CHAPTER 1 :- ABSTRACT

The phenomenon of the falling or rising of the house prices has attracted interest from the researcher as well as many other interested parties. House prices increase every year, so there is a need for a system to predict house prices in the future. House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location. This research aims to predict house prices in Mumbai city with Machine learning algorithms. The previous research works used various regression techniques to address the question of the changes house price. This work considers the issue of changing house price as a classification problem and applies machine learning techniques to predict whether house prices will rise or fall. This work applies various feature selection techniques such as variance influence factor, Information value, principle component analysis and data transformation techniques such as outlier and missing value treatment as well as box-cox transformation techniques. The goal of the project is to create a system that can provide users with location-specific predictions and trends.

CHAPTER 2 :- INTRODUCTION

The aim of this document is to gather and analyze and give an in-depth insight of the complete House price prediction system by defining the problem statement in detail. Nevertheless, accurate prediction of house prices has been always a fascination for the buyers, sellers and for the bankers also. The detailed requirements of the House price prediction system are provided in this document. The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers. Also, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops. The diversity of features makes it challenging to estimate an adequate market price. Apart from providing a summary of the important features of the house, the house description is also a means of raising curiosity in the reader, or in other words to persuade the person. It is possible that there are certain word sequences in the natural language text that seduce potential buyers more than others. Therefore, there might be a relation between the language used in the description and the price of the property. This comparison does not focus primarily on the house characteristics, but on all words within the description.

CHAPTER 3 :- LITERATURE SURVEY

The previous research papers have used different machine learning algorithms to develop this project which include :- Naïve Bayes algorithm, Random Forest and Support Vector Regression. In this project, regression algorithm is used to implement it. The previous papers used hedonic pricing model to estimate house prices in the past decade. The project uses the dataset that consists of 81 attributes and 1460 entries. The past papers used Naïve Bayes algorithm which is a supervised machine-learning algorithm that uses the Bayes' Theorem, which assumes that features are statistically independent.

The theorem relies on the naive assumption that input variables are independent of each other, i.e. there is no way to know anything about other variables when given an additional variable. Regardless of this assumption, it has proven itself to be a classifier with good results. Naive Bayes Classifiers rely on the Bayes' Theorem, which is based on conditional probability or in simple terms, the likelihood that an event (A) will happen given that another event (B) has already happened.

Essentially, the theorem allows a hypothesis to be updated each time new evidence is introduced. Support vector machines are linear discriminant functions (classifier) with the maximum margin is the best. The margin is defined as the width that the boundary could be increased by, before hitting a data point .Random Forests are ensemble classifiers constructed from of a set of Decision Trees, with the output of the classifier being the mode of the output of the Decision Trees. Random Forests combine the "bagging" idea of Breiman with the idea of random selection of features. The algorithm for inducing a Random Forest was developed by Leo Breiman and Adele Cutler.

The artificial neural networks use neurons or perceptrons as the basic units. These perceptrons use a vector of real valued inputs. These inputs are always having a linear combination between themselves. Regression algorithm 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. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

CHAPTER 4 :- PROPOSED SYSTEM

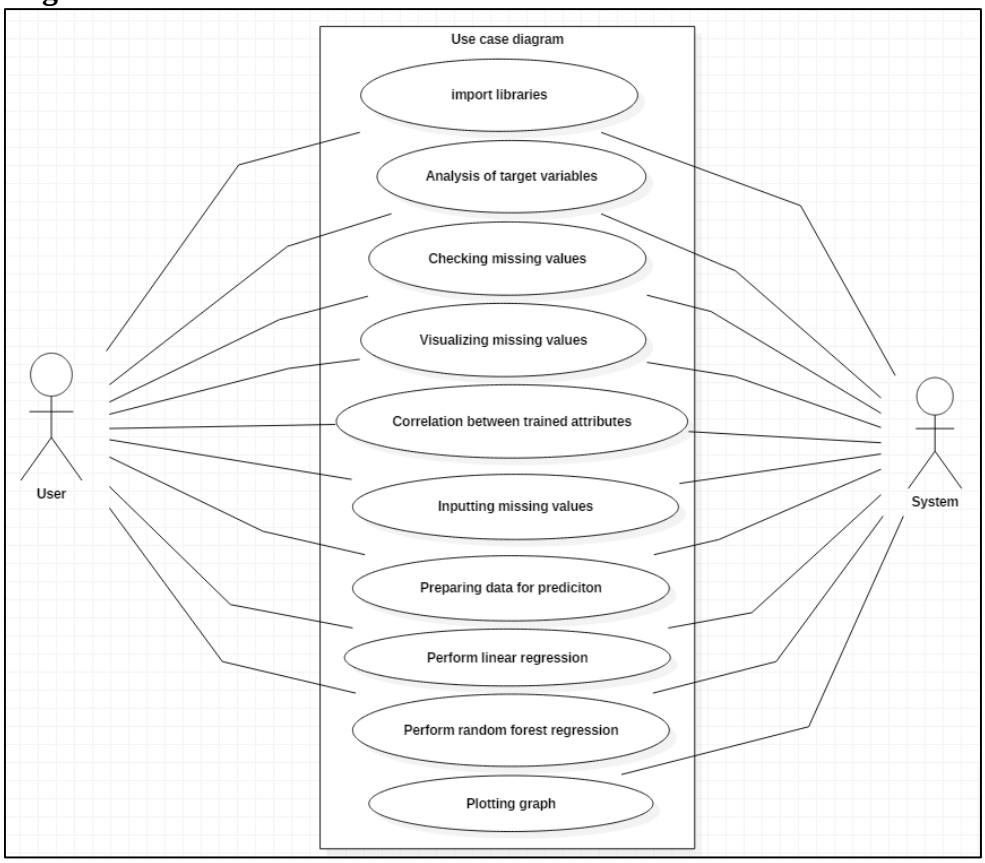
This system will handle all the prediction queries from the user. The dataset used for this system consists of 81 attributes and 1460 entries. The system has used regression algorithm. This project predicts the efficient house pricing for real estate customers with respect to their budgets and priorities. This system give us a good prediction on the price of the house based on other variables.

4.1 Scope of the project

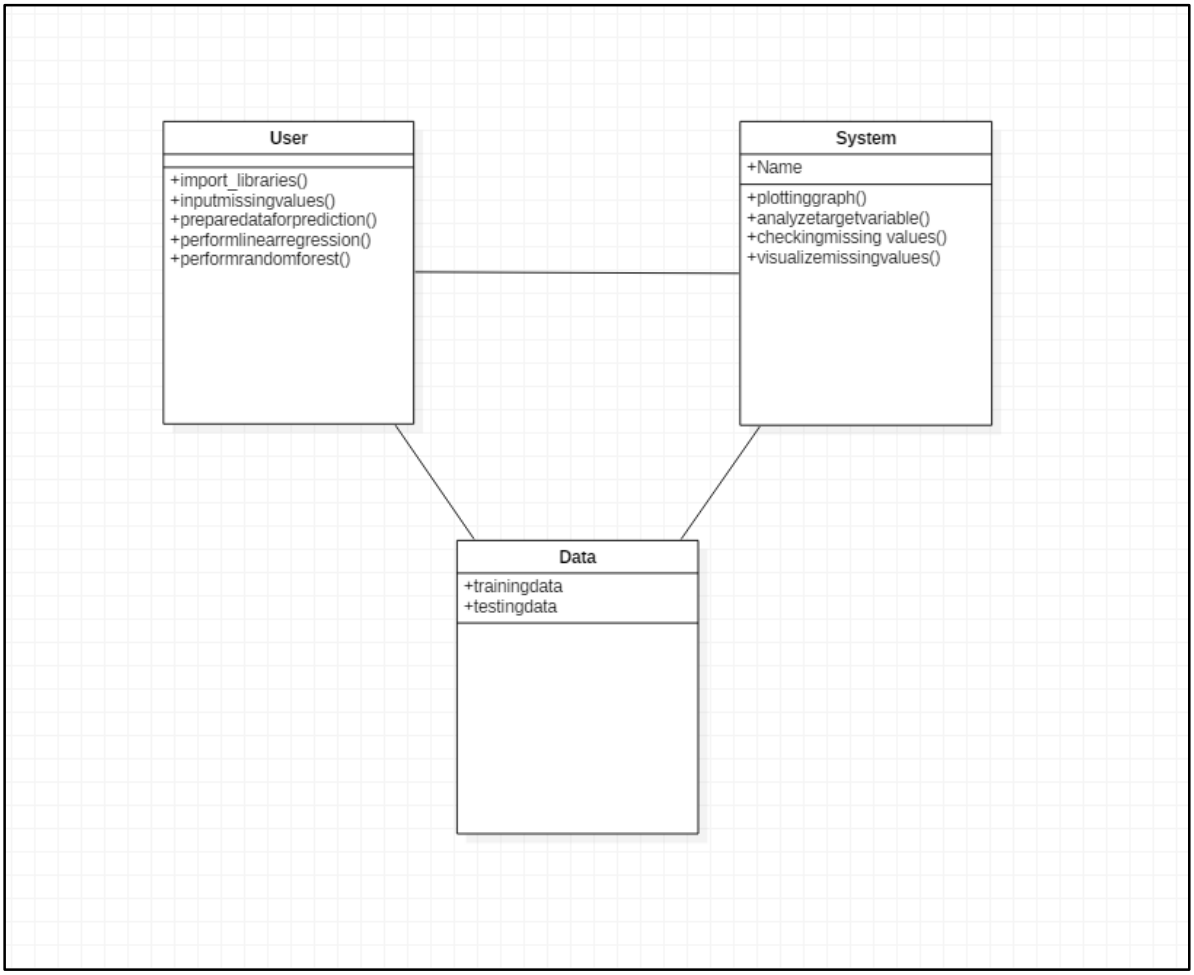
This project can provide users with location-specific predictions and trends on housing market. Obtaining predictions through the system is quick for the users, even though the prediction mechanism may involve computationally intensive tasks. The project has the power to handle prediction tasks from the user. It is built upon a platform which has the necessary computing power and implementation libraries installed.

4.2 System Design

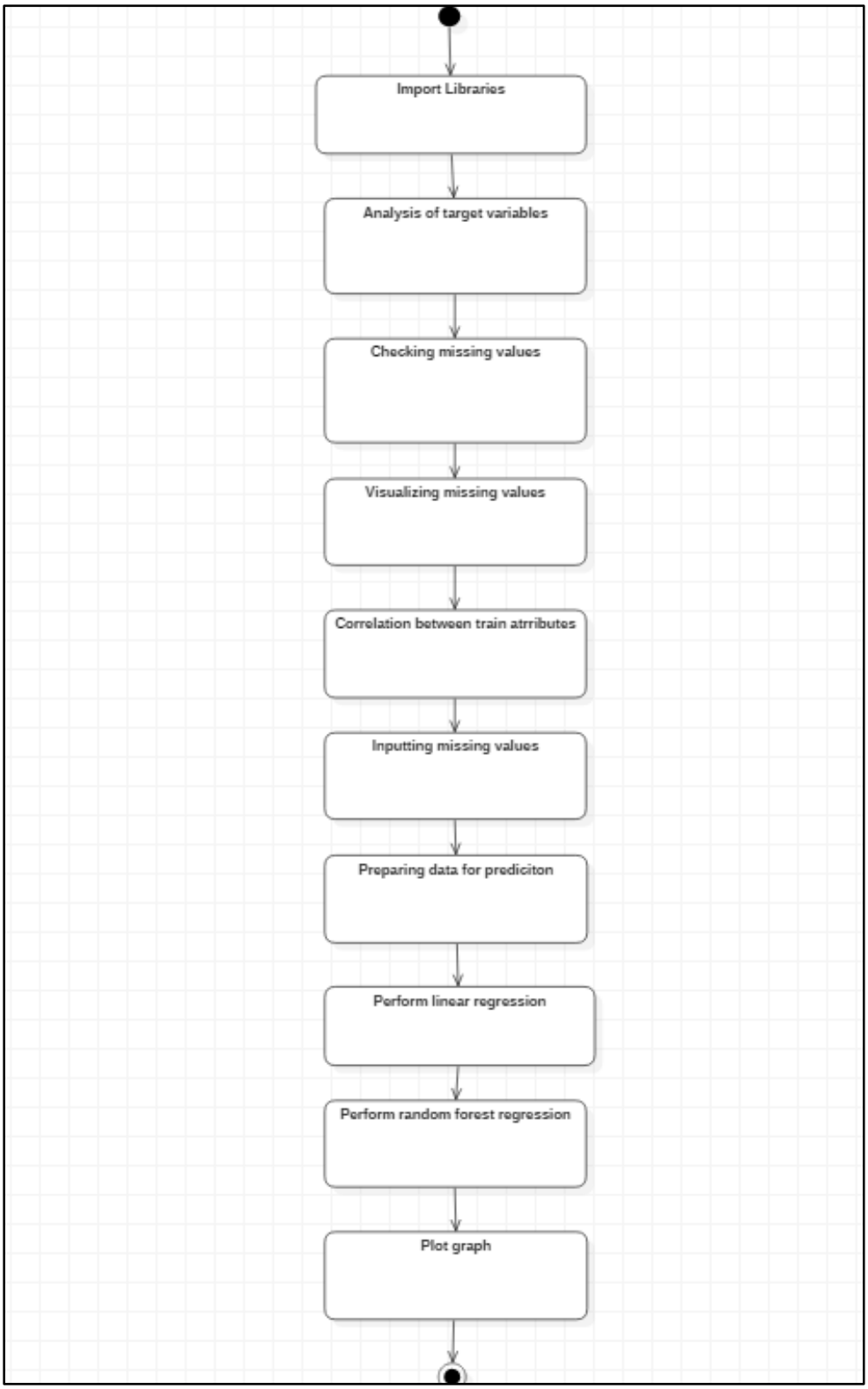
Use Case diagram:



Class diagram:



Activity diagram:



4.3 Implementation Details

4.3.1 Technologies Used

- **Jupyter Notebook:**

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release(October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages,

including Python, R, Julia and Haskell.The Notebook interface was added to IPython in the 0.12 release(December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

- **Python Programming language:**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.[26] Van Rossum led the language community until July 2018.[27][28]Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python features a comprehensive standard library, and is referred to as "batteries included".

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model. Python and CPython are managed by the non-profit Python Software Foundation.

- **Graph plotting(Matplotlib):**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

As of 23 June 2017, matplotlib 2.0.x supports Python versions 2.7 through 3.6. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of Matplotlib to support Python 2.6.

Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

4.3.2 Machine Learning Algorithm:

- **Linear Regression**

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. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

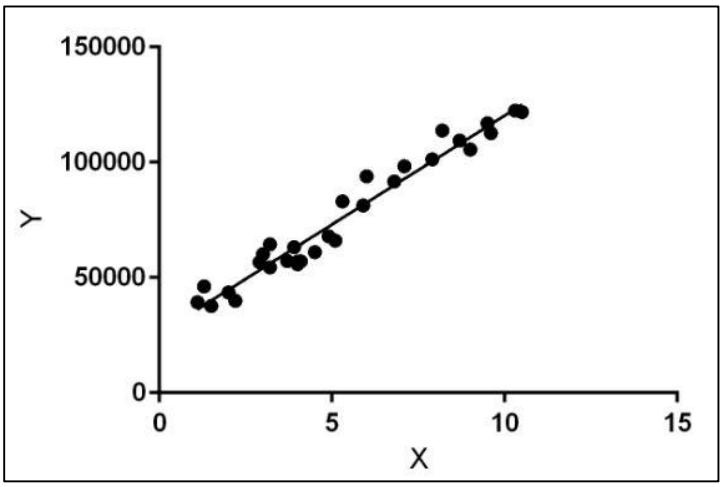


Fig. 1 :- Linear Regression

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

- **Random Forest Regression:**

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**. What is bagging you may ask? Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.

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**. What is bagging you may ask? Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement.

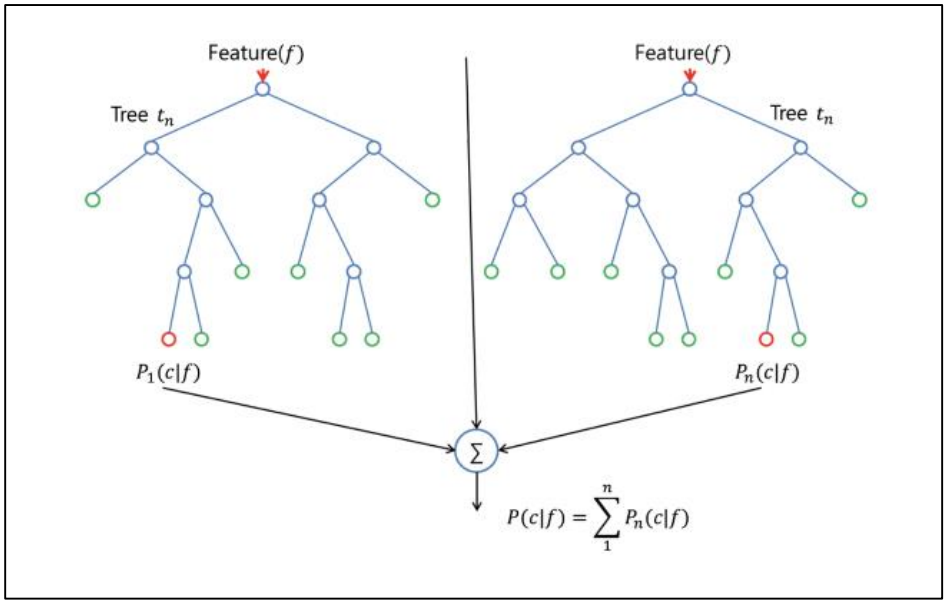


Fig. 2 :- Random Forest Regression

- **Gradient boosting regression:**

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function.[1] Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman,[2][3] simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean.

The latter two papers introduced the view of boosting algorithms as iterative *functional gradient descent* algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

- The project is using the above two mentioned machine learning algorithm for predicting the value of house.
- The accuracy rate using these two model has been shown in the screen shot which is available in result analysis section.

4.3.3 Code:

```
//Preparing data for prediciton
#Take targate variable into y
y = train['SalePrice']
#Delete the saleprice
del train['SalePrice']
#Take their values in X and y
X = train.values
y = y.values
# Split data into train and test formate
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)
//Linear Regression
#Train the model
from sklearn import linear_model
model = linear_model.LinearRegression()
#Fit the model
model.fit(X_train, y_train)
#Prediction
print("Predict value " + str(model.predict([X_test[142]])))
print("Real value " + str(y_test[142]))
#Score/Accuracy
print("Accuracy --> ", model.score(X_test, y_test)*100)
//RandomForestRegression
#Train the model
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=1000)
#Fit
model.fit(X_train, y_train)
#Score/Accuracy
print("Accuracy --> ", model.score(X_test, y_test)*100)
```

```
//GradientBoostingRegressor
#Train the model
from sklearn.ensemble import GradientBoostingRegressor
GBR = GradientBoostingRegressor(n_estimators=100, max_depth=4)
#Fit
GBR.fit(X_train, y_train)
print("Accuracy --> ", GBR.score(X_test, y_test)*100)
```

4.4 Result Analysis:

The two Machine learning algorithms i.e. Linear regression and Random forest regression are used for implementing this project. Various graphs have been used for analyzing the features from the dataset. The dataset have been divided into sets training dataset and testing dataset. The output contains the accuracy of the model used as well as the predicted value and real value. Comparing the output random forest regression is more accurate then linear regression model.

Considering frequency:

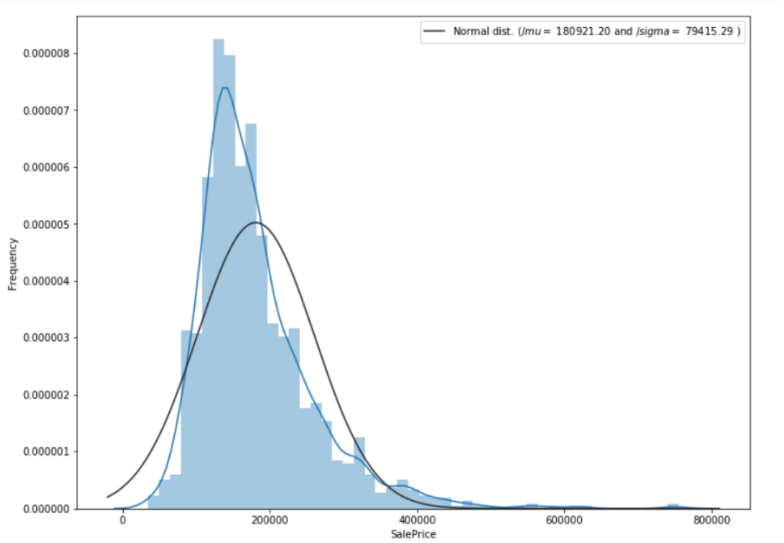


Fig. 3 :- Graph (Frequency v/s Sale Price)

Not considering frequency :

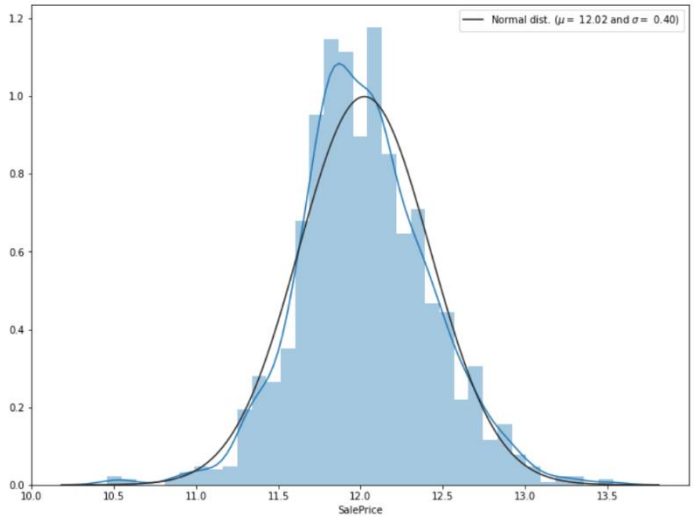


Fig. 4 :- Graph without considering freqeuncy

Null values representation:

Out[172]:

Feature	Null Count
PoolQC	1453
MiscFeature	1406
Alley	1369
Fence	1179
FireplaceQu	690
LotFrontage	259
GarageCond	81
GarageType	81
GarageYrBlt	81
GarageFinish	81
GarageQual	81
BsmtExposure	38
BsmtFinType2	38
BsmtFinType1	37
BsmtCond	37
BsmtQual	37
MasVnrArea	8
MasVnrType	8
Electrical	1
Utilities	0

Fig. 5 :- Features-Null count

```
plt.figure(figsize= (12,6))
sns.heatmap(train.isnull())
plt.show()
```

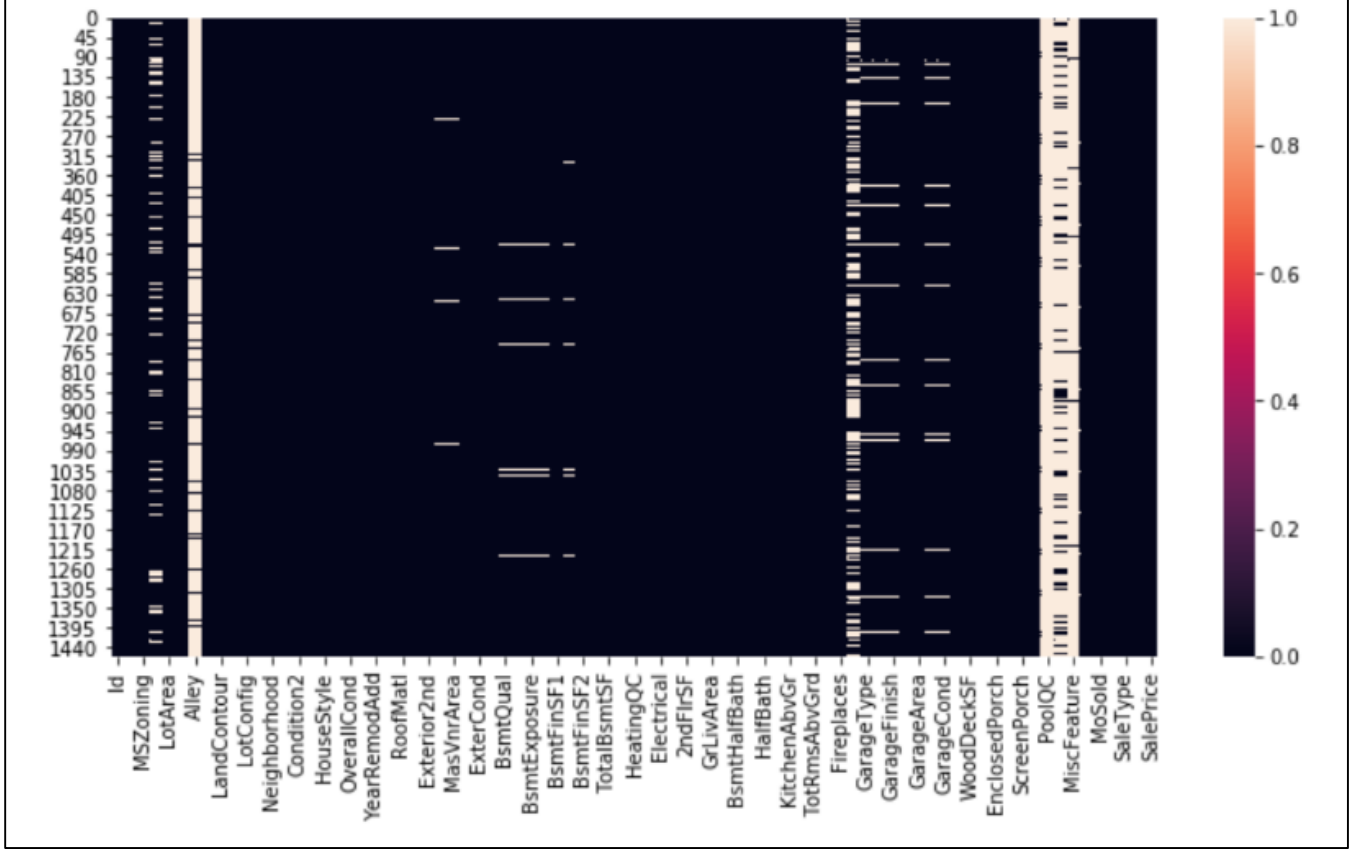


Fig. 6 :- Heatmap with null values

A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors. Here we are displaying features considered in dataset and the white space in the above figure indicates the null values.

```
In [85]: Isnull = train.isnull().sum()/len(train)*100
         Isnull = Isnull[Isnull>0]
         Isnull.sort_values(inplace=True, ascending= False)
         Isnull

Out[85]: PoolQC          99.520548
         MiscFeature     96.301370
         Alley           93.767123
         Fence           80.753425
         FireplaceQu     47.260274
         LotFrontage     17.739726
         GarageYrBlt      5.547945
         GarageType       5.547945
         GarageFinish     5.547945
         GarageQual       5.547945
         GarageCond       5.547945
         BsmtFinType2     2.602740
         BsmtExposure     2.602740
         BsmtFinType1     2.534247
         BsmtCond         2.534247
         BsmtQual         2.534247
         MasVnrArea       0.547945
         MasVnrType       0.547945
         Electrical       0.068493
         dtype: float64
```

Fig. 7 :- Features-Null count(%)

```
In [90]: plt.figure(figsize=(13,5))
         sns.set(style='whitegrid')
         sns.barplot(x='Name', y='count' , data=Isnull)
         plt.xticks(rotation=90)
         plt.show()
```

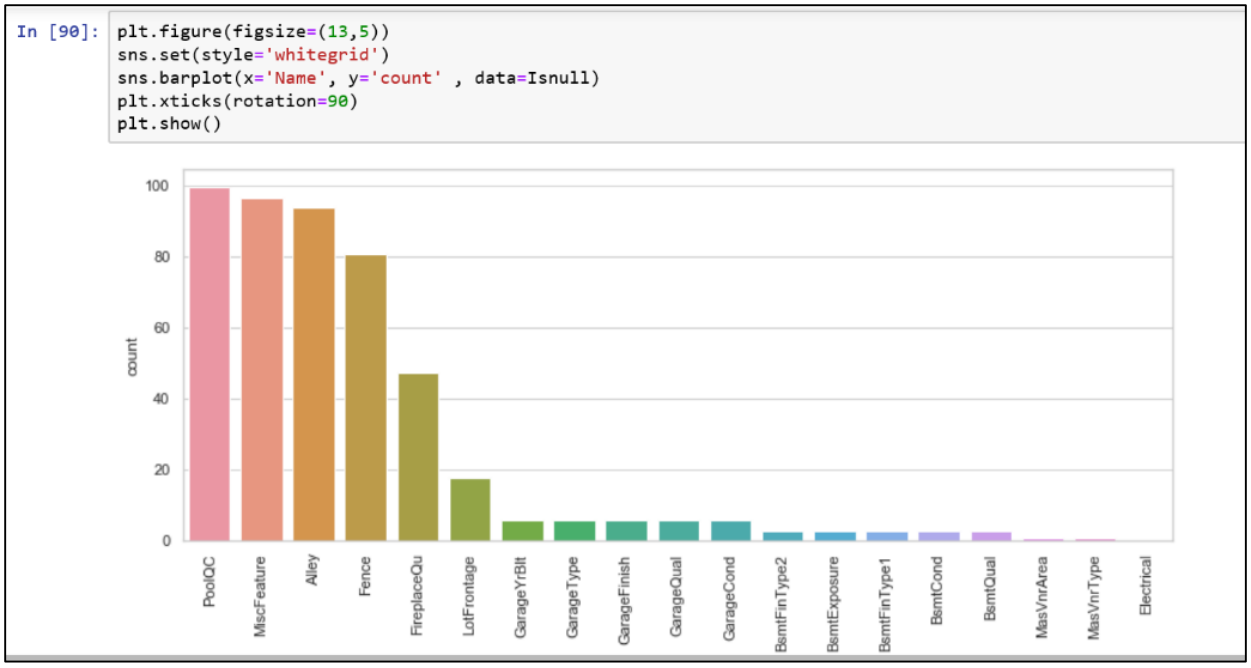


Fig. 8 :- Bar Graph

Correlation matrix(training dataset):

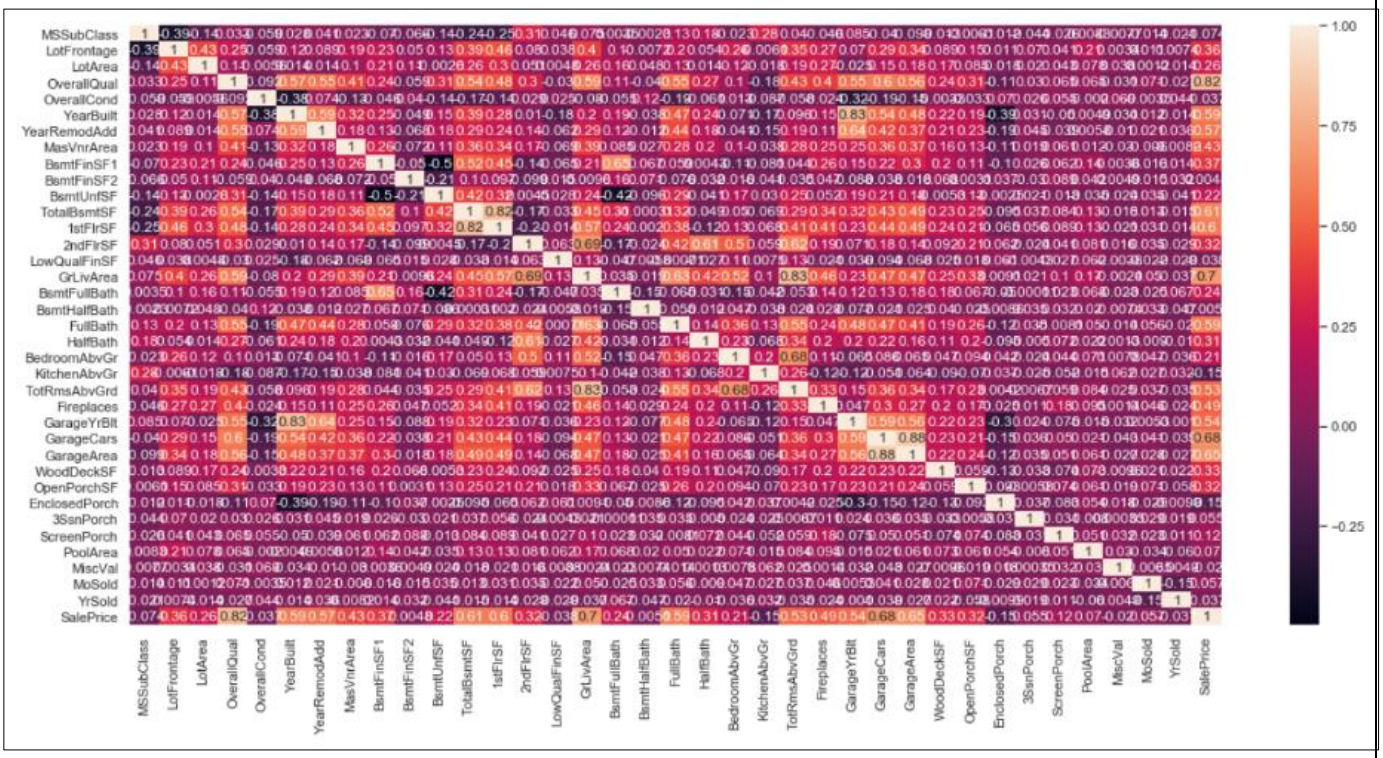


Fig. 9 :- Training dataset matrix

Considering top features for heatmap:

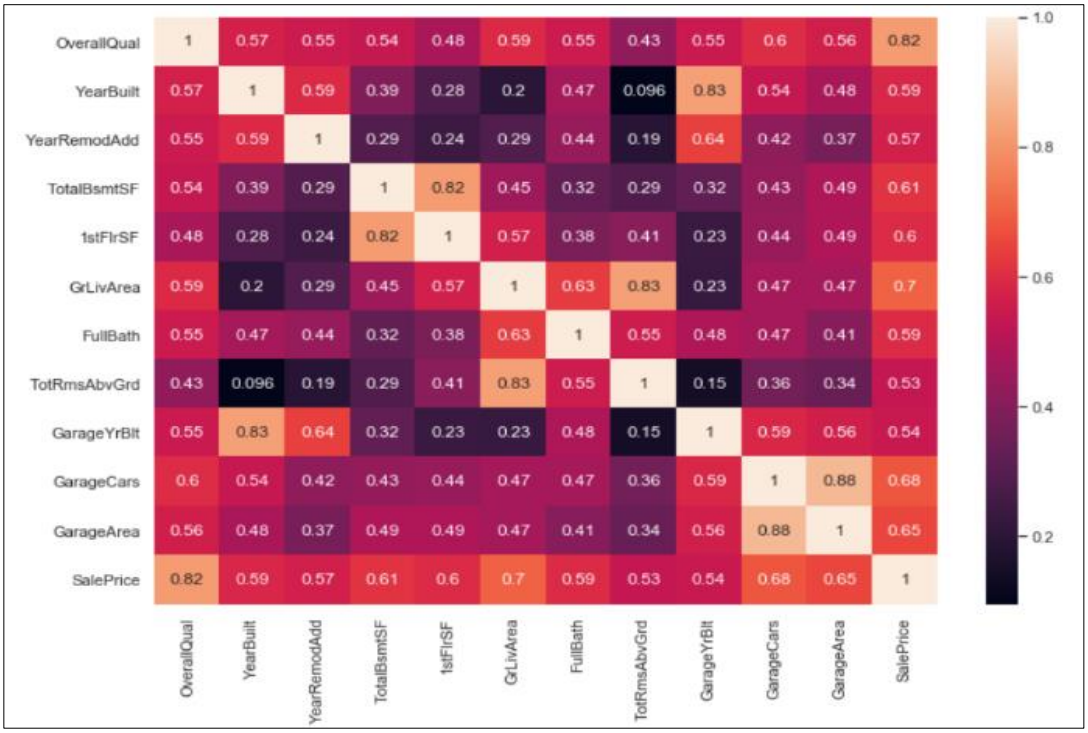


Fig. 10 :- Confusion matrix (Top features)

```
In [97]: sns.barplot(train.OverallQual, train.SalePrice)
Out[97]: <matplotlib.axes._subplots.AxesSubplot at 0x27347806e48>
```

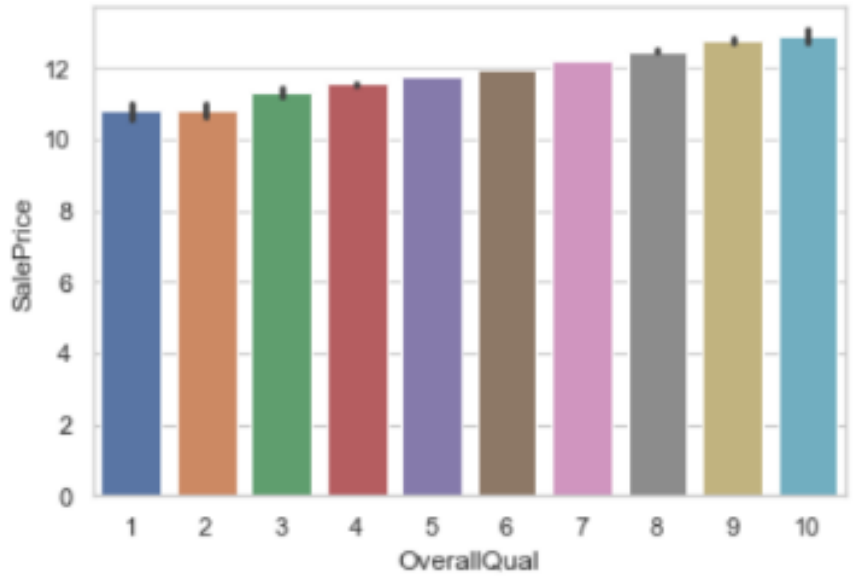


Fig. 11 :- Bar Graph

Box graph**:**

Relationship between categorical features and the target variable.The piece of code below shows how to plot a boxplot of the categorical variables SaleCondition w.r.t. the target variable.

```
In [98]: plt.figure(figsize=(18,8))
         sns.boxplot(x=train.OverallQual, y=train.SalePrice)
Out[98]: <matplotlib.axes._subplots.AxesSubplot at 0x27346568cc0>
```

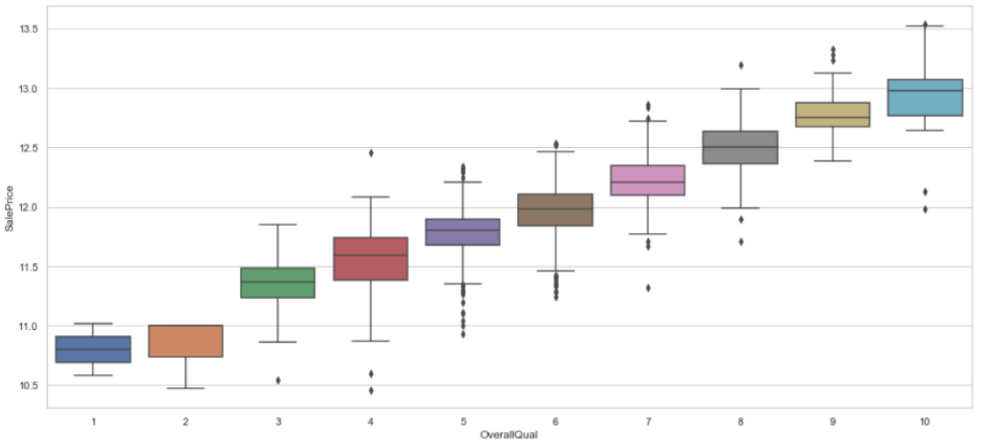


Fig. 12 :- Box Graph

Plotting pairwise:

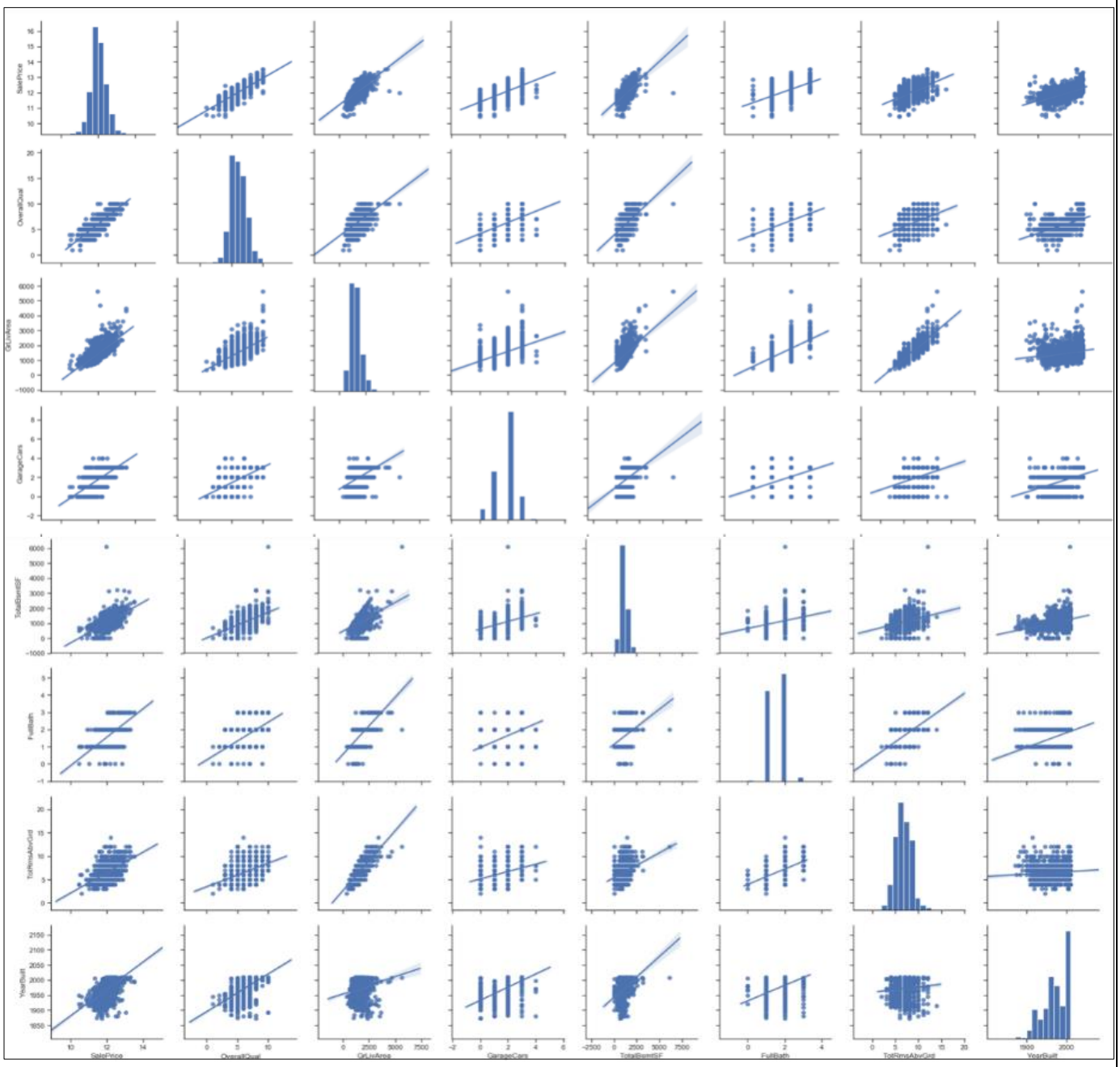


Fig. 13 :- Pair plot

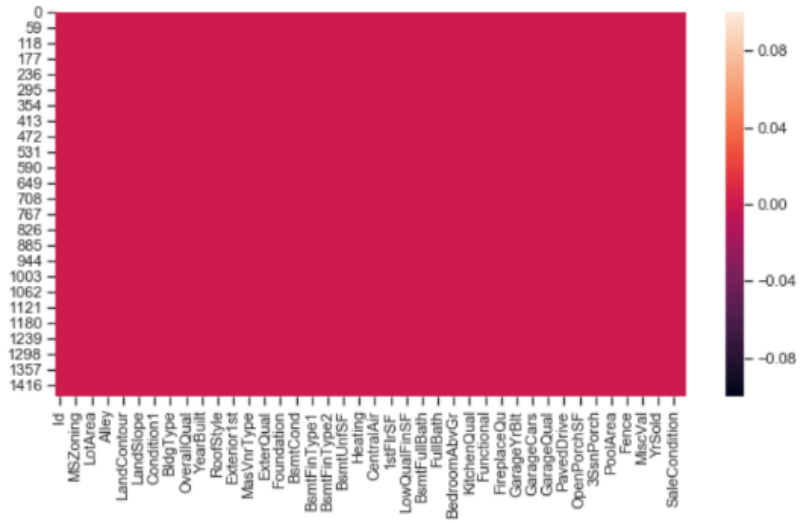
```
Find the most important features realtive to the target

Out[100]: SalePrice        1.000000
          OverallQual      0.817185
          GrLivArea        0.700927
          GarageCars       0.680625
          GarageArea       0.650888
          TotalBsmtSF      0.612134
          1stFlrSF         0.596981
          FullBath         0.594771
          YearBuilt        0.586570
          YearRemodAdd     0.565608
          GarageYrBlt      0.541073
          TotRmsAbvGrd     0.534422
          Fireplaces       0.489450
          MasVnrArea       0.430809
          BsmtFinSF1       0.372023
          LotFrontage      0.355879
          WoodDeckSF       0.334135
          OpenPorchSF      0.321053
          2ndFlrSF         0.319300
          HalfBath         0.313982
          LotArea          0.257320
          BsmtFullBath     0.236224
          BsmtUnfSF        0.221985
          BedroomAbvGr     0.209043
          ScreenPorch      0.121208
          PoolArea         0.069798
          MoSold           0.057330
          3SsnPorch        0.054900
          BsmtFinSF2       0.004832
          BsmtHalfBath    -0.005149
          MiscVal         -0.020021
          OverallCond     -0.036868
          YrSold          -0.037263
          LowQualFinSF    -0.037963
          MSSubClass      -0.073959
          KitchenAbvGr    -0.147548
          EnclosedPorch   -0.149050
          Name: SalePrice, dtype: float64
```

Fig. 14 :- Important features

Eliminating null values:

```
In [111]: #Checking there is any null value or not
          plt.figure(figsize=(10, 5))
          sns.heatmap(train.isnull())
```

```
Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x2734ac69898>
```



Now there is no null values

Encoding str to int

Fig. 15 :- Heat Map

Model implementation:

Linear regression result:

```
In [123]: #Prediction
          print("Predict value " + str(model.predict([X_test[142]])))
          print("Real value " + str(y_test[142]))
          print ("R^2 is: \n", model.score(X_test, y_test))
```

```
Predict value [11.62221633]
Real value 11.767187766223199
R^2 is:
 0.892670867716144
```

Fig. 16 :- Predicted value - Real value

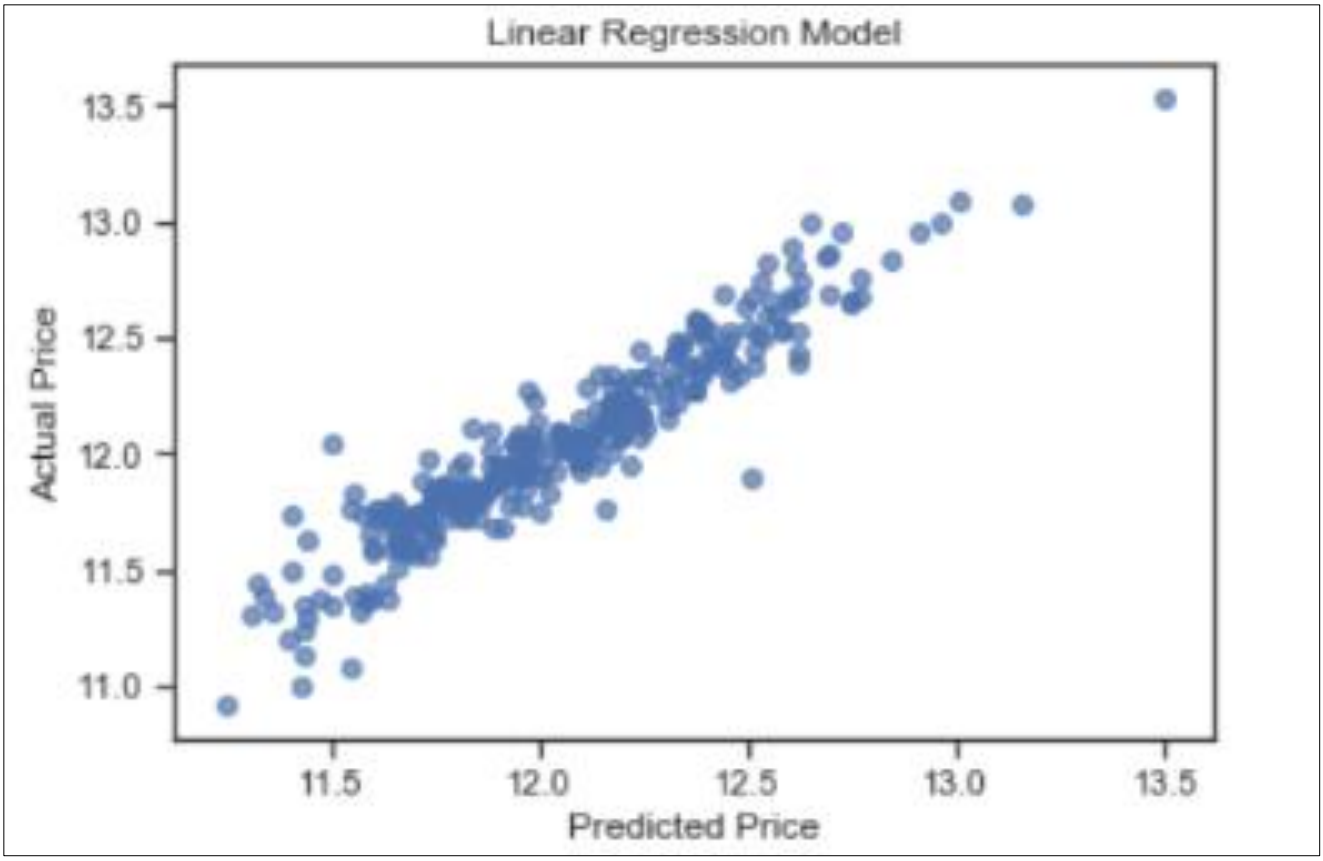


Fig. 17 :- Scatter Graph

Accuracy of the model:

```
In [121]: #Score/Accuracy
          print("Accuracy --> ", model.score(X_test, y_test)*100)

          Accuracy -->  89.2670867716144
```

Fig. 18 :- Linear Regression-Accuracy

Random forest regression:

Accuracy of the model:

```
In [130]: #Score/Accuracy
          print("Accuracy --> ", model.score(X_test, y_test)*100)

          Accuracy -->  89.44394188192113
```

Fig. 19 :-Random Forest Regression-Accuracy

4.5 Conclusion and Future Scope

We have managed to develop a system that provides users with a way to look at future housing price predictions. The regression methods have been explored and compared, before arriving at a pre-diction method. The model have been, so that future price predictions will tend towards more sensible values. We devised a way to utilise as much data as possible in our prediction system. The success of our approach to creating a system for generating predictions can be applied to other problem sets concerned with geographical variations in the prediction model.

Despite having produced a system that met our initial requirements, there are various improvements that can be made in the future. The following factors can be considered to expand the project in the right direction in the future :-

- Consider more factors affecting housing prices :-

One main drawback of our prediction model is the lack of access to information. We would need to seek alternative sources of data besides the Land Registry, which can provide us with information on the area, the number of rooms, etc for each property. Economic factors such as the yearly inflation rate or GDP growth can also be considered.

- Consider heteroscedastic GP regression :-

Under the conventional GP regression method that we are using in our prediction method, the noise level is assumed to be uniform throughout the domain. How-ever, this assumption may not be true. By considering a heteroscedastic treatment of noise, we can treat noise as a random variable. This can potentially better account for the difference in data points across the years in our dataset. An alternative noise model can also be explored, by training the data against the logarithm of the prices.

- Optimise the prediction system through parallelised computations :-

A major concern with the prediction system is the loading time. Moreover, our dataset takes more than one dayto train. Rather than performing the computations sequentially, we can use multiple processors and parallelise the computations involved, which can potentially reduce the training time and prediction time. Another way to approach this problem is to look for alternative APIs that allow us to produce heat maps of similar quality but require fewer data points, or faster libraries to implement GPs.

- Add more functionalities into the application :-

We can provide options for user to select a borough or district to generate the heat maps, instead of entering a postcode and selecting a search radius. We can also look into ways to generate a heat map for the whole of initially in a zoomed-out view, only requesting for more data points when the user zooms in to display a clearer picture.

CHAPTER 5:- REFERENCES

1. Wan Teng Lim, Lipo Wang, Yaoli Wang, and Qing Chang-Housing Price Prediction Using Neural Networks –IEEE
2. LiLi and Kai-HsuanChu - Prediction of Real Estate Price Variation Based on Economic Parameters
3. IJARCCE- House Price Forecasting