Machine Learning

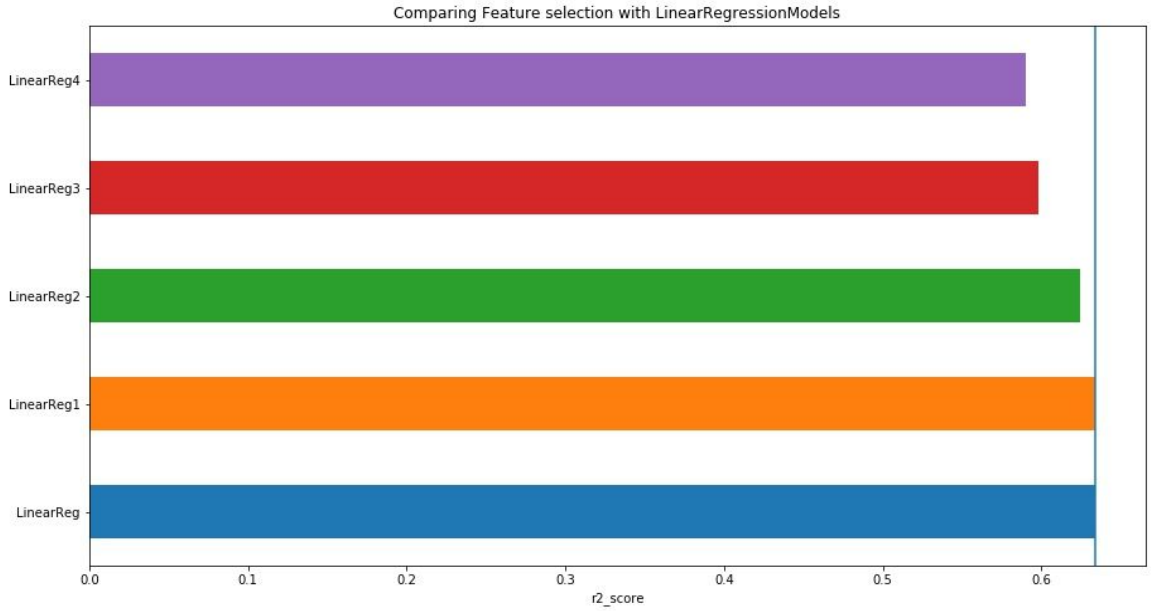
In this project, I use the dataset for supervised machine learning to predict house price. I build 4 different Regression models i.e. Linear Regression, Decision Tree Regression, Gradient Boosted Regression, and Random Forest Regression. I compared the performance of those models using R^2 because it is the ratio between how good our model is vs how good is the naive mean model. I finally recommend the better performing model to predict house price.

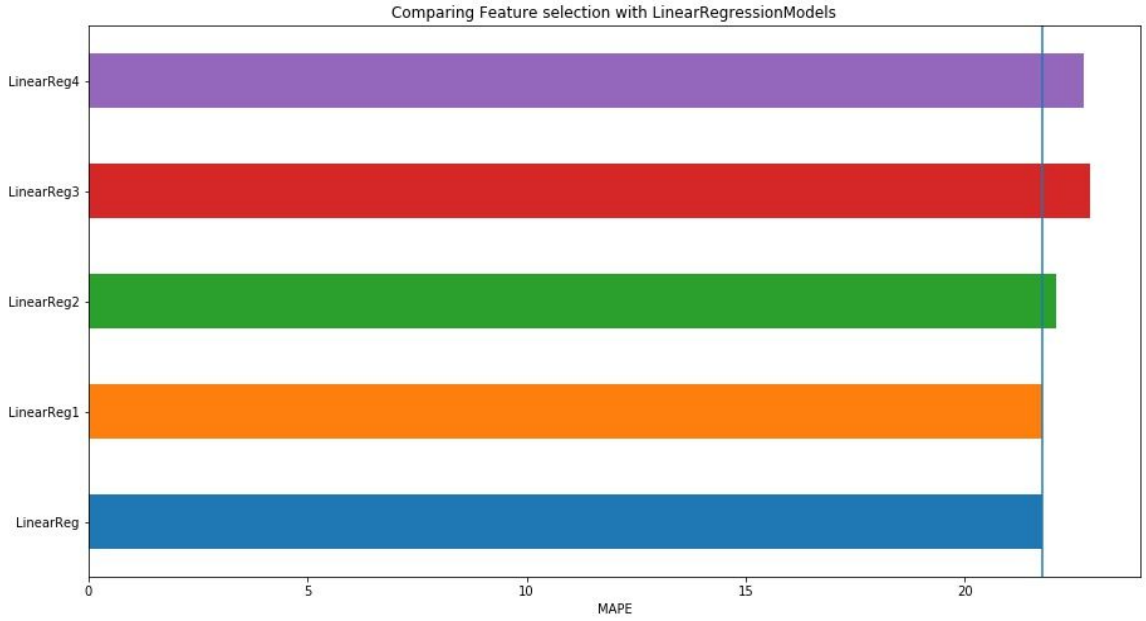
First, I divided the data into independent variable X and dependent variable y. Independent variable X is the features I am going to use to predict the target variable y. I dropped price, id and date column from the new_df dataframe to create the variable X. I used price column from the new_df dataframe to create the variable y. There are different metrics used to measures the performance of the Regression models such as Mean squared errors, Root mean squared errors, R-squared score, Mean absolute deviation, Mean absolute percent errors, etc. In this project, I use Root mean squared error and R-squared score to evaluate the performance of the regression model. In order to save the metrics of the model, I created a dataframe named metrics. Next, I split the data into training and testing set. I kept 80% of the randomly selected data as a training set and 20% of the randomly selected data as a testing set. The model will learn or fit using the 80% of the data, and the rest 20% testing data will be used as an unseen future dataset to predict the house price.

I build a Linear Regression Model using the default parameters, and I fit the model using the training dataset. I used X_test data to predict using the model. Then, I calculated Mean squared error (MSE), Root mean squared error (RMSE), R-squared score (r2_score), Mean absolute deviation (MAD), and Mean absolute percent error (MAPE).
Mean Squared Error (MSE): 13276038923.51
Root Mean Squared Error (RMSE): 115221.6947
r2_score: 0.6338
Mean Absolute Error (MAE): 88288.95
Mean Absolute Percent Error (MAPE): 21.78

I used a backward elimination method of feature selection. Feature selection is the process of selecting a subset of relevant features that may improve the performance of the model. First, I removed the one worst attribute from the feature. I removed waterfront because it has the very weak correlation with the price of the house. Then, I removed condition and sqft_lot from the feature set. Then, I removed sqft_lot15, view, and bedroom from the feature set. I also tried a

univariate feature selection package called SelectKbest from the sklearn library.





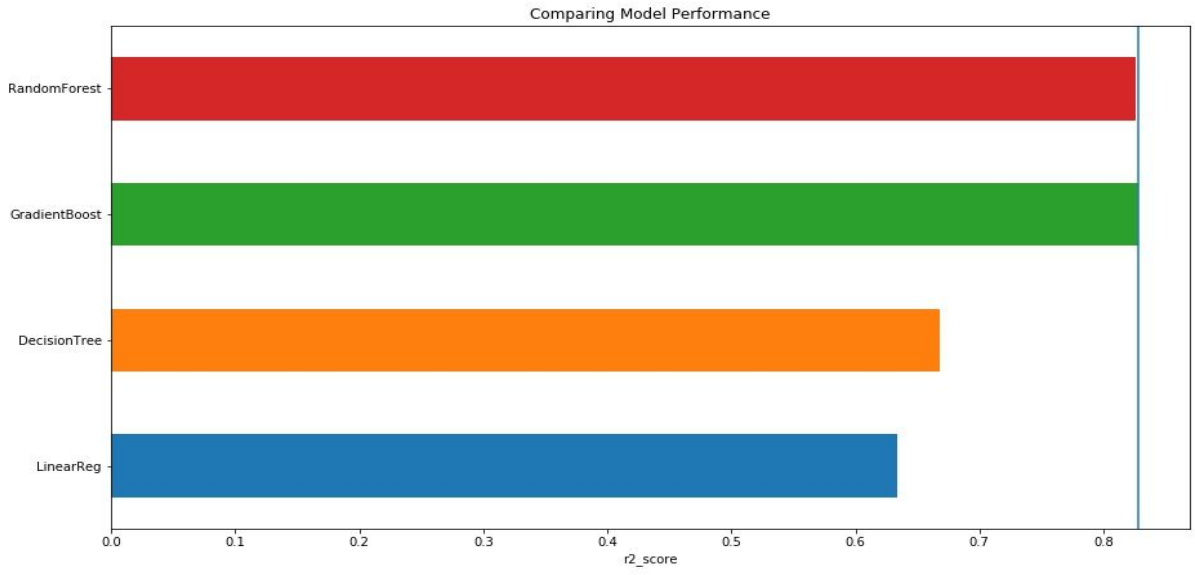
Comparing all different Linear Regression model with feature selection, both barplot suggests us that we should keep all the features to better predict the house price.

I also build a Decision Tree Regressor Model using the default parameters.
Mean Squared Error: 12025421297.16
Root Mean Squared Error: 109660.48
r-squared score : 0.6682893942359164
Mean Absolute Deviation (MAE): 76605.84
Mean Absolute Percent Error (MAPE): 18.45

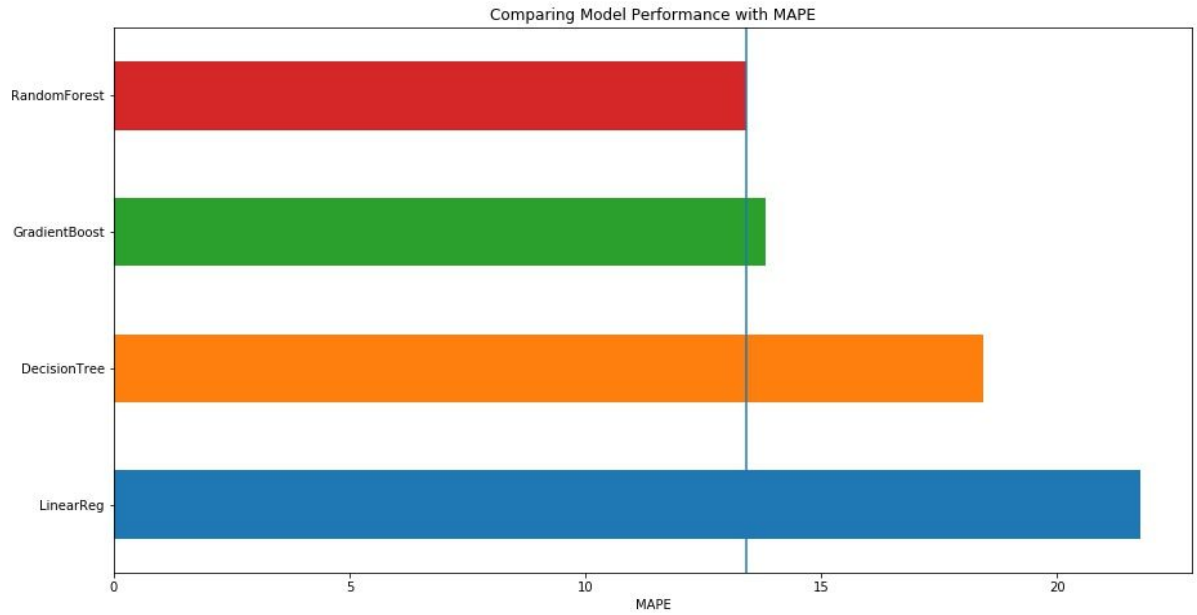
Then, I build a Gradient Boosting Regressor Model using the default parameters.
Mean Squared Error: 6228482868.33
Root Mean Squared Error: 78920.74
r-squared score : 0.8281928113627098
Mean Absolute Deviation (MAE): 56373.65
Mean Absolute Percent Error (MAPE): 13.83

Then, I build a Random Forest Regressor Model using the default parameters.
Mean Squared Error (MSE): 6302252436.45
Root Mean Squared Error (RMSE): 79386.73
r-squared score : 0.8261579431012475
Mean Absolute Deviation (MAE): 54840.62
Mean Absolute Percent Error (MAPE): 13.43

I drew a horizontal bar plot to compare r2_score and Mape of different regressor.

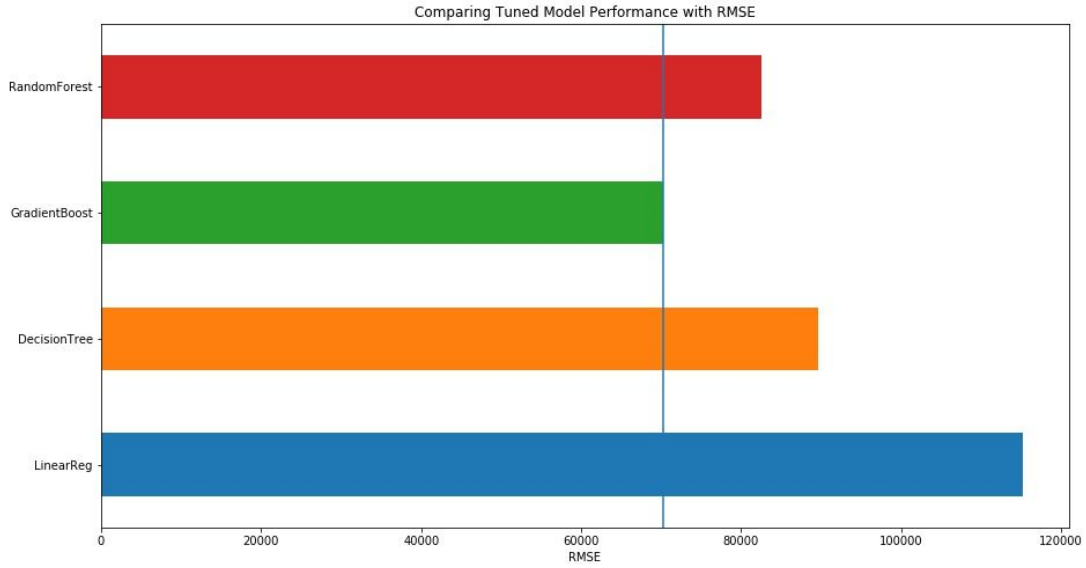


According to the r2_score, Gradient Boosted Regression model is the best performing model.
The random Forest Regression model is the second better performing model for this dataset.

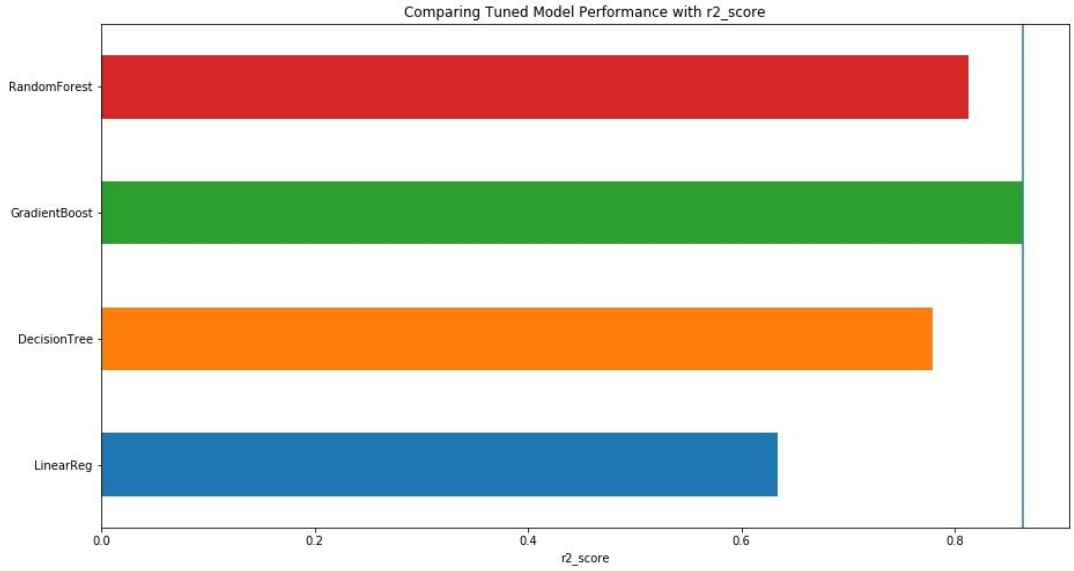


According to the Mean absolute percentage error (Mape), Random Forest Regressor model is the better performing model. Gradient Boosting Regressor model is the second better performing model.

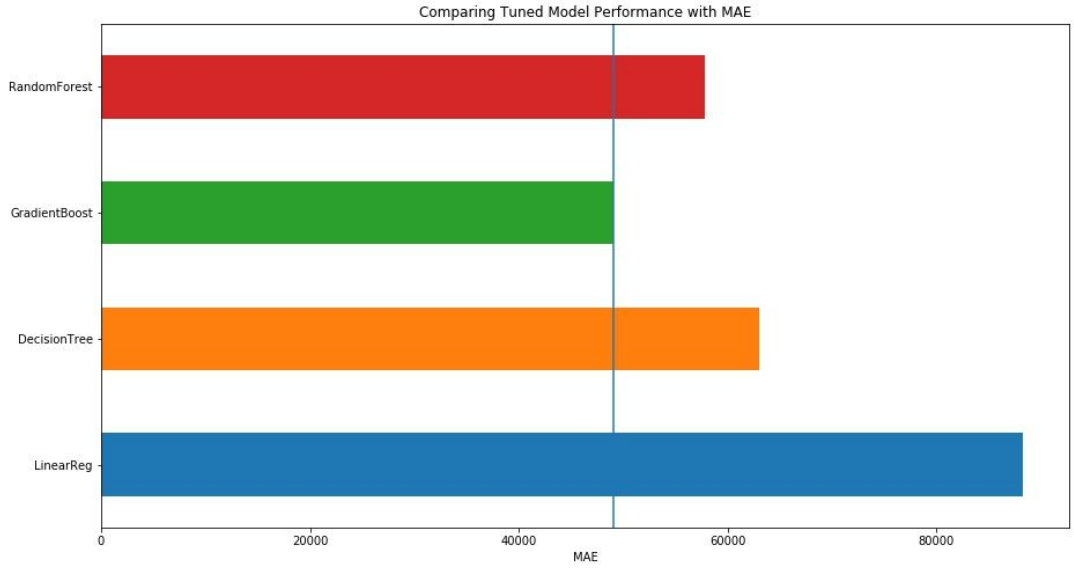
Next, I used GridSearchCV to tune the Hyperparameters of the model to improve the performance of the model. GridSearchCV is a cross-validation method which allows us to use a set of parameters that we want to try with a given model. Each of those parameters is used to perform cross-validation and finally, the best parameters for the model is saved. I created a new dataframe named tuned_metrics to save the metrics of the tuned models. I used the method .get_params() to find all the parameters of the model. For Linear Regression model, I used 'copy_X', 'fit_intercept', and 'normalize' parameters inside the param_grid. The param_grid is a dictionary with parameters names as keys and lists of parameter settings to try as values. I used cv = 5, which is the number of folds used. We can get the best parameter for any Regression model for this data set using the .best_params_ attribute of GridSearchCV. I tuned the Linear Regression model and Decision Tree Regressor model using GridSearchCV. Then, I tuned the Gradient Boosting Regressor model and Random Forest Regressor model using RandomizedSearchCV. RandomizedSearchCV helps us to minimize the computation time because GridSearchCV can take very long computational time to for both Gradient Boosting Regressor and Random Forest Regressor.



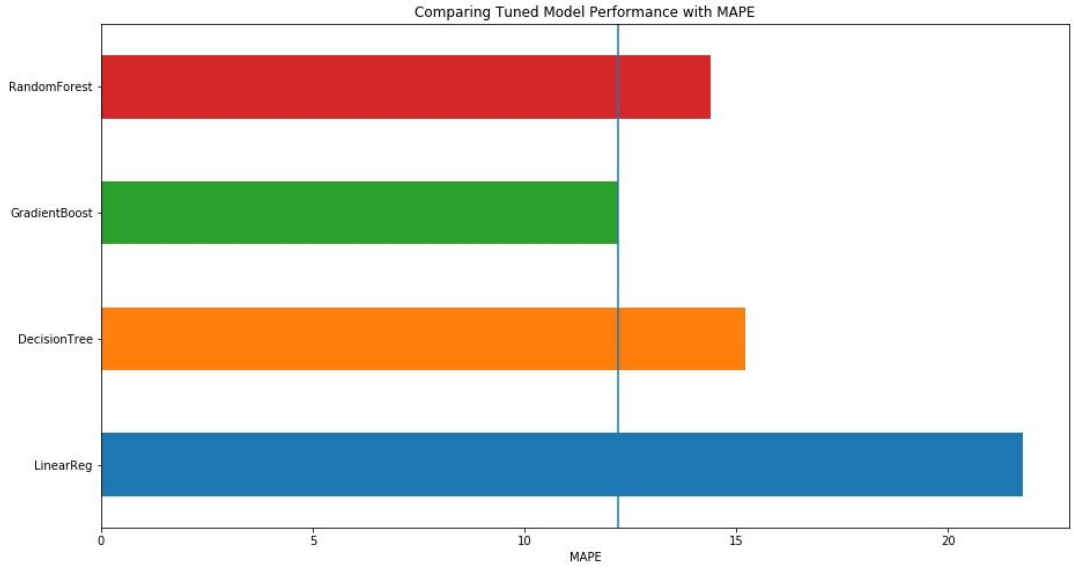
According to the Root Mean Squared Error, Gradient Boost Regression model is the best performing model with the lowest error 115222.



According to the r2_score, Gradient Boost Regression model is the best performing model with the highest score of 0.86.



According to the Mean Absolute Error (MAE), Gradient Boost Regression model is the best performing model with lowest error 49098.4.



According to the Mean absolute percentage error (MAPE), Gradient Boost Regression model is the best performing model with lowest percentage error 12%. All of the metrics suggest that Gradient Boosted Regression model is the better performing model for this dataset.

Gradient Boosting Regression model is a good model to predict house price because it is better than a random guess and it outperforms the other three Regression Model. The model may be improved in the future with more data collection. Many other Regression models which are not

included in this project can also be built and tried. I would recommend this Gradient Boosting Regression model to predict house price.