Final-Report

Outline: -

- **Introduction**
- **Description of Dataset**
- **Description of the machine learning network and training algorithm**
- **Experimental setup**
- **Results**
- **Summary and conclusions**
- **References**
- **Appendix**

1. Introduction

Predicting forest cover types is an important task that can provide valuable information for land managers, conservationists, and scientists who are interested in understanding and managing forest ecosystems. By accurately predicting forest cover types, we can identify areas that are vulnerable to wildfires and other natural disasters, predict the impact of climate change on forest ecosystems, and identify areas that are suitable for timber harvesting or recreational use.

Accurate prediction of forest cover types can provide valuable information for land managers, conservationists, and scientists who are interested in understanding and managing forest ecosystems. For example, accurate predictions of forest cover types can help identify areas vulnerable to wildfires, predict the impact of climate change on forest ecosystems, and identify areas suitable for timber harvesting or recreational use.

Moreover, the dataset is also useful for evaluating the performance of machine learning algorithms and techniques for classification tasks. The dataset is large and contains a diverse range of features, including both quantitative and categorical variables, making it a challenging but realistic problem for machine learning models to solve.

In addition, the forest cover type prediction dataset has been widely used in research and education, making it a popular benchmark dataset for machine learning and remote sensing applications. Its availability and widespread use

have helped to foster innovation and advance the state-of-the-art in machine learning and remote sensing.

2. Description of Dataset:

Our goal is to predict the forest cover type using some variables based on environmental conditions and soil types. We get two datasets available for this problem.

- UCI forest cover prediction dataset: This dataset is imbalanced, and it has around 581012 data points.

In this dataset we have,

- 10 Numerical features
- 44 Categorical features

EDA:

I worked on some of the numerical features and categorical features(soil features)

There is no missing information(Null) in this dataset.

Some of the features which I covered

Hill shade features:

We have 3 features based on the hill shade values for 3 particular times during the day.

This feature is important as it will give us an idea that how much sunlight is available at that spot.

These values range from 0 to 255, so it is giving us the color shade between black and white. (0 is dark and 255 is white)

Based on the violin plot, hill shade 3am looks normal and hillshade_9AM and Hillshade_Noon are left skewed. As the range is fixed, we can say that there is no outlier present.

Horizontal distance to roadways and fire points:

Horizontal distance is the distance of the survey spot from the roadways and fire points.

Main impact of this feature is on the forest quality as if the locations are away from the roads and fire points, those spots are more secure and we will get better quality of forest cover as there will be less pollution and the probability of damage from the fire will be very less.

This feature is explaining the horizontal and vertical distance from the survey location to roadways

We can clearly see that these features are heavily right-skewed and the value range we have is from 0 to 7000.

To normalize that, we can apply log on the feature as it will shift the higher range values towards the left and we will get an approximate normal distribution.

EDA- Correlation plot
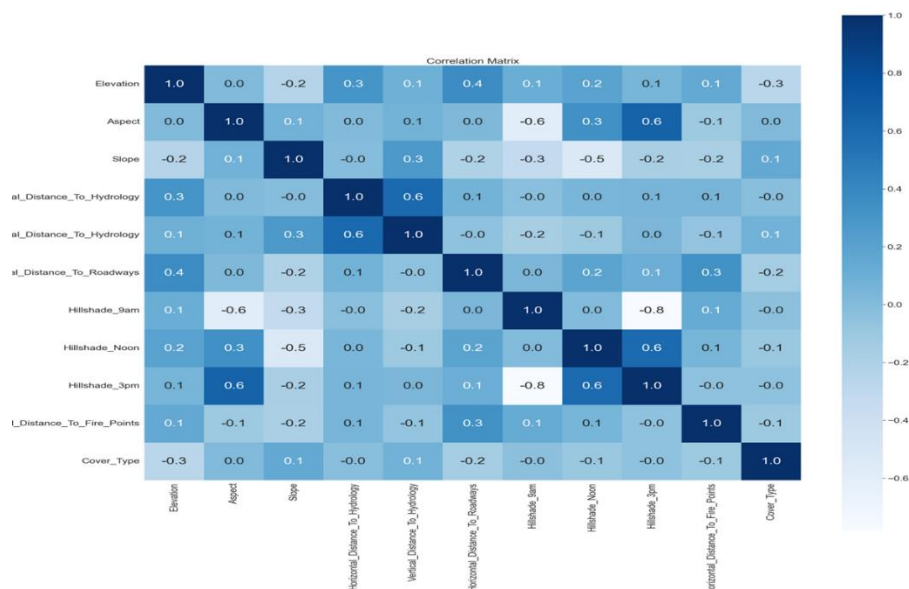
The correlation plot shows us that there are features which are positively and weakly correlated like horizontal distance to hydrology and a few features like elevation that are negatively and weakly correlated with forest cover type (target class).

Feature Engineering on Numerical data:
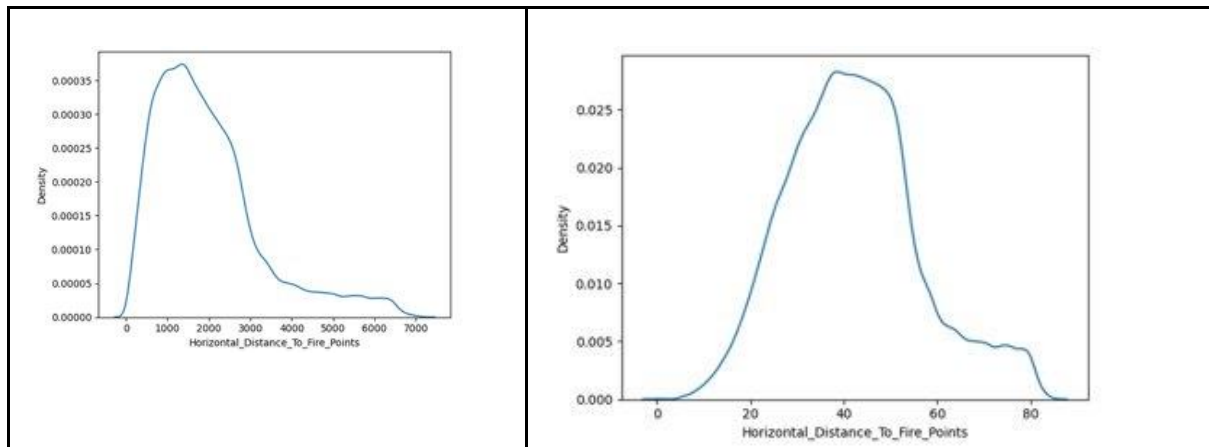
So, we normalize the feature which was right skewed to approximate normal feature using sqrt transform.
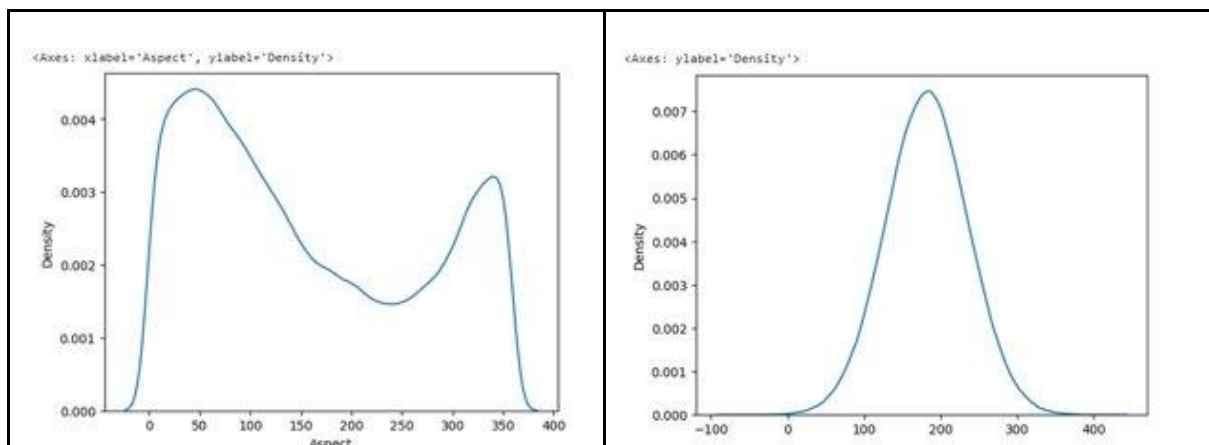
Wilderness area:

We have 4 features based on wilderness area type.

To find the impact of these features on the cover prediction, we mapped the percentage wise class of each wilderness area for particular class type(cover type).

Based on this graph we can say that, Cache la paudre has mostly cottonwood/Willow cover type.

Most of the cover types are observed in wilderness area 3(Comanche peak)

Very small percentage of cover type observed in Neota.

Cover type 1,2,5 and 7 observed in Rawah.

Feature Engineering:

Soil Features:

we have 40 different soil types with the soil texture description.

As we observed that some of the soil description is common, there must be some similarity in these soil types and we can find out the strong relations which are representing the given given types into strong groups.

In case of soil classification, there are 12 main orders of the soil types.

Order – Twelve soil orders are recognized. The differences among orders reflect the dominant soil forming processes and the degree of soil formation. Each order is identified by a word ending in 'sol.' An example is Alfisols.

Suborder - Each order is divided into suborders primarily on the basis of properties that influence soil formation and/or are important to plant growth.

Great Group – Each suborder is divided into great groups on the basis of similarities in horizons present, soil moisture or temperature regimes, or other significant soil properties.

Subgroup – Each great group has a 'typic' (typical) subgroup which is basically defined by the Great Group. Other Subgroups are transitions to other orders,

suborders, or great groups due to properties that distinguish it from the great group.

Family – Families are established within a subgroup on the basis of physical and chemical properties along with other characteristics that affect management.

Series – The series consists of soils within a family that have horizons similar in color, texture, structure, reaction, consistence, mineral and chemical composition, and arrangement in the profile.

The main 12 Soil orders are **Entisols, Inceptisols, Andisols, Mollisols, Alfisols, Spodosols, Ultisols, Oxisols, Gelisols, Histosols, Aridisols, and Vertisols**.

Each order is based on one or two dominants physical, chemical, or biological properties that differentiate it clearly from the other orders. Perhaps the easiest way to understand why certain properties were chosen over others is to consider how the soil (i.e., land) will be used.

As we have 40 different soil types in the dataset, it is really hard to manage this many features in the model and we can reduce the by classifying them into main parent classes.

With the help of various survey sites like soilweb.com, we can easily find out the parent type of the given soil family. We can easily find out the order and sub group with the help of survey information.

Example:

Soil_Type1 is from the Cathedral family.

search the family in

https://soilmap2-1.lawr.ucdavis.edu/sde/?series=mccall#shared-subgroup

Go to the Shared subgroup option for the hierarchical diagram

In this tab we will get the main parent sub group of the soil family.

Go to soil website and search for sub-group. We will get to see the main order of the soil.

Next, we will check if there is any strong relationship between soil order types and forest cover types.

This is how we got the mapping of 40 soil types into 6 major soil types. There is one type for which we do not have any information hence we categorize that as unknown.

We are using the above method to map the order types to the respective soil type.

Important soil types based on predicted forest classes:

As per the soil information, it is observed that we have some soil types which are contributing more in the prediction of a particular soil type.

So, we find out the percentage-wise contribution of each soil type in each class prediction.

There are some soil types, which are helpful in predicting the forest cover type and we will include those features in our model

Based on these observations, we can say that Soil_Type29,Soil_Type10,Soil_Type3,Soil_Type30,Soil_Type38 are important in cover prediction.

Feature reduction:

Used RFE to get the feature importance with random forest.

Random forest works really well on imbalanced data and it uses entropy or gini to evaluate the node and split it based on information gain.

So if the feature is used multiple times to split the nodes, that feature will the important feature.

**We got top 30 features using get_feature_names_out() method.**

PCA:

For feature reduction after using recursive feature reduction, we came across 30 features that we can use from our dataset. To ensure that we are not losing out on important features, we tried PCA and checked the explained variance ratio.
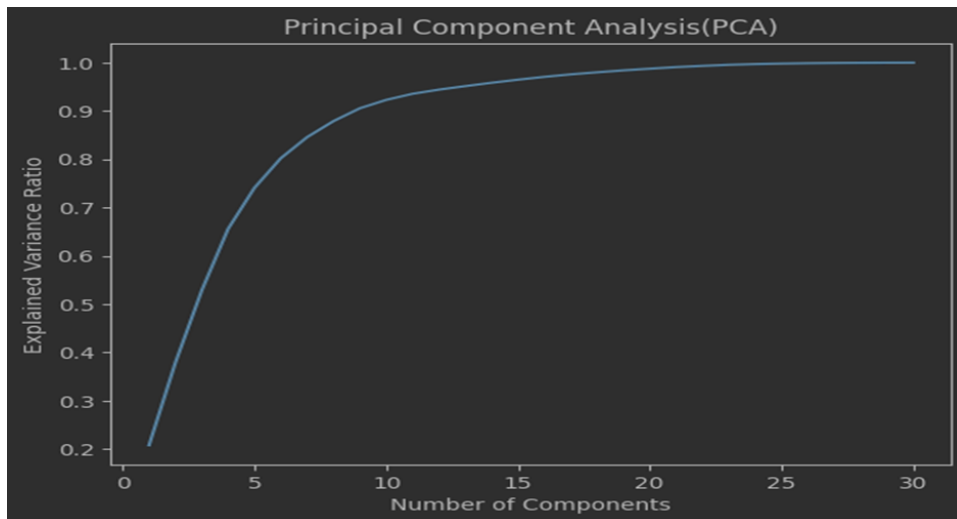
SMOTE: -

Since the dataset was imbalanced with most samples falling under classes 1 and 2, we generated synthetic data using SMOTE and balanced the target feature.
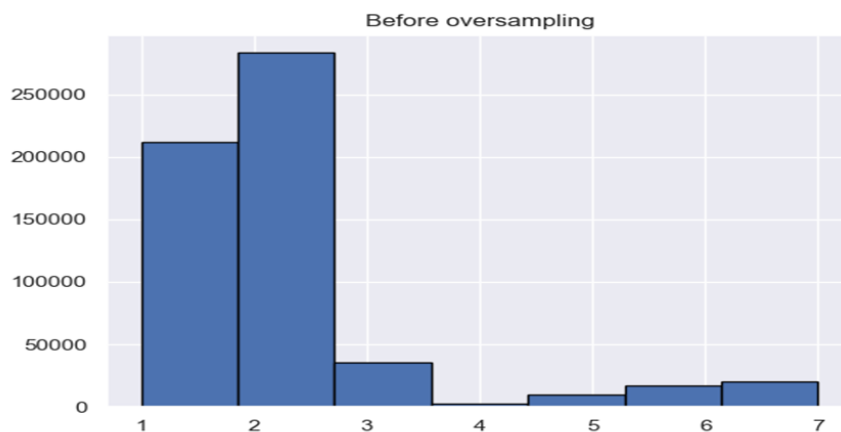
3. Description of the machine learning network and training algorithm

Model Building:

Base model:

Logistic Regression for Multiclass:

Logistic regression is a type of supervised learning algorithm used for binary classification problems. However, it can also be extended to handle multi-class classification problems, where the target variable can have more than two possible values.

One way to handle multi-class classification with logistic regression is to use the "one-vs-all" or "one-vs-rest" approach. In this approach, we train a separate binary logistic regression model for each class. For each model, we treat one class as the positive class and all the other classes as the negative class. We then use the model to predict the probability of the positive class for a given input, and the class with the highest probability is chosen as the predicted class.

Random forest with hyperparameter tuning:

A Random Forest algorithm for multi-class classification works by creating a collection of decision trees, each of which makes a prediction for a given input data point. Each decision tree in the forest is built using a random subset of the training data, and a random subset of the features in the data.

When making a prediction for a new data point, each decision tree in the forest makes a prediction for the class label, and the final prediction is made by aggregating the predictions of all the trees. One common way to do this is to use a majority vote: the class label that is predicted by the most trees is chosen as the final prediction.

To train the Random Forest algorithm, you typically split the dataset into a training set and a validation set. The algorithm is trained on the training set, and the validation set is used to tune the hyperparameters of the algorithm (such as the number of trees in the forest, and the number of features used in each tree).

Random Forest is a powerful and versatile algorithm that can be used for a wide range of classification problems. It is particularly useful when the dataset is large, noisy, or contains many features.

MLP classifier

The MLP (Multi-Layer Perceptron) Classifier is a type of neural network that can be used for classification tasks, including those with multiple classes. The MLP Classifier consists of an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer is determined by the number of features in the input data. The output layer has one neuron for each class, and the output of the MLP Classifier is the class with the highest probability.

The MLP Classifier uses an activation function to introduce non-linearity into the model. The most used activation function is the ReLU (Rectified Linear Unit) function, which returns the input value if it is positive and zero otherwise. Other activation functions, such as the sigmoid or tanh functions, can also be used.

During training, the MLP Classifier adjusts the weights and biases of the neurons in the network to minimize the error between the predicted output and the true output. This is done using a technique called backpropagation, which involves computing the gradient of the error with respect to the weights and biases and using this information to update the weights and biases.

To train an MLP Classifier for multi-class classification, the output layer typically uses the softmax activation function, which ensures that the output values for each class sum to one. The loss function used for training is usually the categorical cross-entropy loss, which measures the difference between the predicted class probabilities and the true class probabilities.

XGBoost classifier

We performed the hyperparameter tuning for the XGBoost classifier like what was done for the Random Forest classifier for the parameters like the number of estimators and depth. Based on many iterations, we found the best values of the number of estimators and depth values to be 100 which gave us the best F1 score of 0.93.
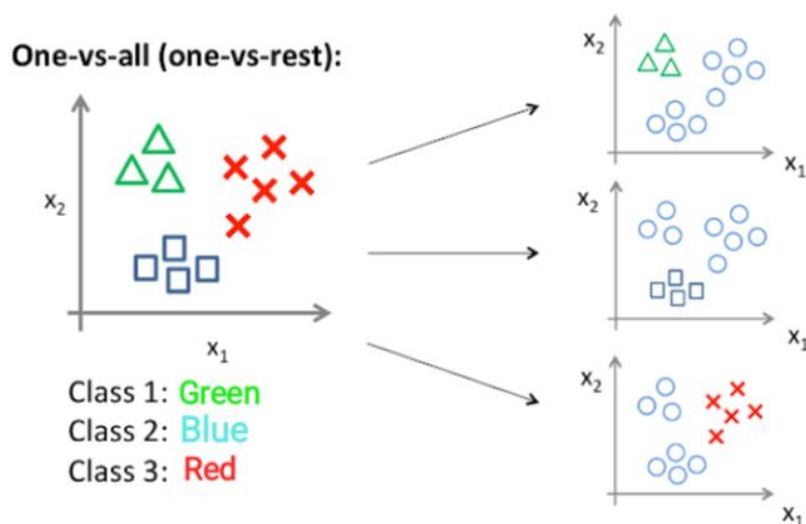
4. Experimental setup

The environment where this project was run contains anaconda3 installed along with python 3.9 version.

5. Results

Logistic Regression for multi-class

- **Used the logistic regression using one vs all approach for multi-class.**
- **It will create sub-models for each class and provides the output based on the higher probability of the class.**


```
pred=best_rf_model.predict(x_test)
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.96	0.94	0.95	63548
1	0.96	0.96	0.96	84956
2	0.93	0.95	0.94	10711
3	0.86	0.89	0.88	813
4	0.81	0.92	0.86	2846
5	0.85	0.92	0.89	5187
6	0.94	0.98	0.96	6243
accuracy			0.95	174304
macro avg	0.90	0.94	0.92	174304
weighted avg	0.95	0.95	0.95	174304

Random forest classification

- **As grid search and randomized cv search were taking too long, we decided to perform the hyperparameter tuning using for loop.**

- **We passed the depth and number of estimators in for loop and check the F1 score for each model using CV and Test data.**

- **We got F1 score around .92 which is a stable model.**

depth vs df1 score

(50, 0.9997)   (100, 0.9996)

(20, 0.9239)

(50, 0.9115)   (100, 0.9111)

(20, 0.8587)

(10, 0.6435)

(5, 0.5033)

(3, 0.4284)

f1 score(train and test)

depth(depth of the tree)

```
pred=best_rf_model.predict(x_test)
print(classification_report(y_test, pred))
```
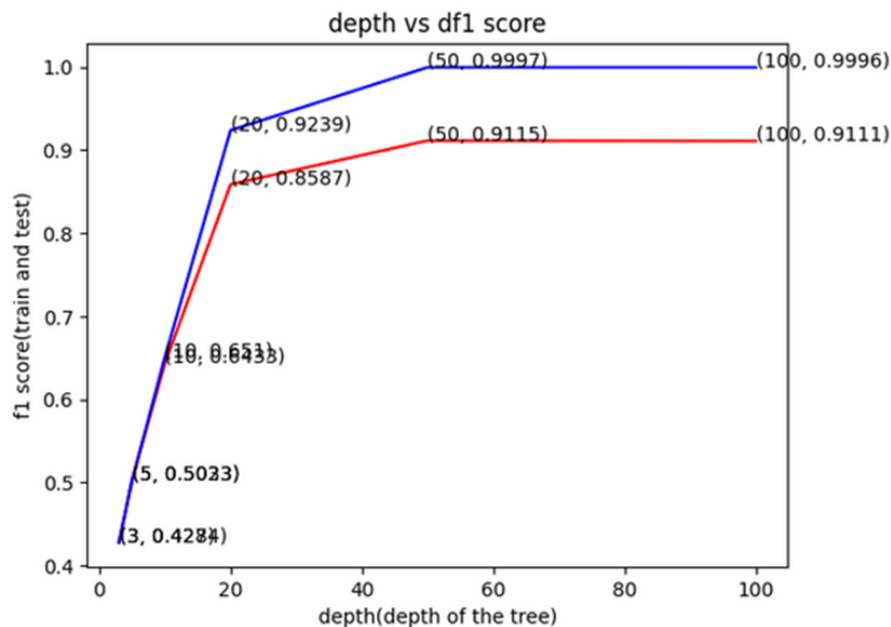
```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     63548
           1       0.96      0.96      0.96     84956
           2       0.93      0.95      0.94     10711
           3       0.86      0.89      0.88       813
           4       0.81      0.92      0.86      2846
           5       0.85      0.92      0.89      5187
           6       0.94      0.98      0.96      6243

    accuracy                           0.95    174304
   macro avg       0.90      0.94      0.92    174304
weighted avg       0.95      0.95      0.95    174304
```
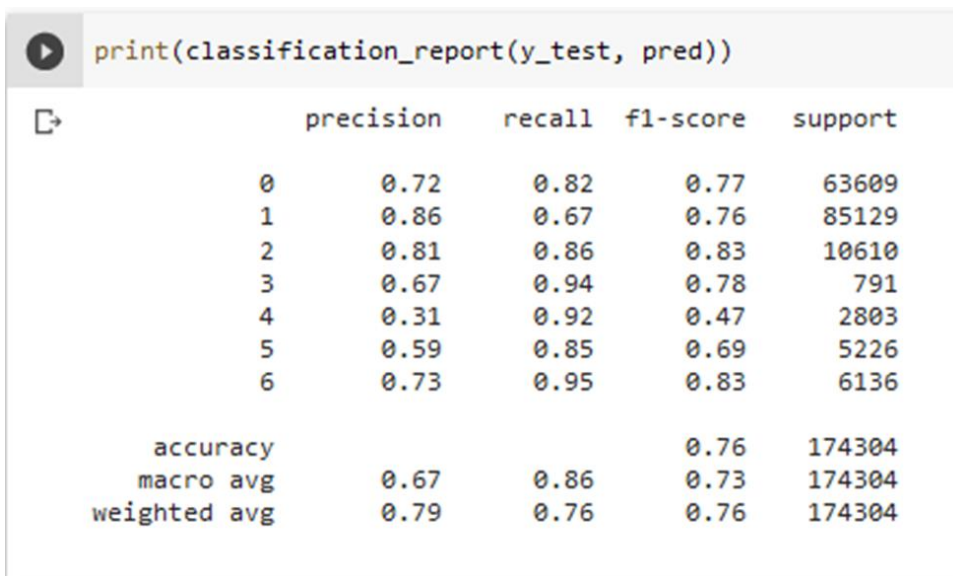
XGBoost classifier

We used a Grid Search method on the parameters mentioned below for the MLP Classifier

After using the Grid search method, which was computationally expensive, we found the best parameters:

The below picture shows the classification report for the MLP classifier: -

```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.82      0.77     63609
           1       0.86      0.67      0.76     85129
           2       0.81      0.86      0.83     10610
           3       0.67      0.94      0.78       791
           4       0.31      0.92      0.47      2803
           5       0.59      0.85      0.69      5226
           6       0.73      0.95      0.83      6136

    accuracy                           0.76    174304
   macro avg       0.67      0.86      0.73    174304
weighted avg       0.79      0.76      0.76    174304
```

6. Summary and conclusions

- Feature engineering and hyperparameter tuning improved results for the Random Forest Classifier and XGBoost Classifier by 1% and 1.2% respectively.
- Soil order type features were also considered important features in feature selection.
- Some other features like weather conditions and geographical information like location could help us in improving our results further.

7. References

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

https://seaborn.pydata.org/examples/many_pairwise_correlations.html

Appendix

https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/wi15/Yerlan_Idelbayev.pdf