Alex Manioudakis
Brown Data Science Institute
https://github.com/Icebox00/CoverType_Final.git

Forest Cover Type Classification Project

Introduction

Purpose and Goal

For issues of natural resource and conservation management, predictive models can support decision-making processes and strategies. With the rise of deforestation and climate change across the world, it is imperative that we maintain a consistent understanding of the relationships between different aspects of ecosystems. While many specialists with years of research can already distinguish between which trees are present in a given ecosystem, crafting a machine learning model that can draw from information on the ecosystem and give predictions streamlines these classifications. Moreover, creating such a model can provide a framework of reference for replications with data from other ecosystems. This project aims to predict the type of tree cover for a given area using several classification models, selecting the model with the best performance on the data.

Forest Cover Type Dataset: Background, Attributes, and Target Variable

The Forest Cover Type dataset was chosen from the UCI Machine Learning Repository. It was collected by the United States Geological Survey in partnership with Colorado State University. The data itself consists of 12 measured attributes across 581,012 samples of 90 square meter areas within the Roosevelt National Forest in Colorado. It includes 4 broad "wilderness areas" within the forest and other attributes such as soil type and elevation. The data is IID and does not contain null values; also, its categorical features are already one-hot encoded to expand 12 attributes into 54 columns of data. The target variable is the last variable listed below. It is the tree cover type of a given sample and has 7 possible values. Each value corresponds to a cover type. See the figure below for a full list of each feature.

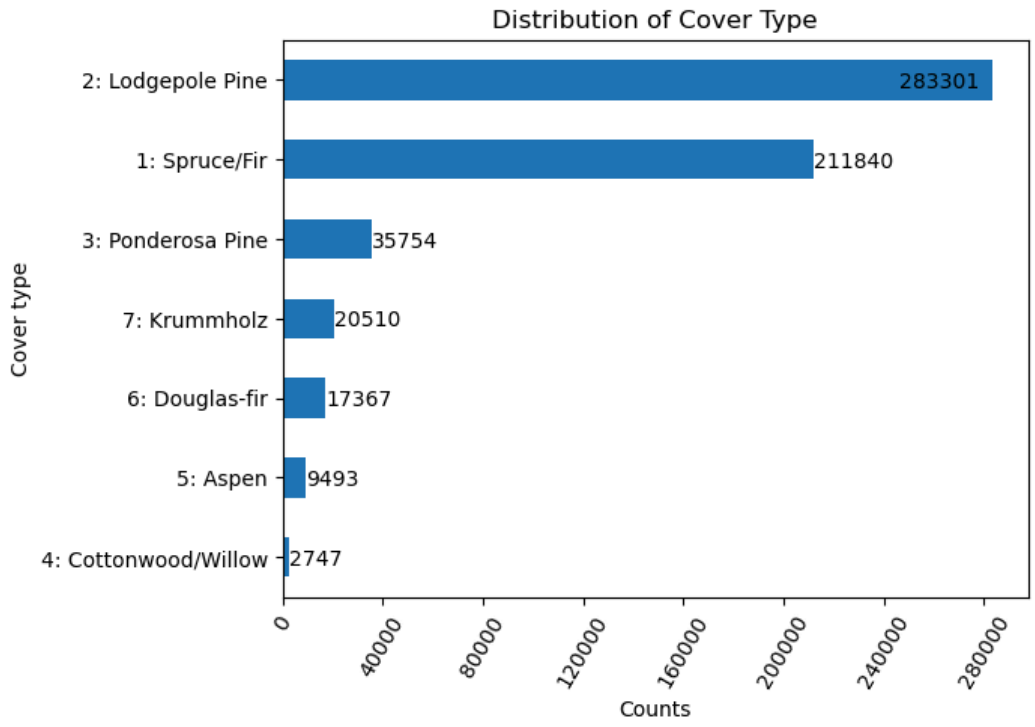
Name	Data Type	Measurement	Description
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horizontal_Distance_To_Hydrology	quantitative	meters	Horz Dist to nearest surface water features
Vertical_Distance_To_Hydrology	quantitative	meters	Vert Dist to nearest surface water features
Horizontal_Distance_To_Roadways	quantitative	meters	Horz Dist to nearest roadway
Hillshade_9am	quantitative	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade_Noon	quantitative	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade_3pm	quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points	quantitative	meters	Horz Dist to nearest wildfire ignition points
Wilderness_Area (4 binary columns)	qualitative	0 (absence) or 1 (presence)	Wilderness area designation
Soil_Type (40 binary columns)	qualitative	0 (absence) or 1 (presence)	Soil Type designation
Cover_Type (7 types)	integer	1 to 7	Forest Cover Type designation

Exploratory Data Analysis

For EDA, I needed to undo the one-hot encoding on the categorical variables for graphing and getting descriptive statistics. For univariate analysis, I used histograms to examine all continuous attributes and showed sample counts across each category for every categorical attribute. I also conducted some bivariate analysis, especially after uncovering the imbalance contained in the dataset across target classes. For my target variable, I also made sure to examine sample counts for each class and produced bivariate graphs with other attributes (especially categorical ones). Lastly, I included correlation matrices to check for multicollinearity, ultimately deciding not to do any feature selection.

Target Variable

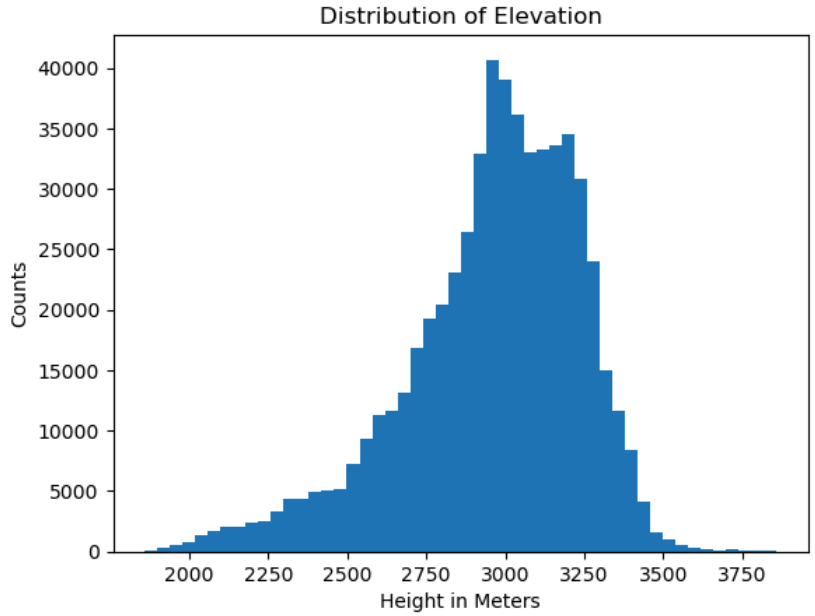
The target variable was the first aspect of the dataset that I examined, as it is crucial to understand its distribution for splitting and implementing the model. The bar chart revealed a large imbalance in the dataset, with the vast majority of data was either a member of class 1 or 2 (Lodgepole Pines or Spruce/Fir). This indicated that data-splits would have to be stratified across classes to ensure all classes were represented across all sets.



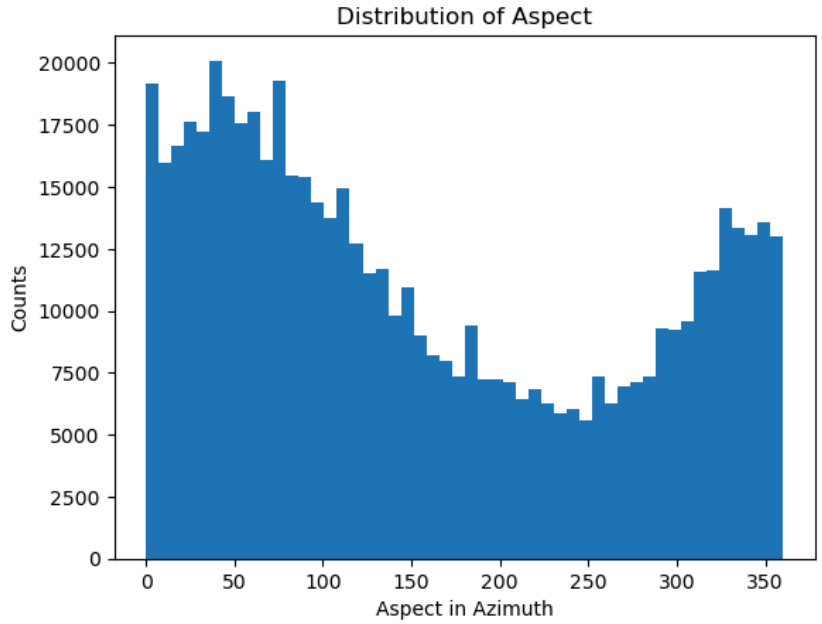
Attributes

Across all attributes and their histograms, nearly none of them had normal distributions. Indeed, the majority contained right-skewed distributions, though none contained egregious outliers that needed to be dealt with on their own (ex. removed from model training). Lastly, a couple features had bimodal distributions, which was often due to the nature of the variables

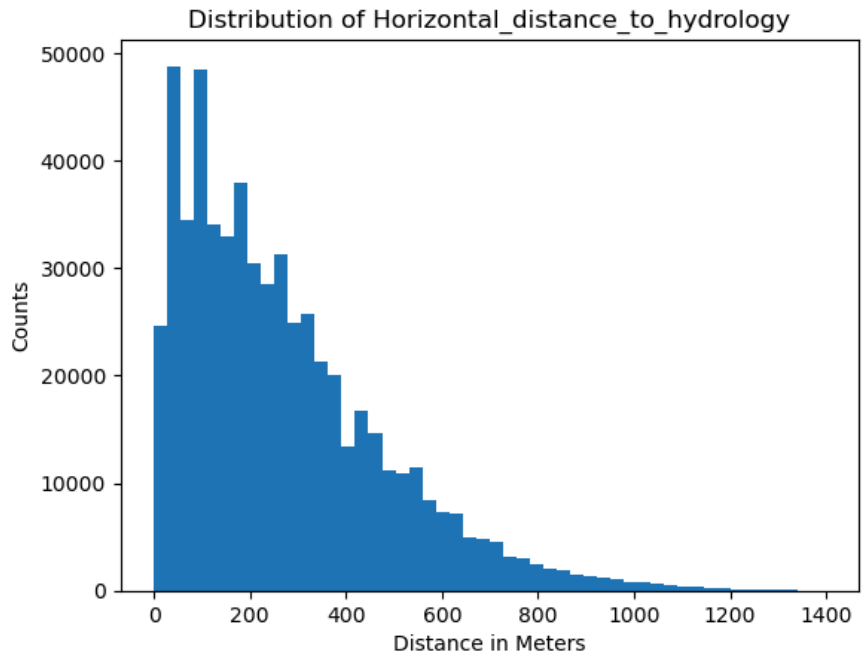
themselves. Nonetheless, knowledge of these distributions was extremely useful later when deciding upon regularization in building the model pipeline and parameters.



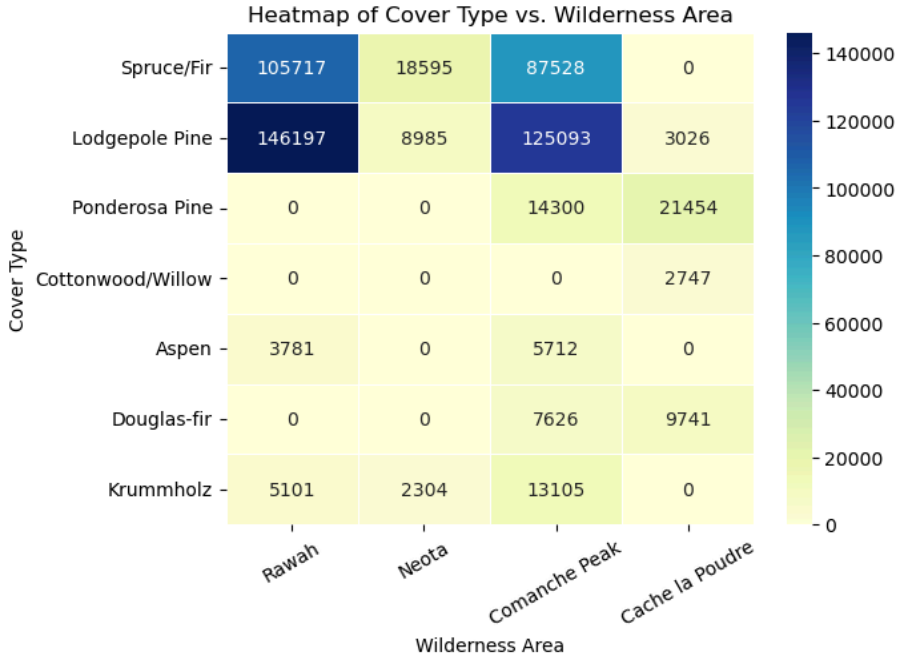
Elevation, as you see above, was the attribute that came closest to a normal distribution. It was still quite concentrated around the mean, and the standard deviation was quite low after normalization.



Aspect was one of the more bimodally distributed attributes. This was largely because of the concept of “back-azimuth” which is the idea that aspect is mirrored across a 360 degree azimuth axis. As such, values closer to 360 are functionally similar to values closer to 0.

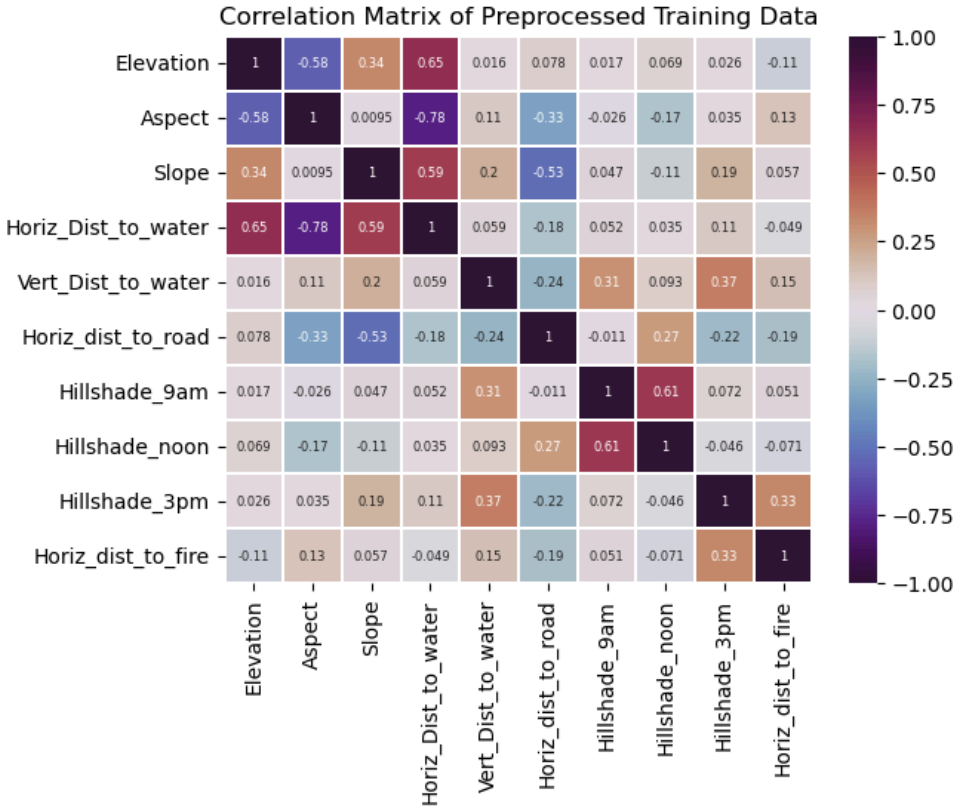


Horizontal distance to hydrology was an attribute that exemplified the common right-skew seen across the dataset. It shows the distance from a water feature (such as river or lake) to the given sample.



After also noticing an imbalance in the wilderness area categorical feature, I wanted to see its relationship with the imbalance target variable. In fact, there is a significant overlap between the two, and this led me to briefly consider removing it from the dataset in preprocessing or to group by wilderness area. While the latter would fundamentally change the outcome and question of the project, the former I considered to avoid potential overdependence

on one feature entirely (one that did not contain any true environmental data, just a name denoting area). However, I chose to keep wilderness area because it remains true to the goal of the project, and it emulates the process of classification of our environment (i.e location is a valid and frequently-used feature for classification in the real world).



For the correlation matrix, I was looking for any features that displayed a correlation coefficient higher than ~85% with another feature. I chose this because values above or at that threshold have caused issues with linear models in particular; however, the sheer number of attributes after preprocessing meant that a coefficient around something as close as 80% would not appear too dominantly in the final model, and I wanted to boost model performance by preserving as many attributes as possible. While some coefficients came close to this threshold, none met it, and thus I felt comfortable with including all variables in the final pipeline.

Methods

Splitting

First, I reduced the size of the data I would use in the pipeline so it would run in a reasonable time (even with large compute clusters). I stratified across the target variable to ensure every class was included in the subsample, and took 20% of the 581,012 data points.

For splitting my subsample,I used a stratified train_test_split function across the target variable to split the data into testing data (20% of subsample) and training & validation data bundled together for cross validation.

For my cross-validation, I used StratifiedKFolds with 4 splits to be included in the grid search function of the pipeline. Through all of this, I used 3 random states to account for randomness in the splitting.

Preprocessing

In the preprocessing stage, I used StandardScaler to normalize the majority of my continuous features except for those with defined minimums and maximums. Aspect and all 3 of the hillshade indexes were preprocessed using MinMaxScaler. The categorical features came one-hot encoded with the dataset. Lastly, all features were normalized with the StandardScaler for consistency.

ML Pipeline

The ML pipeline first included the 2 previous steps, and then finally implemented several classifiers through a GridSearchCV function. I decided to implement 4 classifiers: a logistic regression model with L2 regularization, a KNeighborsClassifier, a XGBoostClassifier, and a RandomForestClassifier, all with a number of parameters for permutation in the gridsearch. See the table below for all models and parameters used.

Model	Parameter	Values
L2 Logistic Regression	Regularization strength	[10, 100, 1000]
KNeighborsClassifier	Number of neighbors	[2, 3, 5]
	Weights	[‘uniform’, ‘distance’]
XGBoostClassifier	Max Depth	[18, 25, 30]
	Learning Rate	[0.1, 0.2, 0.4]
	Num of Estimators	[300, 400, 500]
RandomForestClassifier	Max Depth	[40, 48, 53]
	Maximum Features	[0.5, 0.75, 1.0]

I chose L2 logistic regression to deal with outliers that I noticed in the skews of many of the attributes’ histograms in the EDA process, KNeighborsClassifier due to its potential to perform well with skewed data and its unique accounting for clusters, XGBoostClassifier for its

computational efficiency, and RandomForestClassifier for its prevalence in classification as a practice and its mutually exclusive and exhaustive nature. The parameters for each were chosen after multiple cycles of wide-range tuning until I could simplify the model parameters into 3-4 length lists per parameter for efficiency with the compute cluster.

Results

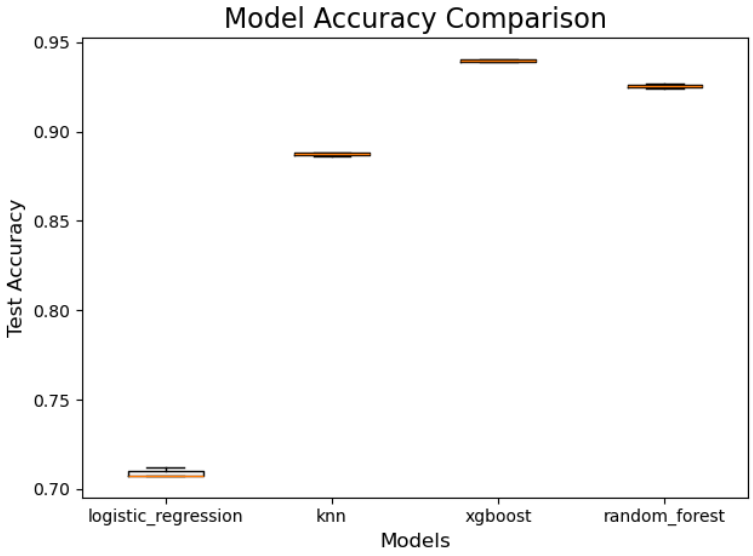
The results of the pipeline revealed some interesting facts about the dataset and each model's performance. I chose accuracy as the best evaluation metric for the data because there is no need to account more for false positives or false negatives because there are no outstanding consequences for the frequency of either of those occurring. I also decided to use a weighted F1 score to add a more holistic picture of the results, even though accuracy is my principal metric.

In addition to evaluation metrics, I wanted to examine the importance of various features in my results, so I used a permutation importance method as well as Shap value graphs for each class to explain their impact on the outcomes.

Accuracies

The **baseline accuracy of the entire dataset was 0.4876**, which was calculated by choosing the most frequent class and taking its proportion of the total data as a baseline. While logistic regression performed the worst according to this metric, it still had an accuracy of around 71%. KNeighbors performed significantly better with an accuracy of around 89%, but still performed worse than random forest and XGboost with around 93% and 94% respectively. All of these accuracies were taken as evaluations on the test data.

```
--- Final Test Scores ---
logistic_regression: 0.7090 ± 0.0022
knn: 0.8874 ± 0.0008
xgboost: 0.9397 ± 0.0008
random_forest: 0.9254 ± 0.0012
```



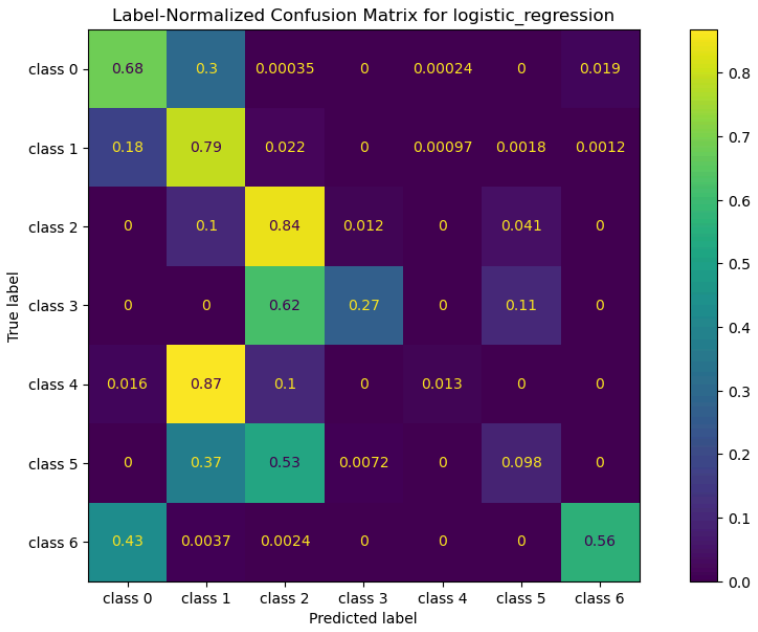
F1 Scores (secondary evaluation metric)

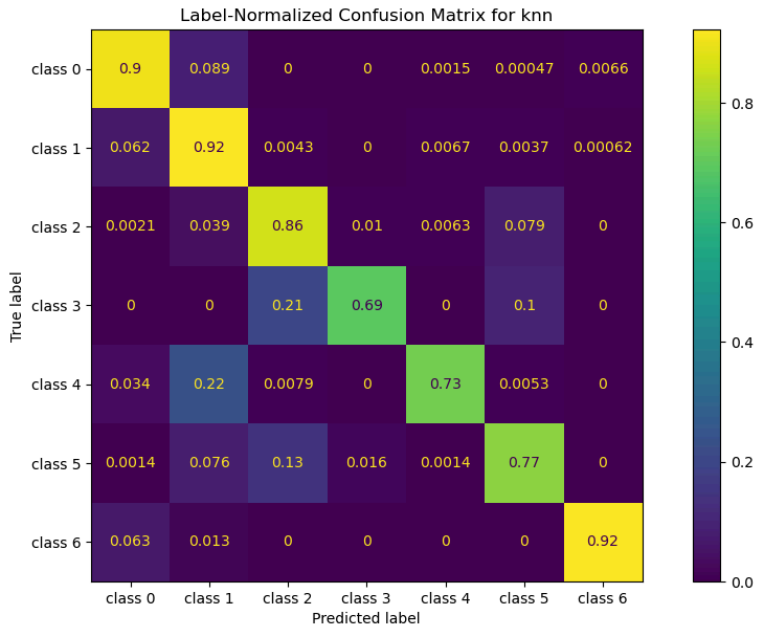
F1 scores were used as a secondary evaluation metric in this project. **The baseline F1 score was 0.3196** and was calculated using a dummy classifier which took the F1 score of the most frequent class (which was class 2). As one can see from the F1 values below, all of the models performed significantly better than the baseline by several

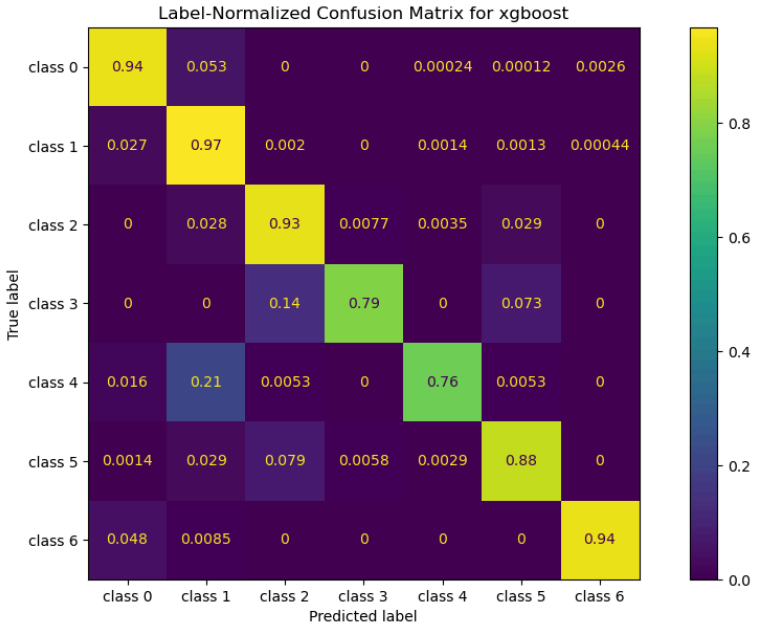
standard deviations. The F1 score was chosen because there was no need to weight recall or precision more than the other.

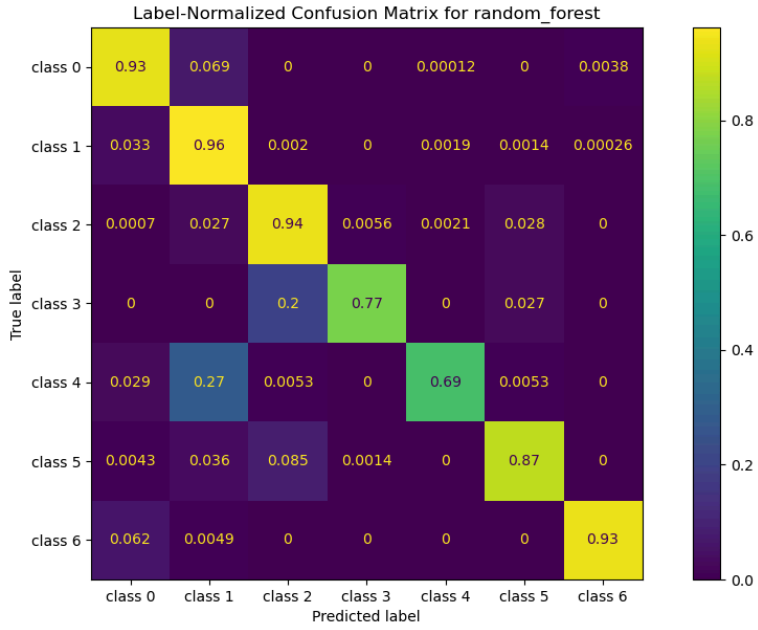
```
--- Final F1 Scores ---
logistic_regression: 0.6982 ± 0.0005
knn: 0.8993 ± 0.0086
xgboost: 0.9466 ± 0.0045
random_forest: 0.9345 ± 0.0059
```

Confusion Matrices



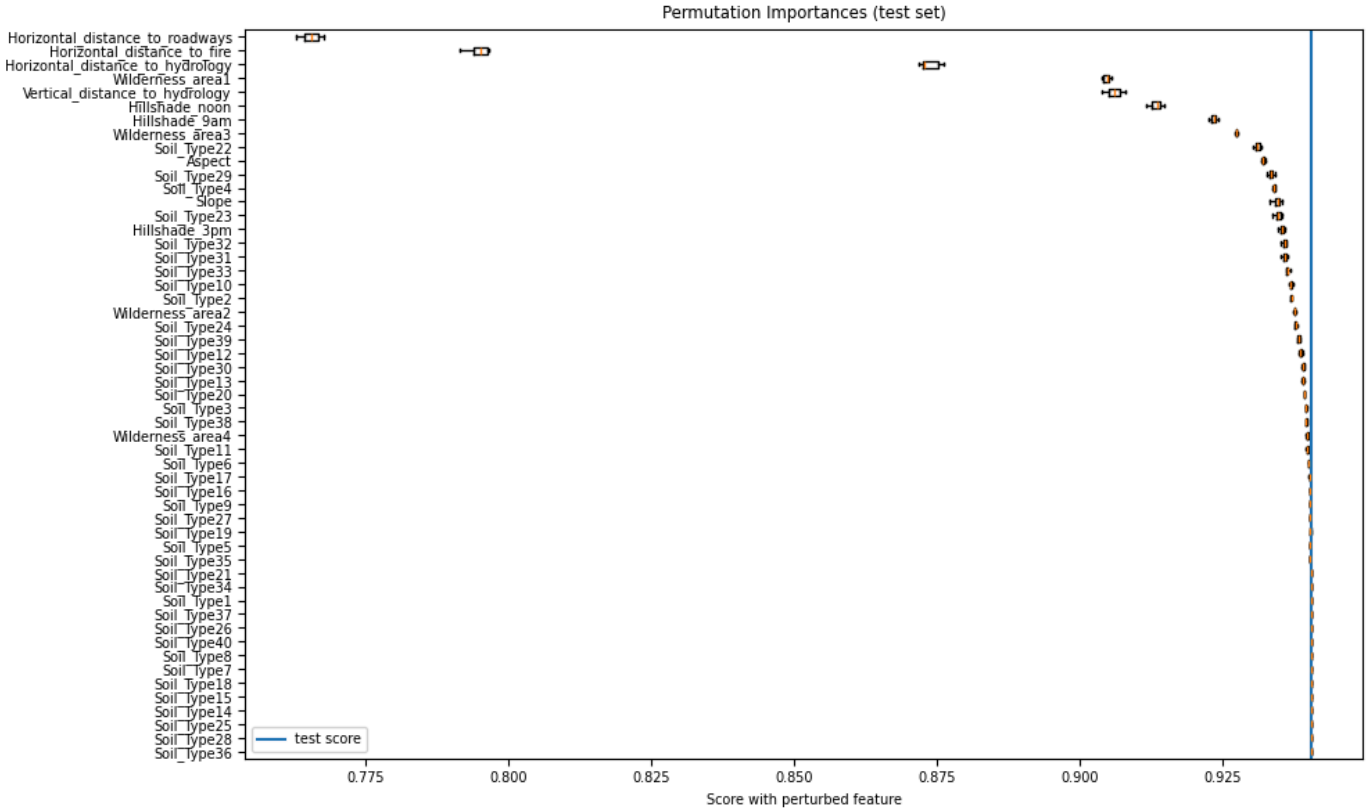






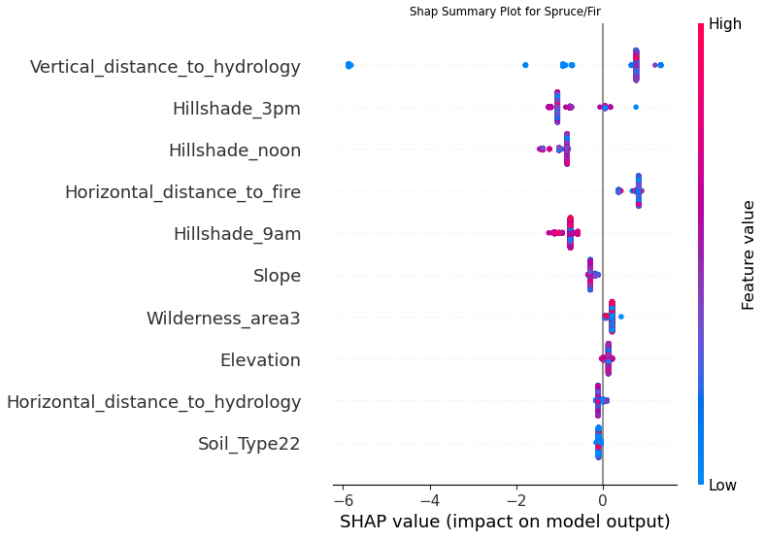
The label-normalized confusion matrices above illustrate the progressive improvement of the model from logistic regression and KNN to the split-based models of xgboost and randomforest classifiers. One can see the accuracy consistently increase with respect to the type of model, and this spurred a major takeaway from the project: how the success of certain models relate to how we classify nature and the environment in real life.

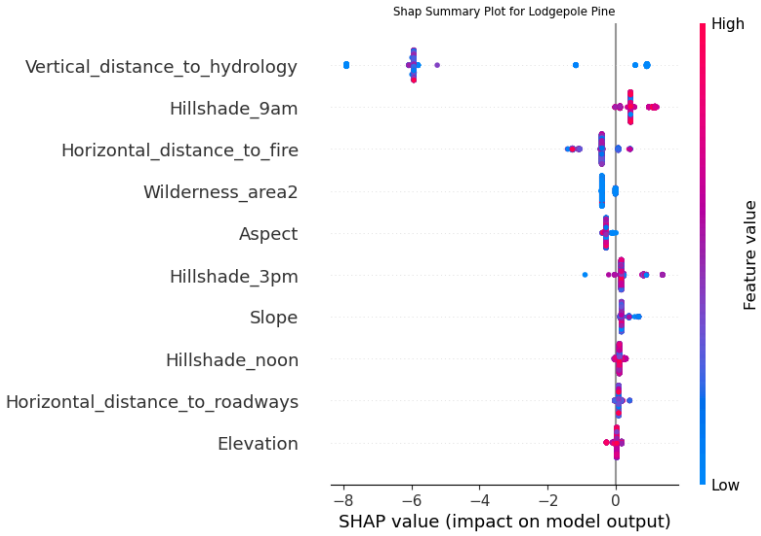
Feature Importance (XGBoostClassifier)



The perturbation importance graph shows the score of our best performing model, XGBoostClassifier, across each perturbed feature. From this, we can glean which features are most significant for the model based on how they affect its performance. As we can see, horizontal distance to roadways, fire points, and vertical distance to hydrology all had the most impact on model performance and are thus the most significant to the success of the classifier.

Taking a further look with shap graphs below, we can see how these variables have an impact on the two largest classes, in particular the vertical distance to hydrology. Other attributes such as hillshade index in the morning also had a significant impact.





Outlook

The results of this project showed how successful XGBoost and RandomForest were at classifying tree cover types given our data. Similarly to how we classify aspects of nature in real life (i.e such as with genus and species trees), XGBoost and RandomForest detected which features were best to split on and continued narrowing them down. This invites further research into how these split-based models perform when classifying other aspects of nature such as species of animals or plants given their attributes. Perhaps, in the world of environmental studies, this relationship is a new frontier that can be explored more. Beyond how the models relate to our real life practices, I believe that the skewed distributions of many attributes greatly benefitted the split-based models, because they provided waterfall-like steps in the form of attributes for the models to split upon. Lastly, the format of the ML pipeline I created provides a great framework for other projects on environmental data in different areas around the world. However, it is important to note that these models in their trained state cannot be generalized to other ecosystems.

For further considerations, I would have liked to use a neural network on this data, as it could have yielded even stronger results with enough tuning. I would have also liked to conduct more localized SHAP value examinations and to perhaps group my splits by wilderness area. All in all, this project found XGBoostClassifier and RandomForestClassifier to be considerable successes when predicting tree cover types.

References

[1] Blackard, J. (1998). Covertype [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C50K5N.