A white circle with black text

Description automatically generated

**Prediction of forest types using forest cover type dataset**

**Student Name**: Atharva Deshmukh

**RIT Email ID**: ad2546@rit.edu

**Course**: DSCI 633: Foundations of Data Science

**Instructor**: Dr. Nidhi Rastogi

**Date**: 15 December 2023

# **Data Science Problem**

The objective of this data science project is to develop a predictive model for classifying forest cover types based on various environmental features. The Forest Cover Type dataset, obtained from the sklearn library, serves as the foundation for this analysis. The primary goal is to create a robust model that accurately identifies different types of forest covers.

# **Data and Model Description**

The samples in this dataset correspond to 30×30m patches of forest in the US. The dataset contains 52 features and 581012 Instances that are divided into 7 classes which are:

* Spruce/Fir
* Lodgepole Pine
* Ponderosa Pine
* Cottonwood/Willow
* Aspen
* Douglas-fir
* Krummholz

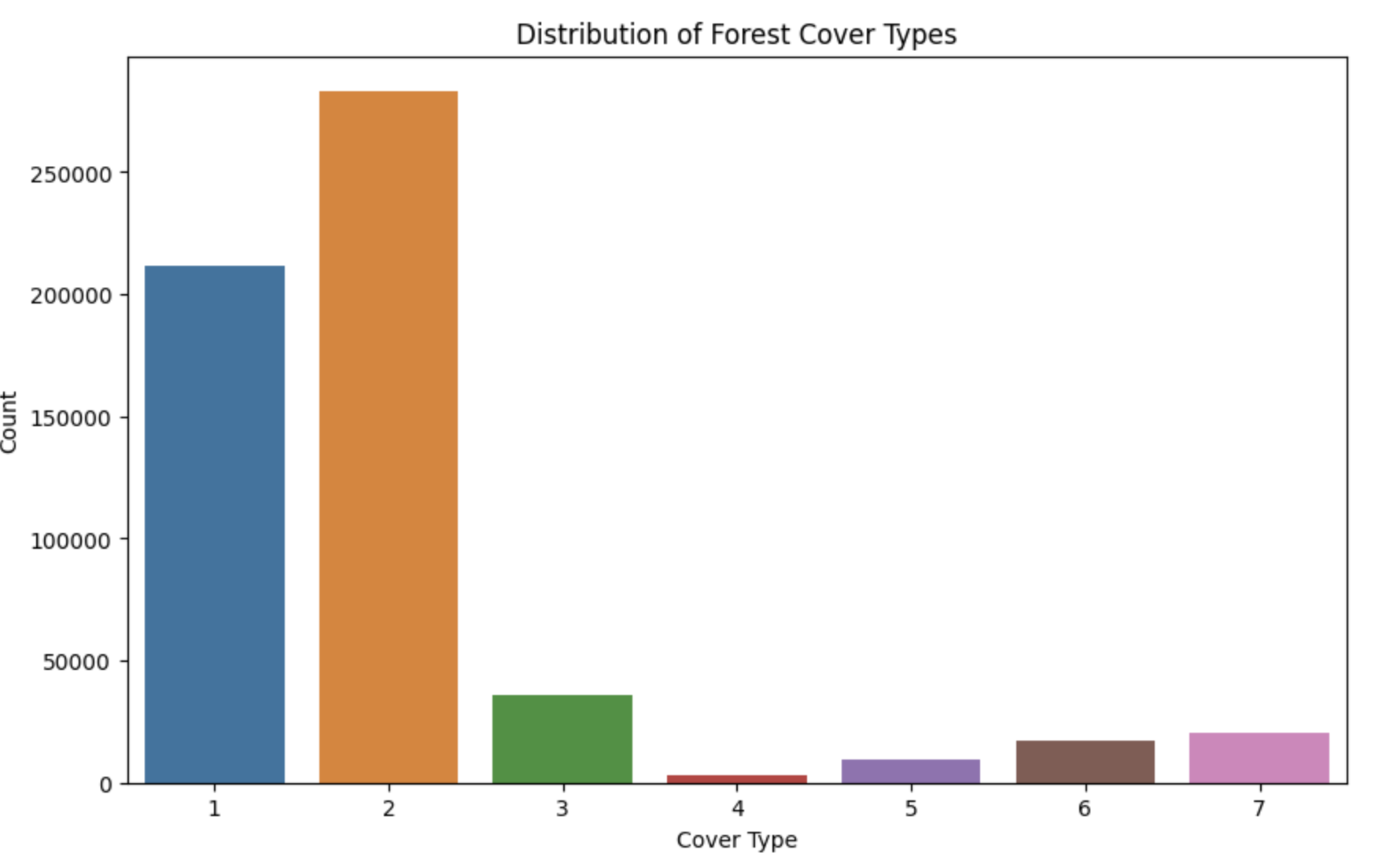


Figure 1 Data Exploration

As Figure 1 shows the data is highly unbalanced among classes. So we need to make the dataset more balanced.

### Correlation Matrix

Through the correlation matrix in Figure 2, we can analyze the following

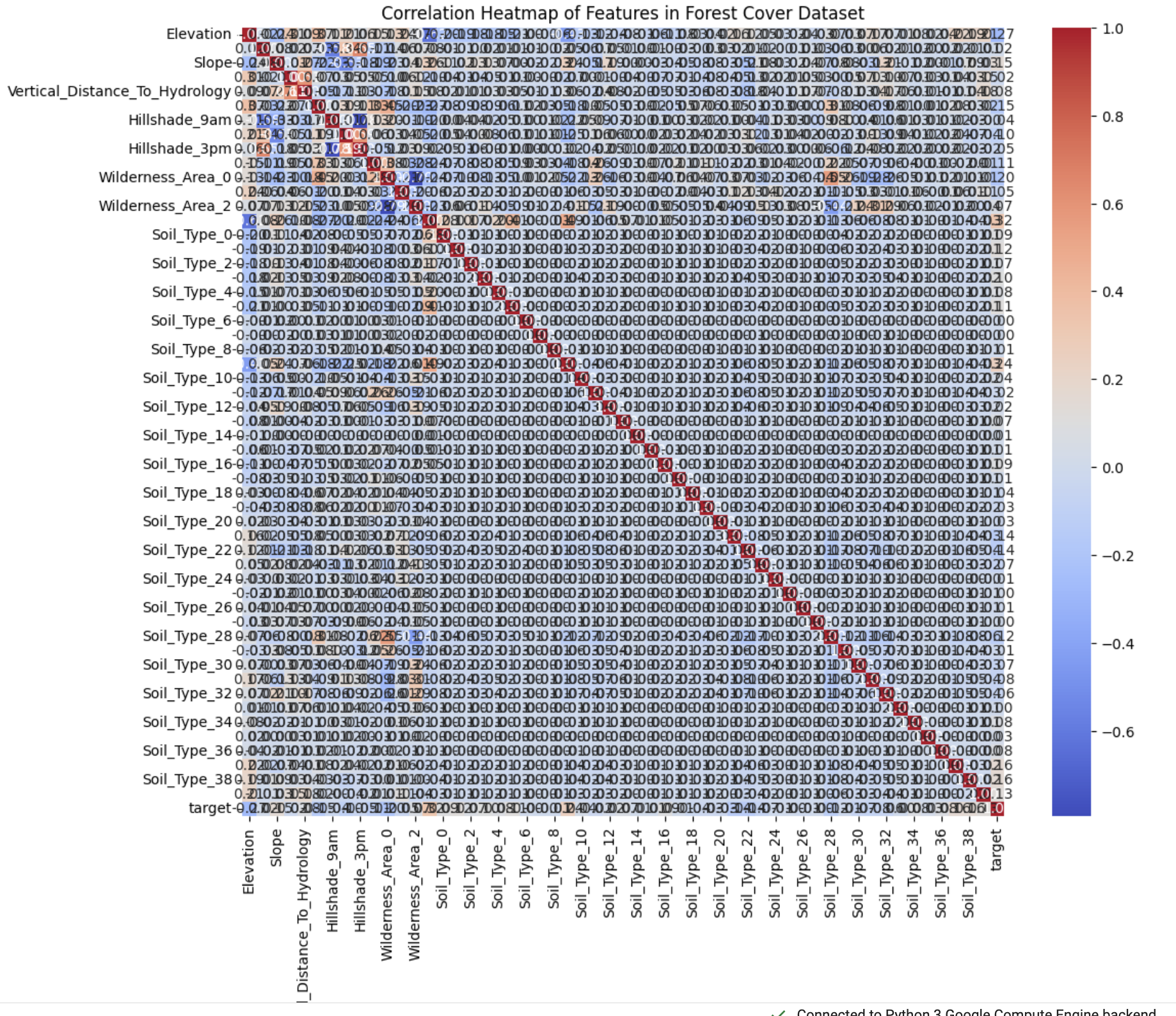


Figure 2 Correlation matrix

1. Elevation is positively correlated with Hillshade\_9am and Hillshade\_3pm. afternoon.
2. The slope is negatively correlated with Elevation.
3. Soil Type\_8 is positively correlated with Soil Type\_10 and Soil Type\_12. This means that these soil types are often found together

### Feature Engineering

To reduce the processing time and improve the efficiency of our models we do some feature engineering

1. We can convert the Elevation Range into low, medium, and high concerning the range of data.
2. We can convert horizontal and vertical Distance To Hydrology to Euclidean Distance To Hydrology.
3. Horizontal Distance To Hydrology, Horizontal Distance To Roadways, and Horizontal Distance To Fire Points can be converted into Total Distance.
4. Skewed features can be shortened to Hillshade 9 am, Hillshade Noon, Hillshade 3 pm.

### Oversampling

As stated before our dataset is very much unbalanced so to deal with it we can use undersampling or oversampling. Both undersampling and oversampling have their pros and cons.

**Undersampling**: This approach will give us more accurate data but will compromise the dataset values.

**Oversampling**: This approach will give us more data to work with but it may not always represent the real-world values.

**SMOTE (Synthetic Minority Oversampling Technique)**

SMOTE is a technique designed to address class imbalance by oversampling the minority class. Rather than simply duplicating existing instances, SMOTE generates synthetic examples to augment the minority class, thereby creating a more balanced dataset.

**Step-1 Identify Minority Instances**

The process begins by identifying instances belonging to the minority class. In our forest cover classification, these would be the less prevalent cover types.

**Step-2 K-Nearest Neighbors:**

For each minority instance, SMOTE identifies its k-nearest neighbors. The parameter 'k' is user-defined and determines the number of neighbors to consider as shown in Figure 3.

**Step-3 Synthetic Instance Generation:**

SMOTE creates synthetic instances along the line segments connecting the minority instance to its k-nearest neighbors.

The dataset after performing SMOTE is completely balanced and has an equal number of classes.

**SMOTE Algorithm Steps**

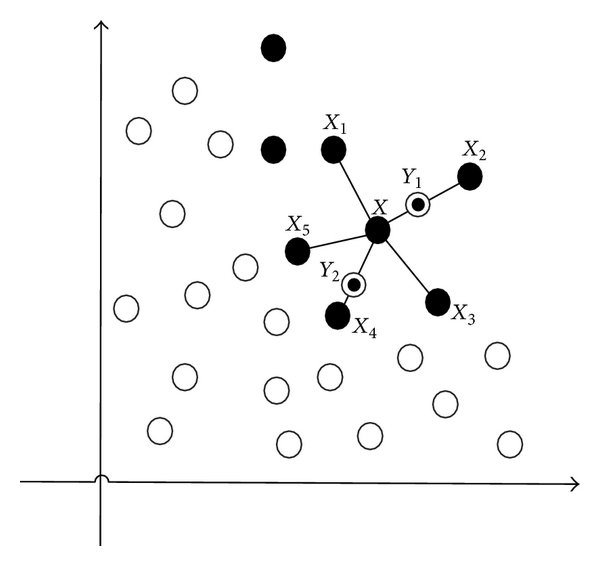


Figure 3: [1]SMOTE working

### Principal component analysis

The core of the analysis lies in the application of PCA. The PCA class from sci-kit-learn is employed, with a specified number of components set to 10 (n\_components=10). This choice aims to strike a balance between reducing dimensionality and retaining sufficient information to capture the variability within the dataset.

Further, we are dividing the dataset into three parts train, validate, and test in the ratio of 80-20-20. This leads us to 15864850 rows in train 1983110 rows in test and validate data respectively. Moving forward all our analysis will be conducted concerning training and validation data unless mentioned otherwise.

# **Analysis Strategy**

We are using three algorithms to predict forest cover types

### 1. Support Vector Machine using Stochastic Gradient Decent

Support Vector Machines (SVMs) are a powerful class of machine learning algorithms known for their effectiveness in handling non-linear relationships, making them particularly well-suited for classification tasks. In the context of forest cover classification, where attributes often exhibit intricate non-linear dependencies (e.g., elevation, soil type, and wilderness area), SVMs can offer a robust solution.

SVMs excel in capturing complex, non-linear relationships within the data. This capability is crucial when dealing with forest cover attributes that may exhibit intricate interactions. Techniques like randomized search can be employed to systematically explore the hyperparameters, finding the combination that optimizes the model's ability to discern between different forest cover types.

However, the reality was that we didn't get the most optimal results using SVM which are stated in Figure 4.

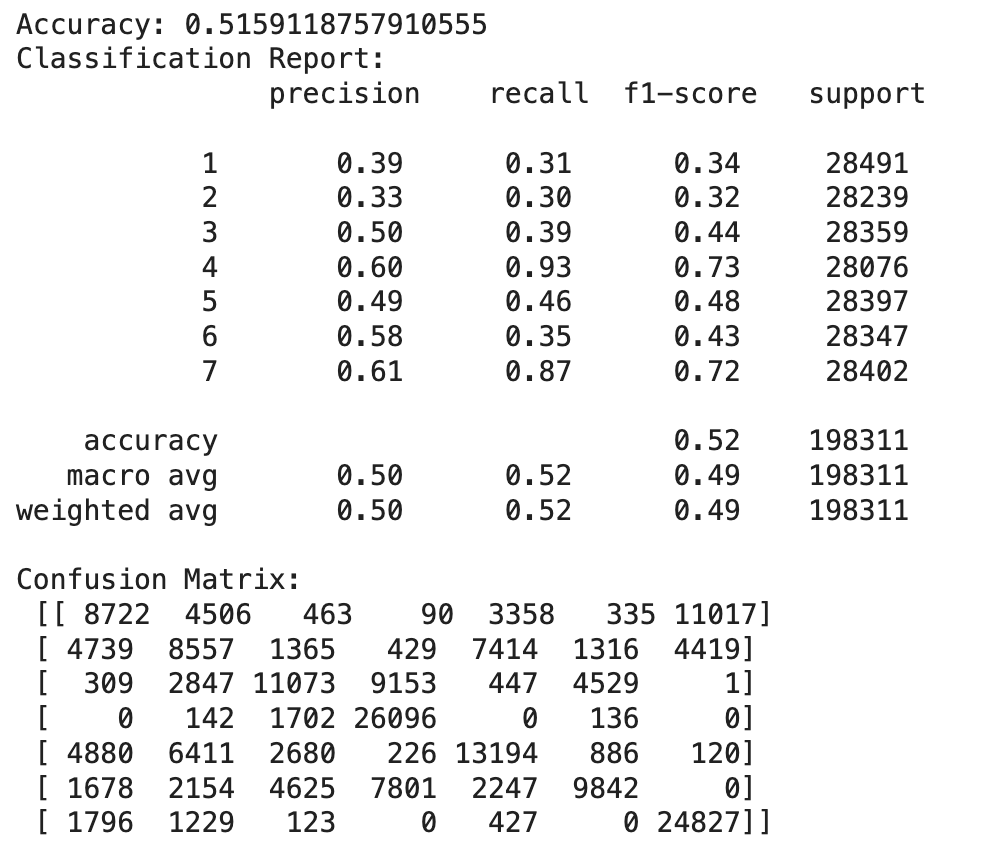


Figure 4: Results from SVM

Hence this is not an optimal model for our classification problem.

### 2. Decision Trees

Decision Trees represent a powerful class of machine learning models characterized by their tree-like structure, making decisions through a series of splits within the dataset. Their unique characteristics position Decision Trees as an excellent choice for classifying forest cover types, especially given the complex and non-linear relationships often present within the data. Decision Trees are well-suited for handling complex and non-linear relationships within the dataset. This is crucial when dealing with the diverse and interdependent attributes characterizing forest cover types.

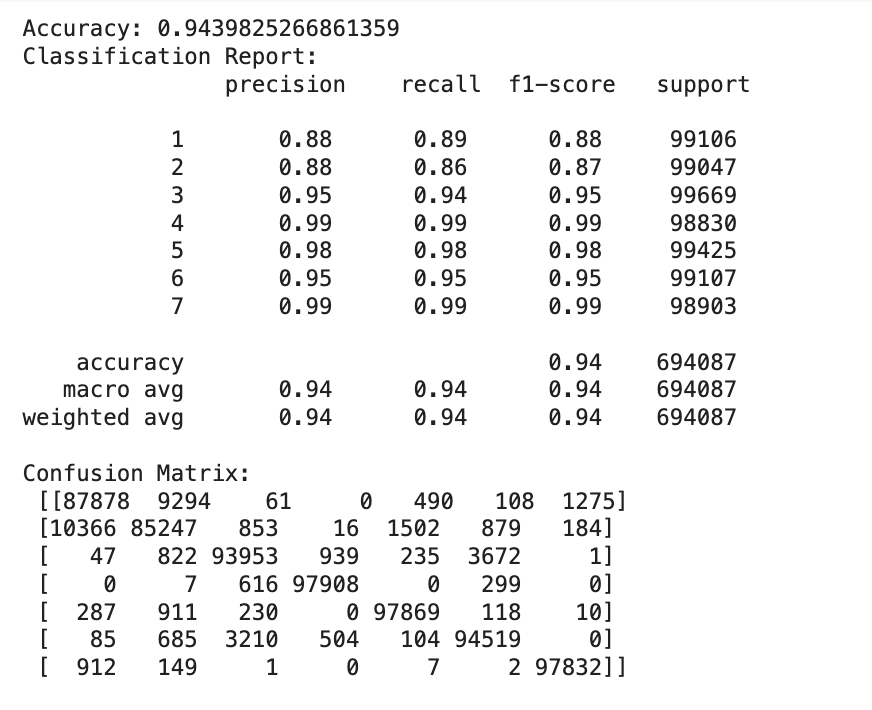


Figure 5: Decision Trees Results

Figure 5 states the accuracy of the model using the best hyperparameters that are:

'min\_samples\_split': 5, 'max\_depth': 30, 'criterion': 'entropy'.

As you can see Decision Trees is a very viable model as it gives a good F1 score for all targets as well as a good accuracy but it can be further improved upon using Random Forest Classification ie an ensemble of Decision Trees.

### 3. Random Forest Classification

Random Forest constructs a multitude of decision trees during training, each utilizing a random subset of the data. This diversity ensures a more comprehensive exploration of the feature space.

Figure 6 will delve into the results we got from the Random Forest Classifier and the hyperparameters used.

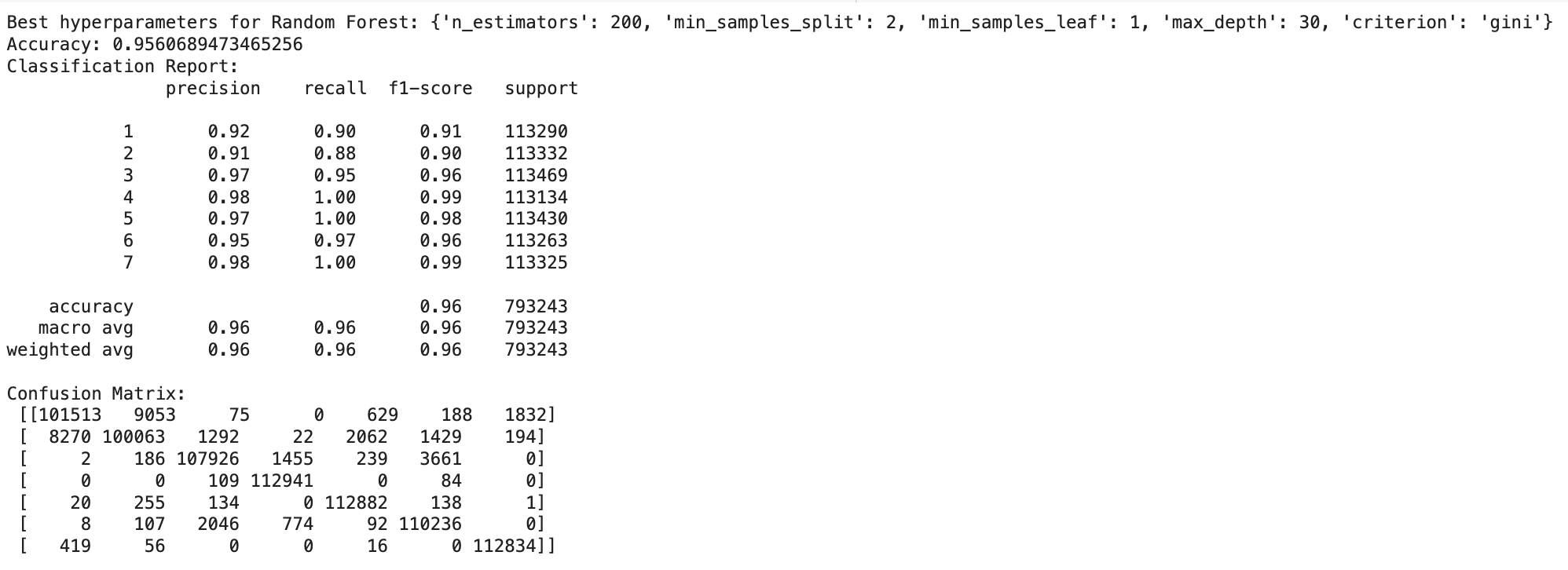


Figure 6: Random Forest Results

As you can see from Figure 6 results we obtained from random forest are more accurate than decision trees.

### Model Choice

Here we have two viable models with not-so-different results. Both Random Forest and Decision Trees are similar in the way they operate but Random Forest Trees take a lot more resources as they are an ensemble of several decision trees. Further, we can use boosting techniques to increase the accuracy of Decision Trees and make it more accurate. Hence for this problem, we will go with the Decision Trees algorithm.

### Further Analysis without using PCA

Without using PCA we can get way better results which are stated in Figure 7

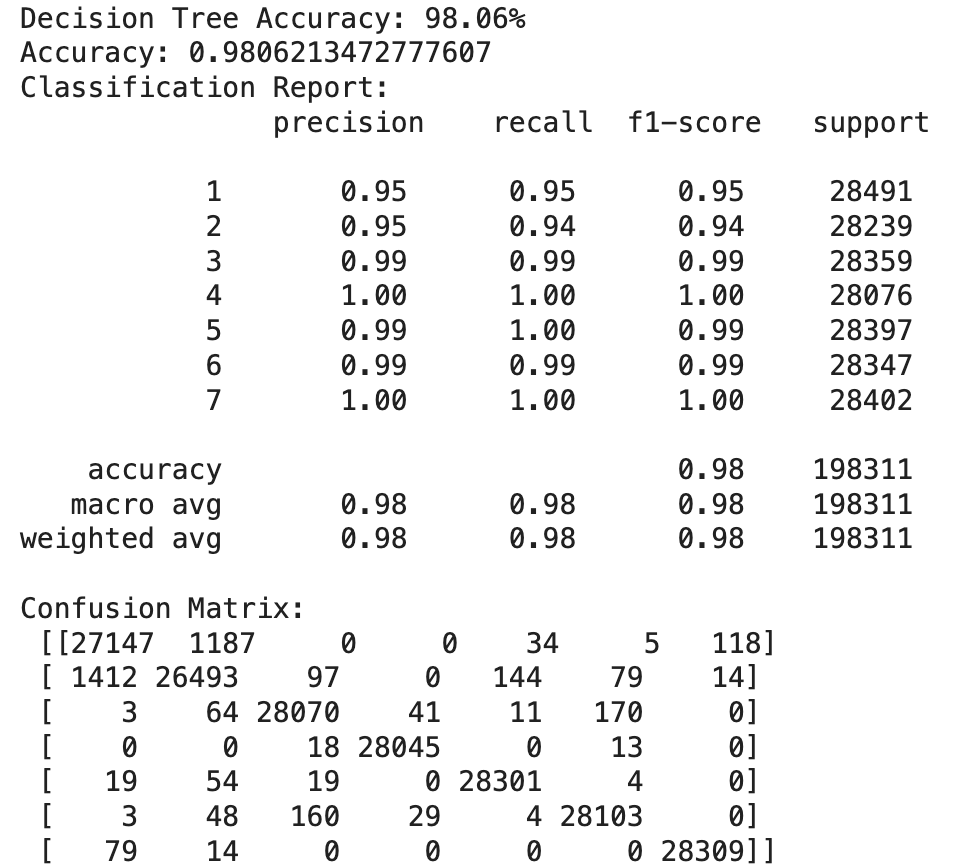


Figure 7: Decision Trees without PCA

Finally, we use ADA boost on Decision Trees without PCA which will be our final model.

### ADA Boost

AdaBoost, or Adaptive Boosting, is an ensemble learning method used for classification tasks, particularly in forest cover classification. It combines predictions from multiple weak learners, often decision trees, to create a robust classifier. AdaBoost falls under the boosting category, sequentially training weak learners and adjusting their importance based on performance. It assigns weights to training examples, focusing more on misclassified instances in each iteration. AdaBoost is known for improving model accuracy, handling complex relationships, and mitigating overfitting. It is adaptive, robust to noise, and effective in capturing intricate patterns within forest cover attributes. Future directions include hyperparameter tuning and exploring different weak learners to enhance AdaBoost's performance.



Figure 8: Decision tree using ADA Boost

Finally, we use our model for testing data and the results are mentioned in Figure 9

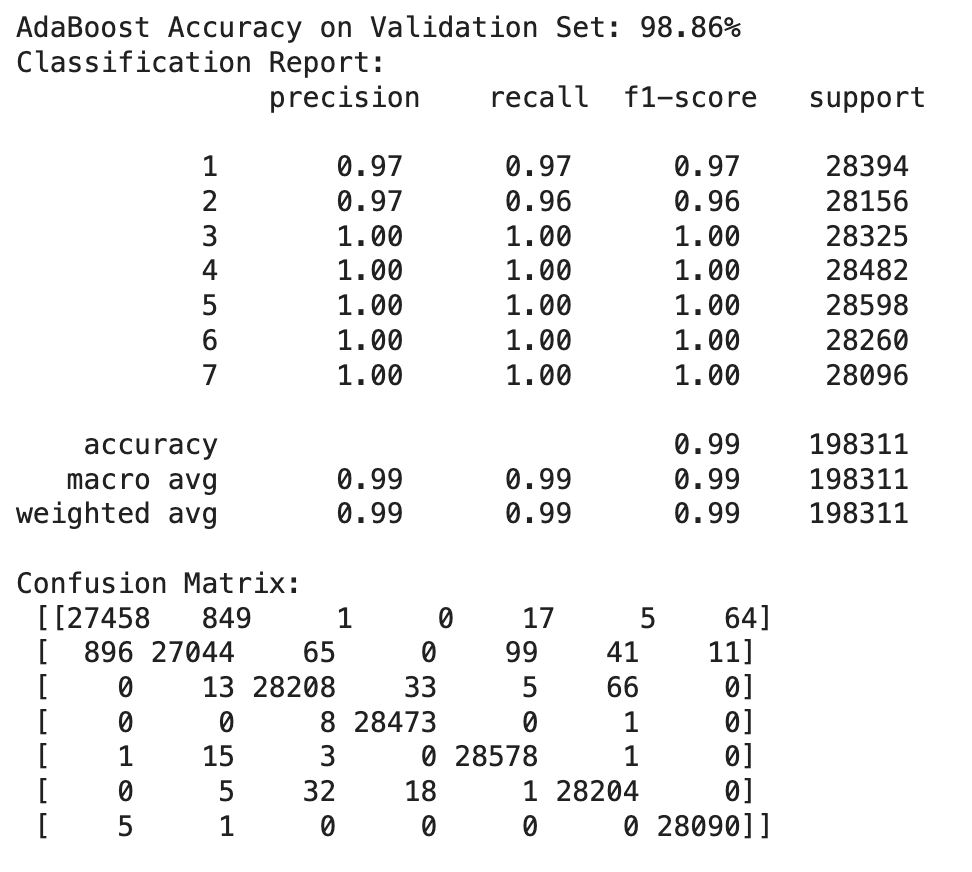


Figure 9 Testing data results

# IV. Conclusion

We can conclude that we are getting the best results using ADA Boost on Decision Trees using 'min\_samples\_split': 5, 'max\_depth': 30, 'criterion': 'entropy' and learning rate as 1.0 hyperparameters. We can conclude this using accuracy and F1 score as factors to measure the feasibility of our model.

# V. Analysis Code



Figure 10: SVM with best hypermeters



Figure 11: Decision Trees with best hyperparameters



Figure 12: Random Forest with Best Hyperparameters



Figure 13: ADA Boost

# VI. References

[1] “Schematic diagram of SMOTE algorithm.,” ResearchGate. Accessed: Dec. 04, 2023. [Online]. Available: https://www.researchgate.net/figure/Schematic-diagram-of-SMOTE-algorithm\_fig1\_282830682

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