****

**FINAL PROJECT REPORT**

**AIDI 1100 – INTRODUCTION TO AI DEVELOPMENT**

**ASHKAN KIANI**

**SONAKSHI KARKERA**

**SURBHI THAKUR**

**ARUN KALAESWARAN**

**Rahul Angalakduti**

**December 2, 2019**

**Contents**

[**Problem statement** 3](#_Toc26195217)

[**Data Source and Tools** 3](#_Toc26195218)

[**Exploratory Data Analysis** 3](#_Toc26195219)

[**1.** **Features and Target variable** 3](#_Toc26195220)

[**2.** **No of categorical and Numerical features** 4](#_Toc26195221)

[3. **Handling Missing values** 4](#_Toc26195222)

[**4.** **Correlation Matrix** 4](#_Toc26195223)

[**5.** **Distribution plots for both Numerical and Categorical variables** 5](#_Toc26195224)

[6. **Dropping features that are not strongly correlated** 6](#_Toc26195225)

[**7.** **Label encoding for Categorical feature** 7](#_Toc26195226)

[8. **Outlier Detection** 7](#_Toc26195227)

[**9.** **Normalizing the dataset** 7](#_Toc26195228)

[**10. Splitting the dataset into Test and Train** 8](#_Toc26195229)

[**Algorithms** 8](#_Toc26195230)

[**1.** **Linear regression** 8](#_Toc26195231)

[**2.** **K Nearest Neighbor** 9](#_Toc26195232)

[**3.** **Decision trees** 10](#_Toc26195233)

[**4.** **Gradient Boosting** 10](#_Toc26195234)

[**Challenges** 11](#_Toc26195235)

[**Evaluation** 11](#_Toc26195236)

[**Conclusion** 12](#_Toc26195237)

**Predicting House Prices**

# **Problem statement**

To predict the Sales Price of the house by implementing Machine Learning algorithms on dataset provided by Kaggle. Different types of people tend to have different priorities for buying houses and are interested in different attributes. The final price of the house depends on numerous features and to predict the price of a house is a difficult task.

# **Data Source and Tools**

The dataset was provided by Kaggle for House Pricing Prediction Competition. The rows and columns present in train file is 1460,81 and in test file is 1459,80. There are total 79 features present in the dataset using which we will implement the price prediction model.

The tools we use are the following; Pandas/NumPy- Data Storage and Organization

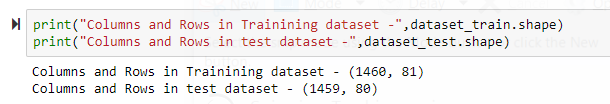
Matplotlib/ Seaborn – To implement graphs and visualizations on the data analysis

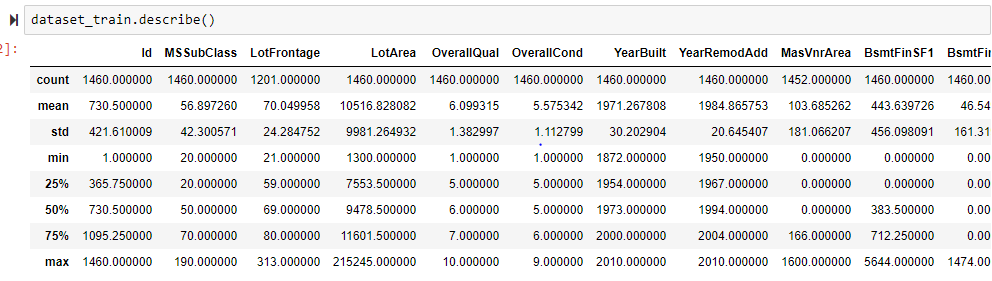
Sklearn- To implement the machine learning algorithms and measuring their accuracy score.

# **Exploratory Data Analysis**

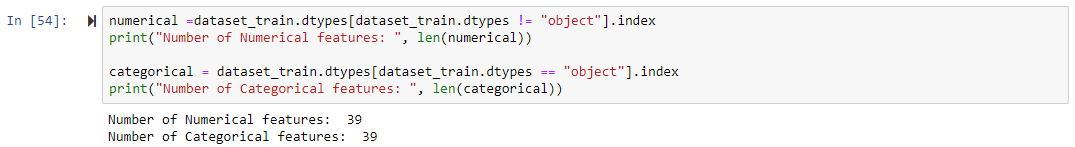
## **Features and Target variable**

There are total 80 features in the dataset and 1 Target variable

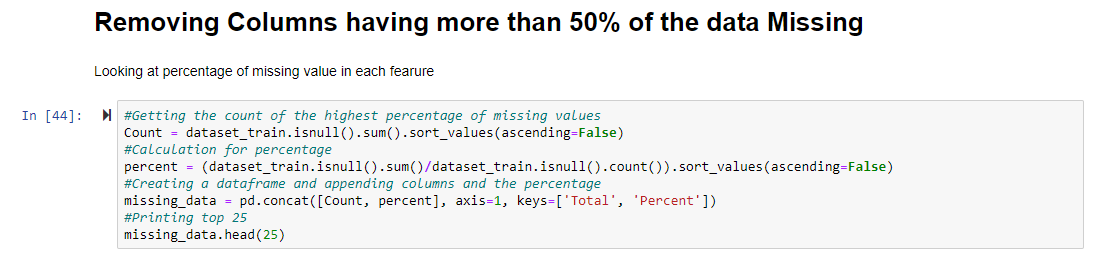




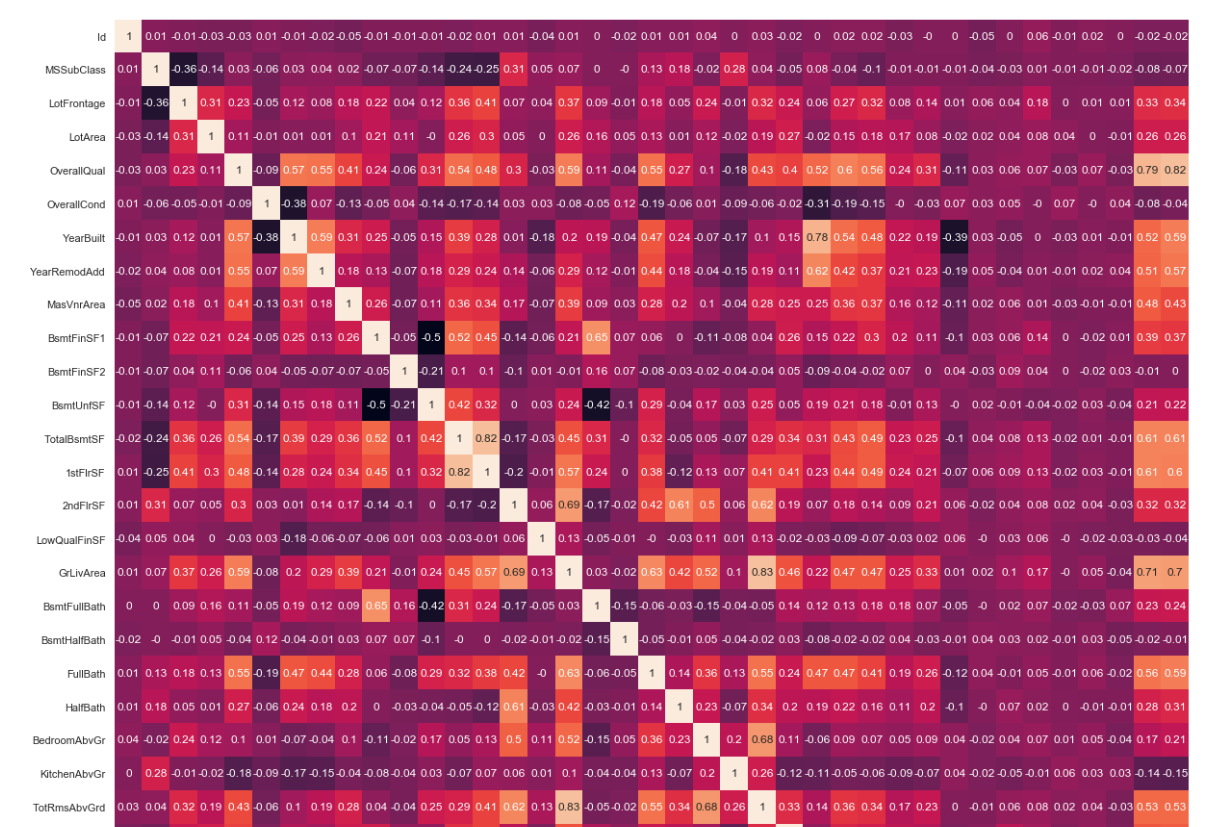
## **No of categorical and Numerical features**

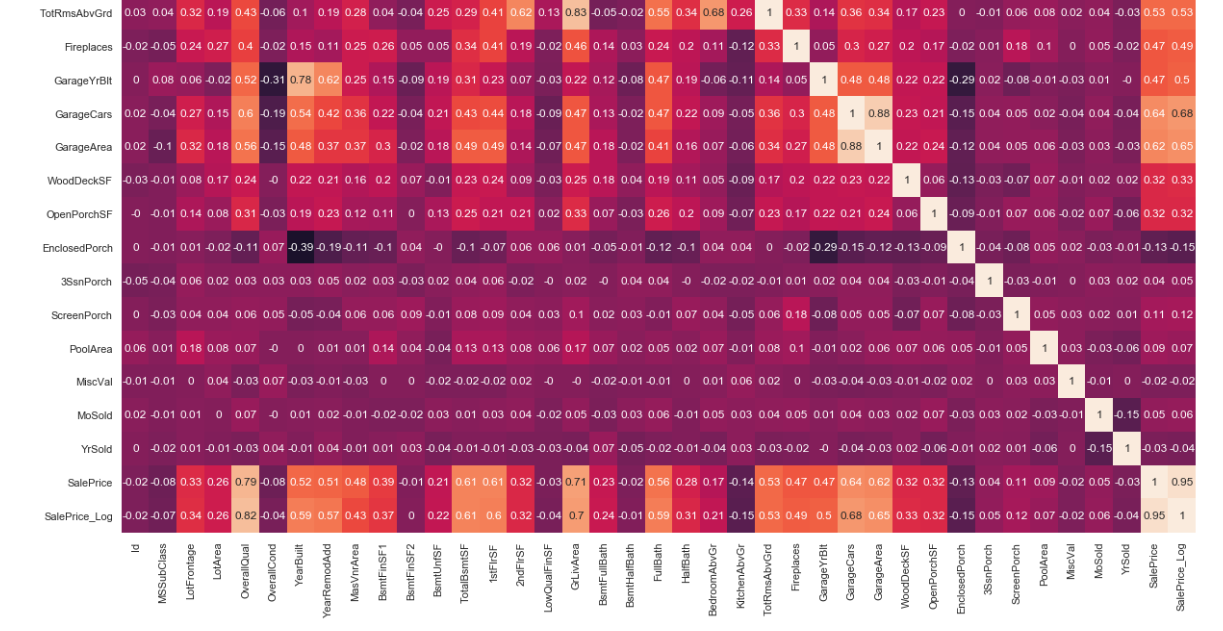


1. **Handling Missing values**



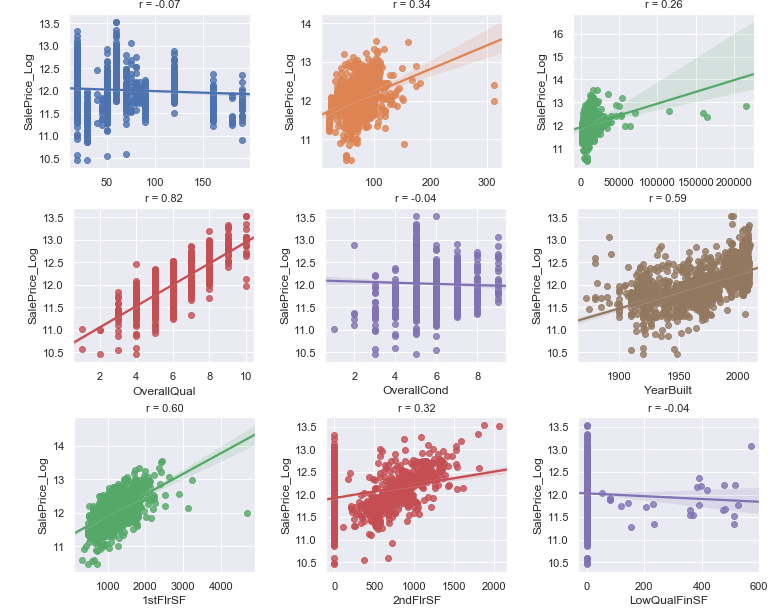
## **Correlation Matrix**



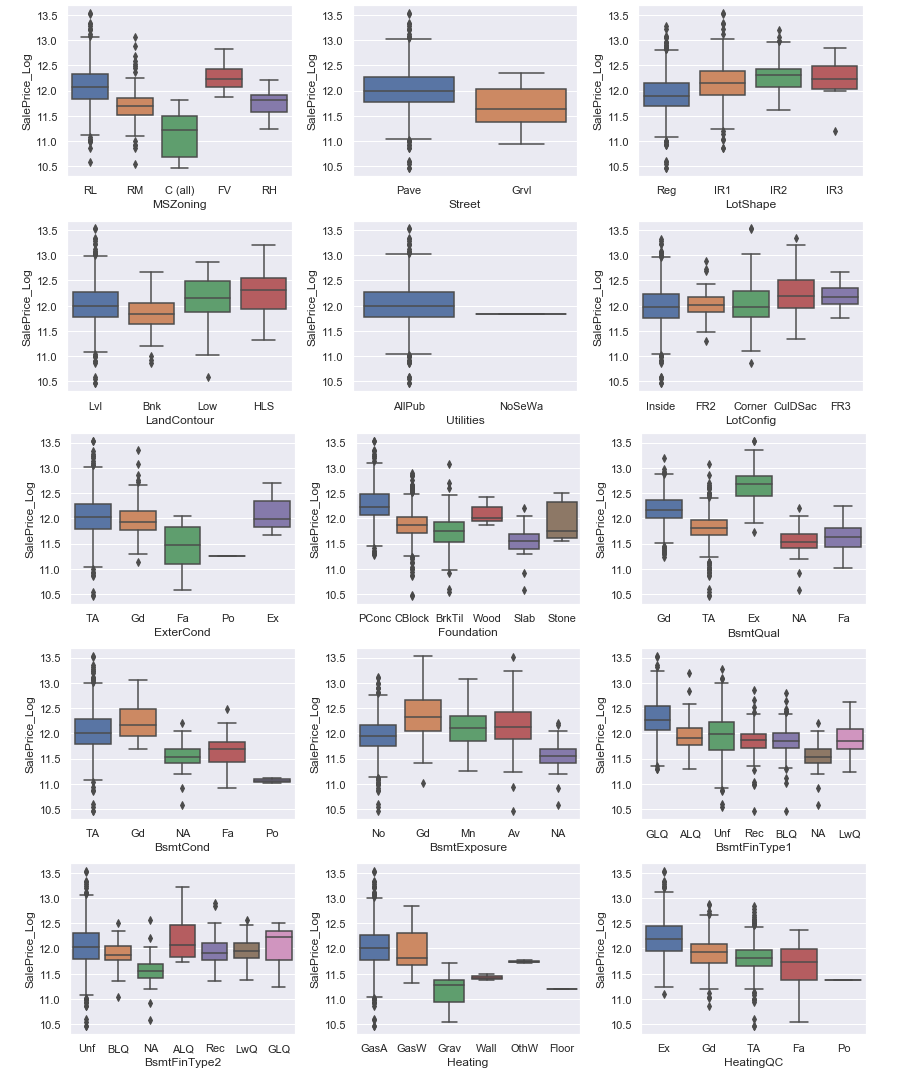


## **Distribution plots for both Numerical and Categorical variables**

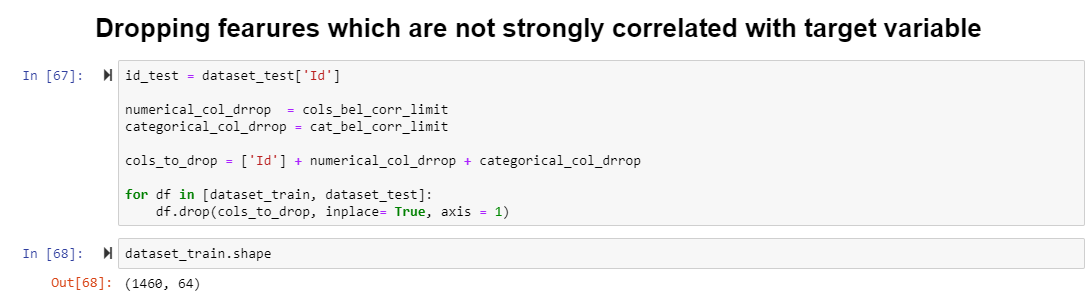
For Numerical Values:-



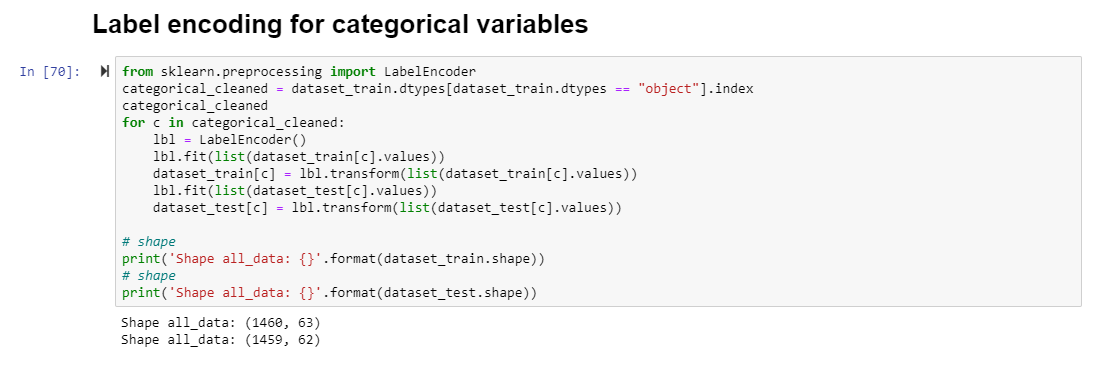
For Categorical Values:-



1. **Dropping features that are not strongly correlated**



## **Label encoding for Categorical feature**

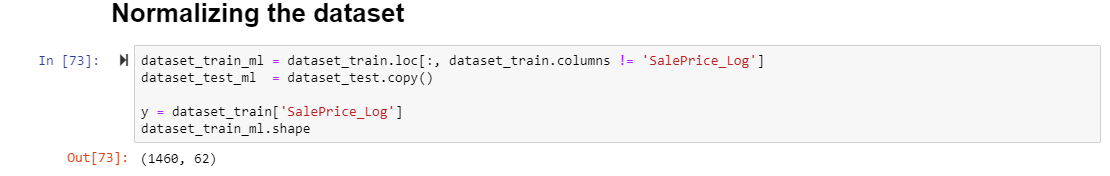


Label encoding is done to provide the numerical label to the categorical variables to convert it into machine-readable form.

## **Outlier Detection**

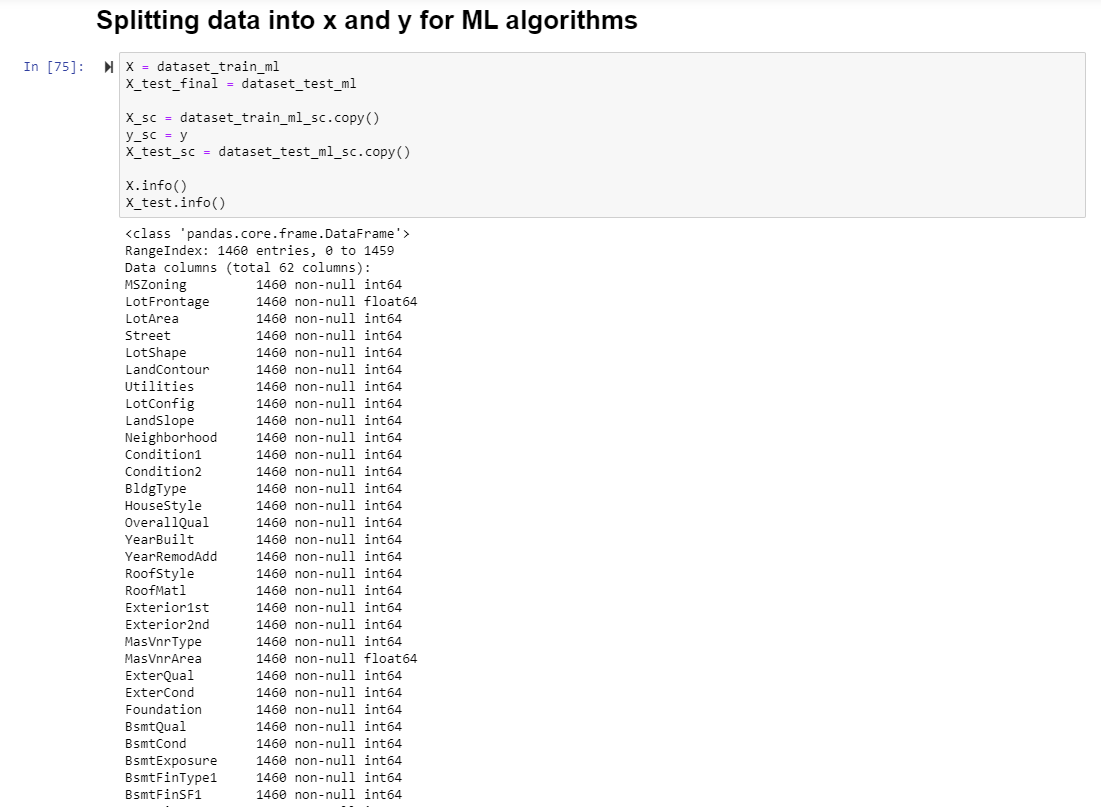


## **Normalizing the dataset**



We used normalization of dataset as there were multiple features with different ranges and we need to make sure that the different features take similar range of values do that gradient descent converge more quickly.

## **10. Splitting the dataset into Test and Train**



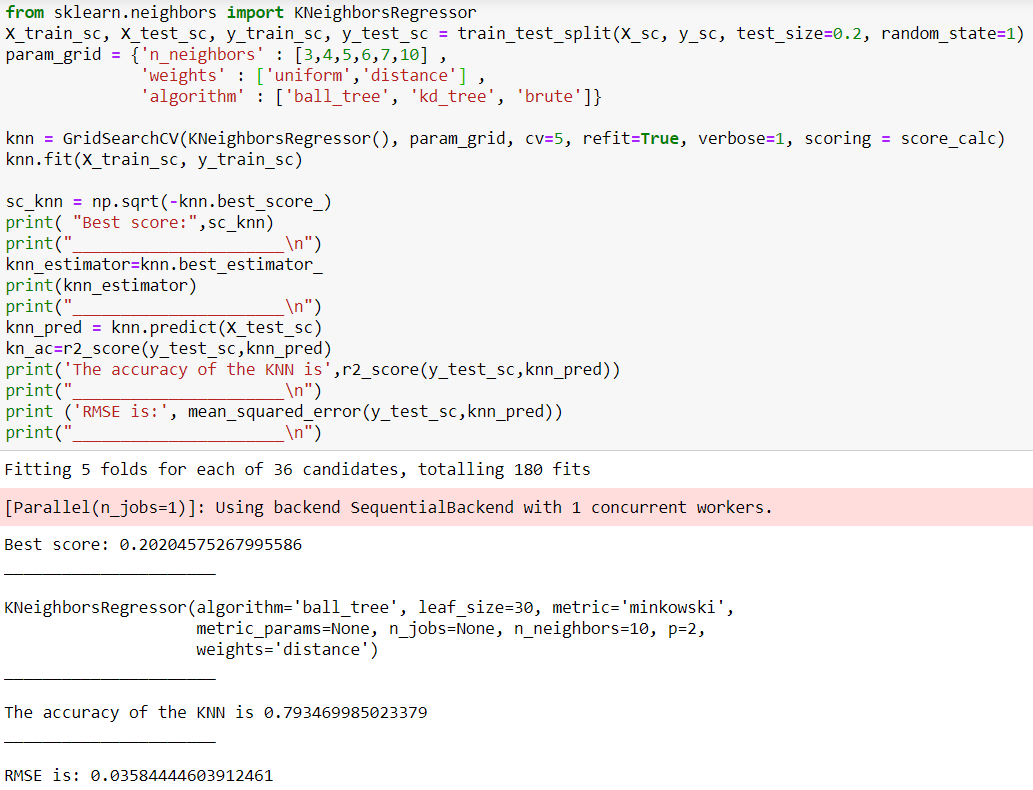
# **Algorithms**

The model we used to help us predict house prices were linear regression, KNN regressor, decision tree regressor and gradient boosting.

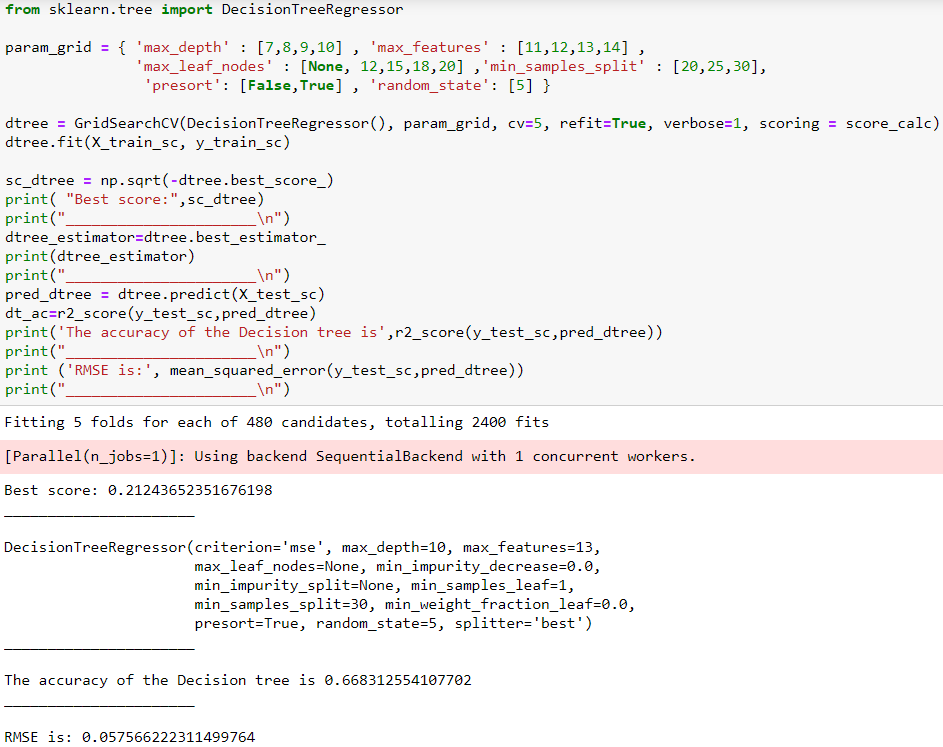
1. **Linear regression** fits a model (line/equation of best fit) that best represents the data. Here specifically it was a multiple linear regression model due to the multiple features we have in our dataset. It is easy to implement, easy to understand and very often used to predict numeric values. However, It is prone to overfitting the data and can not be used if there isn’t a linear relationship.



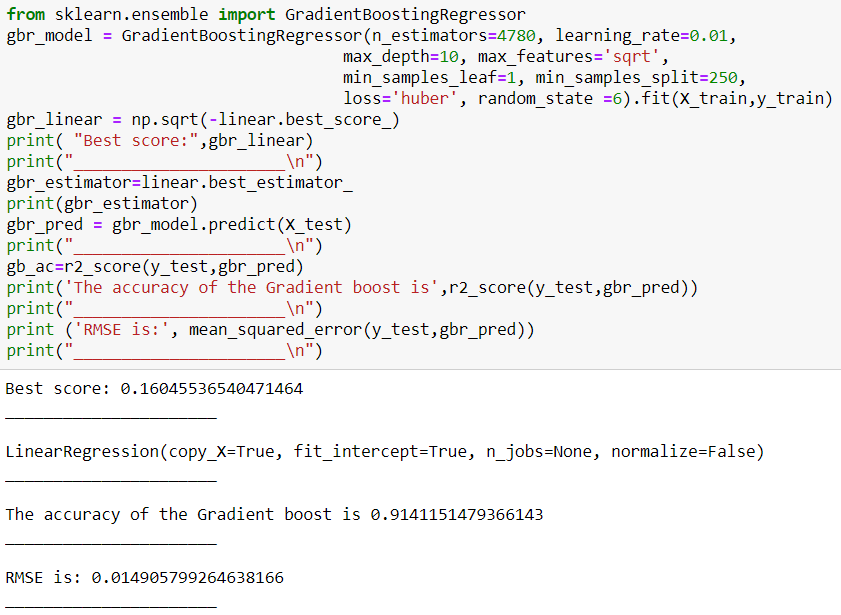
1. **K Nearest Neighbor** algorithm is a robust classifier and is very easy to understand and implement. It is known as the “lazy” learning algorithm. It clusters the data into several classes in order to predict how a new point will be classified. It is simple and Intuitive, makes no assumptions, constantly evolving and widely used and easy to implement. However, it is a very slow algorithm, suffers from curse of dimensionality (works well with a small number of input variables) and is very sensitive to outliers.



1. **Decision trees** is a supervised algorithm that consistently split based on the given parameters until there is an output. It consists of decision nodes and leaves. It is easy to interpret and visually represent, mimics human decision making, can be used for regression or classification and feature selection happens automatically. However decision trees tends to overfit the data and only axis aligned splits of data.



1. **Gradient Boosting** builds on what is learned from a previous decision tree and identifies the errors and corrects itself in the next decision tree. This gradual and sequential learning can be used for regression and classification problems. They tend to give better results than random forest models, however due to the sequential nature it is slower and longer to train and is still prone to overfitting.



# **Challenges**

We faced many challenges during this project. Some of the biggest challenges was during the cleaning of the dataset, feature selection and making sure we trained and tested our models on data that was complete and relevant. To clean the data, we began by removing columns that had more than 50% of their data missing. Then we checked the distribution to see what effect the outliers had on the data. Since our data was skewed to the right, we started to remove the outliers and used logarithmic transformation on our target variable to bring it back to a normal distribution. Then for any missing values in our remaining columns, we filled with it with the mean value of the column. We did this in order to keep data and not lose data nor compromise the data integrity. Finally, for our feature selection, we checked to see which columns had at least a correlation to our target variable of at least 0.2 and kept those features that were strongly correlated.

# **Evaluation**

The methods of evaluation we used were accuracy score and root mean squared error. Accuracy of the model determines the number of predictions it got right. While the root mean squared error tells us how far our data is to our predicted model’s line of best fit. Looking at accuracy we see that Gradient Boosting had the highest score at 91% and Decision Tree was the lowest at 67%. Also, we see that Gradient Boosting had the lowest RMSE at 0.0149 and Decision Tree had the highest RMSE at 0.0575.

# **Conclusion**

Based on our evaluation metrics, Gradient Boosting had the best results in predicting the house prices. Gradient Boosting Algorithm works best with the numerical and categorical data. This algorithm can optimize on different loss functions and provides several parameter tuning options that make the function fit very flexible.