# Project Report: Predicting Forest Cover Type

Auguste Piromalli - Solal Danan - Sami Kalai

MAP541 – Machine Learning II

Master of Science and Technology: Data Science for Business

École Polytechnique

March 2024

## 1 Introduction

In this report, we present a comprehensive analysis of a Kaggle machine learning project focused on predicting cover types in forest areas using different geographical, cartographic, and environmental data.
The dataset contains information obtained from the US Forest Service (USFS) Region 2 Resource Information System (RIS) data, as well as data from the US Geological Survey (USGS). Our main goal is to build an accurate predictive model that can classify the forest cover type from the data. We have two main datasets:

- A training set with 15,120 samples
- A test set with 581,012 samples

One of the challenges of this competition is the larger amount of data in the test set. The datasets consist of both quantitative and qualitative attributes, including among others elevation, aspect, slope, distances to hydrology features and roadways, sunlight exposure (hillshade), wilderness area designations, and soil type designations. There are four wilderness areas in the dataset, each represented by a binary column indicating its presence or absence. Similarly, there are 40 binary columns representing different soil types based on the USFS Ecological Landtype Units (ELUs) for the study area.

**Target Variable:** The forest cover type is the predictor while the other attributes serve as predictors. It includes seven classes: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, and Krummholz. Each cover type is represented by an integer value ranging from 1 to 7.

**Objective:** Our goal is to surpass the previous best accuracy score of 0.75 achieved in the last iteration of this challenge. In our past attempt, minimal data pre-processing and feature engineering were performed, and only the Random Forest classifier [9] was used. This time, we aim to employ more advanced methods, including preparing the data in different ways with advanced feature engineering, fine-tuning model settings with hyperparameter optimization, and using different machine learning models to get better results and improve our predictive performance.

With a clear understanding of our objectives and of the dataset's overall structure, we will proceed with the exploratory data analysis (EDA) phase to glean insights into the dataset's characteristics. Subsequently, we will cover data preprocessing, feature engineering, model evaluation, and testing phases. We used this approach aiming to develop reliable and accurate machine learning models for forest cover type prediction.

## 2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in any machine learning project, serving as the foundation for developing a deeper understanding of the dataset's characteristics and uncovering insights that can guide subsequent modeling strategies. Through EDA, we aim to identify patterns, anomalies, and relationships within the data, enabling us to make informed decisions about feature engineering, model selection, and ultimately, improving the accuracy and efficiency of our predictive models.

Firstly, we detail the distribution of the target variable ('Cover_Type') in order to check if there is no class imbalances during the model training phase. The two plots below helped us to affirm that the distribution of trees' cover types was perfectly uniform distribution in our training dataset.

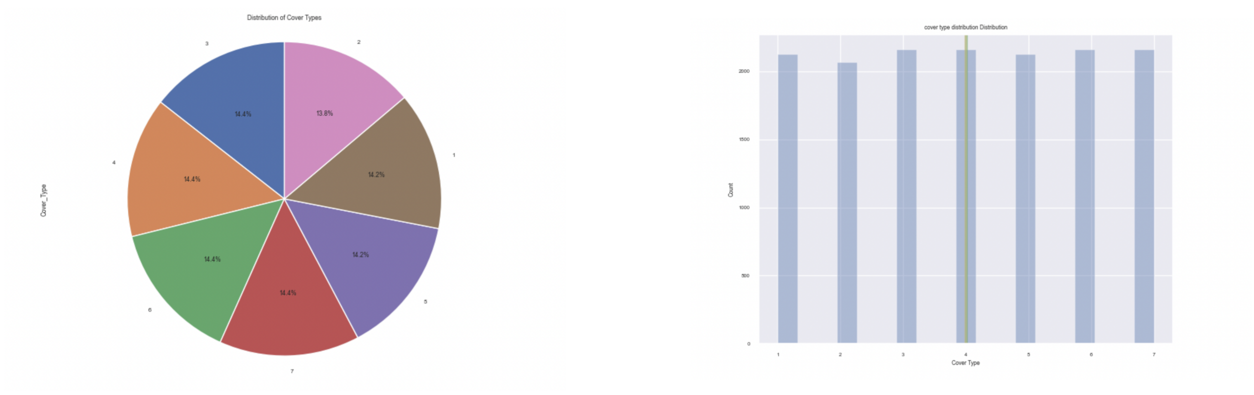

Figure 1: Uniform Distribution of Target Variable 'Cover_Type' in the Training Dataset

## 2.1 Categorical Variable Analysis

In our Exploratory Data Analysis, the categorical variables 'Wilderness Area' and 'Soil Type', represented through one-hot encoding, were scrutinized for their impact on the distribution of forest cover types. Our analytical efforts unveiled distinct associations between certain cover types and their preferred environments.

Particularly noteworthy is the proclivity of Cottonwood/Willow (Cover_Type 4) for Wilderness Area 4, suggesting a unique ecological niche. Additionally, certain species like Spruce/Fir, Lodgepole Pine, and Krummholz (Cover_Types 1, 2, and 7) are most commonly found in Wilderness Areas 1 to 3, which suggests these areas' ecological conditions are particularly conducive to these types.

When examining 'Soil Type,' we observed that some cover types are less selective, demonstrating adaptability by thriving in a variety of soils, while others exhibit distinct soil preferences. This variability speaks to the diverse adaptations and ecological strategies of the species represented in the dataset.

The provided visualizations capture the intricacy of these relationships:

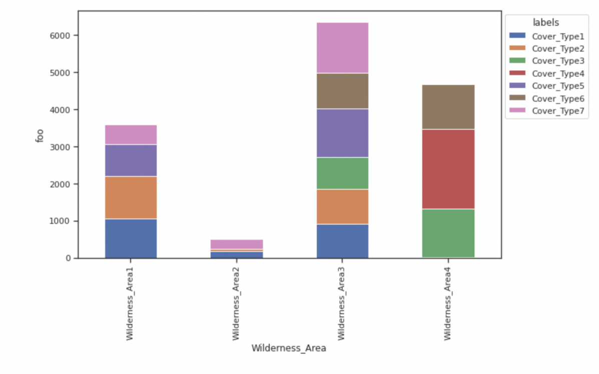

Figure 2: Distribution of forest cover types within each Wilderness Area, revealing species-specific habitat preferences.

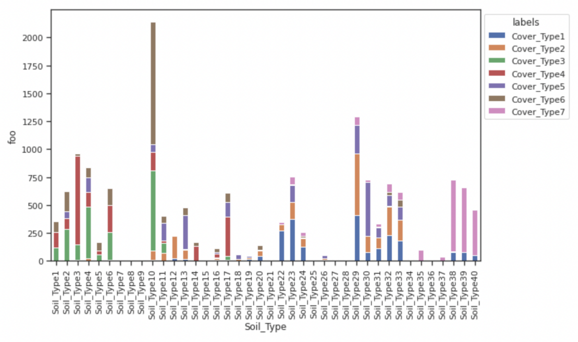

Figure 3: Forest cover type distribution over various soil types, indicating ecological adaptability and soil-type specificity.

These insights, distilled from our data-driven visualizations, emphasize the intricate interplay between terrain, soil characteristics, and the distribution of forest types. This understanding is vital for enhancing the predictive capability of our models, ensuring they reflect the complex realities of ecological systems.

## 2.2 Continuous Variable Analysis

As we advanced our Exploratory Data Analysis, we conducted an in-depth examination of the distribution and relationships of the numerical features within our dataset. By utilizing histograms, box plots, and violin plots, we aimed to discern which features hold the most predictive power for the 'Cover_Type' target variable.

A particularly noteworthy observation was the sigmoidal relationship between the Aspect and Hillshade_3am variables, well-characterized by the logistic function $\frac{1}{1+e^{-x}}$. This finding suggests that the Aspect may exert a nonlinear influence on Hillshade_3am, offering valuable insight into the interaction dynamics of these variables. Additionally, our analysis revealed a pronounced correlation between Hillshade_9am and Hillshade_3pm, visually substantiated and quantitatively supported by a high Pearson correlation coefficient. Given this redundancy, it may be prudent to consider discarding one of these features to streamline the model while preserving its predictive capacity. Indeed, features that are strongly correlated with each other are known as collinear and removing one of the variables in these pairs of features can often help a machine learning model generalize and be more interpretable, thus, reducing overfitting.

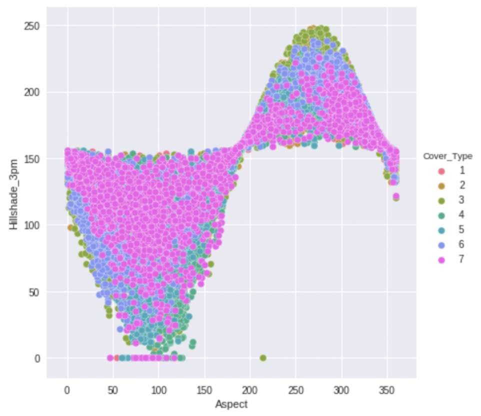

Figure 4: Scatter plot displaying the sigmoidal relationship between Aspect and Hillshade_3am across different Cover_Types.

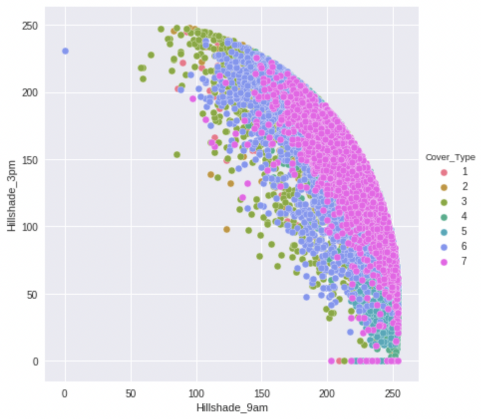

Figure 5: Scatter plot illustrating the correlation between Hillshade_9am and Hillshade_3pm, color-coded by Cover_Type.

The feature 'Elevation' (measured in feet) emerged as an especially influential predictor during our analysis. The visual representations unequivocally illustrated its capacity to differentiate among the diverse cover types. For instance, Cover_Type 7 predominantly occupies the higher elevation zones, around 3400 feet, while Cover_Type 4 is typically found at lower elevations, ranging from 2000 to approximately 2400 feet. The stark variations in elevation across all cover types underscore its importance as a key feature.

However, the remaining features did not demonstrate a clear distinction between forest cover types. This implies that, while individually they may not be as effective, there could be an opportunity for feature engineering. By combining features or creating new ones, we aim to enhance the model's ability to discriminate between classes, thereby improving the predictive performance of our machine learning algorithms.

Our exploratory journey continues with a careful consideration of how to best leverage these insights, through feature engineering and selection, to build a model that is both interpretable and highly accurate in classifying forest cover types.

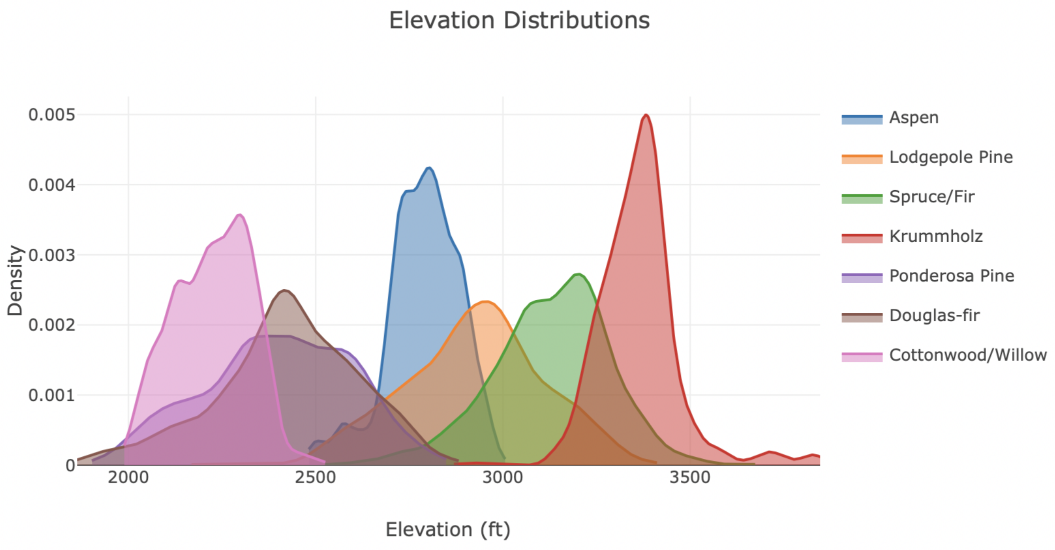

Figure 6: Density plots of Elevation Distributions for different forest Cover Types. The plots reveal distinct elevation ranges for each type, highlighting Elevation as a key feature for differentiating between classes.

## 3 Data Preprocessing and Feature Engineering

Before proceeding with modeling, extensive feature engineering and transformation were conducted to enhance predictive performance and capture meaningful relationships within the dataset.

### 3.1 Data Preprocessing

Outliers and missing values were addressed as the initial step. Outliers were detected using a function called outlier_function, which calculates the first and third quartiles along with the interquartile range (IQR) for a given column of the dataframe. Based on these metrics, upper and lower limits were determined to identify outliers conservatively. Outliers falling outside these limits were considered and subsequently removed to mitigate their potential impact on model performance.
Furthermore, as we believed that a distance could not be negative, we addressed the issue of negative values in the vertical distance to hydrology variable by transforming them into their absolute values, ensuring that all the distances reflect a non-negative measurement.
Once outliers were addressed, we moved to our categorical variables. Although Ordinal Encoding [8] was tested to encode our categorical variables ('Wilderness-Area' and 'Soil-Type'), one-hot encoding [7] was preferred and showed better results. Regarding numerical features, we chose to use Standard Scaler [11] to encode them i.e., to normalize the feature values so that they would have a mean of 0 and a standard deviation of 1. This normalization is a crucial step because it ensures that each feature contributes equally to the distance calculations in models that are sensitive to the scale of the data such as KNN Classifier. Additionally, standard scaling helps to speed up the convergence of gradient-based optimizers by maintaining a consistent scale across all dimensions.

These data-preprocessing steps ensured that the dataset was prepared for subsequent analysis and feature engineering.

## 3.2 Feature Engineering

Feature engineering techniques were applied to derive new features or transform existing ones to improve model performance. This involved creating composite features, such as Euclidean distance to hydrology, and applying mathematical transformations like square roots to certain features to improve linearity or reduce skewness.

**Linear Combinations**
The following features were engineered:

- **Euclidian Distance To Hydrology**: Calculated as the Euclidean distance between horizontal and vertical distances to hydrology features.
- **Mean Elevation Vertical Distance Hydrology**: The mean elevation and vertical distance to hydrology features were combined to capture their combined influence.
- **Mean Distance Hydrology Firepoints**: Calculated as the mean distance between hydrology and fire points.
- **Mean Distance Hydrology Roadways**: Calculated as the mean distance between hydrology and roadways.
- **Mean Distance Firepoints Roadways**: Calculated as the mean distance between fire points and roadways.
- **Square Root Transformations**: Applied square root transformations to features like 'Hillshade_Noon', 'Euclidean_Distance_To_Hydrology', 'Horizontal_Distance_To_Fire_Points', 'Horizontal_Distance_To_Roadways', and 'Mean_Distance_Hydrology_Roadways'.
- **Mean Hillshade**: Calculated the mean of 'Hillshade_9am', 'Hillshade_Noon' and 'Hillshade_3pm'.
- Other linear combinations of distance were tried.

**Interaction Features**:
Additional interaction features were created to capture complex relationships between existing features:

- Elevation and Slope
- Horizontal Distance to Hydrology and Vertical Distance to Hydrology
- Aspect and Slope

These interaction features were derived based on domain knowledge and intuition, aiming to capture non-linear relationships that may enhance the predictive performance of our model.

**Grouped Statistics Features**:
Grouped statistics features were computed to capture aggregated information within categorical groups:

- Mean of Elevation by Soil Type
- Sum of Aspect by Wilderness Area

These features provide aggregated information within specific categorical groups, enhancing the model's ability to capture patterns and relationships.

**Feature Selection**:
Feature selection was performed to identify the most important features for predicting the forest cover type. The top features were selected based on their importance scores after we trained several classifiers. The final list below is based on the Extra Trees classifier. The selected features are:

- Elevation
- log.Elevation
- Mean Hillshade
- Mean Euclidian Distance To Hydrology
- Mean.Elevation.Vertical.Distance.Hydrology
- Mean.Distance.Hydrology.Firepoints
- Mean.Distance.Firepoints.Roadways
- Square Root Transformations of Horizontal Distance.To.Roadways
- Square Root Transformations of Mean.Distance Hydrology.To.Roadways
- Elevation.Soil.Type.mean
- Aspect.Wilderness.Area.sum

## 4 Model Selection and Hyperparameter Tuning

In this section, we present the evaluation of various machine learning models tested for predicting forest cover types. Starting from simple baseline classifiers to more complex ensemble methods, each model was evaluated based on its performance metrics and suitability for the dataset.

### 4.1 Model Selection

- **Baseline Classifier:** [12] We established a baseline performance metric before exploring machine learning algorithms. This baseline, set by a simple dummy classifier, reflects how a simple classifier that was not adapted with domain knowledge or any expertise might approach the problem. The dummy classifier predicts based on basic rules, such as class distribution in the training set, and we evaluate its accuracy on a validation set. In our case, this dummy classifier achieved an accuracy of 0.15 in predicting the cover type. By comparing our model's accuracy against this baseline, which is notably low, we can assess the effectiveness of our approach. If our advanced models can't surpass this baseline accuracy, it suggests that our approach is problematic, calling us to refine our method or the training data.
- **Random Forest Classifier:** [9] Next, a Random Forest Classifier was chosen due to its ability to handle nonlinear relationships and high-dimensional data effectively. We also had a good experience with this classification method in the last iteration of the challenge. This model demonstrated improved performance compared to our baseline classifier (0.73), but there was still room for further improvement.
- **XGBoost Classifier:** [13] We integrated the XGBoost Classifier into our analysis framework due to its renowned efficiency and performance in handling complex datasets. XGBoost, which stands for eXtreme Gradient Boosting, is particularly celebrated for its ability to execute fast and accurate model training, even with large data volumes. It leverages gradient boosting framework at its core, which aids in optimizing predictive accuracy by minimizing errors in successive iterations. In our application, the XGBoost Classifier exhibited promising results (0.77), showcasing significant improvement in predictive performance over the baseline model.
- **Extra Trees Classifier:** [4] Finally, an Extra Trees Classifier was selected for its efficiency and robustness. This classifier offers several advantages that make it well-suited for our task of predicting forest cover types. As an extension of Random Forests, the Extra Trees classifier builds multiple decision trees and aggregates their predictions, resulting in improved accuracy and robustness. One of the key features of Extra Trees is its use of random thresholds for feature splitting. Unlike traditional decision trees which search for the optimal threshold for each feature, the Extra Trees Classifier selects random thresholds. This randomness helps reduce variance and can lead to improved generalization performance, especially in high-dimensional datasets like ours. Additionally, the Extra Trees Classifier benefits from the ensemble learning approach, where multiple models are trained independently and their predictions are combined to make the final prediction. This ensemble technique helps mitigate overfitting and improves our model's ability to capture complex relationships within the data. During our evaluation process, the Extra Trees Classifier demonstrated superior performance compared to other models we tested (0.80). It achieved high accuracy on both training and validation datasets, indicating

its effectiveness in capturing the underlying patterns in the data. Furthermore, the computational efficiency of the Extra Trees Classifier allowed us to train and evaluate multiple models efficiently, enabling us to iterate quickly and experiment with different hyperparameters.

- **Exploring Other Models and Techniques:** In addition to the models mentioned above, other techniques were explored to further improve predictive performance:
  - Neural Network: Different architectures and regularization techniques were tried with a regular neural network. However, due to the limited amount of data available, the results were not promising, and the approach was quickly abandoned.
  - K-Nearest Neighbors (KNN):[6] KNN, a simple and intuitive classification algorithm, was tested. While it showed some potential, its performance was not competitive compared to ensemble methods especially on the time constraint aspect.

  For each of these methods, ensemble methods and blending techniques were experimented with to enhance predictive accuracy.

Table 1: Comparison of Final Models

| **Final Models** | **Accuracy (locally)** | **Accuracy (Kaggle)** |
|---|---|---|
| Baseline | 15% | NA |
| RF | 82% | 73% |
| XGBoost | 83% | 77% |
| ETC | 88% | 80% |

### 4.2 Hyperparameter Tuning with Optuna Optimization

After refining our features and selecting the optimal model, we advanced to fine-tuning the Extra Trees Classifier using Optuna [1], a sophisticated framework for hyperparameter optimization. Initially, we explored hyperparameter tuning through randomized grid search [10]. However, for more refined outcomes, we shifted our strategy to Optuna, leveraging various optimizers. Notably, we favored the Tree-structured Parzen Estimator (TPE) over the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), based on the superior results it delivered.

Indeed, the objective function was defined to maximize the mean accuracy of cross-validation scores. Optuna's CMA-ES (Covariance Matrix Adaptation Evolution Strategy) sampler was tried for optimization but the Tree-structured Parzen estimator (TPE) was preferred. The TPE allows for an efficient exploration of the search space while balancing it with exploitation to lead efficiently to convergence.

The best hyperparameters obtained from the Optuna optimization were used to train the final Extra Trees Classifier model. The model was trained on the entire training dataset, including the selected features. Training accuracy was calculated to assess the model's performance on the training data. The trained model was then used to make predictions on the test set, and the results were submitted for evaluation.

## 5 Model Evaluation

The evaluation of our predictive models incorporated several key performance metrics, with a particular emphasis on accuracy, given its status as the primary criterion for the Kaggle competition. Precision, recall, and the F1-score were also calculated in the process to provide a more nuanced view of the models' performance, taking into account not just the overall rate of correct predictions but also how well the models managed the balance between false positives and false negatives.

To ensure that our model's performance was not only high but also stable across various segments of the data, we applied K-fold cross-validation [3]. This method enhanced the reliability of our evaluation by mitigating the potential biases that could arise from a single, static split of the data into training and test sets. Essentially, the dataset was divided into K equal parts, with the model being trained on K-1 folds and validated on the remaining fold. This process was iterated K times, each time with a different fold held out for validation, ensuring that each data point was used for validation exactly once. The average performance across all K folds then provided us with a robust measure of the model's overall efficacy.

A key tool in our evaluation arsenal was the confusion matrix [2], which offers a visual representation of the model's predictive accuracy across the different categories of forest cover types. The normalized confusion matrix, in particular, provided insights into the percentage of correct predictions for each actual class, with ideally high values along the diagonal indicating accurate classifications. The off-diagonal numbers highlighted instances of misclassification. The matrix from our final model underscored a strong predictive capability for types 3 through 7, showcasing a commendable rate of correctly identified labels for these categories. However, the model displayed some limitations with types 1 and 2, where the accuracy was lower, suggesting an area where future model refinement could be beneficial.

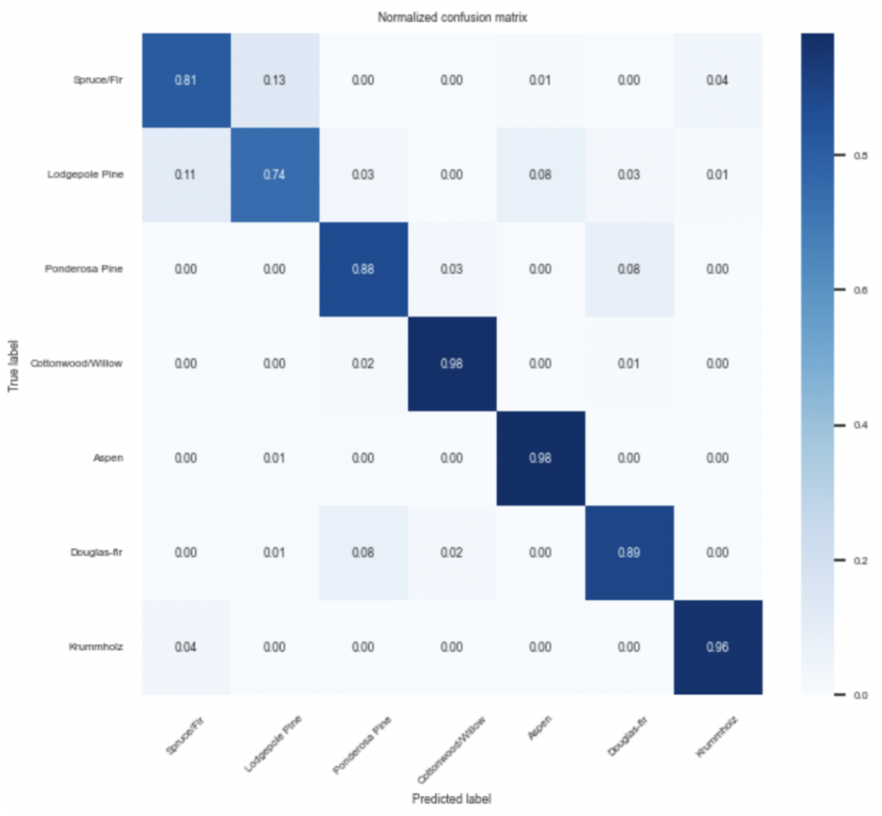

Figure 7: Normalized Confusion Matrix for Forest Cover Type Prediction

## 6 Potential Refinements

Our analysis has identified several avenues for refining our predictive model to enhance its accuracy and reliability further:

- **Improved Discrimination Between Cover Types:** The model currently shows a stronger predictive accuracy for higher numbered cover types. This observation suggests that there may be underlying similarities between cover types 1 and 2 that our model fails to adequately distinguish. To address this issue, a deeper dive into the dataset through advanced feature engineering is recommended. By developing more complex variables that capture the nuanced differences between these two cover types, we can improve the model's ability to differentiate between them. Such improvements aim to bolster the model's accuracy, ensuring its performance remains consistent and reliable.

- **Expansion of the Training Set:** Another significant challenge encountered in this Kaggle competition was the disproportionately small size of the training set compared to the test set. Augmenting the training set with additional observations could markedly enhance the model's predictive capabilities. Expanding the dataset either through data augmentation techniques or by sourcing more observational data will provide a richer foundation for training, allowing the model to learn a broader array of patterns and reducing the risk of overfitting to the training data.

By focusing on these areas for improvement, we aim to advance our model's performance significantly, setting a new benchmark for accuracy in forest cover type prediction.

## 7 Conclusion

In this project, we conducted a thorough analysis of a dataset containing geographical and environmental attributes of forest areas to predict cover types. Through exploratory data analysis, data preprocessing, and feature engineering, we gained insights into the dataset's characteristics and prepared it for modeling.

We evaluated several machine learning models, starting from simple baseline dummy classifiers to more sophisticated ensemble methods. Each model was assessed based on its performance metrics and computational efficiency. Ultimately, the Extra Trees Classifier emerged as the top-performing model, striking a balance between predictive accuracy and computational complexity. With this model, we manage to reach an accuray on the public leaderboard of 0.80512.

However, there are still opportunities for further improvement. Experimentation with additional feature engineering techniques, such as interaction terms or polynomial features, could potentially enhance model performance. Additionally, we realized that it was the different combinations of features that were the main drivers of the variability of performances in our final submissions, holding all else equal in the model.

In conclusion, this project highlights the importance of thorough data analysis, model evaluation, and iterative refinement in building effective predictive models for complex datasets. By leveraging a combination of domain knowledge, feature engineering, and machine learning techniques, we can develop models that accurately predict forest cover types and contribute to the understanding and management of forest ecosystems.

## References

[1] Optuna. *Optuna*. https://optuna.org.

[2] Scikit-learn. *Confusion Matrix Documentation*. https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html.

[3] Scikit-learn. *cross validation Documentation*. https://scikit-learn.org/stable/modules/cross_validation.html.

[4] Scikit-learn. *ExtraTreesClassifier Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html.

[5] Scikit-learn. *Gradient Boosting Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.

[6] Scikit-learn. *KNN Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[7] Scikit-learn. *OneHotEncoder Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.

[8] Scikit-learn. *OrdinalEncoder Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html.

[9] Scikit-learn. *RandomForestRegressor*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

[10] Scikit-learn. *RandomizedSearch Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[11] Scikit-learn. *StandardScaler Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[12] sklearn. *dummy*. https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html.

[13] XGBoost developers. *XGBoost Documentation*. https://xgboost.readthedocs.io/en/stable/.