Machine Learning

In this project, I used supervised machine learning algorithms to predict house price. I tried seven Regression models which are **Linear Regression (LR), Ridge Regression(RR), Lasso Regression(LassoR), Support Vector Regression(SVR), Decision Tree Regression (DTR), Random Forest Regression(RFR),** and **Gradient Boosted Regression(GBR).** I compared the performance of those models using Mean Squared Error (MSE), R2 score, Root Mean Square Error (RMSE), and Mean_Absolute_Error (MAE) (I evaluated the performance of SVR model only according to R2 score). I finally recommended the better performing model to predict house price.

a. Additional Data Preparation Before Applying Models

1. I applied **one-hot encoding** on 'waterfront', 'floors', 'view', 'condition', 'grade', 'basement_present', 'renovated' features.

2. I created the copy of **'h_data'** dataset as **'h_data_copy'.** I applied one-hot encoding on 6 zip codes, which have highest mean of house prices on **'h_data'** dataset and I dropped the "zipcode" feature. I used his version on LR, RL, LassoR, and SVR models. I used both version of "zipcode" feature (one-hot encoding applied version and original version) on tree-based models.

3. I applied **"pandas.profiling"** to see the latest changes and their effects on datasets**. 'renovated1'** feature is highly correlated with yr_renovated feature ($\rho = 0.99997$). I decided to drop this feature.

4. First, I divided the data into independent variables **"X** and **X_copy"** and target variables **"y** and **y_copy"**. Independent variables are the features I am going to use to predict the target variables.

5. I split the dataset into training (70 %) and test (30 %) sets. The model will learn or fit using the 70% of the data, and the rest 30% testing data will be used as an unseen future dataset to predict the house price.

6. I created a new dataframe named **"evaluation_matrix"** to store the metrics of models.

b. Machine Learning Algorithms

b.1. Linear Regression(LR) : I build a Linear Regression Model using the default parameters. I used **X data** for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated Mean Squared Error (MSE), Root Mean Square Error (RMSE), R-squared score (R2_score), Mean absolute error (MAE) and stored them into evaluation matrix data frame. **R2 score is 0.751971**. I also made cross validation (CV) with 5 fold and **CV R2 = 0.749681.**

b.2. Ridge Regression (RR): I build a Ridge Regression Model using the "**alpha = 0.01, normalize = True"** parameters. I used **X data** for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated MSE, RMSE, R2_score, MAE and stored them into evaluation matrix data frame. **R2 score is 0.751886**. I also made cross validation (CV) with 5 fold and **CV R2 = 0.749797.** The results are nearly as same as LR model.

b.3. Lasso Regression (LassoR): I build a Lasso Regression Model using the "**alpha = 1, normalize = True"** parameters. I used **X data** for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated MSE, RMSE, R2_score, MAE and stored them into evaluation matrix data frame. **R2 score is 0. 0.751895**. I also made cross validation (CV) with 5 fold and **CV R2 = 0.749738.** The results are nearly as same as RR and LR model.

b.4. Support Vector Regression(SVR): Before building model, I scaled the data. After scaling, I build a **Support Vector Regression** Models using the "kernel **= 'rbf' and "**kernel **= 'linear'"** parameters separately. I used **X data** for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated MSE, RMSE, R2_score, MAE and stored them into evaluation matrix data frame. **R2 score with** kernel **= 'rbf' parameter is 0.673261, R2 score with** kernel **= 'linear' parameter is 0.746757**.

b.5. Decision Tree (DT): I build a Decision Tree Model using the default parameters, and I fit the model using the training dataset. I used both **X** and **X_copy** data for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated Mean Squared Error (MSE), Root Mean Square Error (RMSE), R-squared score (R2_score), Mean absolute error (MAE) for both datasets, and stored them into evaluation matrix data frame. **R2 score for X data is 0.56713, R2 score for X_copy data is 0.573935. There is no significant change.** I also made cross validation (CV) with 5 fold and **CV R2 for X data is 0.59616, CV R2 for X_copy data is 0.62108.**

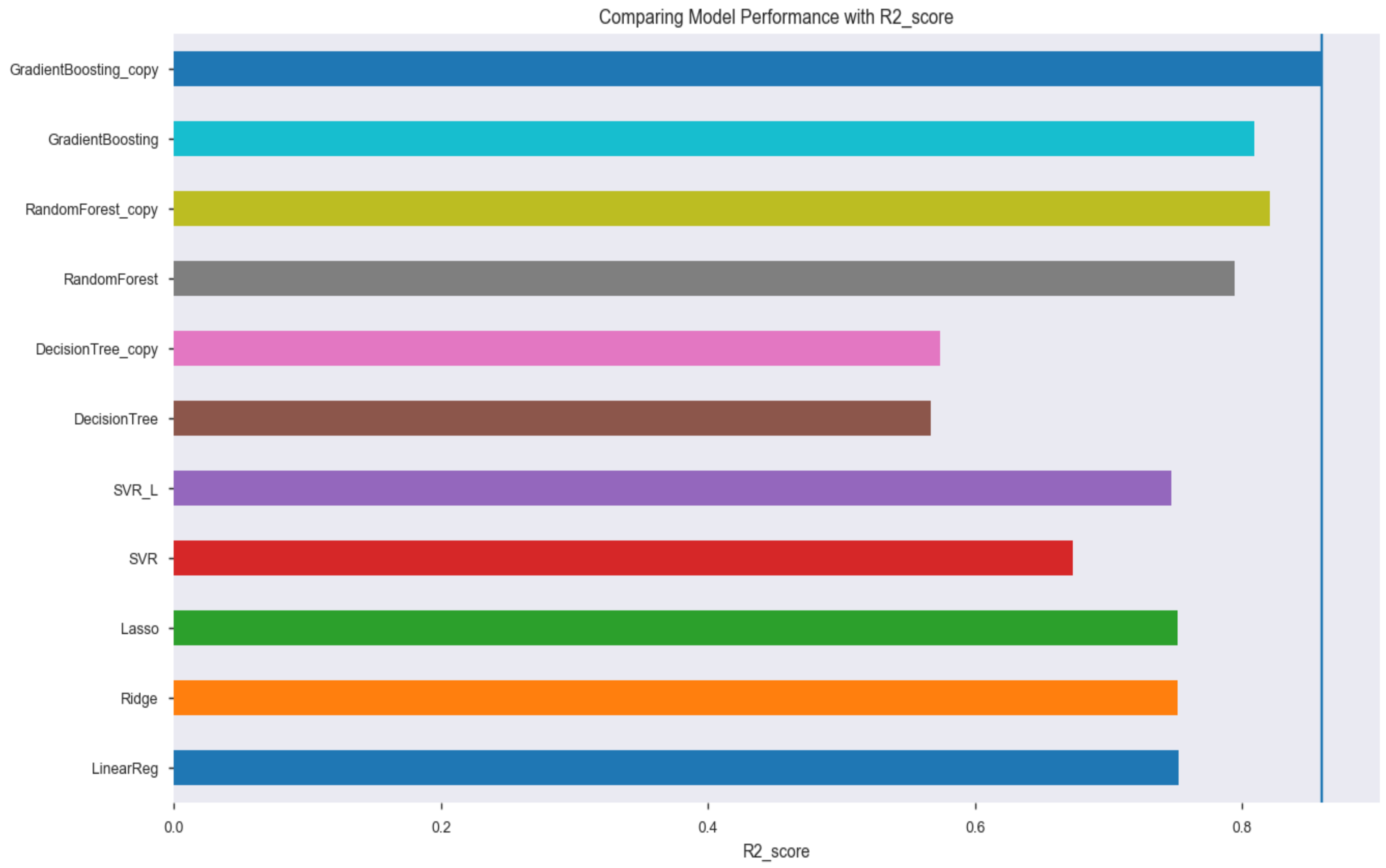
b.6. Random Forest (RF): I build a Random Forest Model using the **"n_estimators=100 and n_estimators=300"** parameters, and I fit the model using the training dataset. I used both **X** and **X_copy** data for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated Mean Squared Error (MSE), Root Mean Square Error (RMSE), R-squared score (R2_score), Mean absolute error (MAE) for both datasets, and stored them into evaluation matrix data frame. **R2 score for X data is 0.794496, R2 score for X_copy data is 0.820588.** I also made cross validation (CV) with 5 fold and **CV R2 for X data is 0.78390.**

b.7. Gradient Boosting Regressor (GBR): I build a Gradient Boosting Regressor Model using the **"n_estimators=300"** parameter, and I fit the model using the training dataset. I used both **X** and **X_copy** data for data split, and I fit the model. I used **X_test** data to predict target variable. Then, I calculated Mean Squared Error (MSE), Root Mean Square Error (RMSE), R-squared score (R2_score), Mean absolute error (MAE) for both datasets, and stored them into evaluation matrix data frame. **R2 score for X data is 0.808864, R2 score for X_copy data is 0.859649.**

According to the R2_score, Mean_Absolute_Error(MAE), and Root_Mean_Squared_Error(RMSE), **Gradient Boosting Regressor model (with h_data_copy dataset)** is the **better** performing model. **Rondom Forest Regressor model (with h_data_copy dataset)** is the **second better** performing model.

	LinearReg	**Ridge**	**Lasso**	**SVR**	**SVR_L**
Mean_Squared_Error(MSE)	3.37764e+10	3.3788e+10	3.37866e+10	0.326739	0.253243
Root_Mean_Squared_Error(RMSE)	183784	183815	183811	0.571611	0.503232
R2_score	0.751971	0.751886	0.751895	0.673261	0.746757
Mean_Absolute_Error(MAE)	125387	125394	125416	0.328629	0.332053

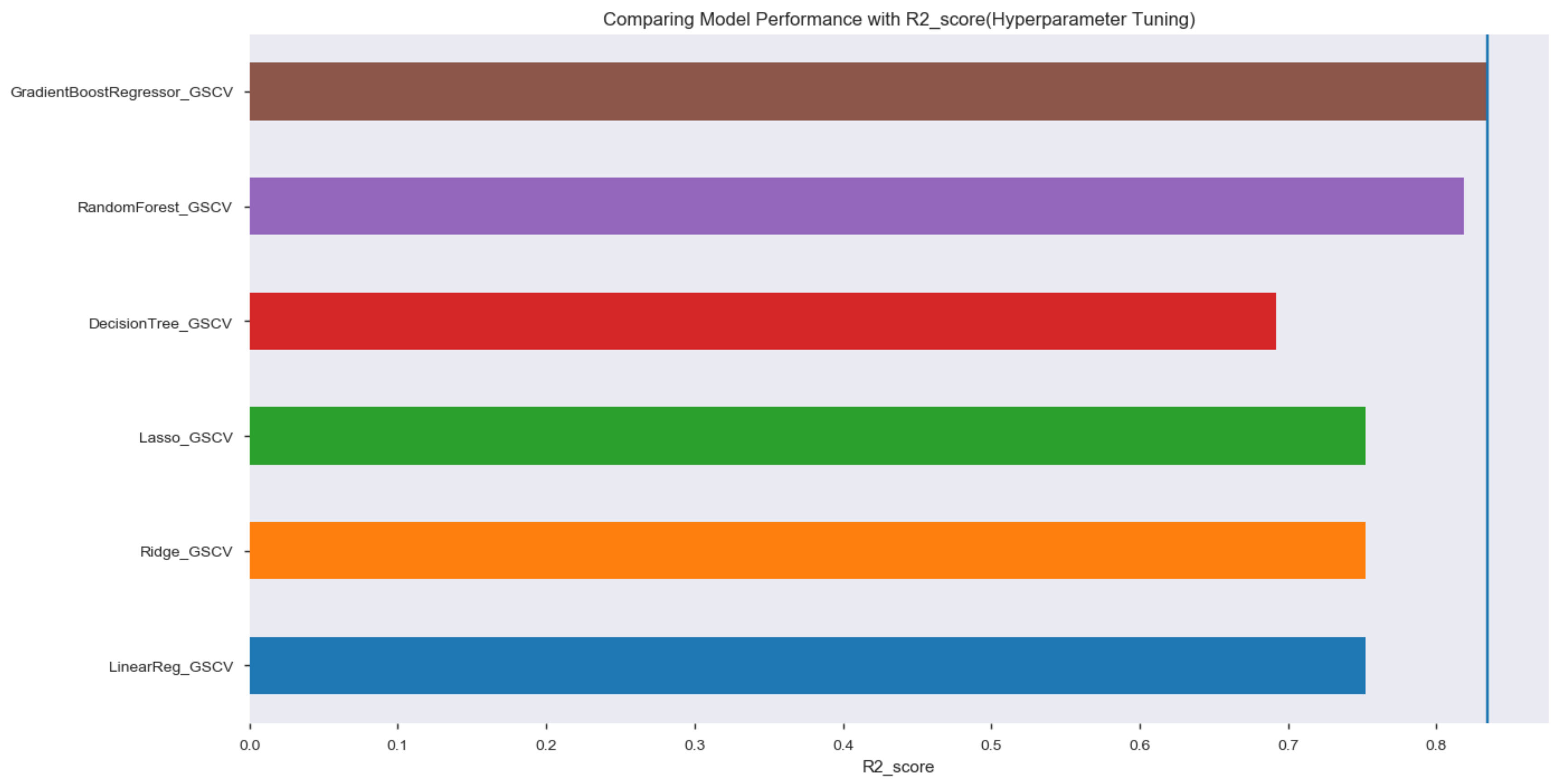
	Decision Tree	**Decision Tree_ copy**	**Random Forest**	**Random Forest_ copy**	**Gradient Boosting**	**Gradient Boosting_ copy**
Mean_Squared_ Error(MSE)	5.89478e+10	5.80211e+10	2.79853e+10	2.44322e+10	2.602865e+10	1.911278e+10
Root_Mean_ Squared_Error(RMSE)	242792	240876	167288	156308	161334	138249
R2_score	0.56713	0.573935	0.794496	0.820588	0.8088645	0.8596496
Mean_Absolute_ Error(MAE)	150482	127169	109315	88127.2	110167	83950.4


Comparing Model Performance with R2_score
GradientBoosting_copy
GradientBoosting
RandomForest_copy
RandomForest
DecisionTree_copy
DecisionTree
SVR_L
SVR
Lasso
Ridge
LinearReg
0.0
0.2
0.4
0.6
0.8
R2_score

c. Hyperparameter Tuning

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. For Ridge and Lasso model, I used "alpha" parameter 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, Ridge, Lasso, and Decision Tree Regressor model using GridSearchCV. Then, I tuned the Random Forest Regressor and Gradient Boosting 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.

	Linear Reg_ GSCV	**Ridge_ GSCV**	**Lasso_ GSCV**	**Decision Tree_ GSCV**	**Random Forest_ RSCV**	**Gradient Boost Regressor_ RSCV**
Mean_ Squared_ Error (MSE)	3.37764e+10	3.3788e+10	3.37639e+10	4.19655e+10	2.47025e+10	2.25726e+10
Root_ Mean_ Squared_ Error (RMSE)	183784	183815	183749	204855	157170	150242
R2_score	0.751971	0.751886	0.752063	0.691836	0.818603	0.834243
Mean_ Absolute_ Error (MAE)	125387	125394	125386	111556	88326.1	83265.6



According to the R2_score, Gradient Boost Regression model is the best performing model with the highest score of 0.834243.