Predicting Forest Cover Type

Supervised Learning Capstone

By: Montanna Colburn

Research Question

- Can we predict forest cover type based on the surrounding characteristics?
  - What are the most commonly seen cover types?
  - Do certain cover types grow best under certain conditions?
  - What are the most important attributes in predicting cover types?

- Practical Use? Could help in forest regrowth and preservation.

Dataset

- Kaggle Dataset
  - https://www.kaggle.com/uciml/forest-cover-type-dataset
  - Multi Classification: 7 potential cover types
  - 581,012 observations of 30m x 30m patches of forest floor in Roosevelt National Forest
    - No nulls, and all int values (aka minimal data cleaning!)
  - The data collected is from areas with minimal human disturbances, so that cover types are more likely correlated to ecological processes

Dataset Continued

- Total of 55 columns
  - Continuous:
    i. Elevation: Elevation in meters.
    ii. Aspect: Aspect in degrees azimuth.
    iii. Slope: Slope in degrees.
    iv. Horizontal_Distance_To_Hydrology: Horizontal distance to nearest surface water features.
    v. Vertical_Distance_To_Hydrology: Vertical distance to nearest surface water features.
    vi. Horizontal_Distance_To_Roadways: Horizontal distance to nearest roadway.
    vii. Hillshade_9am: Hill shade index at 9am, summer solstice. Value out of 255.
    viii. Hillshade_Noon: Hill shade index at noon, summer solstice. Value out of 255.
    ix. Hillshade_3pm: Hill shade index at 3pm, summer solstice. Value out of 255.
    x. Horizontal_Distance_To_Fire_Points: Horizontal distance to nearest wildfire ignition points.

Dataset Continued

- Total of 55 columns
  - Categorical:
    i. Wilderness Area (4 dummy variable binary columns, 0 = absence or 1 = presence)
    ii. Soil Type (40 dummy variable binary columns, 0 = absence or 1 = presence)

Data Exploration/ Visualization

- Distribution of Target Variables: Class imbalance

Forest Cover Type Key:

1.Spruce/Fir

2.Lodgepole Pine

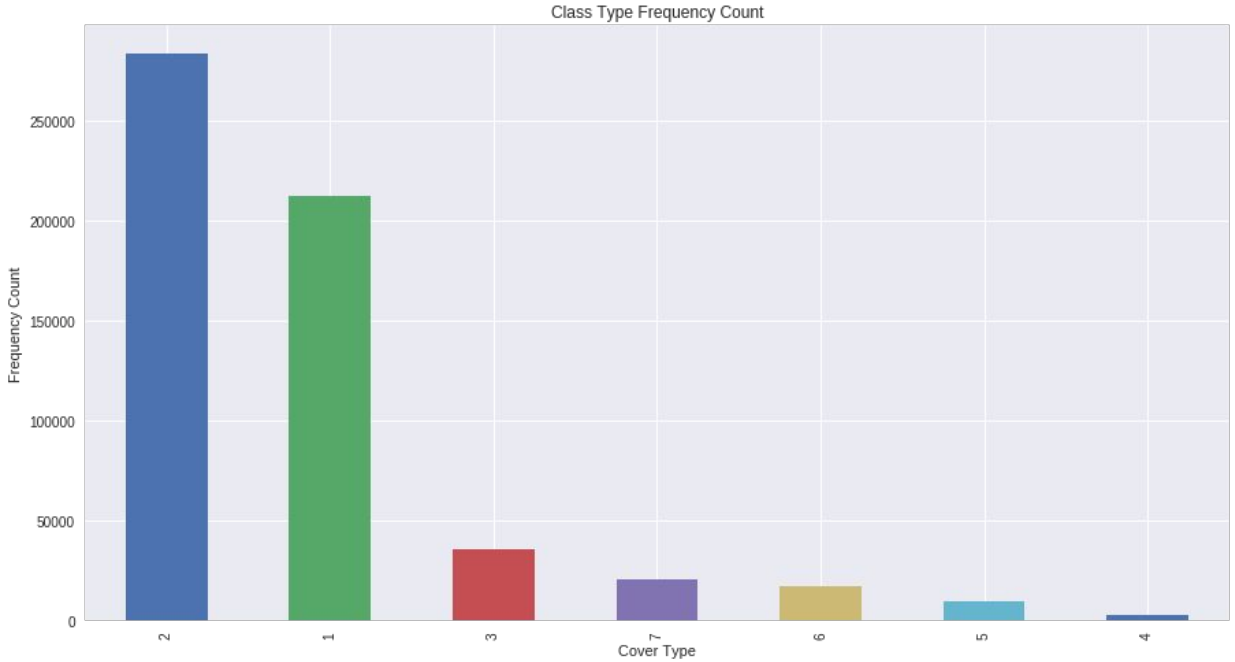
3.Ponderosa Pine

4.Cottonwood/Willow

5.Aspen

6.Douglas-fir

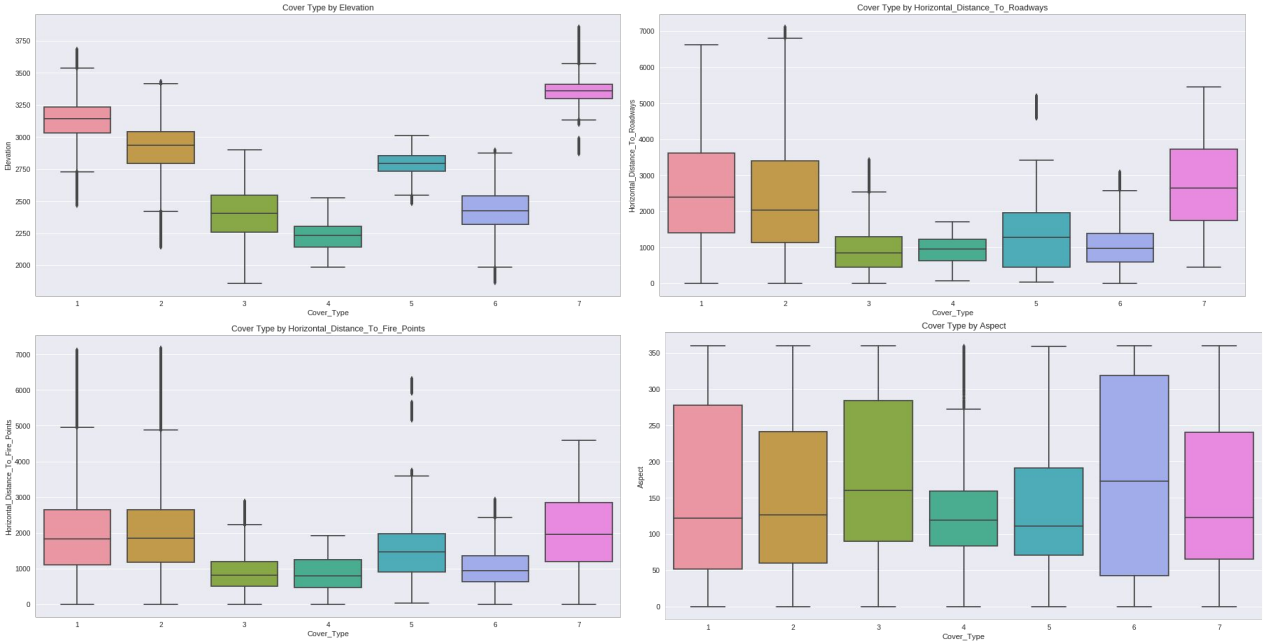
7.Krummholz



Data Exploration/ Visualization

- Distribution of Continuous Variables by Target Variables

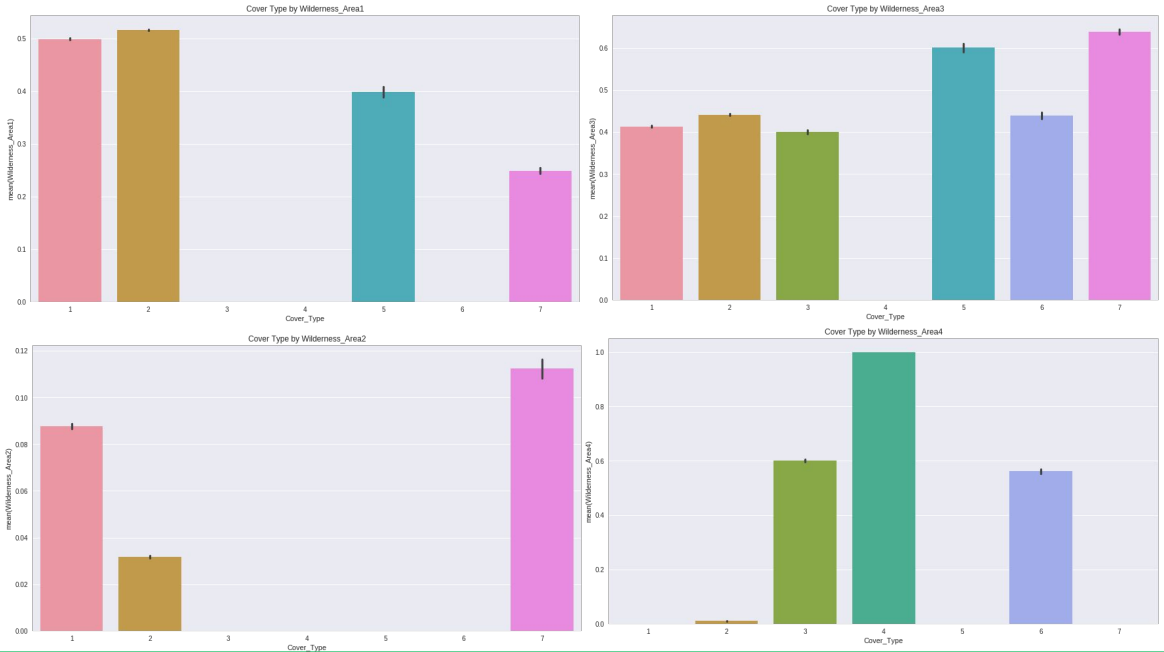
Variables that are likely to be good indicators of cover type due to the variability amongst floor coverings:



Data Exploration/ Visualization

- Distribution of Categorical Wilderness Area by Target Variables

Certain wilderness areas are more likely to have specific kinds of forest cover type than others, so this will likely be a good indicator for our prediction model:

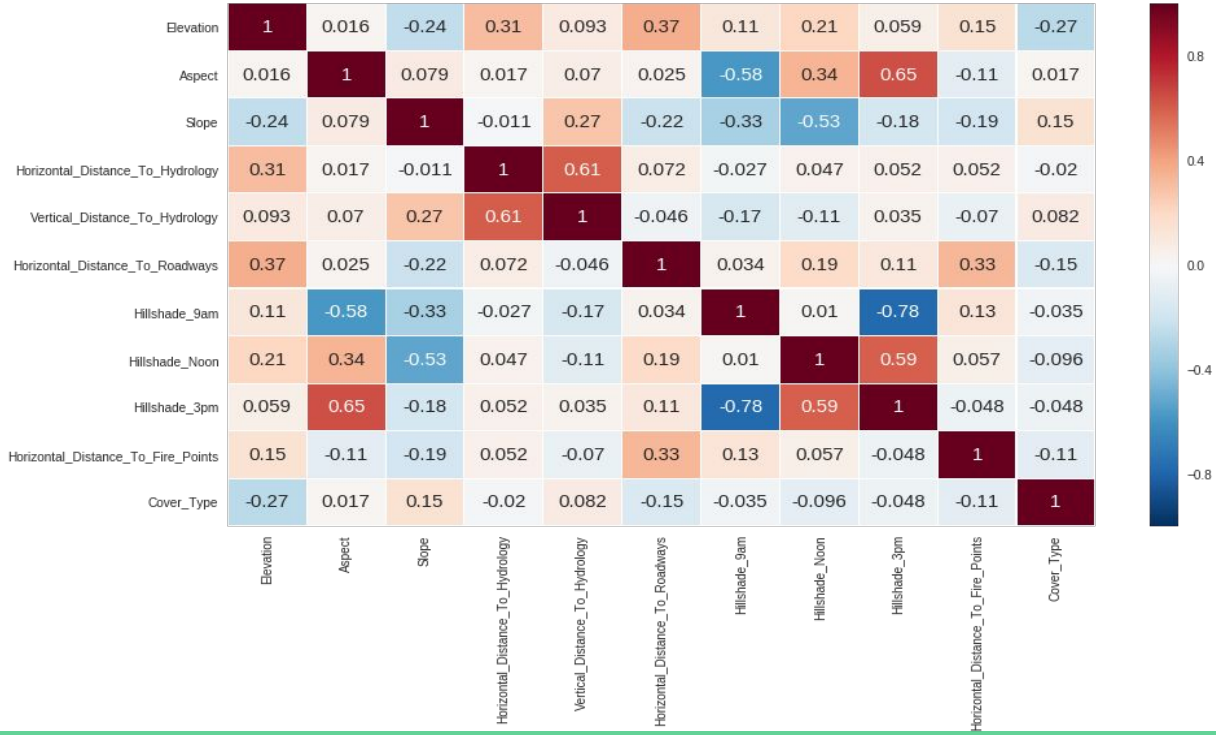


Data Exploration/ Visualization

- Continuous Variable Correlation Matrix

Aspect & Hillshade_3pm, Horizontal_Distance_To_Hydrology & Vertical_Distance_To_Hydrology, Hillshade_Noon & Hillshade_3pm, and Hillshade_9am & Hillshade_3pm are relatively correlated.

We'll use a .90 threshold for removal of correlated variables, and thus we won't be removing any of these, as we won't account for too much variance or overfit our model from this.

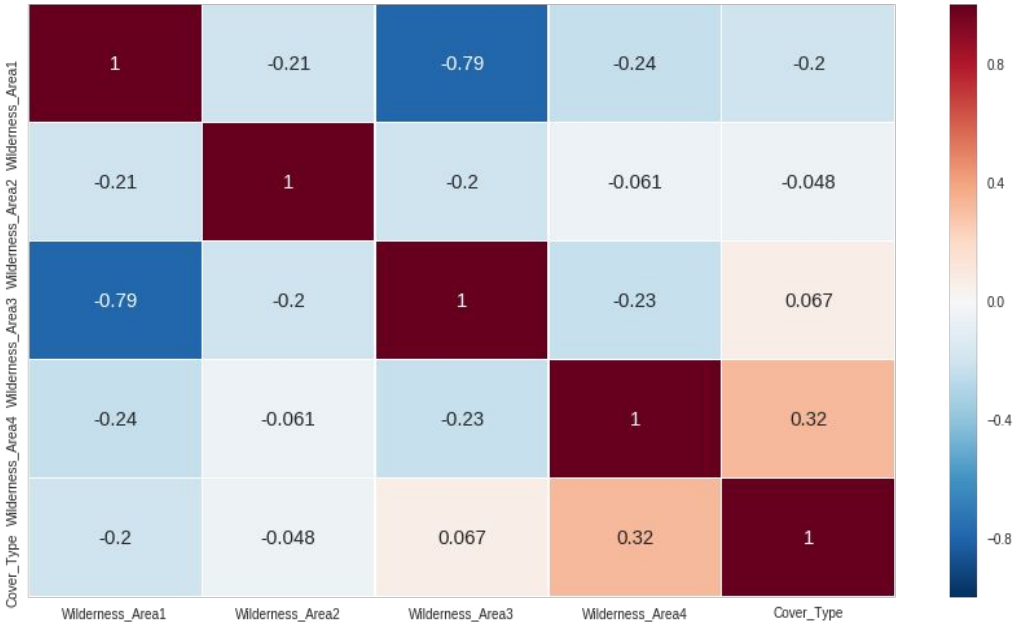


Data Exploration/ Visualization

- Categorical Variable Correlation Matrix: Wilderness Area

Looks pretty good! These aren't too very correlated, other than wilderness area 3 and 1 being inversely correlated.

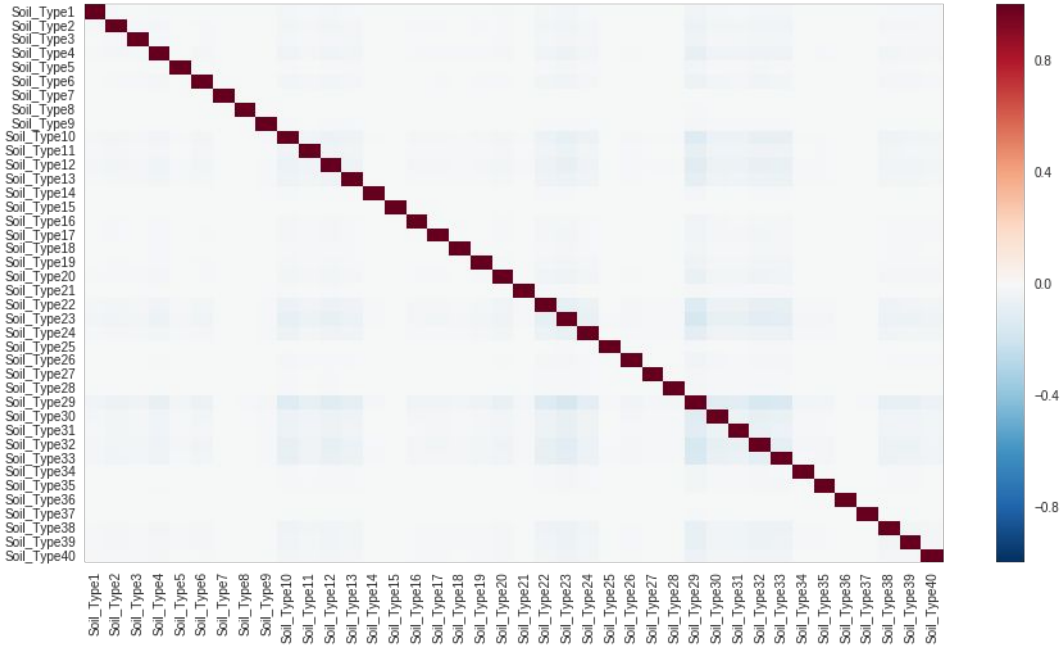
Again, doesn't surpass our .90 threshold for removal, so we'll be keeping all of these.



Data Exploration/ Visualization

- Categorical Variable Correlation Matrix: Soil Type

This looks great! Seems that our soil types differentiate from one another rather well.



Feature Engineering

- Binary variables already one hot encoded, called for less feature engineering
- Performed some minor feature engineering combining correlated variables
  - Will later see how they fare in feature importances

```
means_1 = df[['Hillshade_3pm','Hillshade_9am']].mean(axis=0)
stds_1 = df[['Hillshade_3pm','Hillshade_9am']].std(axis=0)
df['Hillsade_3pm_9am'] = ((df[['Hillshade_3pm','Hillshade_9am']] - means_1) / stds_1).mean(axis=1)

means_2 = df[['Hillshade_3pm','Aspect']].mean(axis=0)
stds_2 = df[['Hillshade_3pm','Aspect']].std(axis=0)
df['Hillsade_3pm_Aspect'] = ((df[['Hillshade_3pm','Aspect']] - means_2) / stds_2).mean(axis=1)

means_3 = df[['Vertical_Distance_To_Hydrology','Horizontal_Distance_To_Hydrology']].mean(axis=0)
stds_3 = df[['Vertical_Distance_To_Hydrology','Horizontal_Distance_To_Hydrology']].std(axis=0)
df['Vertical_Horizontal_Distance_To_Hydrology'] = ((df[['Vertical_Distance_To_Hydrology','Horizontal_Dista
nce_To_Hydrology']] - means_3) / stds_3).mean(axis=1)
```

Models

- Logistic Regression
- Random Forest Classifier
- XGB Classifier

Models

- Logistic Regression
  - Basic Model CV : ([0.64608655, 0.66651129, 0.65699738, 0.67722154, 0.65584881])
  - Best Parameters: {'penalty': 'l1', 'C': 1600}
  - class_weight: 'balanced'
  - Test Score: 0.684612273349225
    --- 0.2860429286956787 seconds ---
  - ROC_AUC: 0.7799067448276092

Clasification report:

	precision	recall	f1-score	support
1	0.698978	0.675273	0.686921	42254
2	0.775744	0.703839	0.738044	56780
3	0.621328	0.708017	0.661846	7110
4	0.332690	0.652174	0.440613	529
5	0.205866	0.383455	0.267903	1922
6	0.346995	0.469405	0.399023	3481
7	0.513749	0.801308	0.626089	4127
micro avg	0.684612	0.684612	0.684612	116203
macro avg	0.499336	0.627639	0.545777	116203
weighted avg	0.704791	0.684612	0.691531	116203

Confusion Matrix:

```
[[28533 10013    36     0   502   259  2911]
 [11471 39964  1504    19  2100  1504   218]
 [    0   276  5034   504    91  1205     0]
 [    0     0   130   345     0    54     0]
 [   26   932   173     0   737    53     1]
 [    0   320  1208   169   150  1634     0]
 [  791    12    17     0     0     0  3307]]
```

Models

- Random Forest Classifier
  - Basic Model CV: ([0.86205227, 0.86834463, 0.86182229, 0.86332831, 0.86416698])
  - Best Parameters: {'bootstrap': False, 'max_features': 8, 'min_samples_split': 4, 'n_estimators': 500}
  - class_weight: 'balanced'
  - Test Score: 0.9606034267617876
    --- 36.92564153671265 seconds ---
  - ROC_AUC: 0.9605518598460694

Clasification report:

	precision	recall	f1-score	support
1	0.968962	0.951602	0.960203	42254
2	0.959278	0.973723	0.966447	56780
3	0.946170	0.964135	0.955068	7110
4	0.888889	0.892250	0.890566	529
5	0.927077	0.853278	0.888648	1922
6	0.921449	0.913243	0.917328	3481
7	0.976938	0.964866	0.970864	4127
micro avg	0.960603	0.960603	0.960603	116203
macro avg	0.941252	0.930442	0.935589	116203
weighted avg	0.960638	0.960603	0.960534	116203

Confusion Matrix:

```
[[40209  1945     4     0     7     4    85]
 [ 1136 55288   123     2   112   110     9]
 [    1    68  6855    41     9   136     0]
 [    0     0    45   472     0    12     0]
 [   17   234    22     0  1640     9     0]
 [    5    85   196    16     0  3179     0]
 [  129    15     0     0     1     0  3982]]
```

Models

- XGB Classifier
  - Basic Model CV: ([0.7013015 , 0.69984941, 0.69755809, 0.69658993, 0.69293668])
  - Best Parameters: {'subsample': 0.7, 'max_depth': 9, 'colsample_bytree': 0.9, 'colsample_bylevel': 0.9}
  - Test Score: 0.8861475177060834 --- 9.213517189025879 seconds ---
  - ROC_AUC: 0.9072641409037657

Clasification report:

	precision	recall	f1-score	support
1	0.886283	0.855848	0.870800	42254
2	0.878556	0.917471	0.897592	56780
3	0.905900	0.911252	0.908568	7110
4	0.899431	0.896030	0.897727	529
5	0.917424	0.578044	0.709224	1922
6	0.873270	0.815570	0.843434	3481
7	0.961907	0.923916	0.942529	4127
micro avg	0.886148	0.886148	0.886148	116203
macro avg	0.903253	0.842590	0.867125	116203
weighted avg	0.886579	0.886148	0.885380	116203

Confusion Matrix:

```
[[36163  5936     2     0     8     2   143]
 [ 4314 52094   167     0    88   109     8]
 [    0   304  6479    38     3   286     0]
 [    0     0    45   474     0    10     0]
 [   17   760    29     0  1111     5     0]
 [    8   188   430    15     1  2839     0]
 [  301    13     0     0     0     0  3813]]
```

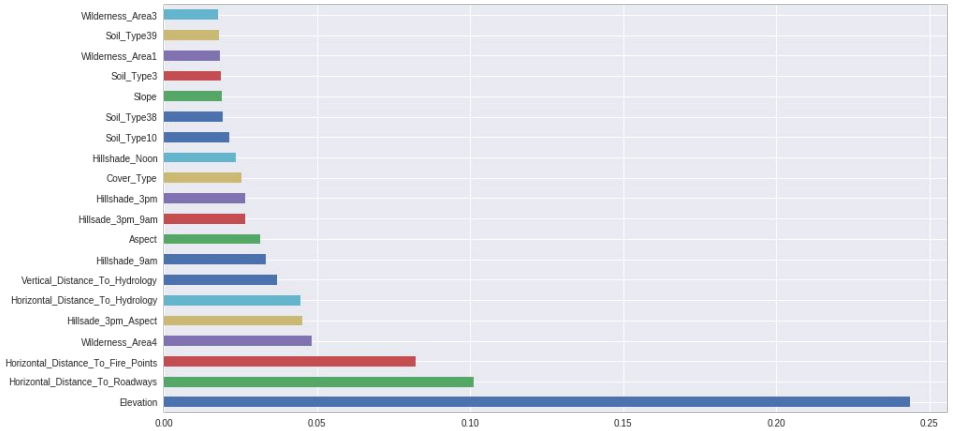
Model Comparison

- Random Forest predicts the forest!

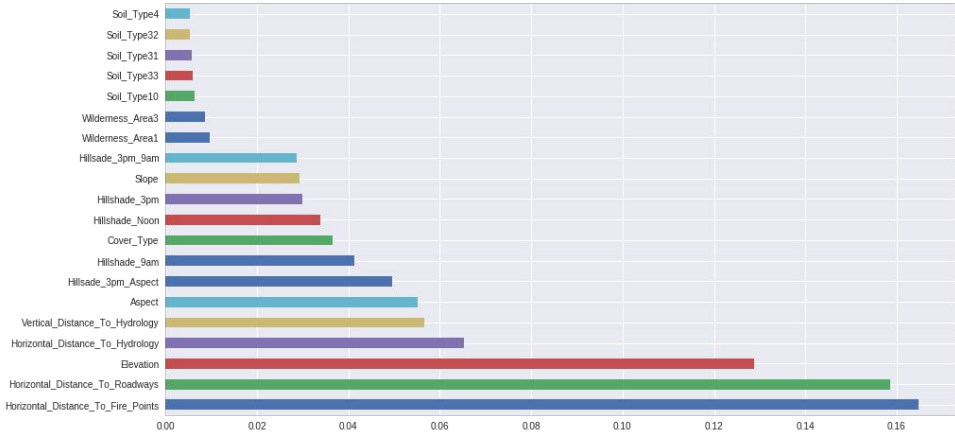
	Model	Test_Accuracy	AUC Scores
1	Random_Forest	0.960603	0.960552
2	Extreme_Boost	0.886148	0.907264
0	Logistic_Regression	0.684612	0.779907

Feature Importances

- Random Forest



- XGBoost



Conclusion

- We can see that our top three important features amongst both our Random Forest and XGBoost Classifier were Elevation, Horizontal Distance to nearest roadway, and the Horizontal Distance to nearest wildfire ignition point.

- Our top performing model was Random Forest with a 96.06% on the test set, and an average of .9605 for an AUC score, though with the longest execution time. Logistic Regression proved to be the least effective predictor with a score of only 68.46% on the test set, though quicker than the rest in speed.

Future Work

- Select the best features found through our XGBoost Classifier and Random Forest model in an attempt to achieve the same results with less computational power necessary and dimensionality reduction.

- Attempt to adjust the tuning parameters on the extreme boosting model, considering the max values ended up as our best params, to see if we could perhaps surpass the score of our Random Forest Classifier.
    - Incorporate scale_pos_weight in XGBoost random search for optimization of class imbalance

Questions?