House Price Prediction in Natural Hazard Prone Areas

Capstone Project 1: Machine Learning

Introduction

In this project, different Regression models i.e. Linear Regression, Decision Tree Regression, Gradient Boosted Regression, and Random Forest Regression were used. The performance of those models using R^2 were compared. Based on these performance score, better performing model were suggested to predict house price.

First, the data was divided into independent variable X and dependent variable y. Independent variable X was used to predict the target variable y. The price, id and date column were dropped from the new_df dataframe to create the variable X. The price column from the new_df dataframe were used to create the variable y. In this project, different metrics were used to 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, Root mean squared error and R-squared score were used to evaluate the performance of the regression model. In order to save the metrics of the model, a data frame was created and it was named metrics. Next, the data was splitted into training and testing set. 80% of the randomly selected data were kept as a training set and 20% of the randomly selected data as a testing set. The model was learned using the 80% of the data, and the rest 20% testing data were used as an unseen future dataset to predict the house price.

Linear Regression Model was built using the default parameters, and the model was fitted using the training dataset. X_test data was used to predict using the model. Then, Mean squared error (MSE), Root mean squared error (RMSE), R-squared score (r2_score), Mean absolute deviation (MAD), and Mean absolute percent error (MAPE) were calculated.

Mean Squared Error (MSE): 54296634556.66
Root Mean Squared Error (RMSE): 233016.3826
r2_score: 0.4637
Mean Absolute Error (MAE): 182963.47
Mean Absolute Percent Error (MAPE): 19.28

Feature Selection

Backward elimination method of feature selection were used. Feature selection is the process of selecting a subset of relevant features that may improve the performance of the model. First, the worst attribute from the feature was removed. The date_sold_month were removed because it has a very weak correlation with the price of the house. Then, year_built_decade_mapped were removed from the feature set. Then, a univariate feature selection package called SelectKbest from the sklearn library was tried. Below are correlation coefficient for different features.

date_sold_month	-0.065022
Year_built_decade	-0.052772
Landslide	-0.002046
Faultzone	0.036062
zip_mapped	0.139833
bathrooms_rounded	0.250558
house_type_mapped	0.317384
Liquefaction	0.361638
bedrooms	0.368206
lot_size_rounded	0.428212
sqft_rounded	0.521310
fire_hazard	NaN

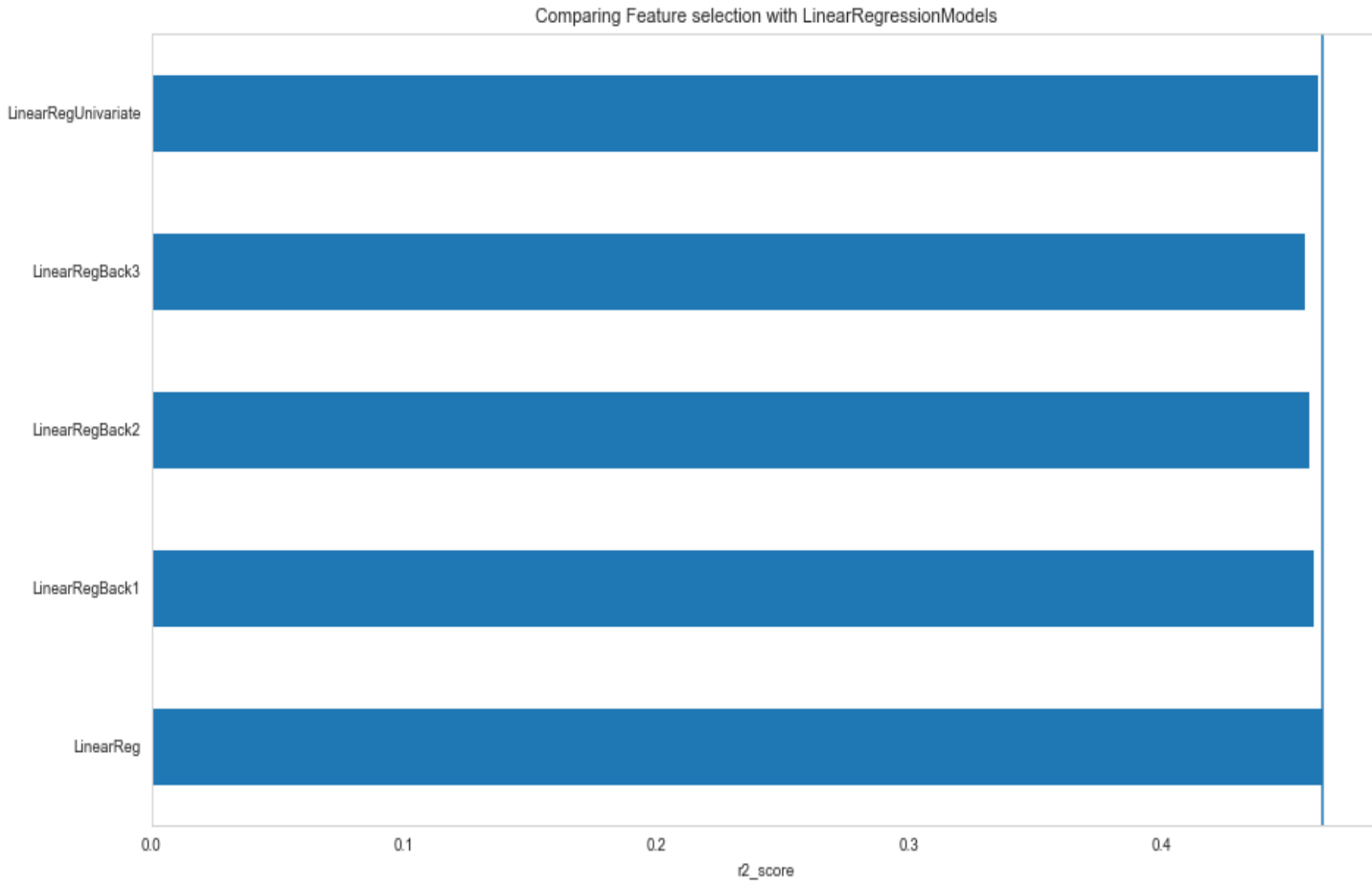

Fig 1a: Comparing Feature selection with Linear Regression models using R2 score

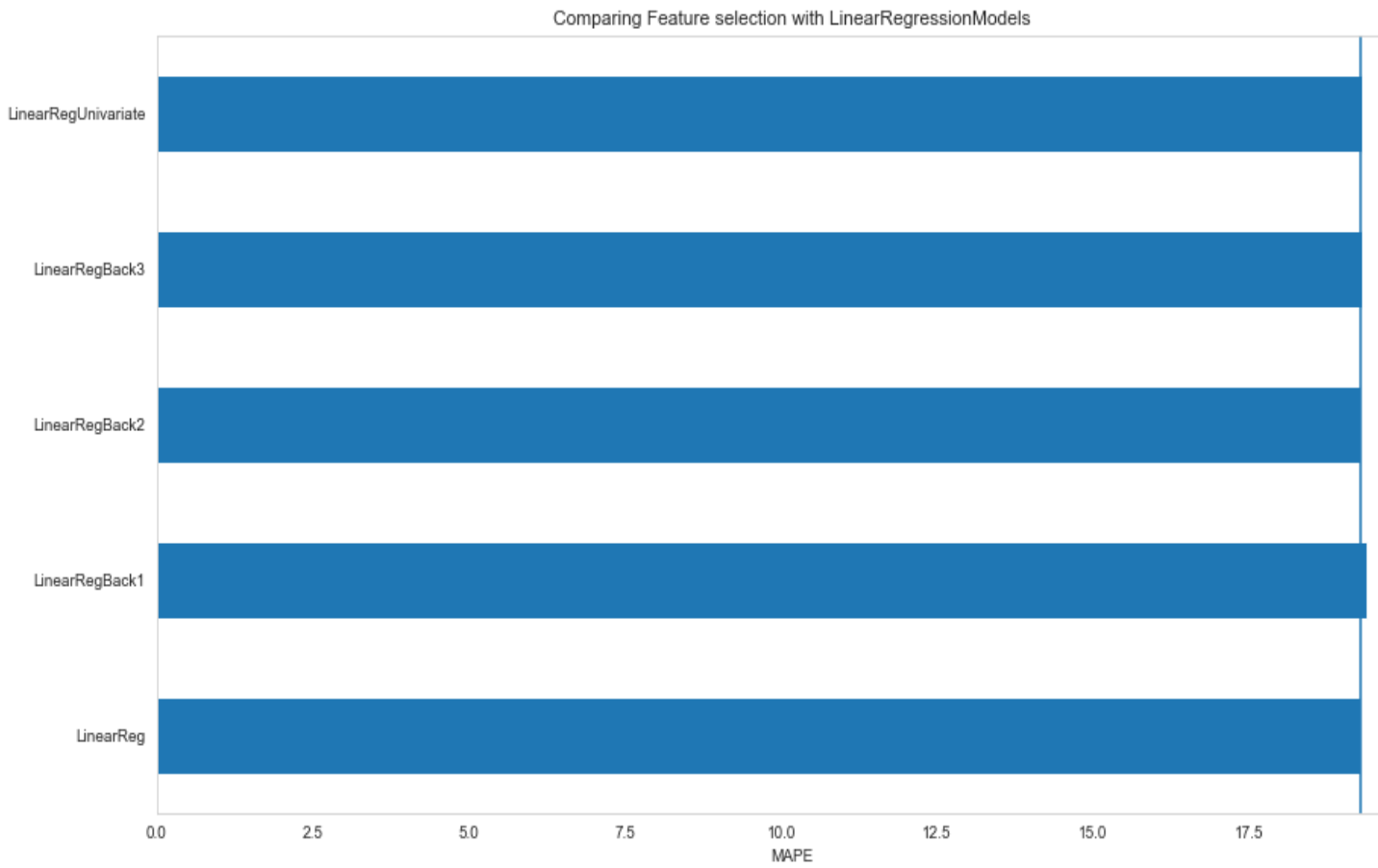

Fig 1b: Comparing Feature selection with Linear Regression models using MAPE score

Comparing all different Linear Regression model with feature selection, both bar plots (Fig 1a and 1b) suggest that we should keep all the features to better predict the house price.

Decision Tree Regression

Decision Tree Regressor Model using the default parameters were built and listed performance score below. The sample decision Tree regression flowchart is shown in Fig 2

Mean Squared Error: 43361291535.89
Root Mean Squared Error: 208233.74
r-squared score : 0.5716779806579388
Mean Absolute Deviation (MAE): 155321.44
Mean Absolute Percent Error (MAPE): 16.13

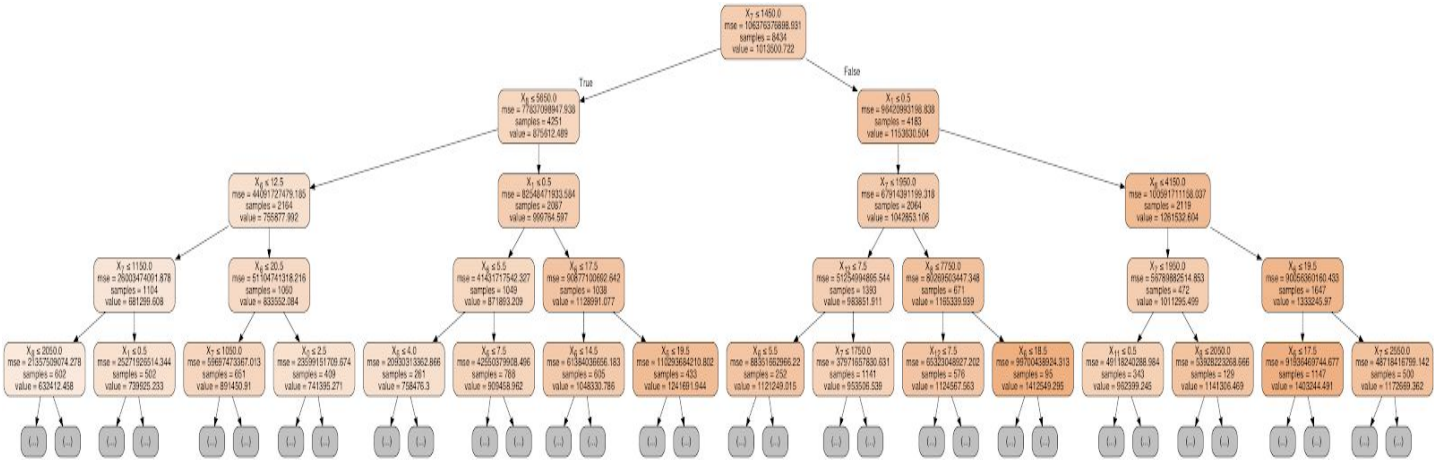

Fig 2. Sample Decision Tree Diagram

Gradient Boosting Regression

Gradient regression Model using the default parameters were built and listed performance score below.
Mean Squared Error: 25679823410.25
Root Mean Squared Error: 160249.25
r-squared score : 0.7463351890632407
Mean Absolute Deviation (MAE): 124156.64
Mean Absolute Percent Error (MAPE): 13.02

Random Forest regression

Random Forest regression Model using the default parameters were built and listed performance score below.
Mean Squared Error (MSE): 25762504105.35
Root Mean Squared Error (RMSE): 160507.02
r-squared score : 0.745518470717669
Mean Absolute Deviation (MAE): 122098.99
Mean Absolute Percent Error (MAPE): 12.92

The horizontal bar plot (Fig 3) is shown below to compare r2_score and MAPE for different regressor.

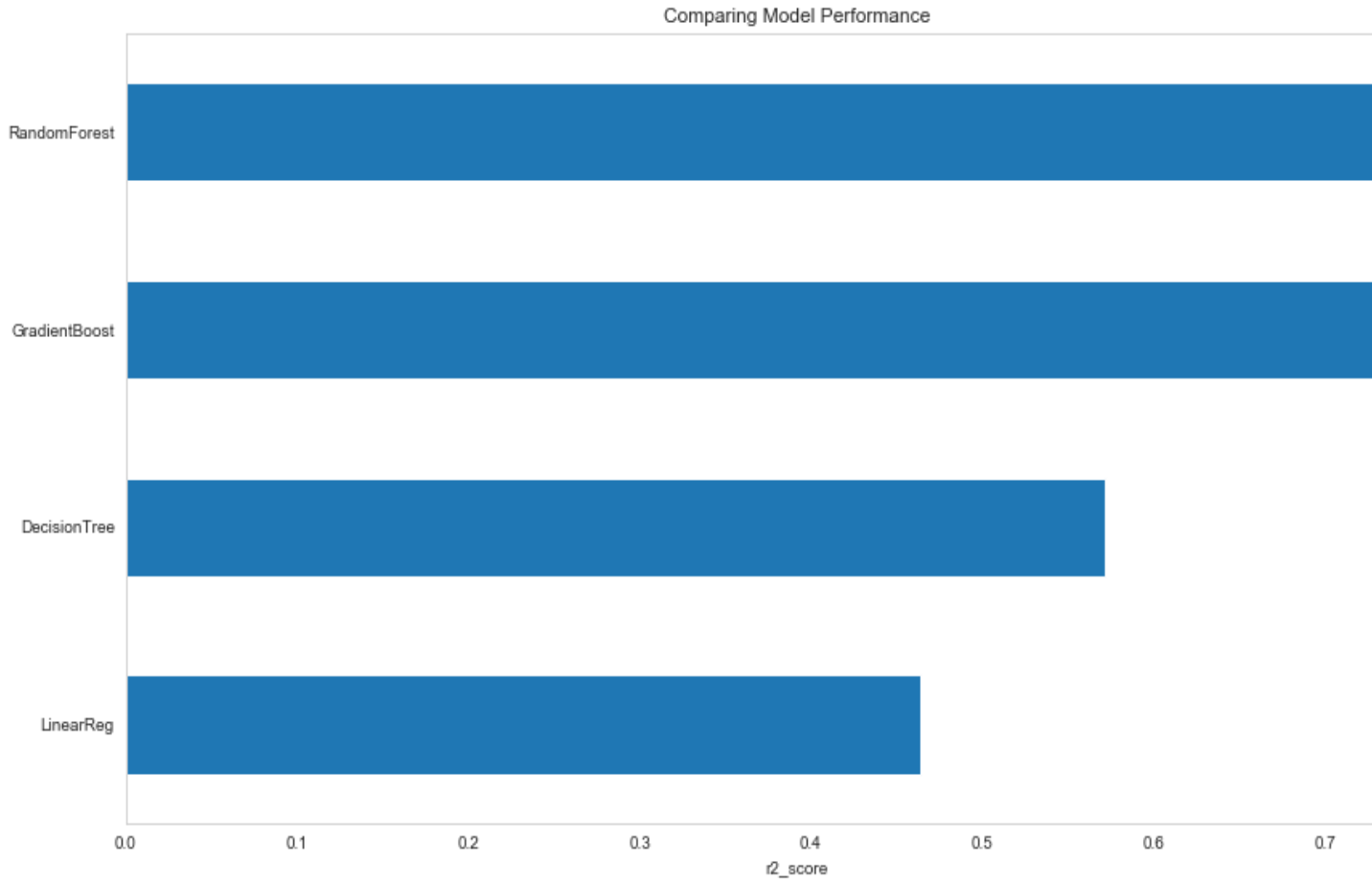

Fig 3: Comparing model performance for different regression models using r2 score

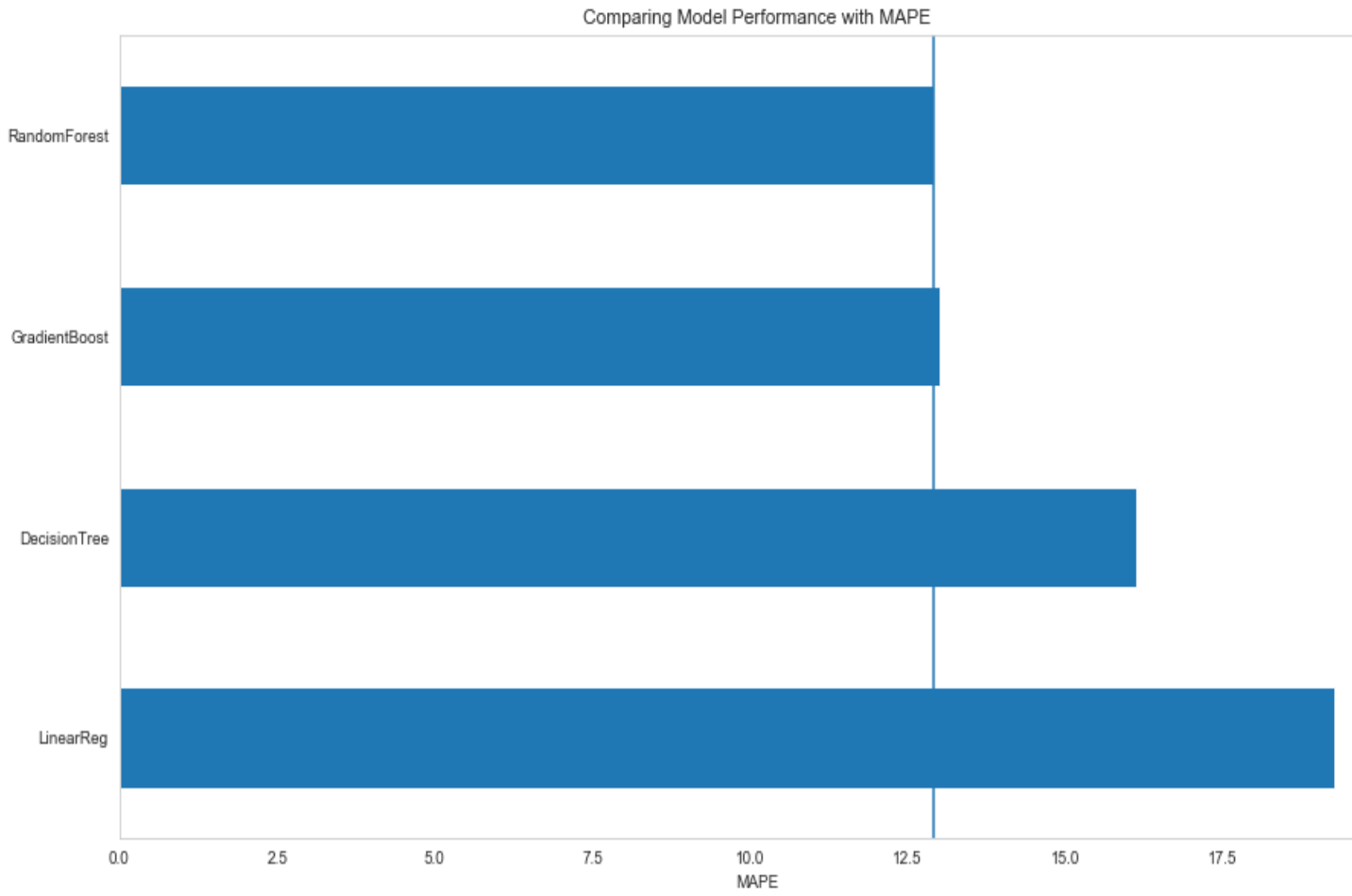

Fig 4. Comparing model performance for different regression models using MAPE score

According to r2_score (Fig 3), Gradient Boosted Regression model was the best performing model. The random Forest Regression model was the second better performing model for this dataset.

According to Mean absolute percentage error (Mape) (Fig 4), Random Forest Regressor model was the better performing model. Gradient Boosting Regressor model was the second better performing model.

Hyperparameter Tuning

Next, GridSearchCV were used 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 was used to perform cross-validation and finally, the best parameters for the model is saved. A new dataframe was created and was named tuned_metrics to save

the metrics of the tuned models. The method .get_params() was used to find all the parameters of the model. For Linear Regression model, 'copy_X', 'fit_intercept', and 'normalize' parameters were used inside the param_grid. The param_grid is a dictionary with parameters names as keys and lists of parameter settings to try as values. cv = 5 was used, 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. The Linear Regression model and Decision Tree Regressor model were tuned using GridSearchCV. Then, the Gradient Boosting Regressor model and Random Forest Regressor model were tuned using RandomizedSearchCV. RandomizedSearchCV helped to minimize the computation time because GridSearchCV can take very long computational time to for both Gradient Boosting Regressor and Random Forest Regressor. Tuned model performance comparison for different regression models using RMSE score is shown in Fig 5.

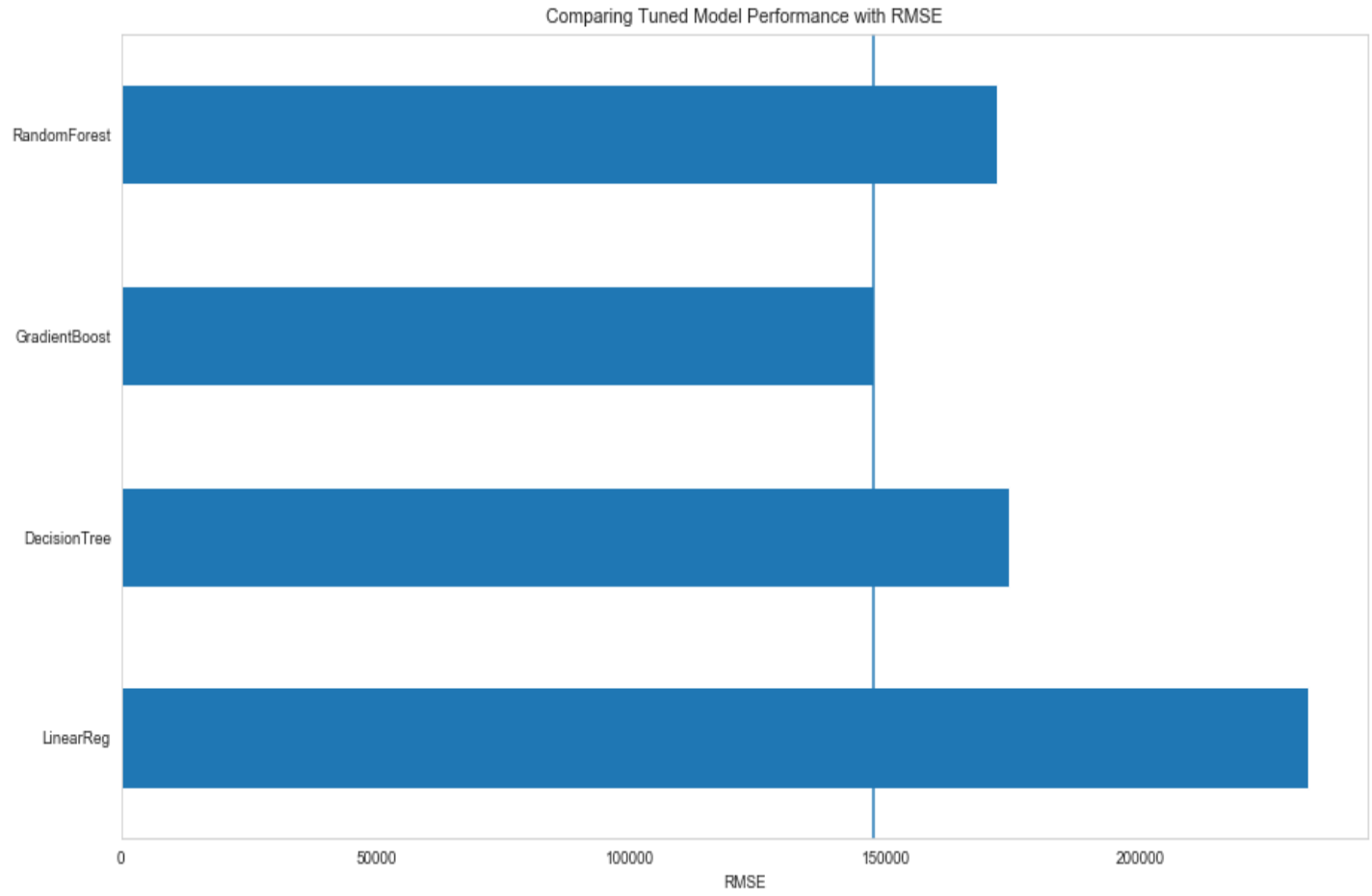

Fig 5. Comparing tuned model performance for different regression models using RMSE score

According to Root Mean Squared Error (Fig 5), Gradient Boost Regression model was the best performing model with the lowest error 147773.

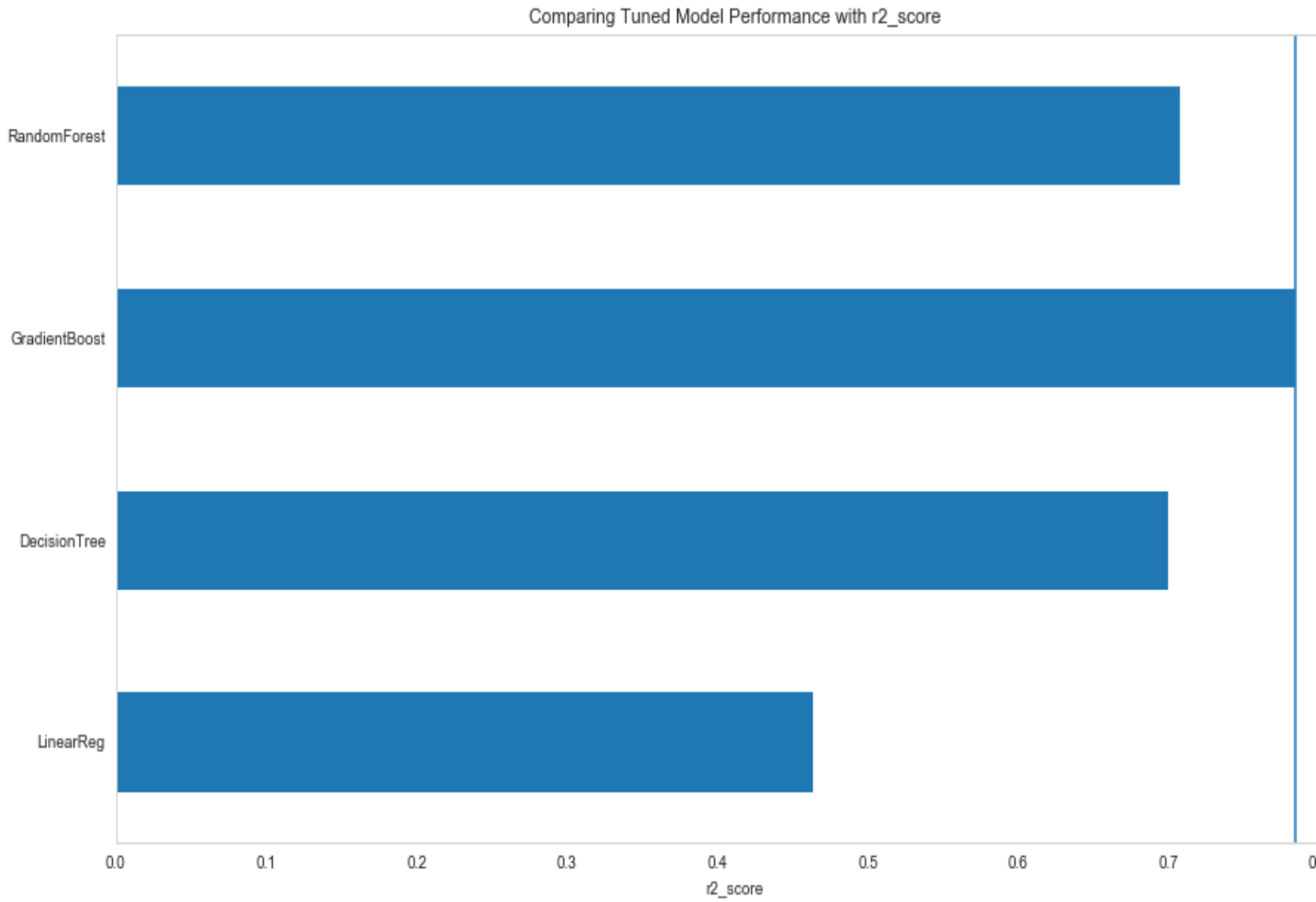

Fig 6. Comparing tuned model performance for different regression models using R2 score

According to r2_score (Fig 6), Gradient Boost Regression model was the best performing model with the highest score of 0.78.

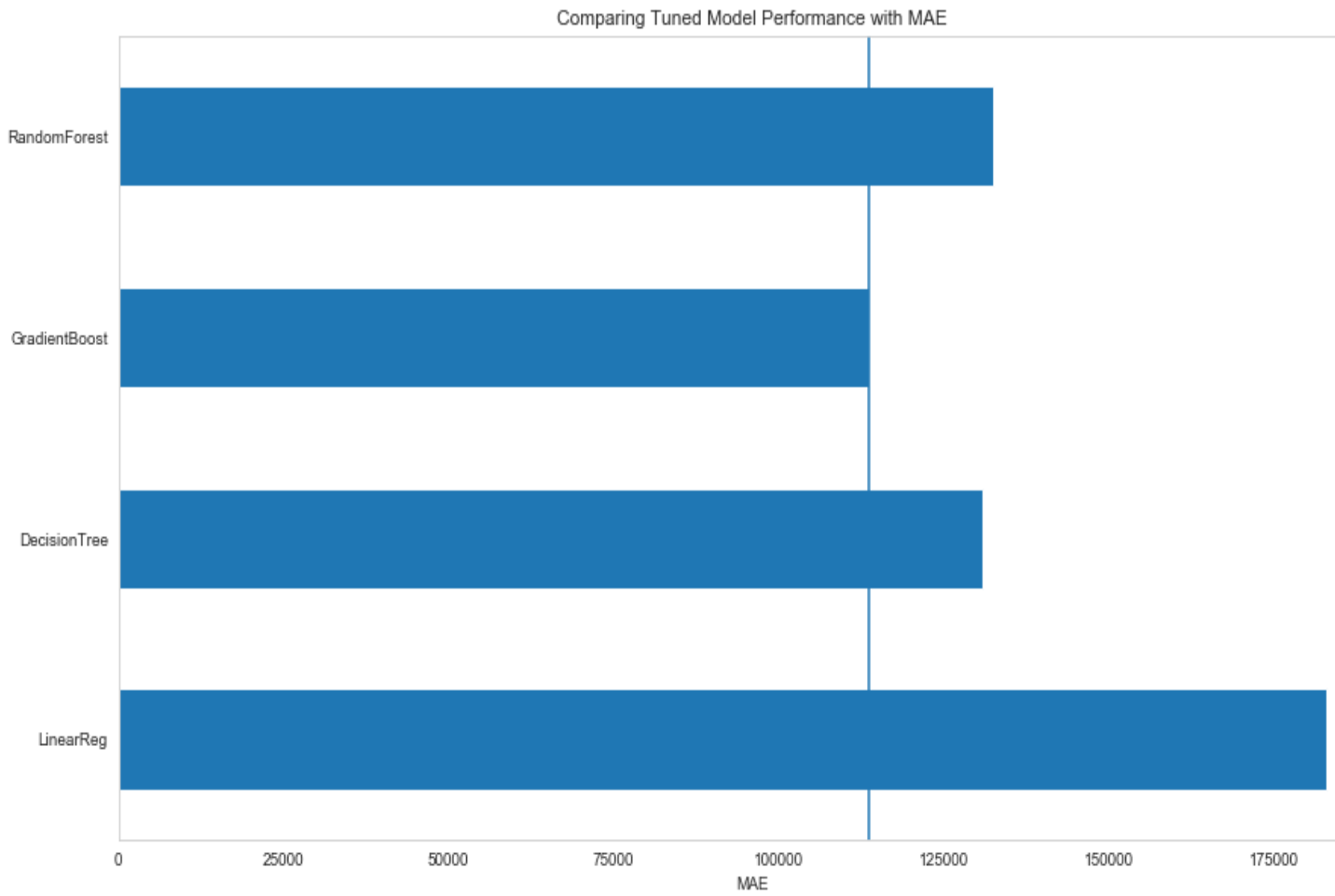

Fig 7. Comparing tuned model performance for different regression models using MAE score

According to Mean Absolute Error (MAE) (Fig 7), Gradient Boost Regression model was the best performing model with lowest error 113737.

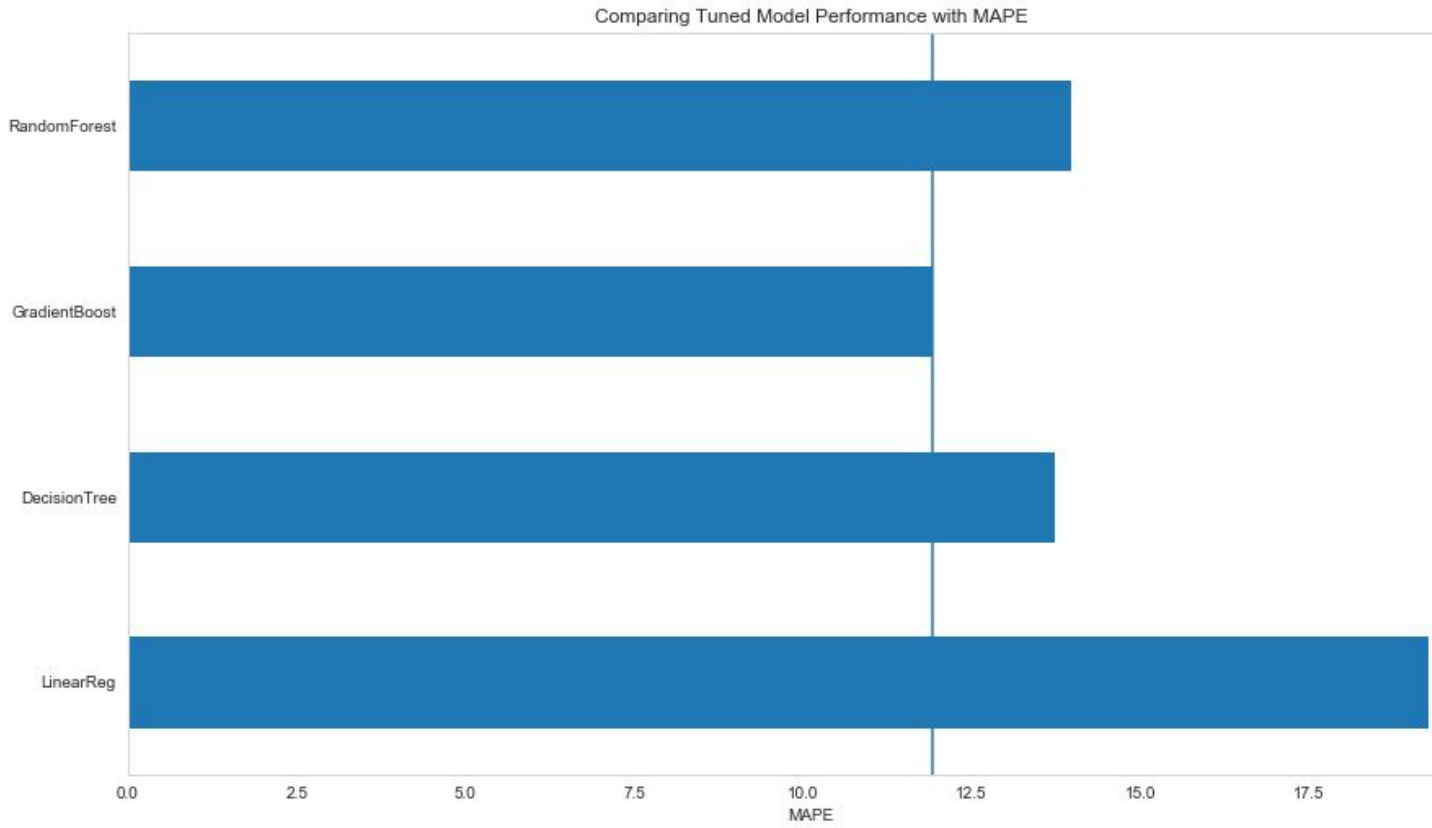

Fig 8. Comparing tuned model performance for different regression models using MAPE score

According to the Mean absolute percentage error (MAPE) (Fig 8), Gradient Boost Regression model was the best performing model with lowest percentage error 12%. All of the metrics suggest that Gradient Boosted Regression model was the better performing model for this dataset.

Summary

Gradient Boosting Regression model was a good model to predict house price because it was better than a random guess and it outperformed the other three Regression Model. The model may be improved in the future with more data collection. Many other Regression models which were not included in this project can also be built and tried. For future work, I would recommend this Gradient Boosting Regression model to predict house price.

Ipython notebook for machine learning:
https://nbviewer.jupyter.org/github/umaraju18/CapStone-Project/blob/master/sanjose_machine_learning_model.ipynb