Forest Cover Type Classification Project Report

Author: Rohit Bansal, Prasanna Kumar M
Roll Number: MT2025105, MT2025090
Institution: International Institute of Information Technology, Bangalore

Submission Date: 12-12-2025

1 Introduction and Dataset Details

The objective of this project is to predict the forest cover type (the predominant kind of tree cover) for 30×30 meter cells using strictly cartographic and environmental variables. This is a multi-class classification problem with seven target classes.

1.1 Dataset Structure

- **Size**: 581,012 rows and 55 columns.
- **Feature Dimensionality**: 54 features (10 raw, 44 binary).
- **Data Quality**: No missing values were found.

2 Exploratory Data Analysis (EDA)

2.1 Target Variable (Cover_Type) Imbalance

The target variable exhibited severe skewness, with Classes 1 and 2 accounting for over 85% of the data, and Class 4 being a critical minority ($\sim 0.5\%$). This mandated the use of **class weighting** across all models.

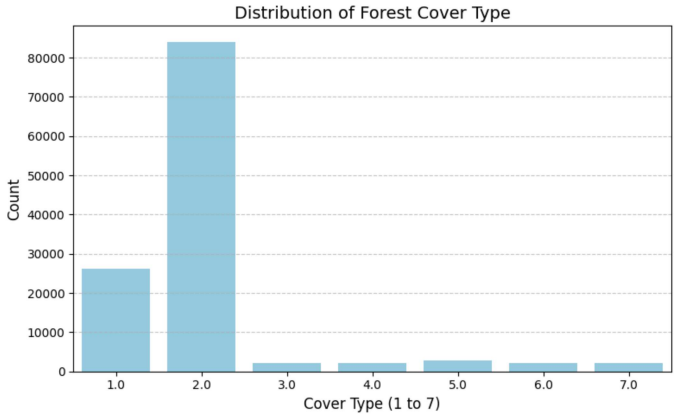

Figure 1: Distribution of Forest Cover Types.

2.2 Feature Correlation and Relationships

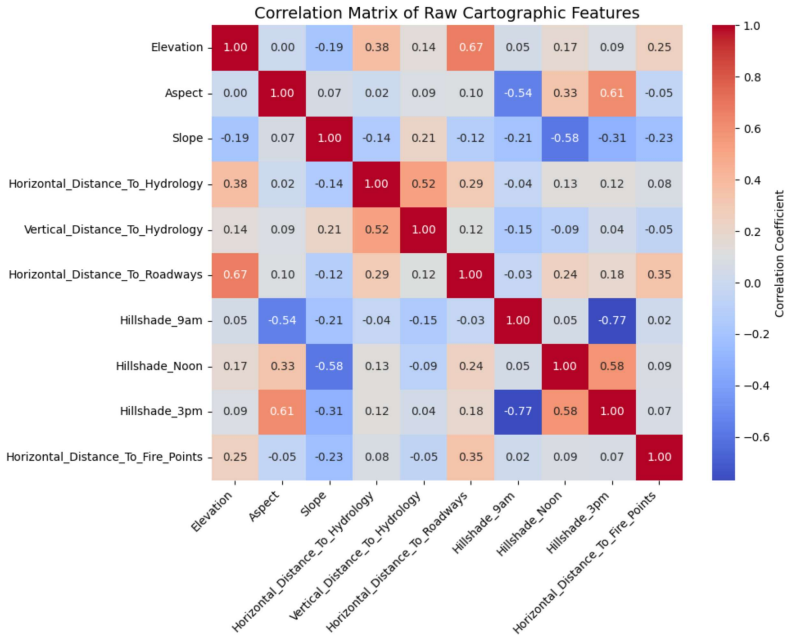

Figure 2: Correlation Heatmap of Raw Cartographic Features.

Analysis of Figure 2:

- **Multicollinearity**: A strong negative correlation ($\rho \approx -0.78$) was identified between `Hillshade_9am` and `Hillshade_3pm`. This redundancy necessitated the use of **PCA** to decorrelate the feature space.

3 Data Preprocessing Strategies

The preprocessing strategy was specialized based on the model: standard steps for DNN/LR, and advanced techniques for SVM feasibility.

3.1 Standard Preprocessing (Applied to DNN and LR)

1. **Train-Test Split**: Data split 80/20, using `stratify=Y`.
2. **Feature Scaling**: `StandardScaler` applied to the 10 raw features.
3. **Dimensionality Reduction**: `PCA(n_components=7)` applied to the scaled features, resulting in 51 total input features (7 PCA + 44 Binary).
4. **Class Weighting**: Weights inversely proportional to class frequency were calculated and applied to model training.

3.2 Specific Preprocessing for SVM Feasibility

5. **Stratified Undersampling**: The full training set ($\sim 460k$) was reduced to a smaller, representative $\mathbf{50,000}$ sample subset. This was **necessary** to make the $\mathcal{O}(N)$ complexity of the Linear SVM computationally feasible for tuning.

6. **Non-Linear Feature Engineering**: `PolynomialFeatures(degree=2)` was applied to the 7 PCA components. This step explicitly created non-linear interaction terms, allowing the fast `LinearSVC` (a linear kernel) to model non-linear boundaries.

4 Model Evaluation: Baseline and Linear Models

4.1 Logistic Regression Performance

LR was trained using class weighting and achieved an expected low accuracy, indicating the data's inherent non-linearity.

Table 1: Classification Report for Logistic Regression Baseline

Cover Type	Precision	Recall	F1-Score
Type 1	0.66	0.66	0.66
Type 2	0.77	0.50	0.60
Type 3	0.68	0.54	0.60
Type 4	0.29	0.89	0.44
Type 5	0.10	0.77	0.18
Type 6	0.31	0.64	0.42
Type 7	0.40	0.88	0.55
Test Accuracy			**58.20%**
Macro Avg F1	0.49	0.70	0.49

4.2 Linear SVM Performance (Tuned C on Sampled Data)

4.2.1 Why Linear SVM Over Kernel SVM?

The decision to use **Linear SVM (`LinearSVC`)** instead of a non-linear **Kernel SVM (e.g., RBF kernel in `SVC`)** was driven purely by computational feasibility:

- **Kernel SVM Complexity**: Standard Kernel SVMs scale poorly, typically $\mathcal{O}(N^2)$ to $\mathcal{O}(N^3)$ with the number of samples (N). Training on the full $\sim 460,000$ training set would take prohibitive amounts of time and memory.

- **Linear SVM Complexity**: Linear SVMs, optimized with libraries like LIBLINEAR, scale much better, closer to $\mathcal{O}(N \times D)$ (where D is the number of features). This made the training and hyperparameter tuning on the 50,000-sample subset feasible within minutes.

4.2.2 Performance Analysis and Rationale for Lower Accuracy

The `LinearSVC` was tuned on C via `GridSearchCV` (optimal $\mathbf{C = 10.0}$).

Table 2: Tuned LinearSVC Model Performance

Hyperparameter Result: Optimal C = 10.0			
Cover Type	Precision	Recall	F1-Score
Type 1	0.72	0.68	0.70
Type 2	0.78	0.69	0.73
Type 3	0.69	0.72	0.71
Type 4	0.52	0.74	0.61
Type 5	0.18	0.56	0.27
Type 6	0.40	0.59	0.48
Type 7	0.53	0.86	0.65
Test Accuracy			**68.98%**
Macro Avg F1	0.54	0.69	0.59

Reason for 68.98% Accuracy (Lower than DNN): The SVM's accuracy, though a **10.78%** gain over LR, is significantly lower than the DNN primarily due to two factors inherent to our approach:

1. **Information Loss from Undersampling**: Training on only $\mathbf{50,000}$ samples meant the model never saw the majority of the variance present in the full $\sim 460,000$ training set. This limits the model's ability to learn robust, generalizable patterns, resulting in higher error rates on the full test set.

2. **Linear Constraint**: Despite using Polynomial Features for engineering, the final model is still a linear classifier in the transformed space. The DNN, being inherently non-linear and trained on the full dataset, could learn optimal complex boundaries that the Linear SVM simply cannot achieve.

5 Model Evaluation: Deep Neural Network (DNN)

5.1 Optimal Hyperparameters Found by Hyperband

Table 3: Optimal Hyperparameters Found by Hyperband

Hyperparameter	Optimal Value
Number of Layers	**3**
Units in Layer 1	**512**
Units in Layer 2	**384**
Units in Layer 3	**64**
Dropout Rate	**0.10**
Learning Rate	**0.001**

5.2 Final DNN Evaluation

Table 4: Classification Report for Tuned DNN Model

Cover Type	Precision	Recall	F1-Score
Type 1	0.89	0.87	0.88
Type 2	0.93	0.86	0.89
Type 3	0.87	0.91	0.89
Type 4	0.58	0.97	**0.73**
Type 5	0.44	0.98	**0.61**
Type 6	0.79	0.86	0.82
Type 7	0.72	0.99	0.83
Test Accuracy			**87.31%**
Macro Avg F1	0.88	0.87	0.88

6 Final Conclusion and Comparative Analysis

Table 5: Model Performance Summary (Macro F1-Score)

Model	Training Strategy	Test Accuracy	Macro Avg F1-Score
Logistic Regression	Full Data, Linear	58.20%	0.49
Linear SVM	Sampled Data, Feature Engineered	68.98%	0.59
Tuned DNN	Full Data, Deep Non-Linear	**87.31%**	**0.88**

Summary of Findings: The project demonstrates that **model complexity** is the dominant factor for performance on this dataset.

1. **Linear Models (LR & SVM)**: Both struggled to capture the inherent non-linearity. The SVM improved significantly over LR due to the **Polynomial Feature Engineering** (raising F1 from 0.49 to 0.59), but was ultimately limited by the necessity of **undersampling** and the constraints of its linear kernel approach. 2. **Deep Neural Network (DNN)**: The DNN's high capacity and inherent ability to learn complex, non-linear representations across the entire dataset were key to its success. The final tuned model achieved the highest performance (**87.31**% accuracy), successfully solving the class imbalance problem while achieving high overall accuracy.