Jackson Brandberg
Brown University, Data Science Initiative
Github Repository: https://github.com/jacksonbrandberg/data1030_project

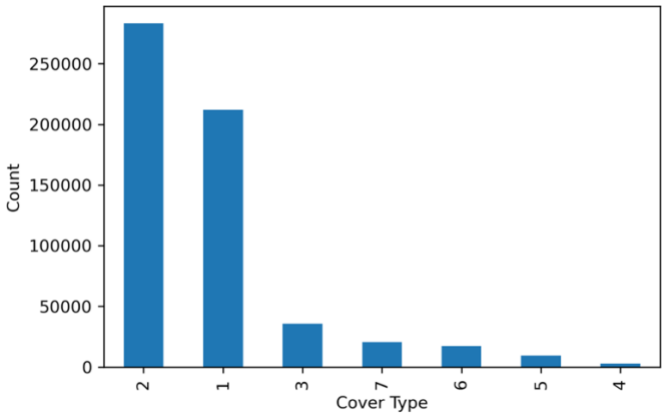
Predicting Forest Cover Types in the Roosevelt National Forest

Introduction:

The forest cover type dataset pulls its data from the Roosevelt National Forest in Colorado. This specific region was selected because it has remained relatively untouched from human development, so the type of forestry can be better attributed to national ecological occurrences. The target variable for this dataset is the type of forest covering, making it a classification problem. There are six possible types the covering can be classified as: spruce/fir, lodgepool pine, Ponderosa pine, cottonwood/willow, spruce/fir and aspen, Douglas-fir, and a seventh type which was unspecified. These covering are numbered 1-7 respectively. Soil type refers to soil types present in the observation, of which 40 can be present or not present. Wilderness area corresponds with the four wilderness areas in the region: rawah, neota, comanche peak, and cache la poudre. Distance to hydrology is the distance in meters to any water source. Hillshade indices are measures based on the shade given at the summer solstice. Lastly, distance to fire points refers to the distance to nearest wildfire ignition points. There are 581012 datapoints and 55 features in this dataset, making it a very large dataset. This dataset is particularly interesting and important as conservation efforts grow in the coming years to combat climate change. With different ecological conditions on the way, knowing which types of forest coverings thrive in certain conditions could be valuable for reforestation.

This dataset has been used in previous publications, such as “Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables" (Blackard and Dean 2000). They seemed to have a lot of success in using this data in a neural network and predicting forest coverings. The dataset has also been used in a Kaggle competition. It seems like it is consistently used to answer the question of what type of forest covering an area will have given certain properties. Among the notebooks, some used kNN, some used Random Forest. Accuracy got as high as 95% in some cases, and I hope to be able to achieve similar results with my own model.

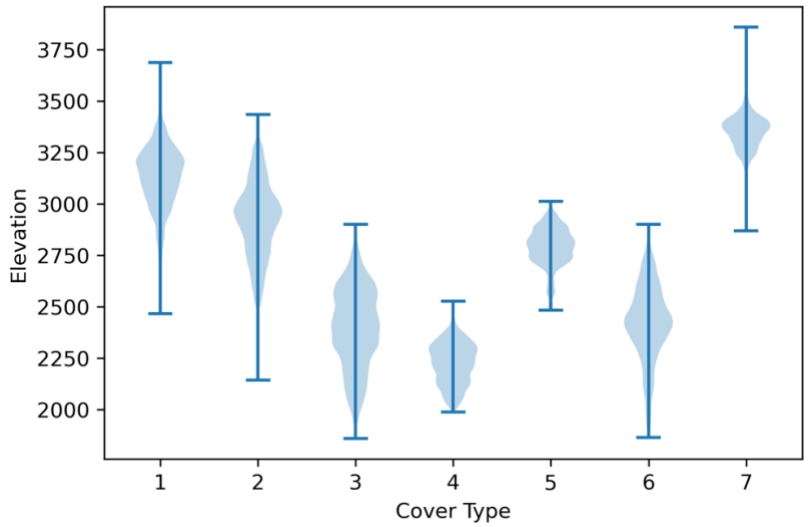
EDA:



The figure above shows the class imbalances of the target variable.

A bar plot for the target variable is pretty valuable in determining how to split the data and what classes are less common.

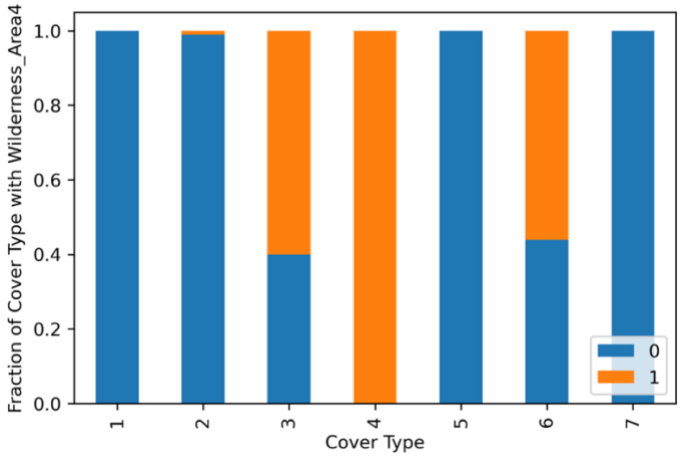
From this, we can see that 4 is the least common cover type, and 1 and 2 are by far the most common. This tell us for later that we will want to stratify the data to ensure that each cover type is included in the training, validation, and test sets. Another feature we should investigate is elevation, and its relation to cover type. In the description, it says that elevation can play a big role in cover type, so a violin plot would help show how that continuous feature is related the various cover types.



Violin plot comparing elevation and the target variable, cover type.

This violin plot gives a lot of insights on how elevation affects the target variable. 4, for example, seems to only be common in lower elevations, and this is helpful to know in classifying this rare cover type. 7 also has very high elevation. This plot shows that elevation may play a big role in determining what cover type the data points to.

Lastly, there are 4 wilderness area features that are binary, indicating whether that type of wilderness area is present in the data point or not. I explored all 4 stacked bar plots with the target variable, but one was particularly notable:



Wilderness area 4 and cover type stacked bar plot

From here, there are a few big takeaways. Cover type 4 is only found in Wilderness Area 4, so if any other wilderness area is present, then it cannot be 4. Also, cover type 1, 5**,** and 7 do not have this wilderness area at all, so if it is present, then those three cover types can automatically be ruled out.

Methods:

I used a stratified split on my data to ensure that even with the imbalance, all cover types would appear in training, validation, and test sets. My dataset is IID, as each data point pertains to a unique 30x30 meter sections of land. There is no group structure because each section of land is not going to be in the dataset more than once. 3 KFolds were employed in the kNN and logistic regression models, allowing for easy cross-validation tuning with a GridSearch CV. In the XGBoost model, 10% of the data was used to build the model for sake of efficiency and it looped through a parameter grid. 10% of data was left for the test set in the KFold approach, and 5% of data was left for testing in the XGBoost approach. Given the large size of the dataset, this should be enough points for testing.

A standard scaler seemed applicable to use on all features and would help normalize the values, all of which were continuous or binary. A label encoder had to be applied to the target variable because it was numbered 1-7, but it needed to be numbered 0-6, so label encoder can fix that. There are 54 features in the preprocessed data, which makes sense because no one hot encoder was applied, therefore there should be no new features created.

Given how large the dataset is, I used one random state to test the machine learning algorithms, as it is unlikely that the randomness in splitting would heavily influence the results. I choose f beta as my evaluation metric, given how imbalanced this classification problem was. I also set the averaging parameter in f beta to weighted to help combat the imbalance. I felt that both precision and recall were important in this metric (I am not trying to minimize false positives or false negatives) so I set beta to 1.

I started with LogisticRegression, as it performs well on large datasets and could serve as a useful baseline model. I used elasticnet as the penalty for regularization in order to apply a balanced regularization that combines both ridge and lasso methods, and the saga solver. I tuned the C and l1-ratio parameters. This helped allow for a spread between over and underfitting the model.

The next model I tested was kNN Classifier. kNN was tuned for the number of neighbors with values of 1,10, and 100 to have a range to show over and underfitting. The weights were set to uniform, as standard Euclidean distance seemed appropriate for model building.

The last model was XGBoost. Given the power of gradient boosting, I was excited and hopeful for the results this model would produce. After obtaining my trained model, I decided to test it on 95% of the data to see how it performed outside of the small sample it was trained and originally tested on. One limitation with this is that some training data will be included in this second testing sample, but it still seemed helpful to see how it performed outside the small sample given XGBoost's tendency to overfit. On an even smaller sample of my data, 1%, I tested out the changes in reg_alpha, reg_lambda, and max_depth parameters, finding max_depth had the largest impact and tuned for that only when training the actual model.

Given one random state, there is some small uncertainty with splitting. Also, there is also uncertainty in the randomness from the XGBoost col samples and subsample at each split. Given the large amount of data, the randomness of these should not affect the results too much.

Results:

Model	Logistic Regression	kNN	XGBoost
F Beta Test Score	0.71479	0.93829	0.99996
Parameter Ranges	C: 0.1, 1, 10, 100 l1_ratio: 0, 0.25, 0.5, 0.75, 1	n_neighbors: 1, 10, 100	max_depth: 1, 3, 10, 30, 100
Best Parameters	C: 1, l1_ratio: 0.5	n_neighbors: 1	max_depth: 30

This table shows the models tested, parameters tuned, and the results

Baseline Score: 0.656 (found by setting everything to cover type 2, which had the highest percentage of points for the target class)
**one random state was used so cannot find standard deviations in model test scores
Logistic Regression:

The model performed decently, giving a test score of 0.715. While not bad, it barely beats the baseline f beta score of 0.656. This model did fine, but not significantly better to attribute any credible results to it.

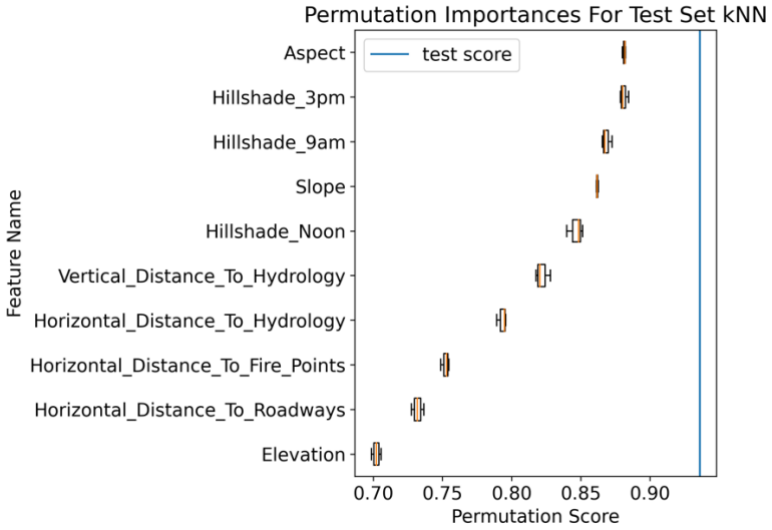
kNN Classififer:

I was a bit surprised to learn the best parameter for n-neighbors was 1. Usually, this would result in overfitting the model, so I was shocked that the cross-validation gave that as the best parameter. Nonetheless, it performed incredibly well on the test set, with an f beta score of 0.938, significantly higher than the previous model. Given the parameter though, if there are any major shifts in new data collection, it might perform poorly due to low bias that comes with n_neighbors = 1

XGBoost:

XGBoost performed extremely well. The best max_depth was a tie between 30 and 100, getting 1.0 validation f_beta score on both. On the test set, the tree with 30 max depth got a f_beta score of 0.99996. While these results are exciting, it's important to remember that XGBoost is prone to overfitting. With this in mind, I tested the model on 95% of the data, and it got a score of 0.91. It is still a very strong score, but it is not near the level of the one from the training and original test score, showing the overfitting that comes with this model.

Feature Importance:



Permutation plot for feature importance: kNN model

Feature permutation scores were calculated for both XGBoost and kNN. For both models, elevation was by far the most important feature. These findings are helpful, as elevation is a relatively easy feature to figure out if we wanted to apply these results for reforestation efforts, we know elevation can be a great indicator of a forest type that would be successful. It also coincides with some of the EDA we performed at the beginning, where elevation showed stark differences between cover types. Distance to roadways, fire points, and hydrology were the next best features (respectively) for both models. For XGBoost, each importance type was also measured. For gain and cover, some of the binary features (wilderness area and soil) were very important, but none of them were significant in total gain or total cover. Given the context, this makes sense. While soil type and wilderness area can be mildly important in determining cover type, larger geological features such as proximity to water and elevation would have a larger impact on the type of environment that is suitable to different trees.

Outlook:

The models that were built do have some limitations. As mentioned, the XGBoost model was only trained on a small sample of the data, and is prone to overfitting, so ideally it would be trained on all the data over a few days. Even then, the two best performing models are overfitting to some level, so it might not remain relevant when new data comes in especially given the changing climate. Also, multiple random states could help solidify the validity of these models. To improve the model, feature engineering could be applied to try and achieve a better result. Feature selection could also come into play, which could improve computational efficiency while maintaining accuracy. Although the data are not available, data on current tree covers in the area (these data are from 2000), more recent data could help build a better model.

References:

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science

Blackard, Jock A. and Denis J. Dean. 2000. "Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables." Computers and Electronics in Agriculture 24(3):131-151.

https://archive.ics.uci.edu/ml/datasets/covertype