Fabricating an Ensemble of Machine Learning Models to Predict Forest Cover Types

Student Name: Swetha Srinivasan

RIT Email ID: ss9577@rit.edu

Course: DSCI 633: Foundations of Data Science
Instructor: Dr. Nidhi Rastogi
Date: 15 December 2023

I. Introduction

In recent years, the analysis of environmental data has become important for sustainable management of land, resources, biodiversity conservation, and ecological understanding of the nature. This project focuses on using advanced machine learning and deep learning techniques to classify forest cover types based on a dataset obtained from Scikit-learn's Forest Cover Types dataset. The dataset shows a multitude of environmental attributes.

The dataset was split into 80 percent training and 20 percent testing sets, and preprocessing techniques, including PCA, SMOTE, and Standard Scaler, were applied to address class imbalance. Classification models like Gradient Descent, Logistic Regression, Decision Tree, Random Forest Classifier, Ada Boost Classifier, and Multi-Layer Perceptron (with L2 Model Regularization) were implemented and thoroughly evaluated.

II. Key Challenges

Imbalanced Classes:
The forest cover types shows significant data imbalance, potentially leading to biased predictions (favoring the major class). A major weightage is constituted in classes 1 and 2, whereas the other classes have significantly less data.

Model Interpretability:
While model accuracy is necessary, the interpretability of the classification model is equally important for practical applications. Ensuring that the model's decision-making process aligns with domain knowledge is a key consideration.

III. Problem Statement

The goal of this project is to construct a robust and accurate classification model capable of classifying the diverse forest cover types present in the given area. The dataset comprises the instances where each data point represents a 30×30m patch of forest within a region, with its corresponding forest cover type (categorical represented in the form of integers).

IV. Data and Model Description

Data Exploration:

Initiated with a comprehensive exploration of the Forest Cover Types dataset to understand the structure of the dataset, the distribution of each features, their potential significance to the model, and the distribution of the target variable (forest cover types). Statistics and Visualization helped understanding the dataset.

- Number of Samples : 581012
- Class Labels - Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, Krummhol

Data Splitting:

The dataset was divided into an 80 percent training set and a 20 percent testing set to facilitate model training and evaluation. Keeping in mind the behemoth size of the dataset, it is pertinent to assign a meagre percentage to test set. Since the training set in its original form without any pre-processing will require lots of training samples.

Data Visualization:

In the data visualization phase of the project, several insightful plots were generated to better understand the forest cover types dataset.

- A bar plot was employed to showcase the distribution of different cover types, providing a clear overview of the dataset's class distribution.
- Subsequently, a scatter plot was utilized to depict the relationship between Elevation and Slope, with each point color-coded by its respective cover type. This scatter plot not only visualizes the interplay between two numeric features but also highlights potential patterns or clusters within the data.
- Subsequently, a scatter plot was utilized to depict the relationship after pruning the feature using PCA. In addition to the feature representation, the scatter plot analyzes the correlation when 54 features were present and after it was trimmed down to 10 features.
- Additionally, a heatmap of the correlation matrix was created to explore the relationships among numerical features, aiding in the identification of potential multicollinearity or dependencies.
    - Dark red cells indicate a strong positive correlation between the feature and type. This means that as the number of features increases, the type (class) also increases.
    - Dark blue cells indicate a strong negative correlation between the feature and type. This means that as the number of features increases, the type(class) decreases.
    - Light red/blue cells indicate a weaker positive/negative correlation, respectively.
    - White cells indicate no correlation between the feature and type.

These visualizations serve as valuable tools for both exploratory data analysis and communicating key insights in the subsequent stages of the project.

Preprocessing:

i) ***Imbalanced Data Handling:*** Applied Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance in the training set. After applying SMOTE it gave a highly balanced data, which might give raise to a model that assigns similar weights to each class. This can lead to almost equal probabilities while arriving at a classification category, thereby making it futile. Hence I used the highest and lowest class of the dataset (class 1 & 2) as a reference and made sure all the other classes

have values distributed randomly between these two classes. This involved calculating the class size proportioned between these two classes, and then initiating a random variable (**γ**) such that the re-sampled class will have a distribution in between these two major classes. Implementing this scheme, did not give raise to a perfectly balanced class.

Reason for Using Smote: Unlike simple duplication of minority class instances, SMOTE generates synthetic samples by interpolating between existing instances, preserving information and introducing diversity. This method mitigates the risk of overfitting, enhances decision boundaries, and provides algorithms with a more nuanced understanding of the minority class. While other approaches, such as undersampling or resampling with replacement, have their merits, SMOTE stands out for its ability to adapt to diverse dataset characteristics and improve model generalization on imbalanced datasets, making it a suitable choice for this classification task.

ii) ***Standardization:*** Standardized numerical features using Standard Scaler to ensure uniform scaling. Standard Scaler was applied in the preprocessing stage to address potential disparities in the magnitudes of features within the forest cover types dataset. This technique standardizes the features by removing the mean and scaling to unit variance, ensuring that all features contribute uniformly during model training. The decision to use Standard Scaler is driven by considerations such as the sensitivity of certain algorithms to feature scales. When it comes to the individual features, I did not experiment with the features independently according to their need to apply Standard Scaler. Although this could have 50-50 chances of optimizing the model, I focused on applying pre-processing to the overall set of features rather than applying independent ones.

iii) ***Dimensionality Reduction:***

Reason for inducing dimensionality reduction and choosing PCA: This is crucial for computational efficiency and mitigating the curse of dimensionality. With the help of a mechanism capable of transforming the original features into a new set of uncorrelated variables, or principal components, PCA allows for the reduction of the dataset's dimensionality while retaining most of its variability. Moreover, PCA aids in identifying and prioritizing the most informative features, reducing redundancy and multicollinearity. Additionally, the use of PCA is particularly beneficial for enhancing the performance of machine learning models, as it streamlines computations, accelerates training times, and often improves the interpretability of the model results by focusing on the most significant components.

Summary of pre-processing on forest cover types

Applied Principal Component Analysis (PCA) for dimensionality reduction. PCA and Standard Scaler were applied both to the training and testing dataset because in the former 54 features were reduced to 10, and hence the test set needs to have reduced features too. A similar reasoning goes along for Standard Scaler too since the values will be normalized. However, for

the SMOTE, the test set was not altered in any way since oversampling was only performed to reduce the imbalance in order for the model to learn the correlations, not to alter the real-world data (here the test set) in any way.

Model Implementation:

The implemented models have been briefed in this section along with a tabular representation of the results towards the end.

- *Logistic Regression :* Logistic Regression is less prone to overfitting, making it a robust choice when the dataset is limited or when dealing with imbalanced classes. Logistic Regression can serve as a baseline model against more complex algorithms. Starting with a simple model, it helps establish a performance benchmark and assess whether more sophisticated models bring substantial improvements. This is particularly useful when decisions based on prediction certainty are crucial. With our dataset, the model produced its best accuracy of 72.28% before any pre-processing techniques.

- *Gradient Descent:* Gradient Descent is applied to the dataset. The baseline accuracy before any preprocessing stands at 21.52%. After addressing class imbalance with SMOTE, the accuracy sees a modest improvement to 22.84%. Subsequent application of Principal Component Analysis (PCA) leads to a further increase in accuracy to 23.80%. These results suggest that both SMOTE and PCA contribute positively to the model's performance. However, these accuracies are very less compared to other models.

- *Decision Tree:* The Decision Tree model applied for forest cover type classification demonstrates excellent performance, yielding an accuracy of 94%. The classification report further highlights the model's proficiency across different cover types. Notably, the model excels in distinguishing between cover types 1, 2, and 7, achieving precision and recall scores of 94% or higher. The macro and weighted average metrics, both at 90%, reinforce the robustness of the model across all classes. This comprehensive evaluation suggests that the Decision Tree model effectively captures the diverse patterns.

- *Random Forest Classifier:* The implementation of Random Forest Classification for forest cover type classification demonstrates outstanding performance, achieving an impressive accuracy of 95.51%. Notably, the model excels in correctly identifying instances of cover types 1, 2, and 3, as indicated by the high values along the diagonal of the matrix.

- *Ada Boost Classifier:* The AdaBoost Classifier implementation for forest cover type classification achieved an accuracy of 47.50%. Despite the relatively lower accuracy compared to other models, the classifier exhibits diverse performance metrics across different evaluation aspects. Notably, the model excels in correctly identifying instances of cover type 1 but faces challenges in achieving higher accuracy for other classes. While the accuracy is lower than some other models, these results emphasize the trade-offs

inherent in the AdaBoost Classifier, highlighting its strengths and areas for refinement in the context of forest cover type classification.

- *Multi Layer Perceptron with L2 regularization:* The Multi-Layer Perceptron (MLP) with Model Regularization (L2) was applied for forest cover type classification, demonstrating an accuracy of 72.79% before preprocessing. The regularization technique aids in controlling overfitting, contributing to the model's generalization ability. The results suggest that the MLP with L2 regularization shows promise for forest cover type classification, with potential for further optimization and refinement through preprocessing techniques and hyperparameter tuning.

MODEL	INSTANCE OF DATASET	ACCURACY (IN PER CENT)	PRECISION	RECALL	F1 SCORE
GRADIENT DESCENT	RAW	17.98	-	-	-
	SMOTE	22.24	-	-	-
	PCA	18.09	-	-	-
LOGISTIC REGRESSION	RAW	72.28	0.71	0.72	0.71
	SMOTE	59.76	0.70	0.60	0.63
DECISION TREE (WEIGHTED AVERAGE)	RAW	94	0.94	0.94	0.94
	SMOTE	0.94	0.94	0.94	0.94
	PCA	0.42	0.43	0.42	0.42
RANDOM FOREST	RAW	95.51	0.96	0.96	0.95
	SMOTE	95.79	0.96	0.96	0.96
	PCA	43.79	0.42	0.44	0.42
ADA BOOST (WEIGHTED AVERAGE)	RAW	47.50	0.58	0.47	0.45
	SMOTE	41.40	0.62	0.41	0.47
	PCA	37.48	0.43	0.37	0.38
MLP (WITH L2)	RAW	72.78	0.69	0.72	0.71
	SMOTE	57.37	0.68	0.57	0.60
	PCA	47.93	0.45	0.48	0.46

Table1. Accuracy, Precision, Recall, F1 Score depiction of the models in three instances: RAW (Before any pre-processing), SMOTE (After applying SMOTE oversampling), and PCA (After Applying Dimensionality Reduction using PCA).

Confusion matrices:

1) Logistic Regression

Before pre-processing:

29644	12009	9	0	1	19	875
10178	45304	691	4	35	258	30
0	701	5830	169	5	416	0
10	1904	66	0	6	9	0
0	824	1894	33	8	730	0
1675	39	0	0	0	0	2301

After SMOTE:

25053	9966	14	0	1624	222	5678
9633	32870	691	18	10309	2355	624
0	55	3480	966	379	2241	0
0	0	32	458	0	36	0
1	338	28	0	1525	103	0
366	3	0	0	18	0	3628

2) Random Forest Classifier

Before pre-processing:

40160	2299	0	0	9	2	87
1237	55015	96	0	76	62	14
2	98	6867	23	6	125	0
28	401	17	0	1537	12	0
1	107	228	19	5	3129	0
153	25	0	0	0	0	3837

Accuracy after SMOTE:

40133	2219	5	0	35	6	159
1172	54809	166	0	192	135	26
2	44	6901	32	8	134	0
0	0	49	464	0	13	0
11	165	11	0	1800	8	0
1	30	151	21	5	3281	0

82	12	0	0	1	0	3920

Accuracy after PCA :

28829	13664	5	0	0	0	59
28918	27006	248	0	0	262	66
136	3829	2574	61	0	519	2
0	167	268	25	0	66	0
525	1469	0	0	0	1	0
156	1879	933	1	0	520	0
3137	857	0	0	0	0	21

4) Ada Boost Classifier

Before pre-processing:

33756	3947	145	0	0	193	4516
30506	16725	1028	0	2	5590	2649
74	367	634	0	0	6045	1
0	0	0	0	0	526	0
50	208	126	0	0	3105	0
3043	0	1	0	0	0	971

Accuracy after SMOTE:

8141	9570	286	0	6048	319	18193
3897	23485	1943	0	14812	5829	6534
0	161	1599	0	208	5153	0
0	0	0	0	0	526	0
12	520	63	0	1132	268	0
0	101	681	0	125	2582	0
812	1	0	0	1	0	3201

Accuracy after PCA :

34121	5875	147	0	444	2	1968
35094	16356	1793	22	768	283	2184
108	1730	3737	589	1	956	0
0	40	219	111	0	156	0
564	1210	213	0	1	1	6
127	1509	1075	309	0	469	0
3101	150	4	0	7	0	753

5) Multi-Layer Perceptron with Model Regularization

Before pre-processing:

30641	11134	64	0	0	0	718
9999	44926	1544	0	0	0	31
0	473	6648	0	0	0	0
0	0	526	0	0	0	0
0	1809	186	0	0	0	0
3	898	2588	0	0	0	0
1615	24	12	0	0	0	2364

Accuracy after SMOTE:

24809	10661	21	0	734	97	6235
12483	31787	862	24	8748	1911	685
0	116	3439	1003	368	2195	0
0	0	73	448	0	5	0
1	556	78	0	1295	65	0
0	246	1159	231	117	1136	0
427	19	0	0	1	0	3568

Accuracy after PCA :

22506	17816	1	0	0	0	2234
20541	33211	809	30	0	0	1909
16	2708	4397	0	0	0	0
0	7	519	0	0	0	0
178	1803	0	0	0	0	14
21	1266	2201	0	0	0	1
2658	548	0	0	0	0	809

Performance Evaluation:

Metrics such as accuracy, precision, recall, and F1-score are computed for each model to assess its performance across different aspects of classification. As depicted in Table 1, this is exactly what we desired. Confusion matrices provide a detailed breakdown of predictions, revealing areas of strength and potential improvement. Models are compared based on their accuracy and other relevant metrics. Strengths and weaknesses of each model are identified to inform model selection and further optimization.

VI. Analysis Code

1)Dataset Description:

```
# Display information about the dataset
print("Number of samples:", data.shape[0])
print()
print("Number of features:", data.shape[1] - 1)  # Subtract 1 for the target variable
print()
print("Unique classes", unique_classes)
print()
print("Class distribution:\n\n", data['target'].value_counts())
```

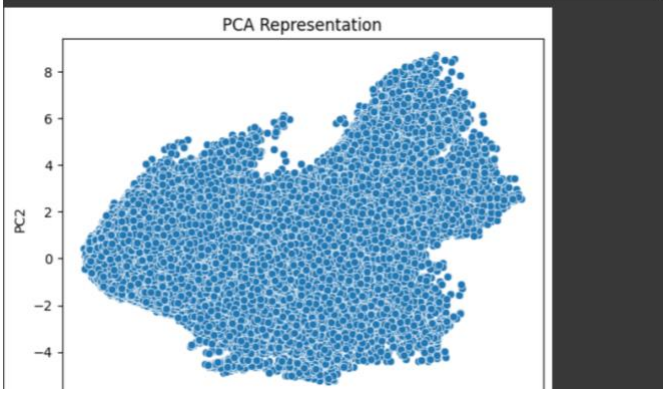
```
Number of samples: 581012

Number of features: 54

Unique classes [5 2 1 7 3 6 4]

Class distribution:

 2    283301
1    211840
3     35754
7     20510
6     17367
5      9493
4      2747
Name: target, dtype: int64
```

2)Train Test Split:

```
# Split the data into features (X) and target variable (y)
X = data.drop('target', axis=1)
y = data['target']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

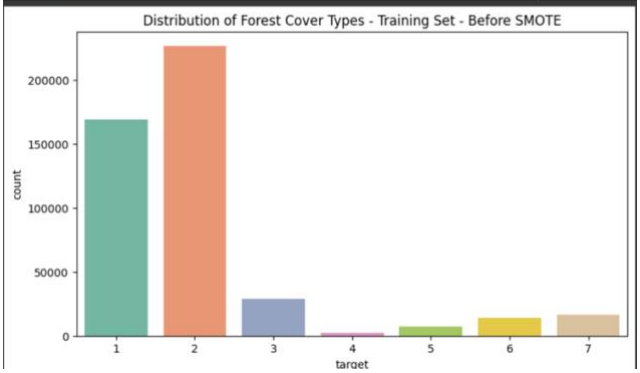
3) Scatter Plot:

```
# Create a DataFrame for easy plotting
df_pca = pd.DataFrame(data=X_train_pca, columns=[f'PC{i+1}' for i in range(10)])

# Assuming 'y' is the target variable
# Combine the PCA components and target labels
df_final = pd.concat([df_pca, y_train], axis=1)

# Plot PCA representation
sns.scatterplot(x='PC1', y='PC2', data=df_final)
plt.title('PCA Representation')
plt.show()
```



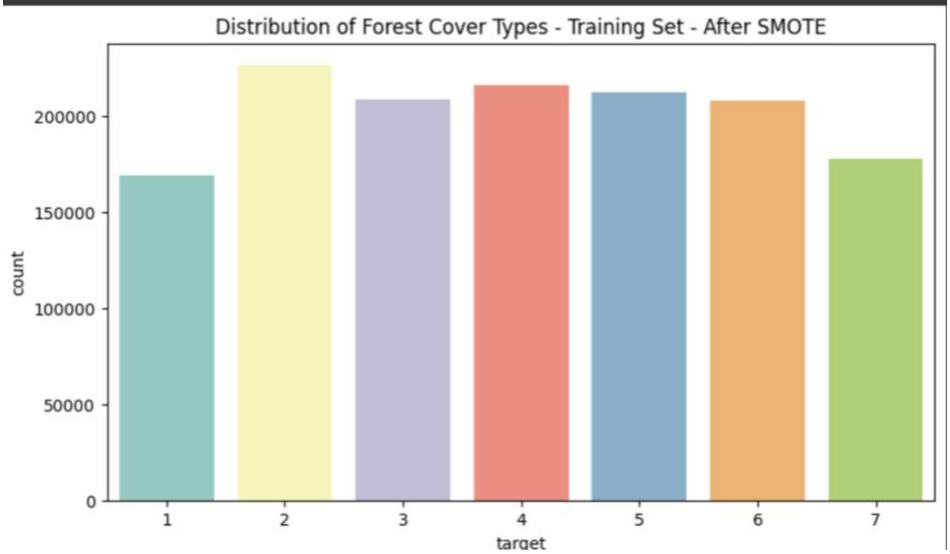
4) Before SMOTE:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Data distribution
# Visualizing the distribution of the target variable

plt.figure(figsize=(9, 5))
sns.countplot(x='target', data=pd.DataFrame({'target': y_train}),palette = "Set2")
plt.title('Distribution of Forest Cover Types - Training Set - Before SMOTE')
plt.show()
```



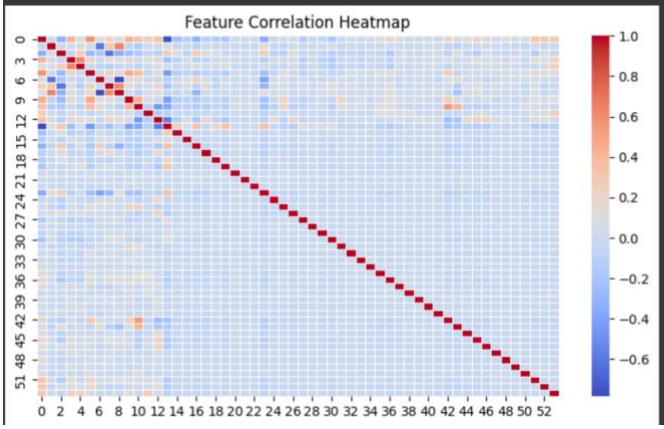
5) After SMOTE:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Data distribution
# Visualizing the distribution of the target variable
plt.figure(figsize=(9, 5))
sns.countplot(x='target', data=pd.DataFrame({'target': y_train_resampled}),palette = "Set3")
plt.title('Distribution of Forest Cover Types - Training Set - After SMOTE')
plt.show()
```



6) Heat Map:

```
# Feature correlations
# Heatmap visualizes the correlation matrix - where each cell represents the correlation coefficient between two features

correlation_matrix = pd.DataFrame(X_train_data).corr()
plt.figure(figsize=(9, 5))
sns.heatmap(correlation_matrix, cmap="coolwarm", annot=False, fmt=".2f", linewidths=0.5)
plt.title('Feature Correlation Heatmap')
plt.show()
```



VI. Conclusion and Output

- A Dataset of a large size such as this requires pre-processing such as data balancing and Normalization patently
- Oversampling the dataset with a random gamma factor proves essential to induce generalisability
- Although we have more accuracy for Random Forest, I am comparing the F1 score and not accuracy.
- Decision Tree Classifier after over-sampling proves to be the best amongst machine learning and deep learning methods
- Principal Component Analysis is an efficient technique only if accuracy is not principal.