DS-GA 3001: Machine Learning

Predicting Forest Cover for Campground Planning

Alex Simonoff, Anna Khazan and Rob Hammond

Introduction

For our project, we predicted the forest cover type of plots of land in Colorado's Roosevelt National Forest. In an attempt to classify regions that are hospitable for tourism and camping we classified regions based on characteristics of the cover types that make them more favorable for camping, i.e., forest types that are less likely to contain bears and have more tree cover. Favorable conditions were determined via research on the flora and fauna present in each cover type.

Data Source

We used Kaggle's Forest Cover Type Prediction dataset (https://www.kaggle.com/c/forest-cover-type-prediction), which consists of 54 continuous and binary features for 30 x 30 meter cells collected by the US Forests Service (USFS) and classification data provided by the US Geological Survey (USGS). The features describe the 7 unique types of forest cover: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-Fir and Krummholz.

Feature Engineering

To start, we calculated several new features based on the given data to better describe the cover types:

- Absolute degrees azimuth to True North
- Distance above sea level
- Elevation buckets (Elevation is trimodal)
- Euclidean distance to hydrology
- Soil description binary features
- First level interactions between every variable

Creating all of these features increased our feature space to a $5,000$ x $15,000$ dimension matrix. Any complex model would be intractable to run on a laptop so we reduced the dimensionality by removing features with 0 standard deviation as it provides no modeling value, resulting in a feature space of $2,800$ features.

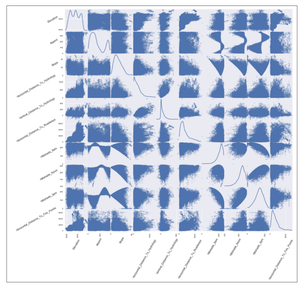

Figure 1: Correlation scatter plots used to inform feature engineering.

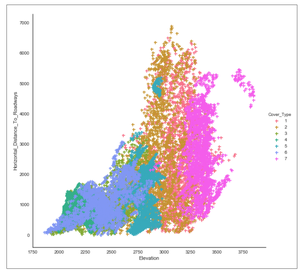

Figure 2: Example of highly correlated features.

Feature Selection

In order to save on computation time (as 2.8K features run on a single SVM takes several hours) we chose to perform two types of feature selection. Using the original data and MinMax scaling, standardized scaling and normalization on our continuous features we ran our interacted and feature engineered data through Random Forest, Extra Trees and XGBoost classifiers.

After fitting each model we stored the feature importances, sorted the features by the median feature importance they earned across the models and output the top 100. We ran into memory issues running feature selection on 2.8K features (15K rows) therefore we quartered the data into 4 sets of equally sized subsets, completed the process above to determine the top 100 features in each of these quarters and then ran the process one last time on the 400 features determined by combining the top 100 from each quarters feature selection.

Figure 3: Median feature importance for XGBoost.

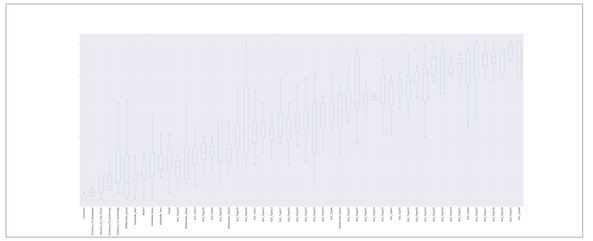

Figure 4: Median feature importance for original features after running 72 models with default hyperparameters.

Model Implementation

We split the provided training dataset of 15,120 observations into training and validation datasets, and tested our models on a testing dataset of 565,892 observations using Kaggle's submission API.

Baseline Models - We first tested versions of our data using Logistic Regression and Support Vector Machines (SVM) to have a benchmark for further algorithms. Using grid search ended up being prohibitively time-intensive, so we used the default hyperparameters instead.

Tree Based Models - After creating our baseline models we realized there was a lot of room for improvement. Using grid search over a number of parameters for the Extreme Gradient Boosting (XGBoost) and Random Forests we were able to make substantial improvements to our model. Similar to our baseline we had to carefully choose our data, and the number of hyperparameters to ensure our models would be able to run.

Results

Dataset	No Transform	MinMaxScaler()
Original Features	0.17989	0.54544
Engineered (Excl. interactions)	0.47318	0.54073
Top 100 Features	0.45303	0.54668

Table 1: Logistic Regression Performance

Dataset	No Transform	MinMaxScaler()
Original Features	0.02687	0.53176
Engineered (Excl. interactions)	0.05936	0.52973
Top 100 Features	0.04755	0.55526

Table 2: SVM Performance

Model	Dataset	Kaggle Score
XGBoost	Original	0.73838
XGBoost	Engineered	0.74069
XGBoost	Top 100	0.75907
Random Forests	Original	0.51607
Random Forests	Engineered	0.53788
Random Forests	Top 100	0.54041

Table 3: Tree Based Model Performance

Conclusions

With each of our models there was definite improvement when using our data set with feature engineering as opposed to the base data set provided by Kaggle. The improvement is due to the robust feature set used in the models, which allows for more ways to better understand the data. This is not surprising because we explicitly captured certain feature interactions and created a more robust feature set. As we expected, XGBoost produced the best Kaggle scores in our suite of models, which is typical of state of the art models.

Project Group Advisor: Brian d'Alessandro