Jon Rippe
CSCE A415 – Machine Learning
HW #3

Random Forest Regressor & Linear Regression

Data Inspection, Visualization, & Cleaning

There are 7610 rows missing Price data. Because this is the value I'm trying to predict, I can't use these rows; chicken/egg situation where my own Price predictions are used to train and test my Price predictions.

The following can be removed based on an initial glance at the data:

- SellerG: seller name is not related to price.
- Date: date ranges are narrow (Jan 2017 to Sept 2017) and should not be related to price.
- Postcode: suburb and lat/long can be used for geographic location.
- Regionname: suburb and councilarea are more granular.

Checkpoint 1

A few things to note after visualizing some data:

- Rooms and Bedroom2 are practically the same. Will drop Bedroom2 as it is incomplete.
- Propertycount is per Suburb and doesn't appear to be correlated with Price. Will drop.
- Type appears to be correlated with Price (houses are more expensive, then townhouses, then units). Method may also be correlated with Price. Will create dummy columns to better visualize.
- There are extreme outliers in Landsize, BuildingArea, and YearBuilt. These may be typos or rare, one-off, situations. Will drop outliers.
- I wish there wasn't so much missing YearBuilt data.

Checkpoint 2

After the first round of cleaning:

- Method doesn't appear to have a huge impact on Price. Will drop.
- Type is definitely correlated with price, but only h and u. Will keep all three for filling missing data but will drop t before feeding into models.
- I expected Landsize to have a bigger impact on Price and BuildingArea, but that doesn't appear to be the case. It will be dropped.

Filling Missing Data

The Suburb column gives a good granular geographic location and should give an easy and effective way of filling in some missing data. I'll be using the Suburb attribute to fill some missing information:

Round 1

- Bathroom: Median value of properties in the same Suburb of the same Type with the same Rooms value
- Car: Median value of properties in the same Suburb of the same Type with the same Rooms value
- BuildingArea: Median value of properties in the same Suburb of the same Type with the same Rooms value
- YearBuilt: Median value of properties in the same Suburb of the same Type
- CouncilArea: Mode value of Suburb

Round 2: Same as Round 1 but using the less granular CouncilArea instead of Suburb.

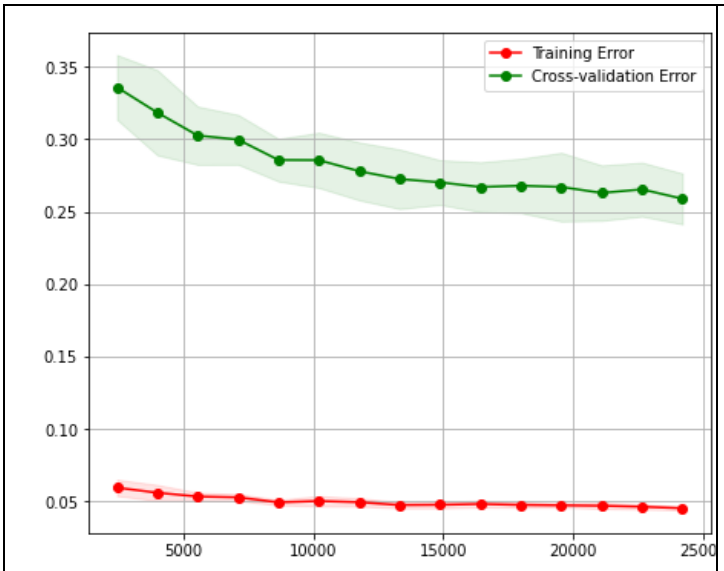
Round 3: Same as Round 1 but not narrowing to Type/Rooms.

Checkpoint 3

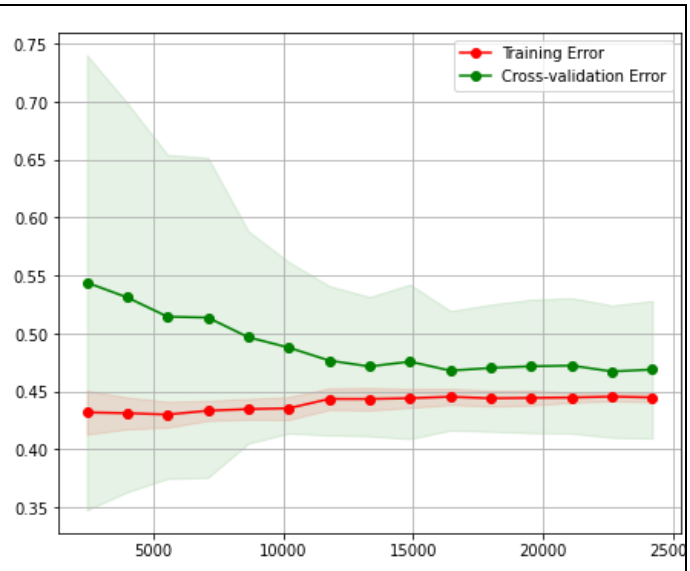
Latitude and Longitude can be filled in using the Address/Suburb attributes and geolocation. (**Note: 6000+ geolocation requests using a free service takes several hours. I've run it and stored the results in a new csv for easy access. The code is modified to use the csv file.)**

Initial Model Runs

The forest regressor performs better than the linear regression model at the expense of speed.



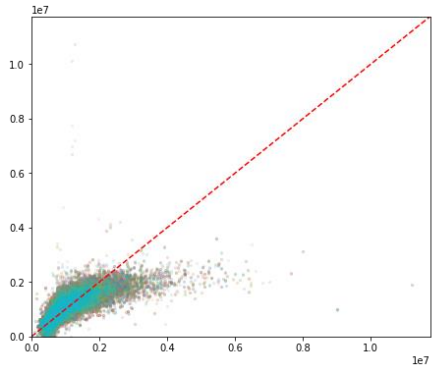
Convergence of testing and training error rates for the random forest regressor may happen given a much larger sample size. A larger gap between the two suggests the model is overfitting the model to the training data. However, even with increased overfitting, the model shows increased accuracy.



Training and validation error rates in the linear regression model converge around 10-15K samples, suggesting an increase in sample size would likely not be beneficial beyond this.

The scatter plot of the linear regression predicted vs. true values shows a non-linear trend.

Applying a polynomial feature to the model could help straighten this trend.

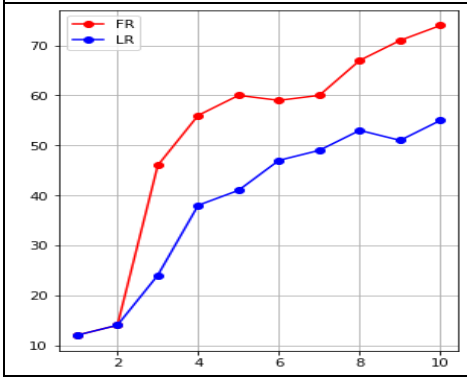
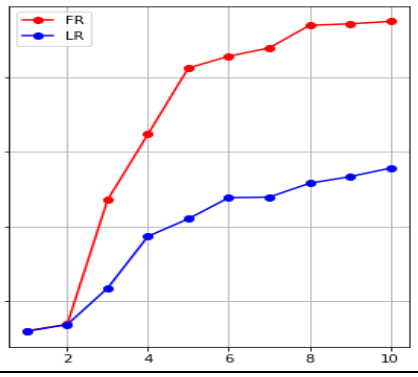
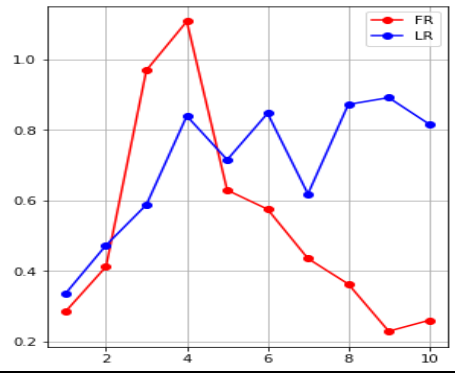


Checkpoint 4

Modified Models

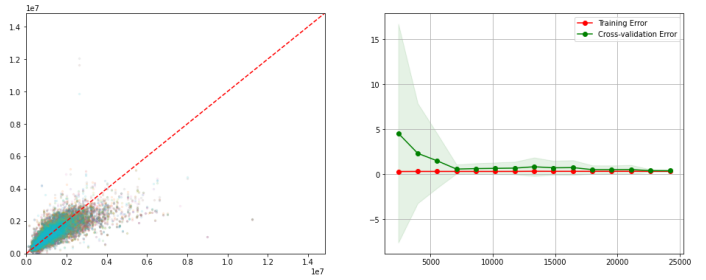
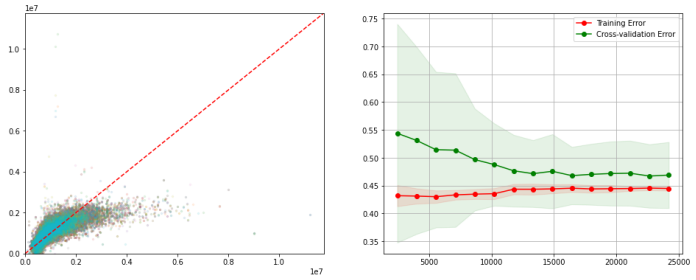
Evaluating performance based on k-best attributes:

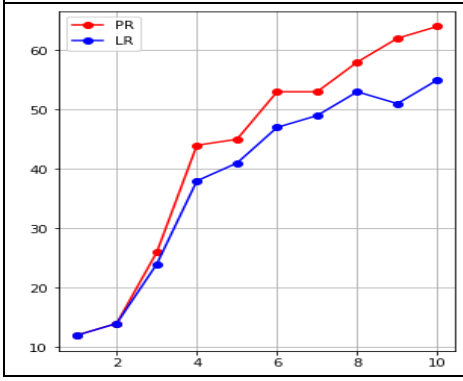
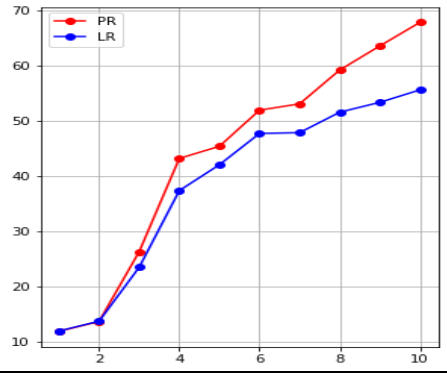
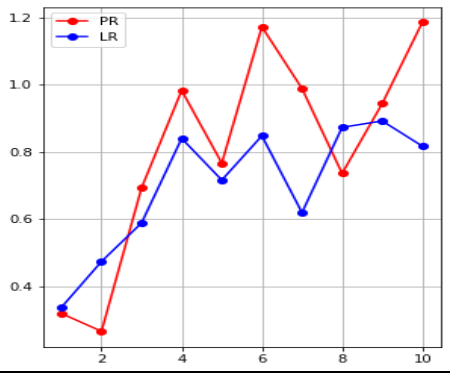
Both models perform better with more attributes with diminishing returns at ≥8.

Accuracy	Mean	Std Dev
		

Evaluating the linear regression model with polynomial features:

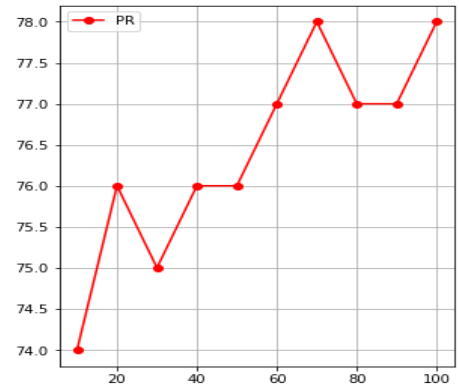
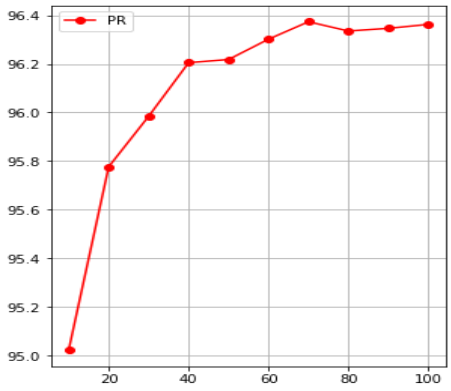
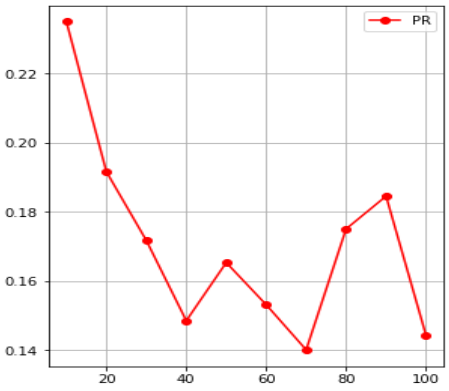
Adding a polynomial feature to the linear model increased accuracy by 10% but also increased runtime 5x (20ms to 100ms). Using a polynomial feature also helps fit training and test error rates, so fewer samples are needed to maximize the accuracy of the model.

Polynomial degree=2	Linear
	

Accuracy	Mean	Std Dev
		

Evaluating the random forest regressor based on tree count:

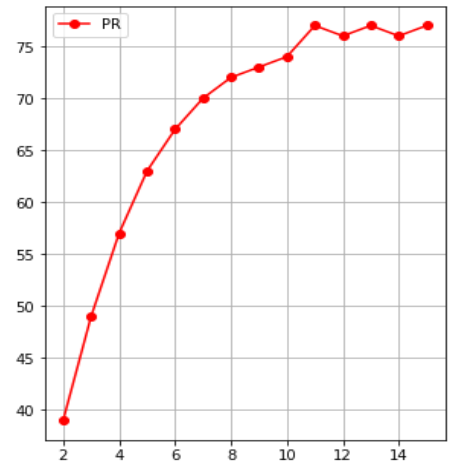
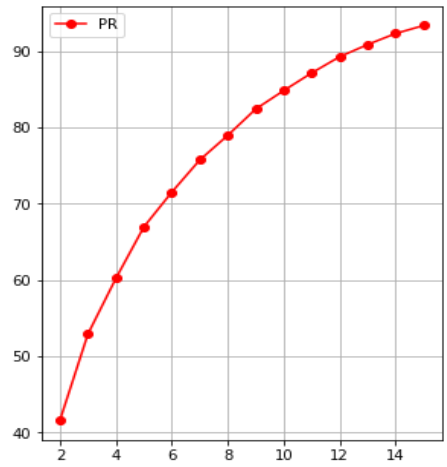
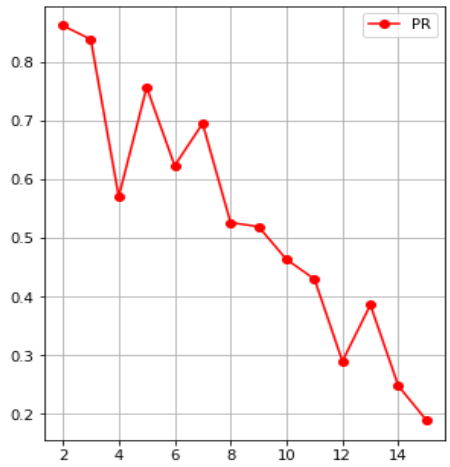
The random forest regressor shows diminishing returns around n=40 – leveling off drastically around n=70. I would have ideally gone into the 100's or 1000's, but time is an important factor and the results up to 100 look conclusive enough.

Accuracy	Mean	Std Dev
		

Evaluating the random forest regressor based on tree depth:

Because a tree count of 70 gave the best results in the previous test, I've used that as the tree count for this test.

There doesn't seem to be an increase in overall model accuracy beyond a tree depth of 11, however precision does continue to get better as tree depth increases. The error rate charts shown in the Jupyter Notebook also show more overfitting with deeper trees.

Accuracy	Mean	Std Dev
		

Final Thoughts

A linear regression model, even when evaluating polynomial features, is a better option if speed is important and processing power is limited.

There was a lot of important missing data in the original dataset that could greatly increase the accuracy of either model. This is one situation where collecting more data would be a valid way to improve model accuracy. Most real property information is public record, so missing information in the dataset could be filled using datamining.