

Brian Elinsky

Professor Lawrence Fulton, Ph.D.

2020SP_MSDS_422-DL_SEC55

1 May 2020

Assignment 2: Evaluating Regression Models

PROBLEM DESCRIPTION

Given a set of explanatory variables for a house, my business objective is to predict its market value. In this fictional scenario, I'm advising a real estate brokerage firm. The firm currently uses conventional methods to price houses. This model will complement those methods. The model will be used to help ensure that the company buys houses below market value, and sells them above market value, ensuring a profit.

DATA PREPARATION

I dropped categorical variables that were missing over 10% of the data. This included: 'Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'. For numeric features, I imputed missing values with medians. For categorical features, I imputed missing values with the most frequent value. I encoded categorical features to dummy arrays. I did not scale any variables.

RESEARCH DESIGN, MODELING METHODS AND EVALUATION

My plan was to build random forest regressors with varying depths, then compare their test errors as the number of trees increases. This would allow me to determine the optimal depth and number of trees. A max depth of 20 outperformed 10, and was as good as

higher depth numbers. Next, I held depth constant at 20, but varied the number of features. Unexpectedly, `max_features=auto` outperformed `log2` and `sqrt`.

Using the same pre-processing pipeline, I next evaluated gradient boosting regressors. The models with the lowest learning rates seemed to perform best. It looked as if the test error would continue to decrease had I included more than 1000 trees in the forest.

Since many of the best models from my first set of experiments used parameters on the bounds of my grid search, I conducted a much larger grid search to select the final models.


The final random forest performed marginally better than my linear regression models from a few weeks ago. However, the gradient boosting model performs substantially better. Submitting both models to Kaggle resulted in a Log RMSE of 0.14935 for the random forest, and a Log RMSE of 0.12768 for the gradient boosting model (Account User ID 4810027)(<https://www.kaggle.com/brianelinsky>)

MANAGEMENT RECOMMENDATIONS

I would recommend deploying the final version of the gradient boosting model to production. I would advise management to not replace the conventional pricing methods with this model, rather to use the model to supplement the existing methods. With a mean housing price of ~\$180k, a model with an RMSE of ~\$25k will give you a rough estimate of the housing price, but not a hyper-accurate one. Additional work could be done to improve the model. That could include building an ensemble model of different regressors.

>_

kaggle competitions submit -c house-prices-advanced-regression-techniques -f submission.csv
-m "Message"


?

4 submissions for [Brian Elinsky](#)


Sort by

Most recent

All

Successful

Selected

Submission and Description	Public Score	Use for Final Score
<div>rf_predictions.csv</div> <div>2 days ago by Brian Elinsky</div> <div>RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=10, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None, oob_score=False, random_state=79, verbose=0, warm_start=False)</div>	0.14935	<input type="checkbox"/>
<div>rf_predictions.csv</div> <div>2 days ago by Brian Elinsky</div> <div>RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=10, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None, oob_score=False, random_state=79, verbose=0, warm_start=False)</div>	Error 	<input type="checkbox"/>
<div>gb_predictions.csv</div> <div>2 days ago by Brian Elinsky</div> <div>GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse', init=None, learning_rate=0.01, loss='ls', max_depth=3, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=6400, n_iter_no_change=None, presort='deprecated', random_state=79, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False)</div>	0.12768	<input checked="" type="checkbox"/>
<div>predictions.csv</div> <div>14 days ago by Brian Elinsky</div> <div>Ridge(alpha=10.0, copy_X=True, fit_intercept=False, max_iter=10, normalize=True, random_state=None, solver='cholesky', tol=0.01)</div>	0.15529	<input type="checkbox"/>

No more submissions to show

Housing Prices Kaggle Competition Lab Notebook

2020-04-13

Problem Framing

- Given a set of explanatory variables for a house, my business objective is to predict its market value. In this fictional scenario, I'm advising a real estate brokerage firm. The firm currently uses conventional methods to price houses. This model will complement those methods.
- The model will be used to help ensure that the company buys houses below market value, and sells them above market value, ensuring a profit.
- Each row in the dataset represents a single housing transaction. This is a supervised learning problem, because we are trying to predict price, and price is labeled in our dataset. Regression models will be used, not classifiers, because the predicted variable is a number.
- My machine learning system will be a batch system. I will train it using all of the available training data. Then, if the results justify, we will deploy it to production. It does not need to be an online model because the model does not need to learn on the fly.
- Performance of my model will be measured by the root mean squared error (RMSE) between the logarithm of the predicted price and the logarithm of the observed price. Using the logarithm of the price evenly penalizes errors on expensive and cheap houses. This performance measurement aligns with the business objective: consistent profit margin per house regardless of the sales price.
- I do not know the predictive accuracy required to meet the business objective. I would need to know the effectiveness of the conventional pricing methods. I would also want to know if the errors between my model and the conventional models are correlated. If they are not correlated, then I have more room for error.
- This problem is similar to most pricing problems.
- I do not have access to any housing human expertise on this project. If this were a real project, I could discuss the model with the people who employ the conventional pricing methods.
- If I had to solve this problem manually, I would probably create a linear regression in excel. Or, I would estimate the value of a house to be the average price of its neighbors. A K-Nearest Neighbors might also produce a good model.
- Assumptions I'm making:
 - The list of transactions is a representative sample of all of the transactions in Ames, Iowa.
 - Many of the features are qualitative. I'm assuming that the assessors were calibrated. That

is, they were trained to provide consistent qualitative assessments. e.g. Person A would have made the same qualitative assessment as Person B on the same house.

2020-04-15

Experiment Plan

I plan to build a preprocessing pipeline using Scikit-Learn pipeline objects. Then build a simple linear regression model and assess its accuracy with K-fold cross validation.

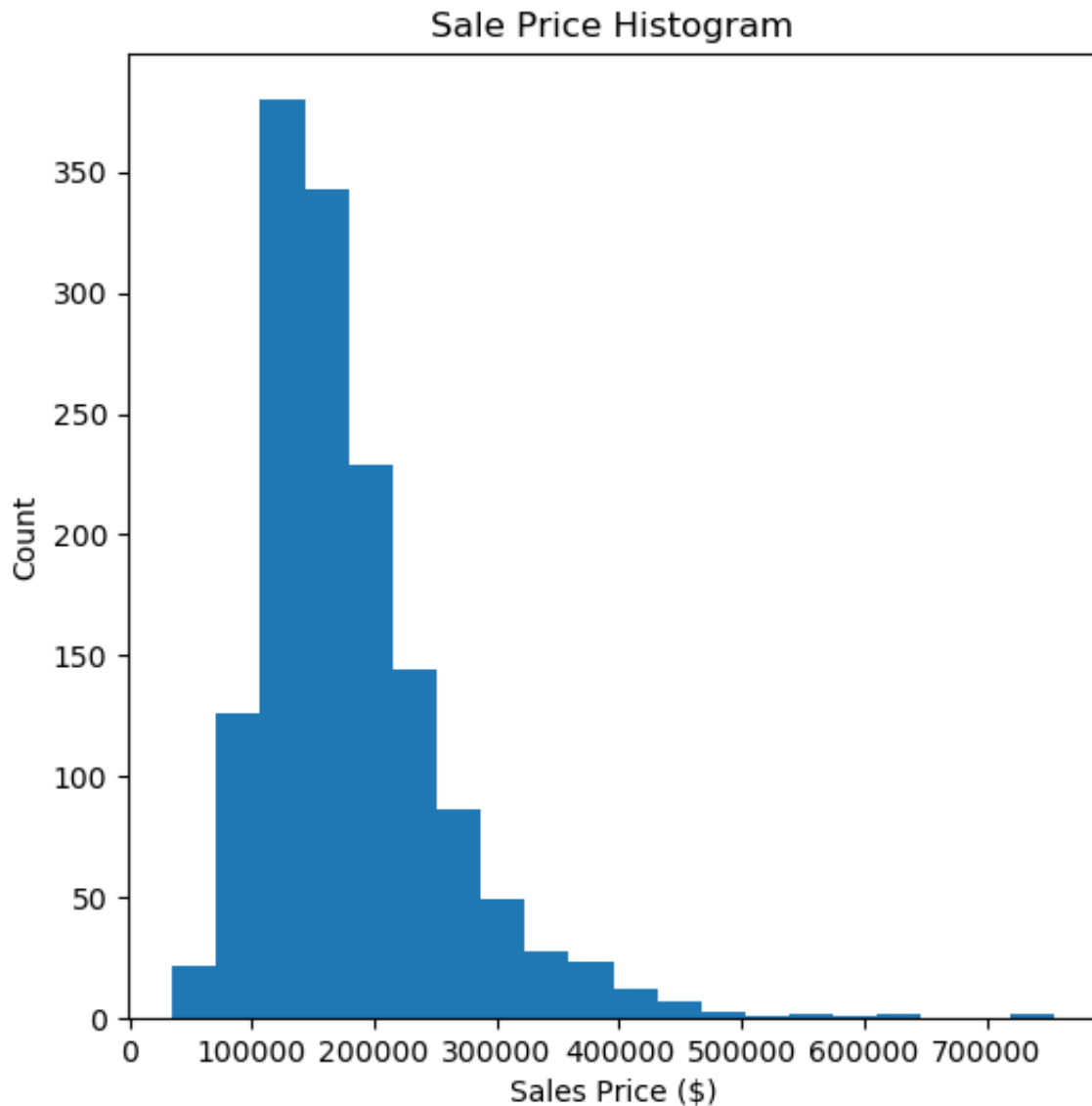
Experimental Results

Preprocessing Decisions

- I dropped categorical variables that were missing over 10% of the data. This included: 'Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'.
- For numeric features, I imputed missing values with medians.
- For categorical features, I imputed missing values with the most frequent value.
- I encoded categorical features to dummy arrays.
- Numeric variables were scaled using a robust scaler.
- A standard linear regression has no hyperparameters.

Results and Interpretation

Metric	Value
RMSE Scores	\$23,925.60, \$29,158.84, \$23,263.94, \$41,767.23, \$29,711.50, \$43,453.35, \$24,383.14, \$22,673.29, \$61,916.91, \$22,487.48
Mean	\$32,274.13
Standard Deviation	\$12,292.94



The average root mean square error of this basic linear model is \$32,274.13. Generally a lower RMSE is better than a higher RMSE. This is an OK model, but definitely not great. This is definitely a good baseline to compare future models to. In practice, I do not think you would want to deploy a model with this poor of a fit.

2020-04-18

Experiment Plan

I would like to build simple Ridge Regression, Lasso Regression, and Elastic Net models. I will use default hyperparameters. This should help me decide which model to choose for fine tuning. I do not have many strong expectations, but I do think that adding some regularization should improve on the standard linear regression.

Experimental Results

Preprocessing Decisions

- I decided to use the same pre-processing pipeline that I built for the linear regression.

Ridge Regression Results and Interpretation

Metric	Value
Scores	\$22341.07, \$24373.09, \$27642.86,\$38633.56 \$31945.70, \$27179.42, \$27325.23, \$25306.76 \$57088.56, \$25630.72
Mean	\$30,746.70
Standard Deviation	\$9,791.03

The ridge regression did perform slightly better than the standard linear regression.

Lasso Regression Results and Interpretation

Metric	Value
Scores	\$21131.29, \$25663.75, \$23090.21, \$41703.26, \$29562.71, \$43187.42, \$24237.69, \$22631.90, \$61024.91, \$22424.98
Mean	\$31,465.81
Standard Deviation	\$12,418.53

When I fit this model with the default hyperparameters, I got a convergence warning. It suggested I increased the number of iterations. I did that, and got the same warning. I suspect this is a product of the lasso regression. In a Lasso regression, the gradients get smaller as you approach the global optimum. I suspect that the model is bouncing around near the global optimum.

Elastic Net Regression Results and Interpretation

Metric	Value
Scores	\$21131.29, \$25663.75, \$23090.21, \$41703.26, \$29562.71, \$43187.42, \$24237.69, \$22631.90, \$61024.91, \$22424.98
Mean	\$33,778.98
Standard Deviation	\$10,492.77

Surprisingly, the Elastic Net model performed the worst. I plan on moving forward with the ridge regression and fine tuning that model.

Fine Tuning the Ridge Regression

I did a grid search over the ridge regression, varying the following parameters: alpha, normalize, max_iter, and tol. The best set of parameters were:

```
Ridge(alpha=10.0, copy_X=True, fit_intercept=False, max_iter=10, normalize=True,
      random_state=None, solver='cholesky', tol=0.01)
```

This model produced the following scores on the training set:

Metric	Value
Scores	\$20824.86, \$24028.96, \$25201.07, \$38864.15, \$32577.72, \$25759.74, \$25427.44, \$23884.19, \$57740.97, \$23928.35
Mean	\$29823.75
Standard Deviation	\$10537.98

And the following results on the Kaggle test set:

Log RMSE: 0.15529

Percentile: 61st

Future Work

Additional work could include building an ensemble model, or attempting to model this dataset with a K-nearest neighbors model. Additionally, it may be valuable to treat some of the pre-processing steps as hyperparameters.

2020-04-27

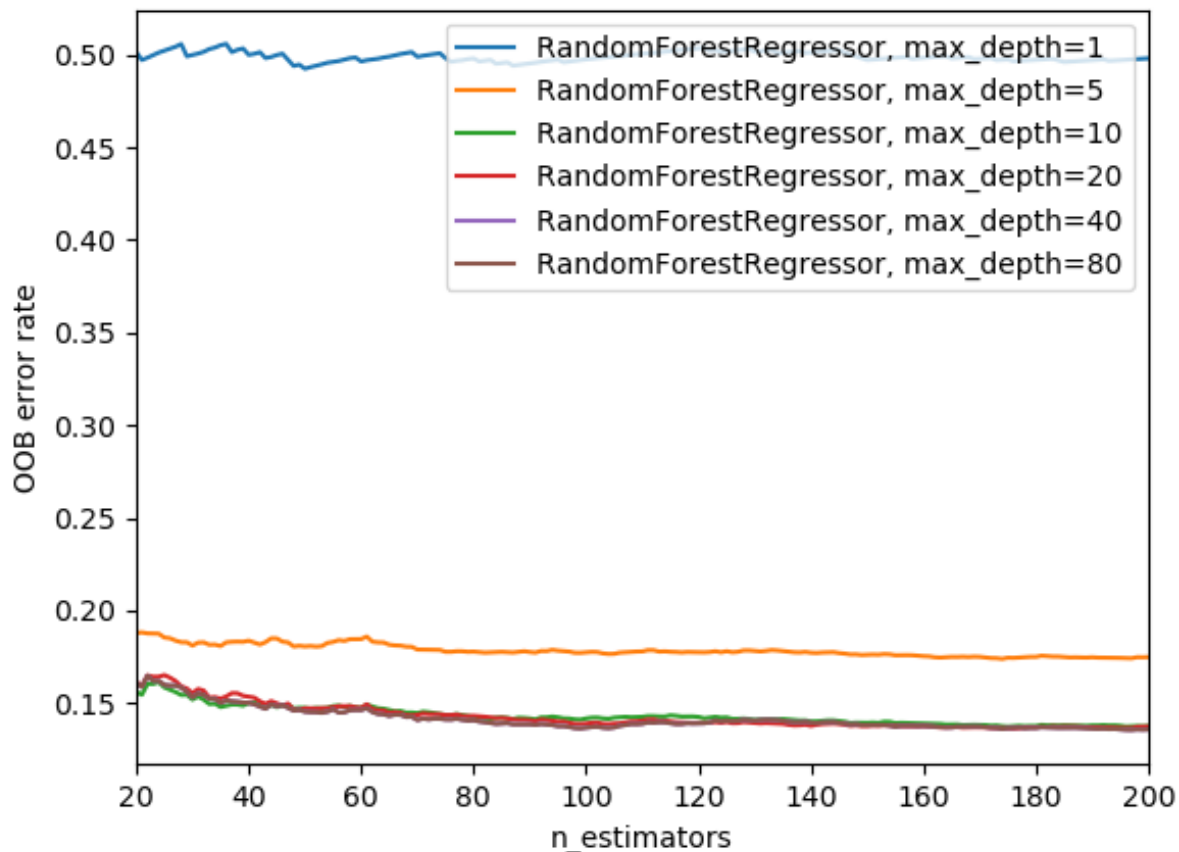
Experiment Plan

I plan on building Random Forest Regressors with varying depths, and comparing their test errors as the number of trees in the forest increases. This will allow me to determine the optimal depth and optimal number of trees in my Random Forest Regressor.

Preprocessing and Model Decisions

- I dropped categorical variables that were missing over 10% of the data. This included: 'Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature'.
- For numeric features, I imputed missing values with medians.
- For categorical features, I imputed missing values with the most frequent value.
- I encoded categorical features to dummy arrays.
- I created RandomForestRegressors with depths of 1, 5, 10, 20, 40, and 80.
- I calculated test error using the out of bag error.
- I calculated the test error for all models for 20 to 200 trees.

Results and Interpretation



- A random forest of stumps doesn't appear to be very effective with this dataset.
- Once you get to a max depth of 10, there isn't much additional benefit to adding more depth.

- Most of the benefit comes from 100 trees. But the test error does appear to continue to decrease after that, albeit very slowly.

Future Work

- Complete the same experiment, but instead of varying depth, vary the number of features.
- Do the same experiment comparing a random forest to Adaboost, and Gradient Boosting.
- Consider adding second degree polynomial features.
- Consider building an ensemble model.
- Build a feature importance graph.

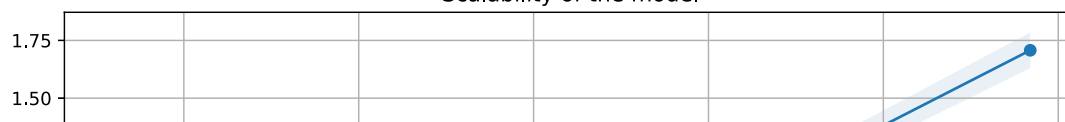
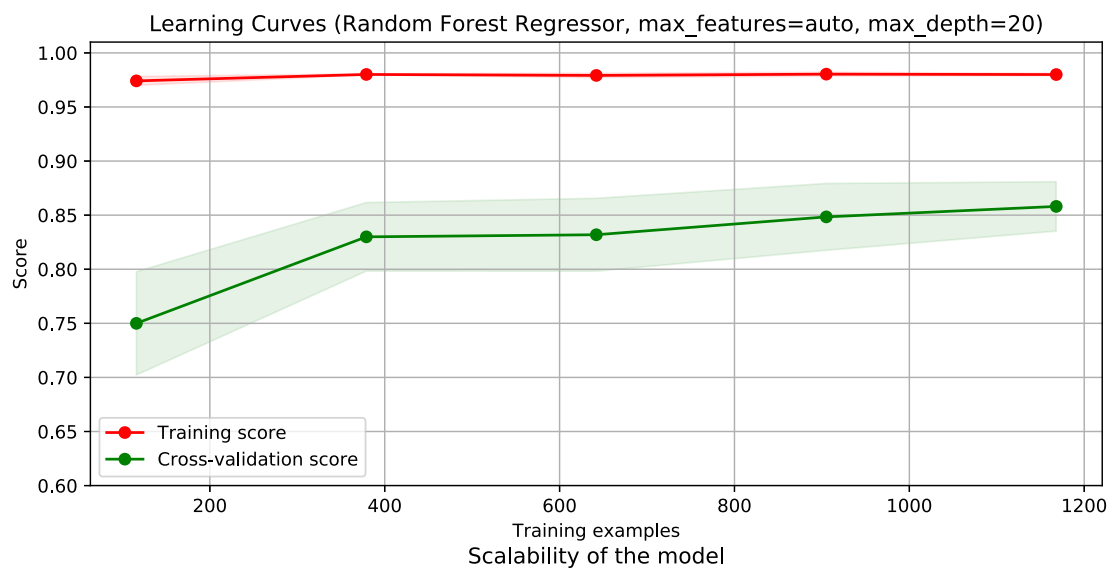
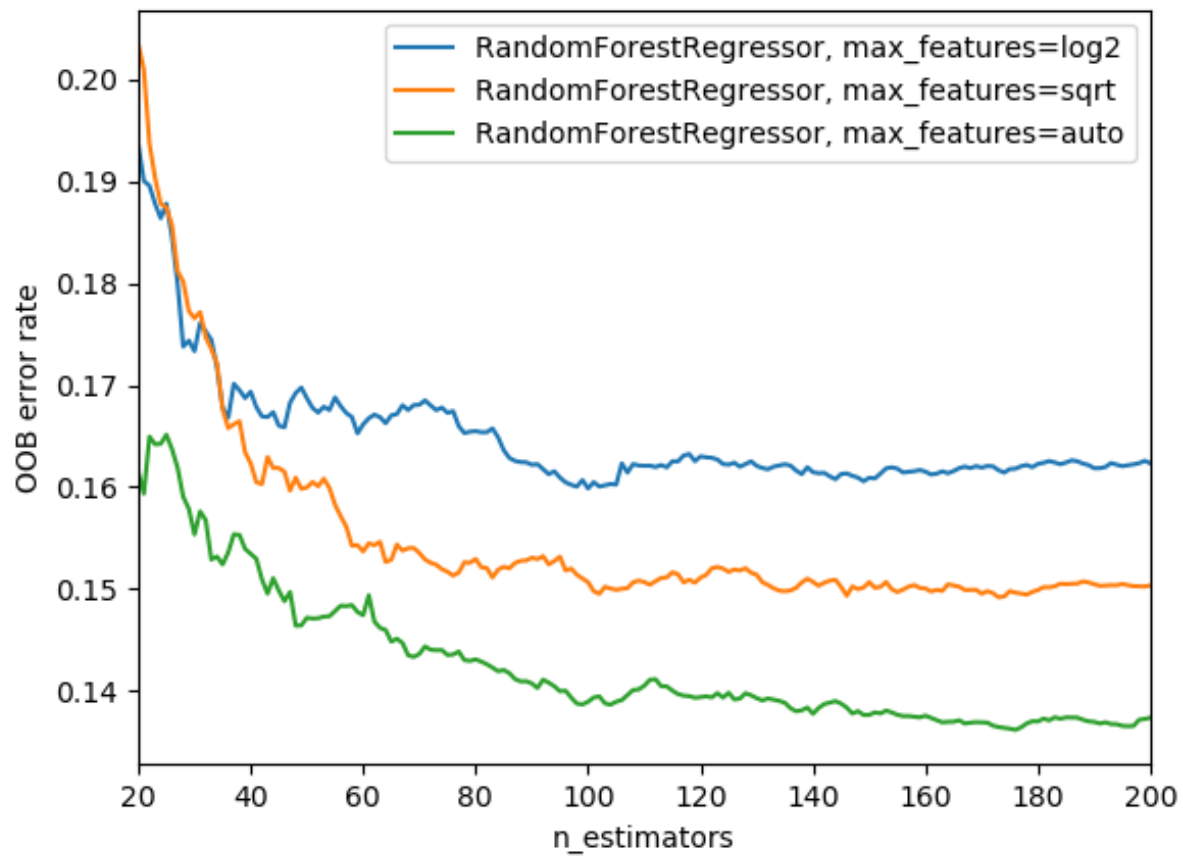
2020-04-29

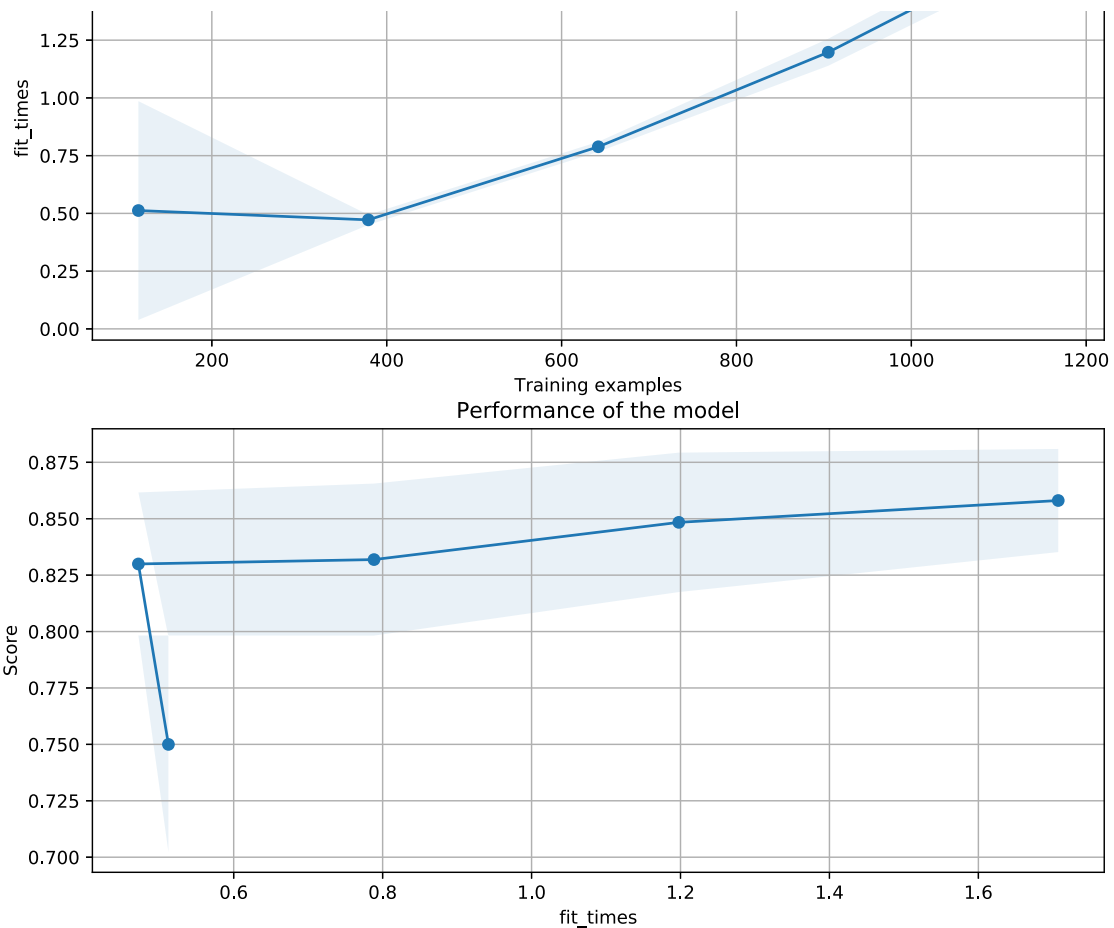
Experiment Plan

I plan on comparing the test error of the following models:

- Random Forest Regressor, max_depth=20, max_features=log2
- Random Forest Regressor, max_depth=20, max_features=sqrt
- Random Forest Regressor, max_depth=20, max_features=auto

Results and Interpretation





Max_features=auto performed the best. I expected log2 to perform better.

Because there is a gap between the training score and the cross validation score, I think this model is likely overfitting the data. However, the cross validation score does continue to slowly increase. So adding more data will likely provide additional benefit. Given the results of the experiment yesterday, I don't think simplifying the model by reducing max_depth would help.

Next Steps

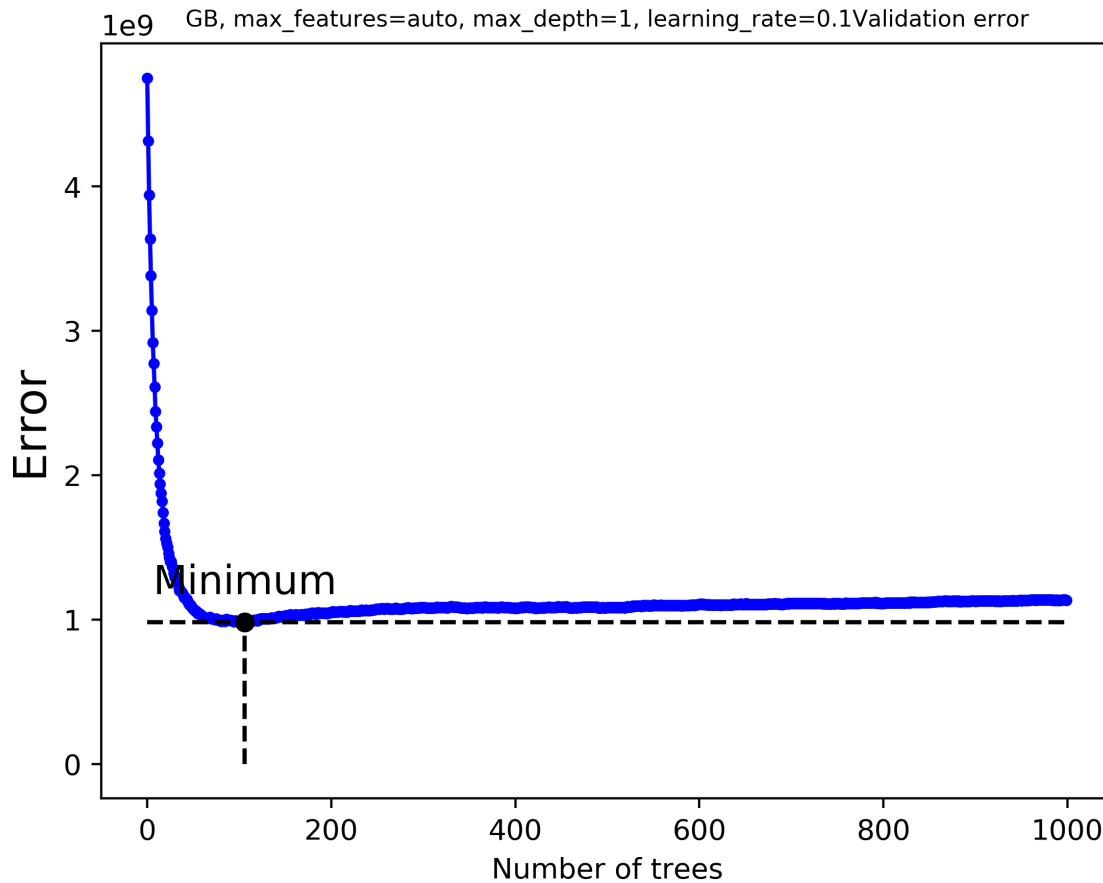
I'd like to evaluate gradient boosting regressors on this dataset.

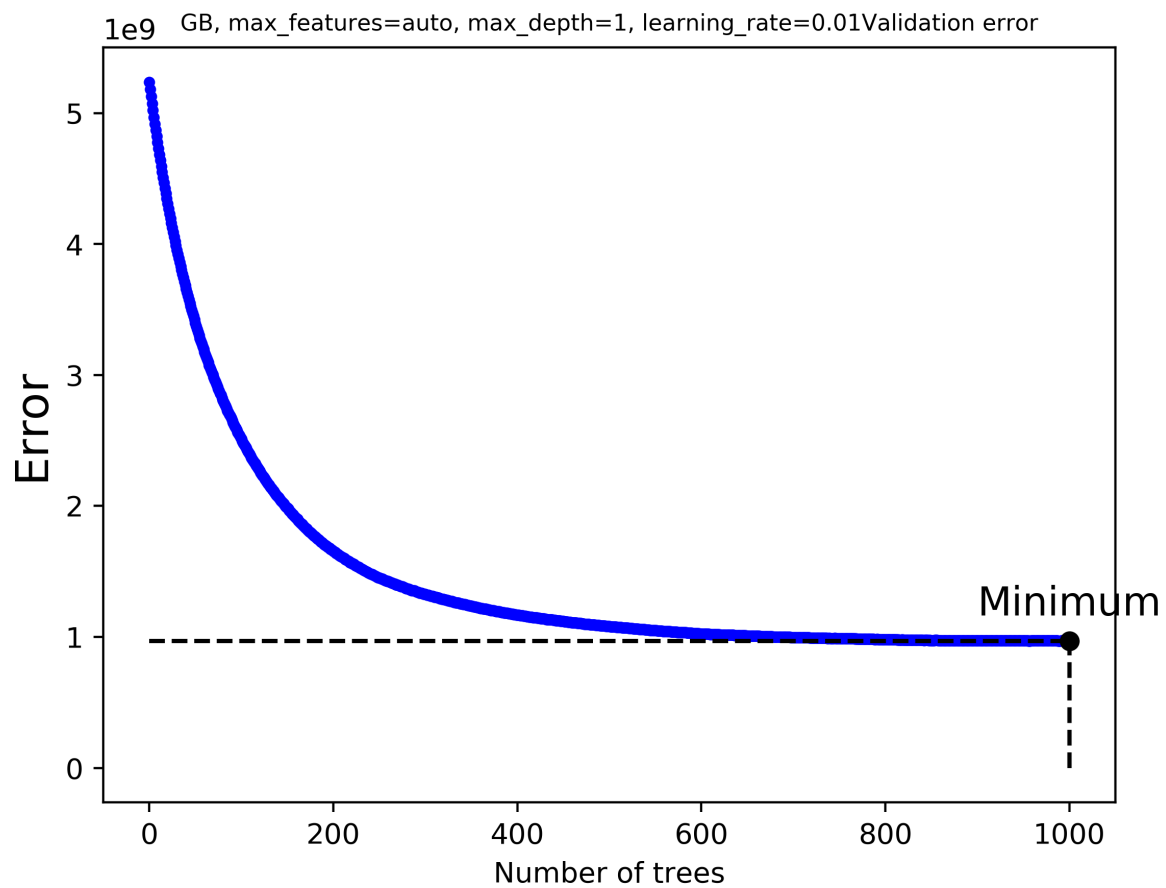
2020-04-30

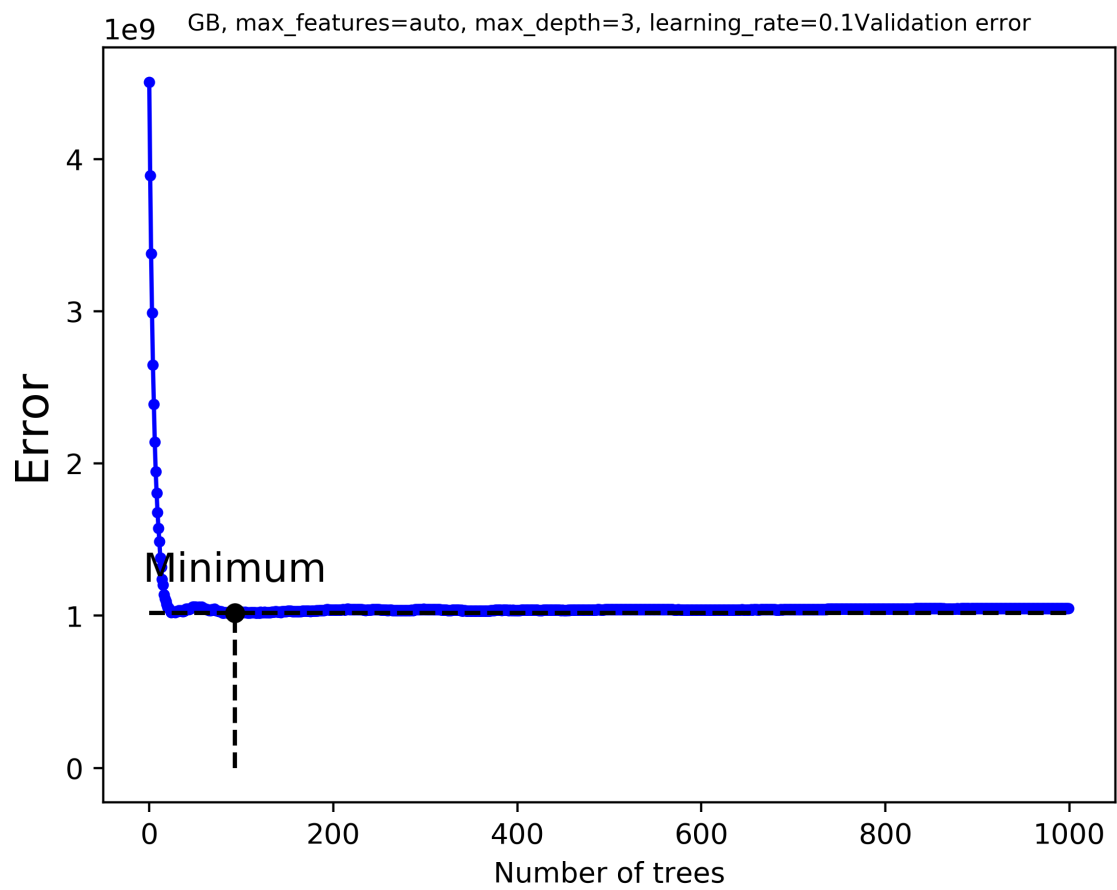
Experiment Plan

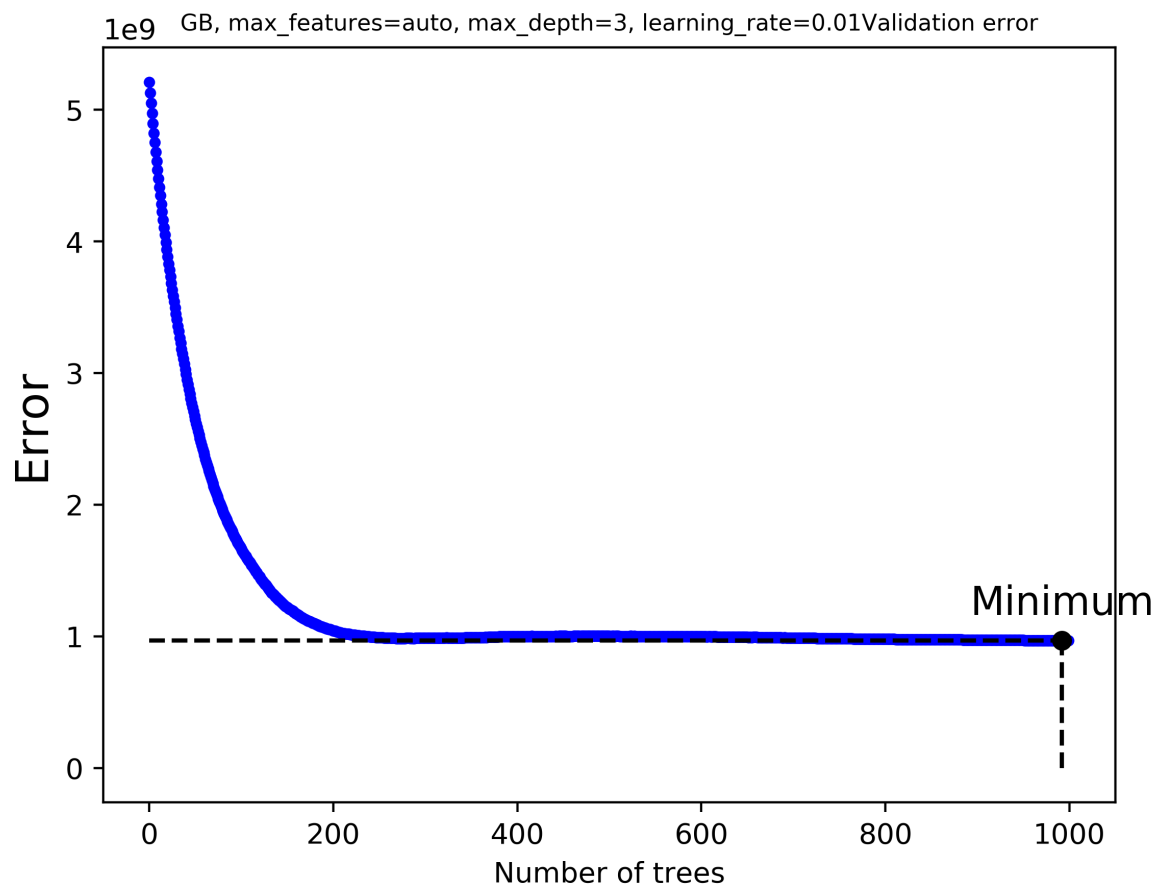
I plan on building a gradient boosting regressor. I will use the same preprocessing pipeline as on previous experiments. I'll try out 12 different models, varying `max_depth`, `learning_rate`, and `max_features`. I'll plot the test error vs the number of trees.

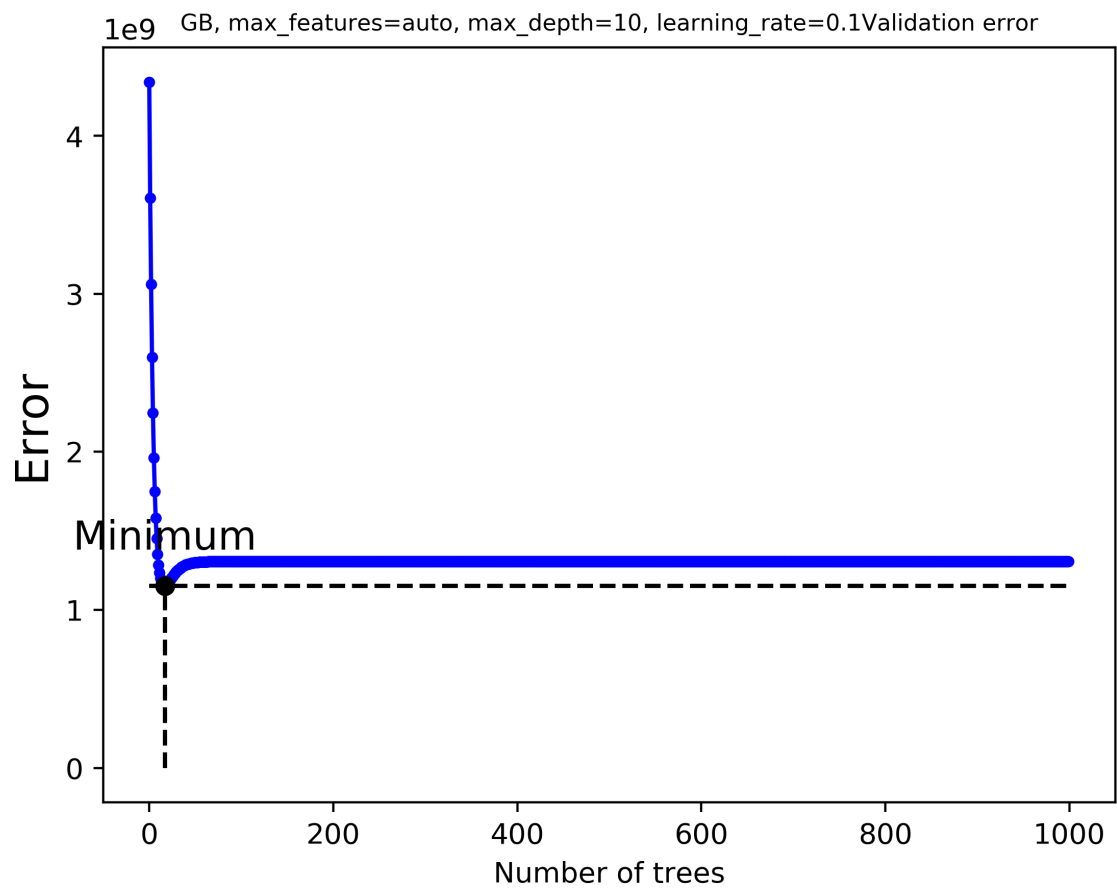
Results and Interpretation

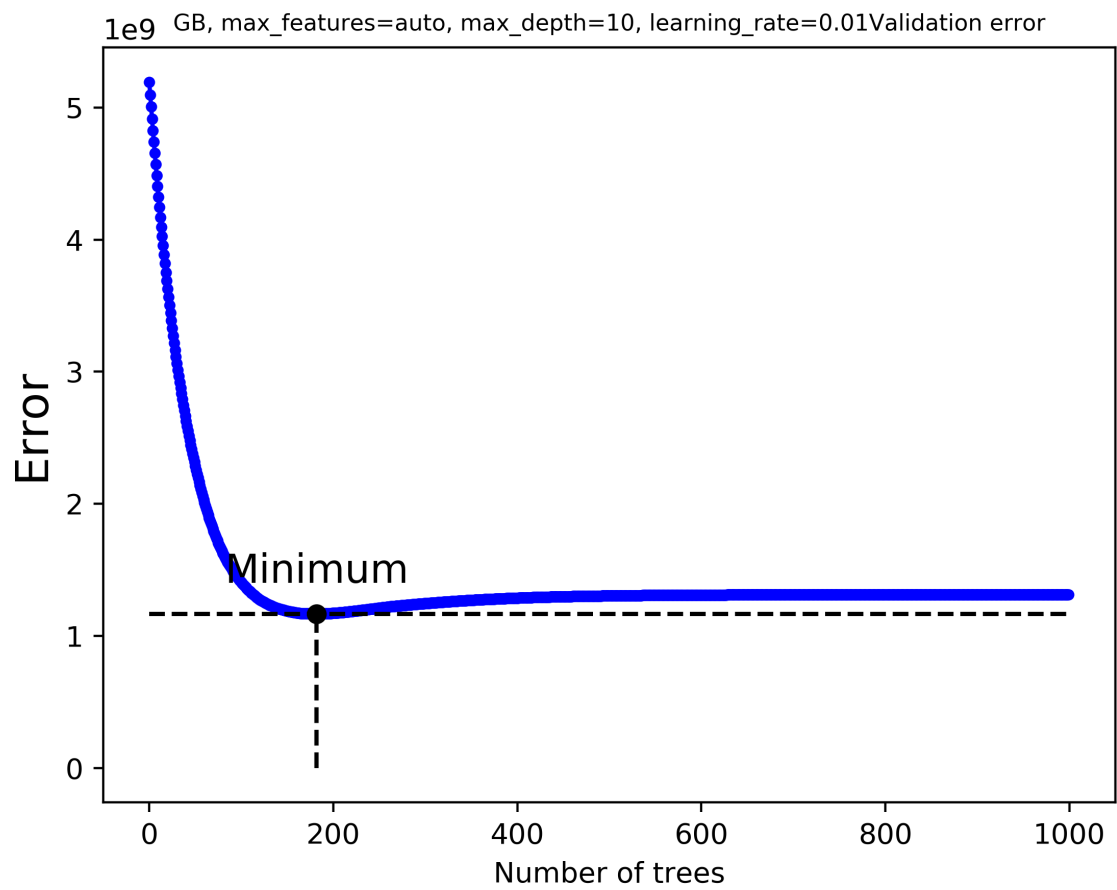


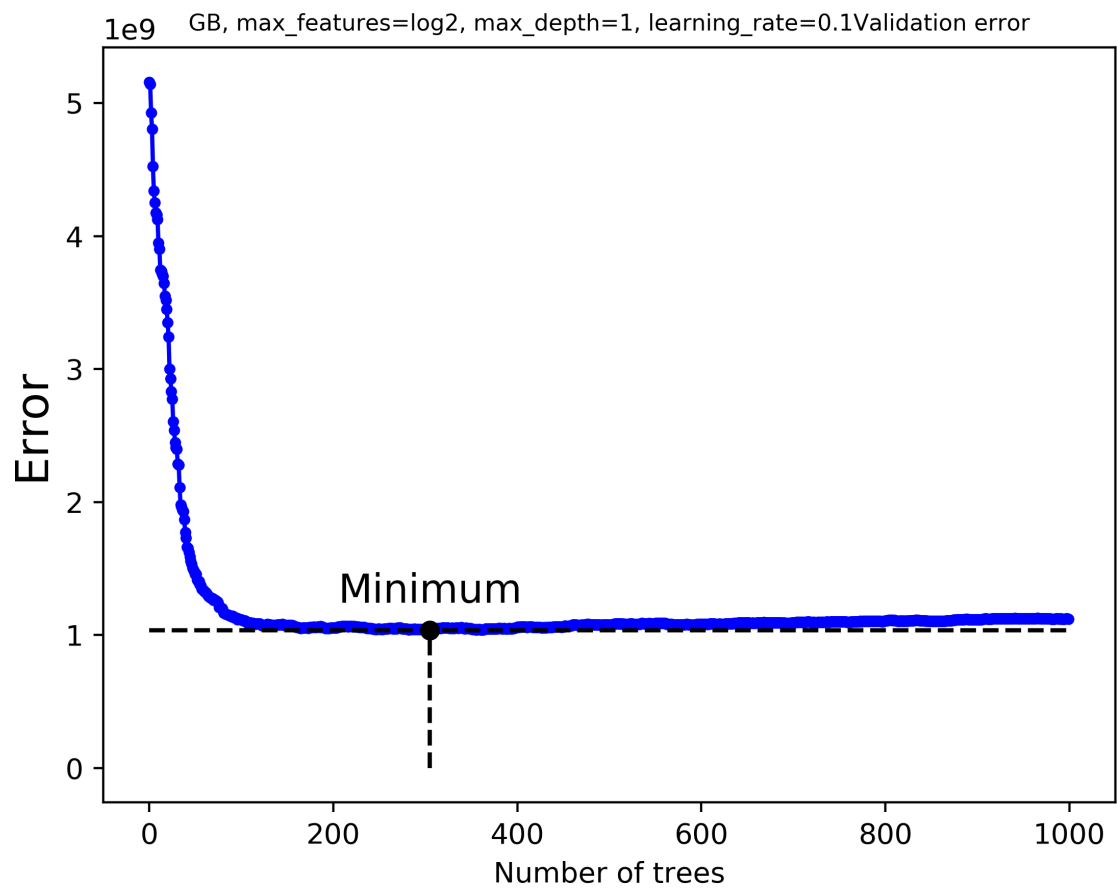


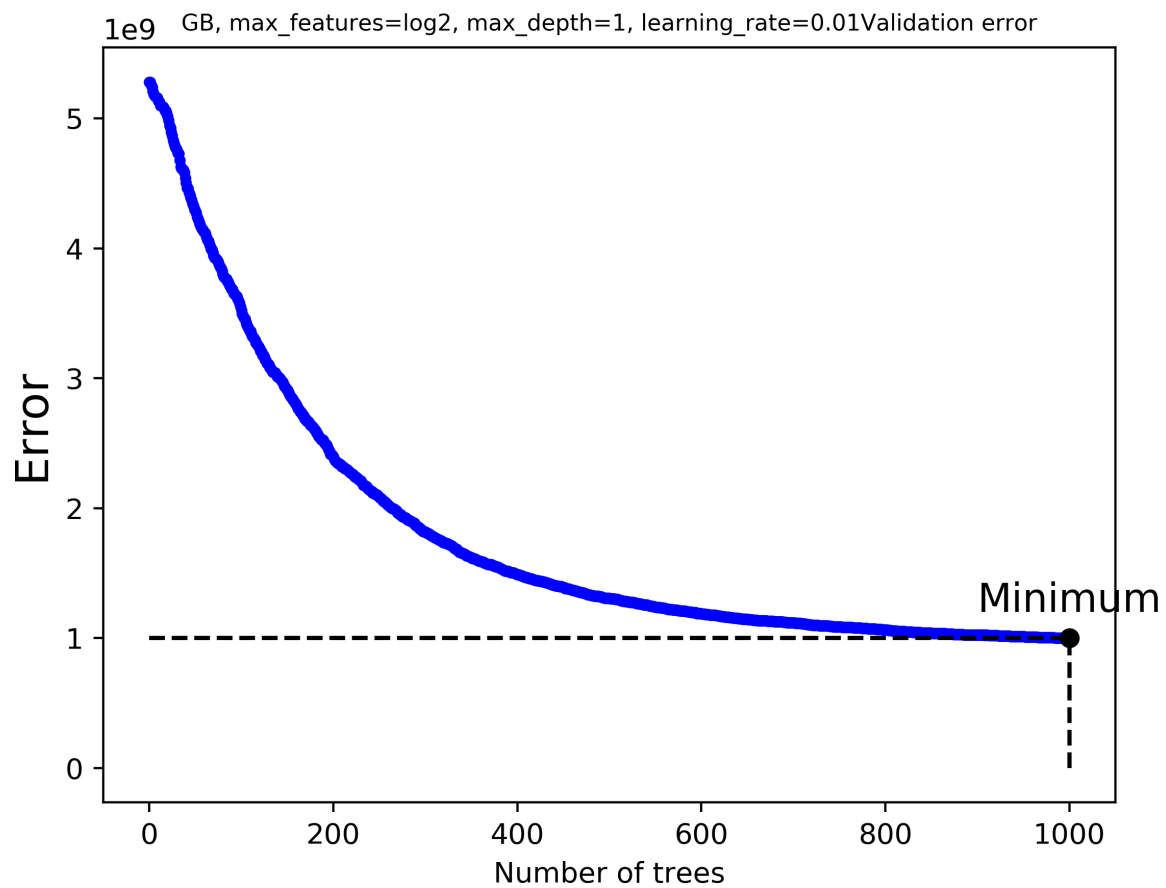


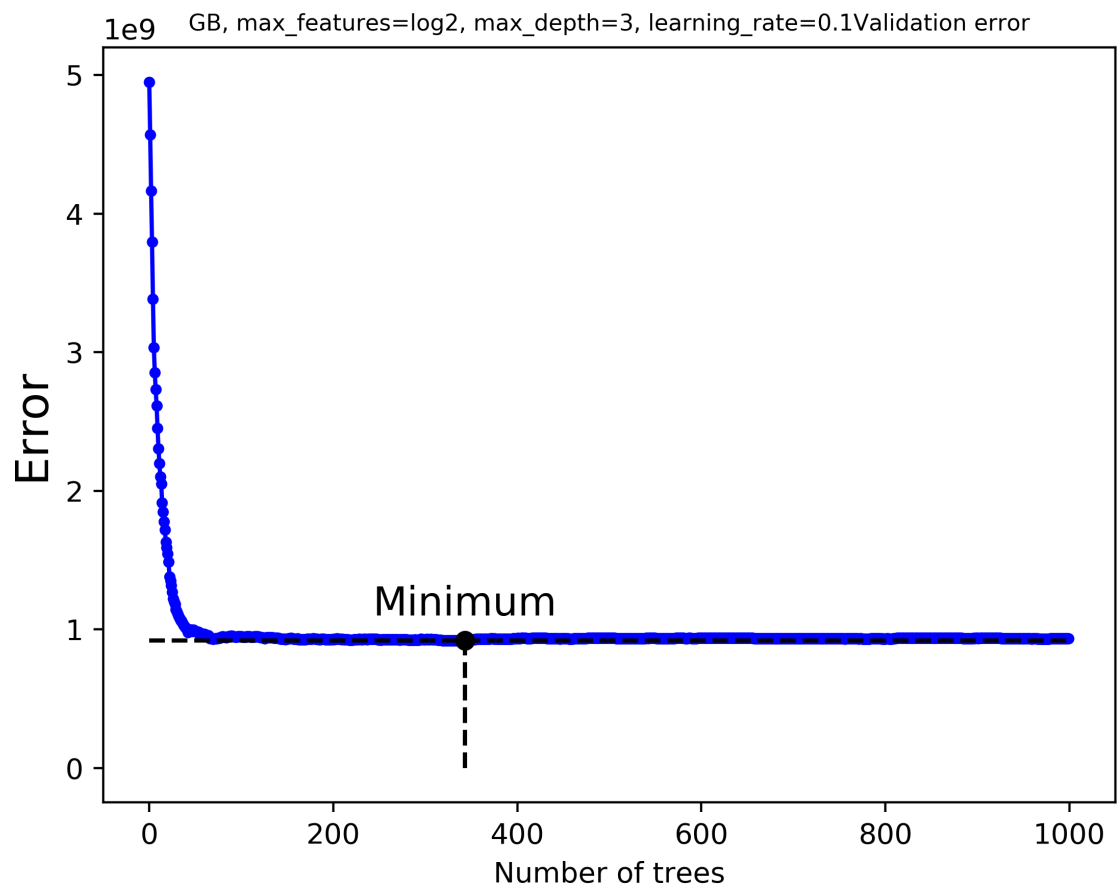


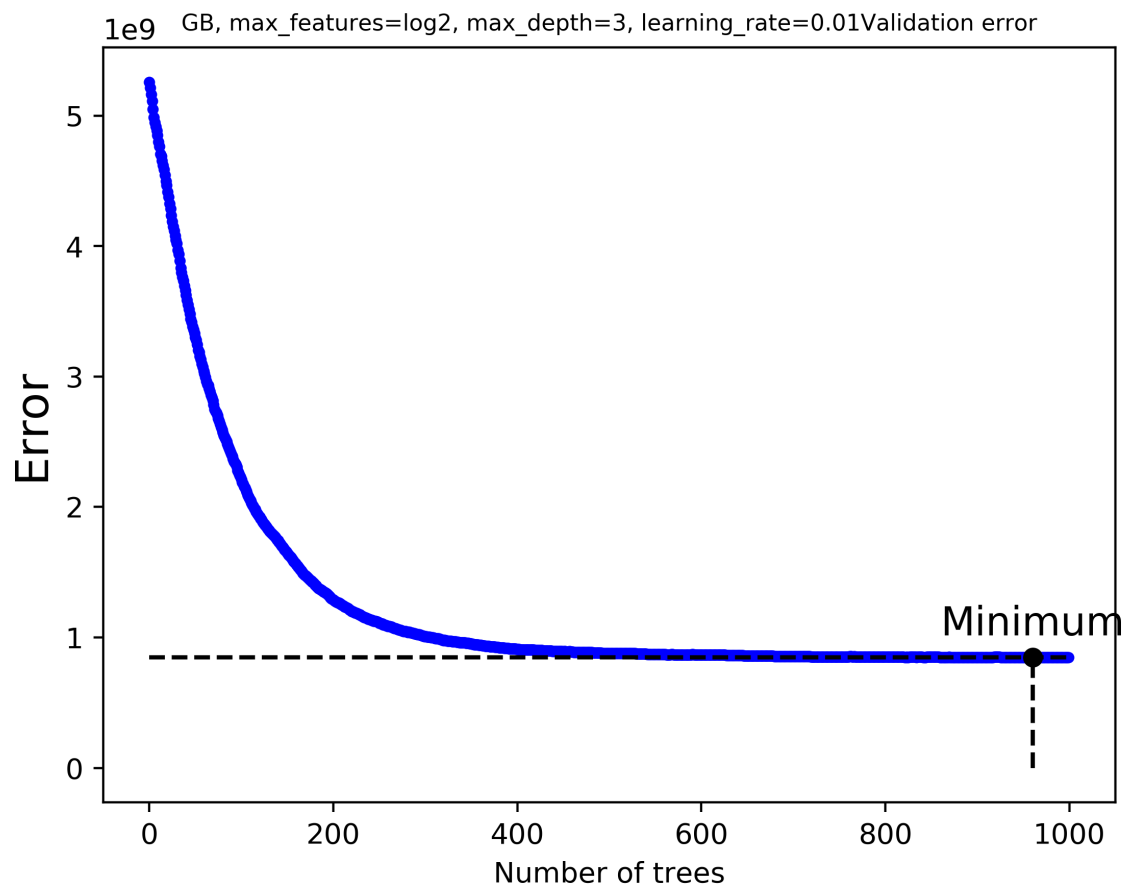


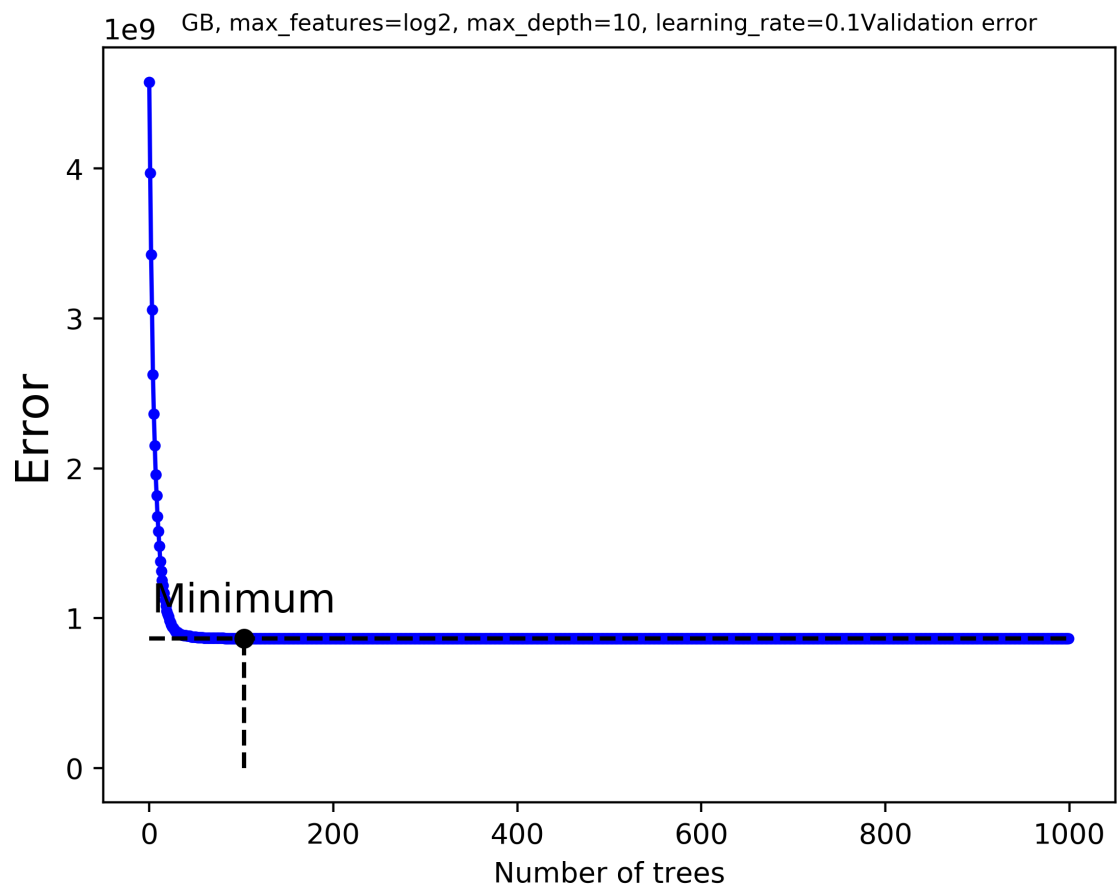


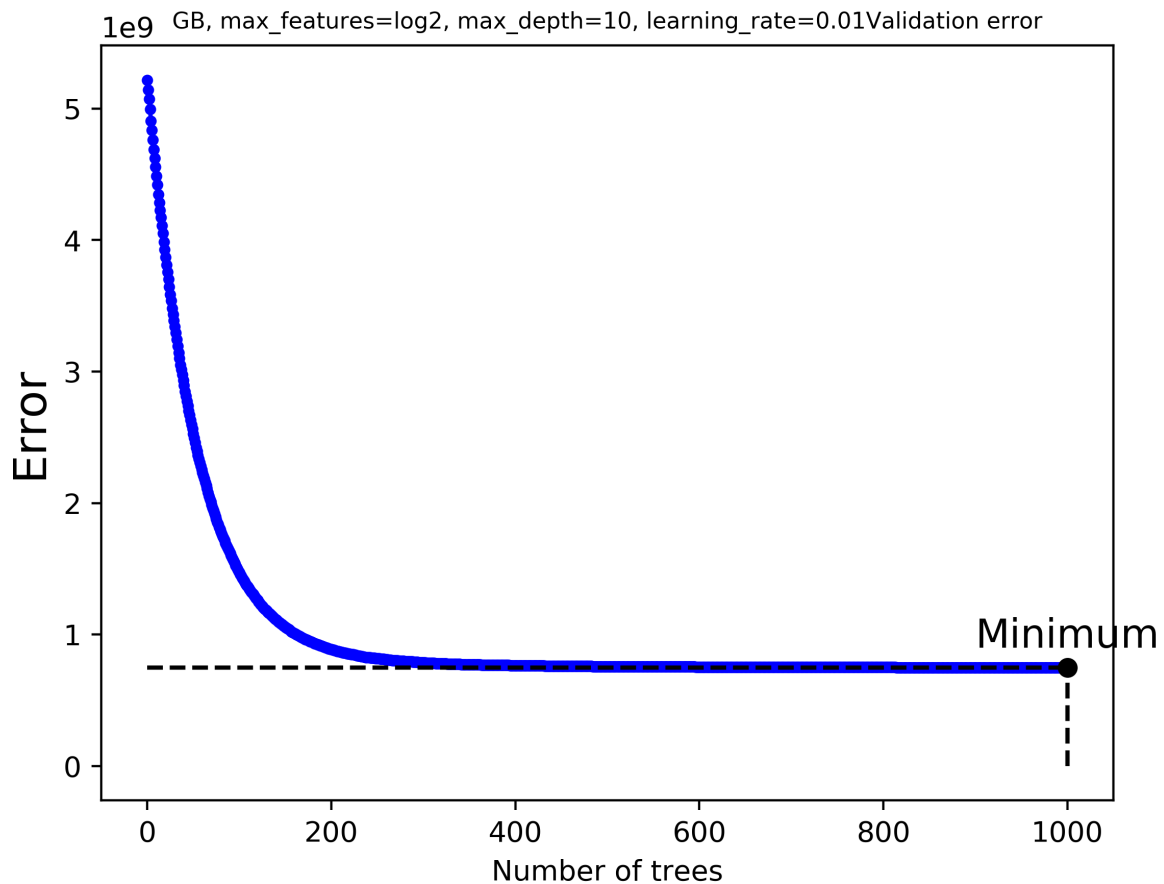












The best model ended up being the last one I tested.

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='ls',
                           max_depth=10, max_features='log2',
                           max_leaf_nodes=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=1000, n_iter_no_change=None,
                           presort='deprecated', random_state=79, subsample=1.0,
                           tol=0.0001, validation_fraction=0.1, verbose=0,
                           warm_start=True)
```

The max_depth and learning_rate parameters were on the edge of the ranges that I choose. So I'm not confident that this is the best model that I can build.

Next Steps

I'd like to do a grid search over a larger space.

2020-05-01

Experiment Plan

I plan on doing a fairly large grid search for both a random forest regressor and a gradient boosting regressor. I'll take the best models from each grid search and submit the predictions to Kaggle.

Results and Interpretation

The following are the best models

Gradient Boosting Model and Scores

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                           init=None, learning_rate=0.01, loss='ls', max_depth=3,
                           max_features='sqrt', max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=6400,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=79, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)
```

Metric	Value
RMSE Scores	19487.55, 21543.60, 19490.61, 39987.68, 27558.00, 21616.68, 21173.44, 18978.27, 30454.24, 21783.84
Mean	24207.39
Standard Deviation	6324.72

And the following results on the Kaggle test set:

Log RMSE: 0.12768

Random Forest Model and Scores

```
# runall.py
```

```
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.impute import SimpleImputer
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.pipeline import make_pipeline, FeatureUnion
from sklearn.preprocessing import OneHotEncoder

from library.library import *
from library.transformers import ColumnSelector, TypeSelector, CategoricalEncoder

# Load training data
data_file = get_dataset_file_path('2020-04-13', 'train.csv')
train = pd.read_csv(data_file)

# Remove label
X_train = train.drop(columns='SalePrice')
y_train = train['SalePrice'].copy()

# Variables to use in the model
x_cols = ['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'LotShape', 'LandContour', 'Utilities',
          'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual',
          'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMat1', 'Exterior1st', 'Exterior2nd',
          'MasVnrType', 'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure',
          'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
          'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath',
          'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
          'Functional', 'Fireplaces', 'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea',
          'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
          'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition']

# Build preprocessing pipeline
preprocess_pipeline = make_pipeline(
    ColumnSelector(columns=x_cols),
    CategoricalEncoder(),
    FeatureUnion(transformer_list=[
        ("numeric_features", make_pipeline(
            TypeSelector(np.number),
            SimpleImputer(missing_values=np.nan, strategy='median'),
        )),
        ("categorical_features", make_pipeline(
            TypeSelector("category"),
            SimpleImputer(strategy="most_frequent"),
            OneHotEncoder()
        ))
    ])
)

X_train = preprocess_pipeline.fit_transform(X_train)

gb_param_grid = [
    {"max_features": ["auto", "log2", "sqrt"],
     "max_depth": [1, 3, 10, 20, 40, 80],
     "learning_rate": [0.001, 0.01, 0.1],
     "n_estimators": [100, 200, 400, 800, 1600, 3200, 6400],
     "random_state": [79]}
]

rf_param_grid = [
    {"max_features": ["auto", "log2", "sqrt"],
     "max_depth": [1, 3, 10, 20, 40, 80],
     "n_estimators": [100, 200, 400, 800, 1600, 3200, 6400],
     "random_state": [79]}
]

gb_grid_search = GridSearchCV(GradientBoostingRegressor(), gb_param_grid, cv=10, scoring="neg_mean_squared_error",
                              n_jobs=-1,
                              return_train_score=True)
gb_grid_search.fit(X_train, y_train)
print(gb_grid_search.best_estimator_)

gb_final_model = gb_grid_search.best_estimator_

rf_grid_search = GridSearchCV(RandomForestRegressor(), rf_param_grid, cv=10, scoring="neg_mean_squared_error",
                              n_jobs=-1,
                              return_train_score=True)
rf_grid_search.fit(X_train, y_train)
```

```

print(rf_grid_search.best_estimator_)

rf_final_model = rf_grid_search.best_estimator_

# Evaluate final model on training data
gb_scores = cross_val_score(gb_final_model, X_train, y_train, scoring="neg_mean_squared_error", cv=10)
gb_reg_rmse_scores = np.sqrt(-gb_scores)
print("GB Scores")
display_scores(gb_reg_rmse_scores)

rf_scores = cross_val_score(rf_final_model, X_train, y_train, scoring="neg_mean_squared_error", cv=10)
rf_reg_rmse_scores = np.sqrt(-rf_scores)
print("RF Scores")
display_scores(rf_reg_rmse_scores)
# Evaluate final model on the test set

# Load test data
data_file = get_dataset_file_path('2020-04-13', 'test.csv')
X_test = pd.read_csv(data_file)
output = X_test["Id"]

# Preprocess data
X_test = preprocess_pipeline.transform(X_test)

# Make final predictions
gb_final_predictions = gb_final_model.predict(X_test)
rf_final_predictions = rf_final_model.predict(X_test)

# Output predictions
gb_final_predictions = pd.Series(gb_final_predictions, name="SalePrice")
rf_final_predictions = pd.Series(rf_final_predictions, name="SalePrice")

output = pd.concat([output, gb_final_predictions], axis=1)
output.to_csv("gb_predictions.csv", index=False)
output = pd.concat([output, rf_final_predictions], axis=1)
output.to_csv("rf_predictions.csv", index=False)

```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                        max_depth=10, max_features='auto', max_leaf_nodes=None,
                        max_samples=None, min_impurity_decrease=0.0,
                        min_impurity_split=None, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        n_estimators=200, n_jobs=None, oob_score=False,
                        random_state=79, verbose=0, warm_start=False)
```

Metric	Value
RMSE Scores	24902.07, 26523.00, 21978.73, 38355.29, 32829.27, 25498.82, 24202.52, 24174.37, 39793.86, 28119.71
Mean	28637.77
Standard Deviation	5907.13

And the following results on the Kaggle test set:

Log RMSE: 0.14935

The random forest performs marginally better than my linear regression models. However, the gradient boosting model performs substantially better.