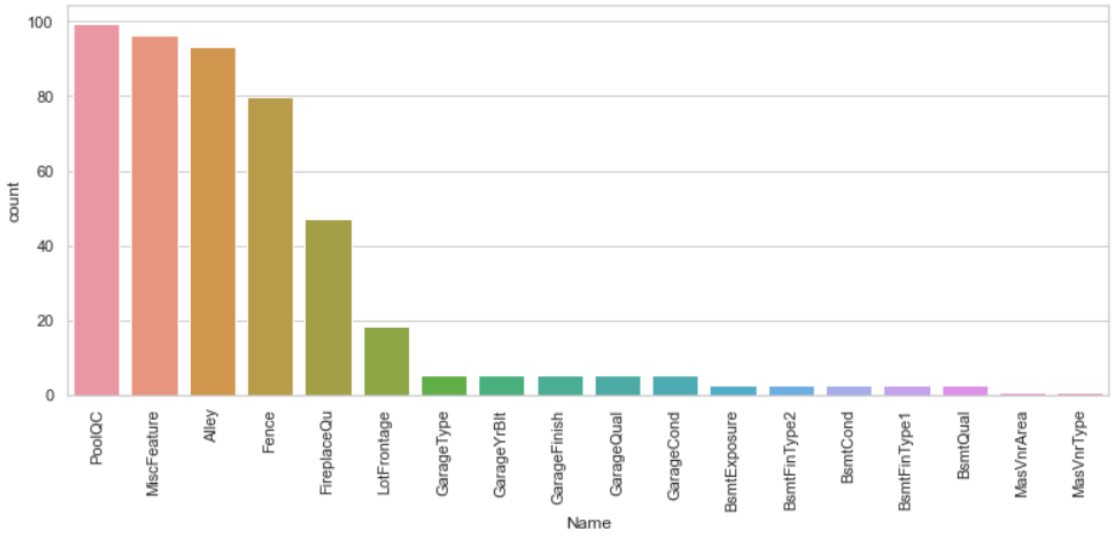
```
#Convert into dataframe
Isnull = Isnull.to_frame()
```

```
Isnull.columns = ['count']
```

```
Isnull.index.names = ['Name']
```

```
Isnull['Name'] = Isnull.index
```

```
#plot Missing values
plt.figure(figsize=(13, 5))
sns.set(style='whitegrid')
sns.barplot(x='Name', y='count', data=Isnull)
plt.xticks(rotation = 90)
plt.show()
```

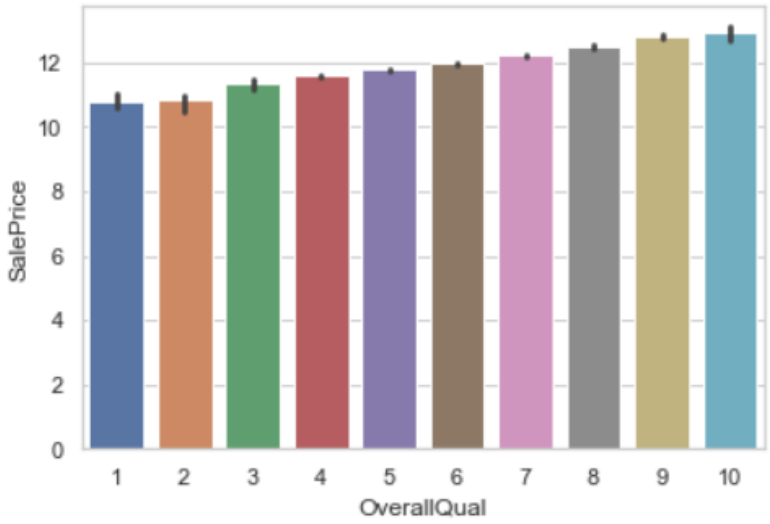



```
1 #unique value of OverallQual
2 train.OverallQual.unique()
```

```
array([ 6,  8,  7,  5,  9,  1,  2,  4,  3, 10], dtype=int64)
```

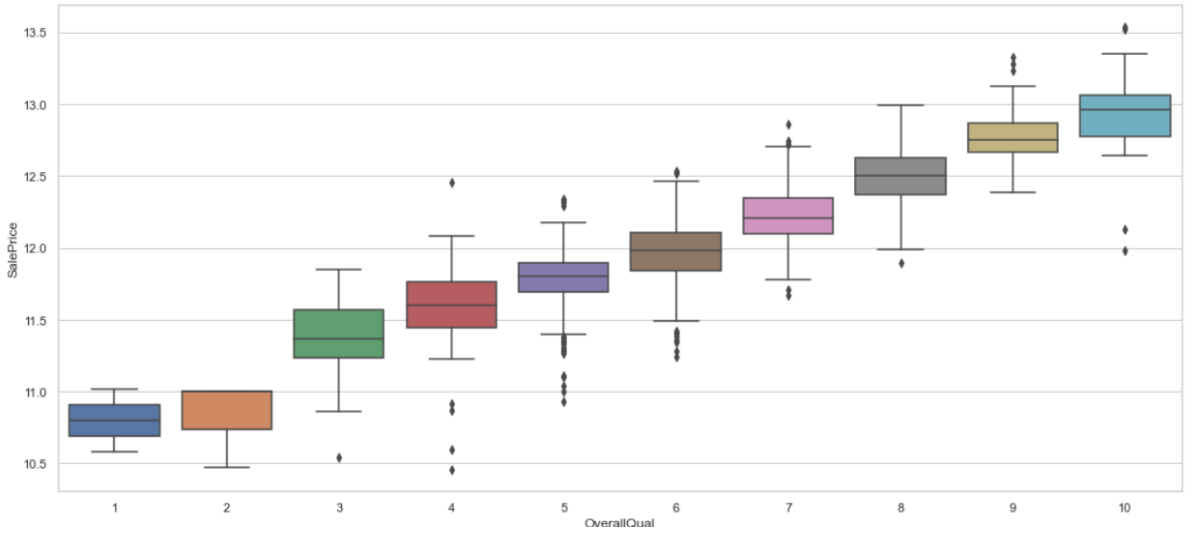
```
1 sns.barplot(train.OverallQual, train.SalePrice)
```

```
<AxesSubplot:xlabel='OverallQual', ylabel='SalePrice'>
```



```
#boxplot
plt.figure(figsize=(18, 8))
sns.boxplot(x=train.OverallQual, y=train.SalePrice)
```

```
<AxesSubplot:xlabel='OverallQual', ylabel='SalePrice'>
```



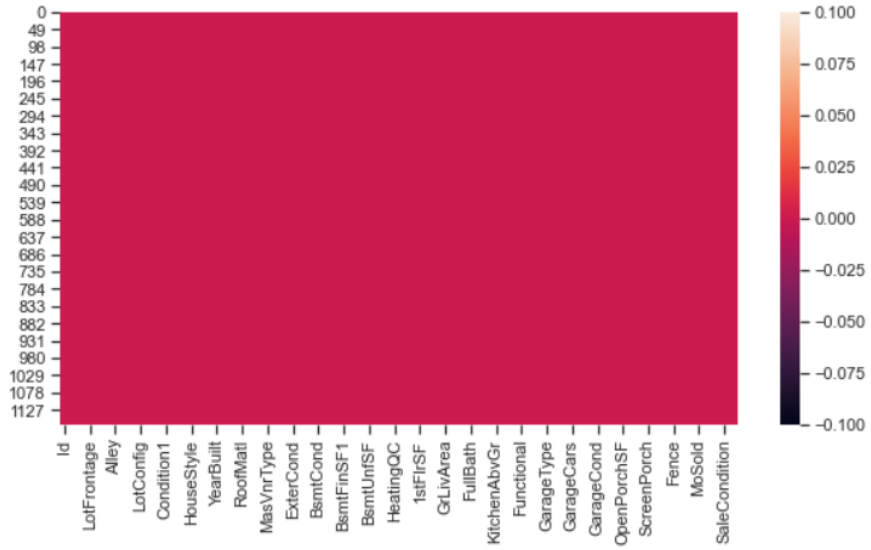

```
col = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt']
sns.set(style='ticks')
sns.pairplot(train[col], size=3, kind='reg')
```

- # Interpretation of the Results

The results were interpreted from the visualizations, preprocessing and modelling.

```
#Checking there is any null value or not
plt.figure(figsize=(10, 5))
sns.heatmap(train.isnull())

```

```
<AxesSubplot:>
```



Now, there is no any missing values

CONCLUSION

- Key Findings and Conclusions of the Study

  So, our Aim is achieved as we have successfully ticked all our parameters as mentioned in our Aim Column. It is seen that circle rate is the most effective attribute in predicting the house price and that the GradientBoosting Regressor is the most effective model for our Dataset with accuracy score 89.04002619286328

- Learning Outcomes of the Study in respect of Data Science

  . Linear Regression is a machine learning algorithm based on supervised learning.

  • It performs a regression task. Regression models a target prediction value based on

  independent variables.

  • It is mostly used for finding out the relationship between variables and forecasting.

  A Random Forest is an ensemble technique capable of performing both regression and

  classification tasks with the use of multiple decision trees and a technique

  called Bootstrap Aggregation, commonly known as bagging.

• Bagging, in the Random Forest method, involves training each decision tree on a

different data sample where sampling is done with replacement.

• The basic idea behind this is to combine multiple decision trees in determining the

final output rather than relying on individual decision trees.

Gradient boosting

Gradient boosting is **a machine learning technique used in regression and classification tasks**, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.

- Limitations of this work and Scope for Future Work

The request contains a list of features, that matches the public dataset's features, that is desired to be available when the data is sent. There is no guarantee that the data will be available in time nor contains the exact requested list of features. Thus, there might be a risk that the access will be denied or delayed. If so, the study will be accomplished based only on the public dataset. Moreover, this study will not cover all regression algorithms; instead, it is focused on the chosen algorithm, starting from the basic regression techniques to the advanced ones