Uppsala University

Introduction to Machine Learning

1DL034

Forest Cover Type Classification Project

Project Group 20:
Hamit Efe Çınar
Hwee Hwa Jovi Tan
Kim Sitthikesorn
Mehmet Firat Dundar

March 8, 2024

1 Introduction

The Covertype dataset is a widely used dataset in the field of machine learning and environmental science. It contains information about the Roosevelt National Forest in Colorado, including various attributes such as elevation, slope, aspect, soil type, and wilderness area. The dataset is often utilized for predictive modeling tasks, particularly in the context of classification.

The main objective of the project using the Covertype dataset is to predict the forest cover type based on the provided attributes. This is a classification problem, where the goal is to assign each observation (or data point) to one of the seven possible classes of forest cover types. The classes typically represent different types of tree species that dominate the forest area, such as spruce/fir, lodgepole pine, ponderosa pine, cottonwood/willow, aspen, Douglas-fir, and krummholz.

Given the diverse range of features available in the dataset, including both categorical and continuous variables, the challenge lies in effectively leveraging this information to build a predictive model that can accurately classify the forest cover types.

2 Feature Extraction and Preprocessing

Feature extraction and preprocessing are crucial for making models perform better. They refine the dataset, making it easier for modelling. Feature extraction finds the most important parts of the data, helping the model understand patterns better. Preprocessing, like cleaning and normalising the data, makes it more reliable. It ensures all features are treated equally, so the model isn't biassed. By doing this, the model becomes stronger and gives more accurate predictions. Overall, feature extraction and preprocessing are vital for building good models.
In our implementations, we executed the following steps:

2.1 Data Cleaning

This involved handling missing, error and duplicate values, as well as checking for any categorical values. In this case, no such data requiring handling or feature encoding was found in the dataset.

2.2 Feature Selection

Given that the dataset contains both continuous and binary features, they were separated. A heatmap of the correlation was plotted (Figure 1) to analyse the features for selection. From this analysis, features with high correlations were identified as potential choices for selection.

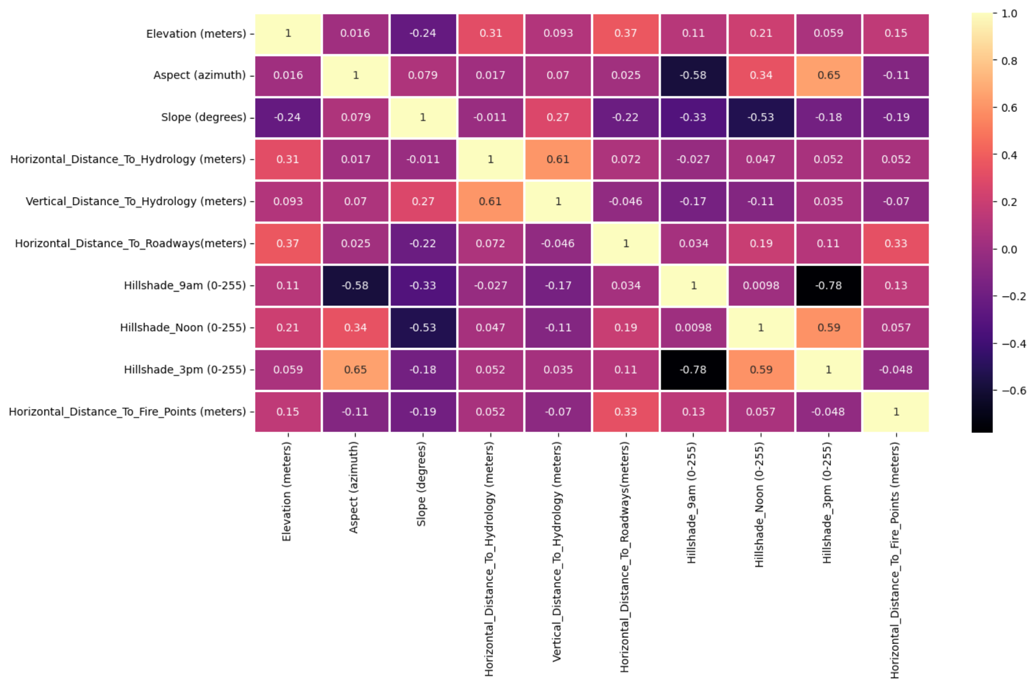
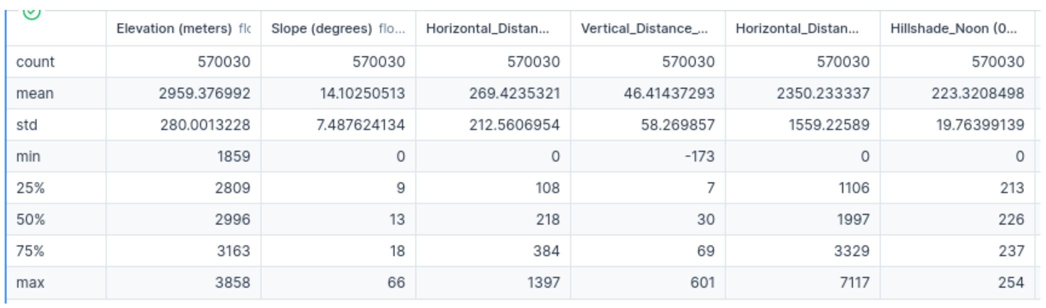


Figure 1: Depicts a heatmap illustrating the correlations among the continuous features.

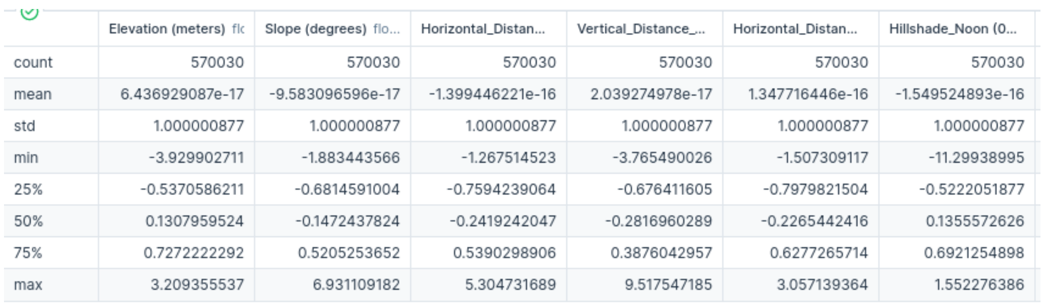
2.3 Feature Scaling and Standardization

Differences in scales among features were observed by examining their mean values. Features with larger scales typically have higher mean values compared to those with smaller scales. Therefore, features were scaled and standardised to ensure consistency in their representation (Figures 2 & 3).



	Elevation (meters) flc	Slope (degrees) flo...	Horizontal_Distan...	Vertical_Distance_...	Horizontal_Distan...	Hillshade_Noon (0...
count	570030	570030	570030	570030	570030	570030
mean	2959.376992	14.10250513	269.4235321	46.41437293	2350.233337	223.3208498
std	280.0013228	7.487624134	212.5606954	58.269857	1559.22589	19.76399139
min	1859	0	0	-173	0	0
25%	2809	9	108	7	1106	213
50%	2996	13	218	30	1997	226
75%	3163	18	384	69	3329	237
max	3858	66	1397	601	7117	254

Figure 2: Table illustrating the dataset's characteristics before standardization.



	Elevation (meters) flc	Slope (degrees) flo...	Horizontal_Distan...	Vertical_Distance_...	Horizontal_Distan...	Hillshade_Noon (0...
count	570030	570030	570030	570030	570030	570030
mean	6.436929087e-17	-9.583096596e-17	-1.399446221e-16	2.039274978e-17	1.347716446e-16	-1.549524893e-16
std	1.000000877	1.000000877	1.000000877	1.000000877	1.000000877	1.000000877
min	-3.929902711	-1.883443566	-1.267514523	-3.765490026	-1.507309117	-11.29938995
25%	-0.5370586211	-0.6814591004	-0.7594239064	-0.676411605	-0.7979821504	-0.5222051877
50%	0.1307959524	-0.1472437824	-0.2419242047	-0.2816960289	-0.2265442416	0.1355572626
75%	0.7272222292	0.5205253652	0.5390298906	0.3876042957	0.6277265714	0.6921254898
max	3.209355537	6.931109182	5.304731689	9.517547185	3.057139364	1.552276386

Figure 3: Table illustrating the dataset's characteristics after standardization.

In algorithms like k-NN, the class of a data point is determined by examining the classes of its nearest neighbours in the feature space. This is achieved by calculating the distance between data points using certain measures. However, when features have varying scales, those with larger scales tend to exert a disproportionately higher influence on the distance calculation. This imbalance can potentially bias the algorithm's decisions. Feature scaling is implemented to address this issue by ensuring that all features contribute equally to the distance calculations, thus preventing such biases.

2.4 Data Splitting

In this step, the available dataset was divided into subsets. These subsets are typically used for various purposes such as training and testing. The primary objective of data splitting is to accurately evaluate the model's performance and prevent overfitting.

After the preprocessing steps, all features except for 'Aspect (azimuth)', 'Comanche Peak Wilderness Area (3/4)', 'Hillshade 9am (0-255)', '3502 (8/40)', '5101 (14/40)', '5151 (15/40)', '7103 (21/40)', '7701 (25/40)', '7710 (28/40)', '8707 (36/40)', and '8708 (37/40)' were utilised. This serves as an initial setup before considering any potential optimization steps.

It's noteworthy that the accuracy of the trained model was tested both before and after the preprocessing and feature selection steps for comparison. The accuracy scores were 0.92 before and 0.96 after preprocessing. These results were obtained using the RandomForestClassifier, which outperformed other classifier methods such as SVC, LinearSVC, DecisionTree, and KNeighbors.

3 Imbalanced Data Consideration

When we look at the histogram of the class labels in the dataset, we notice that there are imbalances in the dataset. Specifically, classes 1 and 2 have many more instances than the others. To tackle this issue, we can use techniques like Oversampling, Undersampling, or Synthetic Data. Oversampling involves adding more instances of the minority class, either by copying existing ones or making new synthetic samples. This helps balance the class distribution, so the model doesn't favour the majority class too much. Undersampling, on the other hand, reduces the number of instances in the majority class to match the minority class. This can be done by randomly selecting some instances from the majority class. However, be cautious, as this might cause us to lose important information from the majority class. Synthetic Data techniques like SMOTE create fake data points for the minority class by filling in the gaps between existing instances. This balances the classes without throwing away any data.

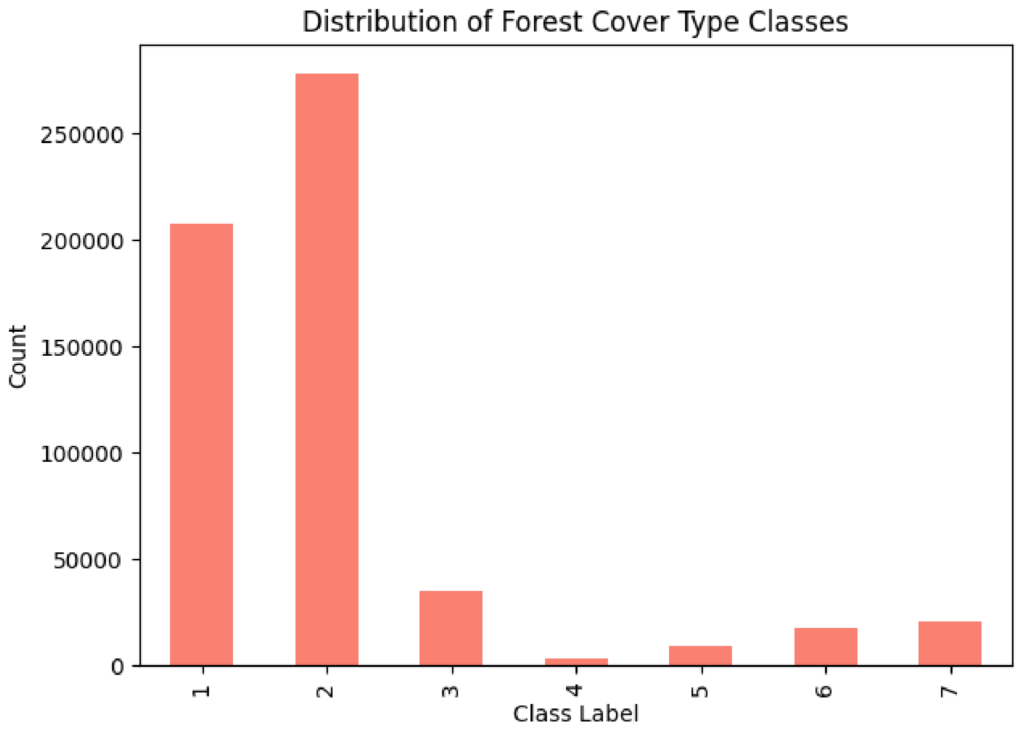


Figure 4: Histogram illustrating the count of instances for each of the seven class labels.

4 Algorithm Exploration

The following categories of machine learning models were explored for this dataset. Logistic Regression, Support Vector Machines, Nearest Neighbors, Decision Trees and Random Forest.

Since Linear Regression is typically not used for multiclass classification, it was not explored. Since Clustering is typically meant for unsupervised learning, it was not explored.

We compared the accuracy of the models using default parameters provided by the scikit-learn library. We also varied the dataset used, ”Raw” for no preprocessing, ”Standardization and Manual Removal” for after using StandardScaler and removing high correlation features, ”Previous Steps + PCA” for all previous preprocessing and after using Principle Component

Analysis with n_components="mle" from the scikit-learn library, and "Previous Steps + Downsampling" for all previous preprocessing including PCA and downsampling using ClusterCentroids.

Both Logistic Regression and SVC Linear SVM models failed to converge so we trained the models again after increasing the maximum number of iterations. The SVM RBF and Linear Kernels were exlcuded from the "Raw" and "After Previous Steps + PCA" datasets due to long training times. Standardization and Manual Removal appeared to have the best improvement. The accuracies are shown below:

Model	Accuracy	
	Raw	Standardization and Manual Removal
Logistic Regression	0.70628	0.72243
SVM RBF Kernel	NIL	0.83669
SVM Linear Kernel	NIL	0.72512
SVM LinearSVC	0.64562	0.71192
Nearest Neighbors	0.95827	0.93660
Decision Tree	0.93807	0.94044
Random Forest	0.95408	0.96117

Table 1: Model Accuracy Comparison

Model	Accuracy	
	Previous Steps + PCA	Previous Steps + Downsampling
Logistic Regression	0.72242	0.70627
SVM RBF Kernel	NIL	0.76609
SVM Linear Kernel	NIL	0.69219
SVM LinearSVC	0.71188	0.64583
Nearest Neighbors	0.93660	0.75656
Decision Tree	0.90560	0.73907
Random Forest	0.94702	0.81589

Table 2: Model Accuracy Comparison

5 Hyperparameter Tuning

Out of all the models we explored, the Random Forest model had the highest accuracy. We performed Grid Search & Randomized Search to find the best hyperparameter for the model.

5.1 Grid Search

For the Random Forest model, we constructed a grid to iterate through (Table 3).

Hyperparameter	Choices			
n_estimators	100	200	300	
max_depth	None	10	20	30
min_samples_split	2	5	10	
min_samples_leaf	1	2	4	
bootstrap	True		False	

Table 3: Random Forest Hyperparamter Grid

The best parameters found were: {'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300} with an accuracy of 0.96563.

5.2 Randomized Search

In addition, we did a randomized search for the Random Forest model hyperparameters (Table 4).

Hyperparameter	Choices or Range	
n_estimators	100 to 500	
max_depth	None	5 to 50
min_samples_split	2 to 20	
min_samples_leaf	1 to 20	
bootstrap	True	False

Table 4: Random Forest Random Hyperparamters

The best parameters found were: {'bootstrap': False, 'max_depth': 48, 'min_samples_leaf': 1, 'min_samples_split': 9, 'n_estimators': 499} with an accuracy of 0.96216.

6 Evaluation Metrics

For the evaluation metrics section, we first recalculate the accuracy of the best model that we found in the previous section. To observe the impact of the features on the model, we tested the feature importance for each of them. Then, we tried different threshold values of feature importance to elect the unnecessary features to further improve the model.

We first tried the values 0.005, 0.003, 0.002, 0.001 respectively and observed that the model precision was decreasing for threshold values less than 0.002 and for the values more than 0.003 (Table 5). So, we deduced that there must be a local maxima between. After trying couple of values in between, we found the 0.027 as the best result for this number of decimal points. At the end, we elected the features: 4704 (11/40), 2704 (3/40), 7102 (20/40), 6102 (17/40), 8703 (35/40), 2702 (1/40), 7101 (19/40), 6101 (16/40), 7790 (34/40), 7709 (27/40), 2706 (5/40), 7702 (26/40), 6731 (18/40), 4201 (9/40), 3501 (7/40).

0.005: 96.52

Threshold Value	Accuracy
0.005	96.52%
0.003	96.70%
0.0027	96.75%
0.002	96.68%
0.001	96.59%

Table 5: Accuracy Results with Different Threshold Values

The performance of the RandomForestClassifier model was evaluated using several metrics: accuracy, precision, recall, F1 score, confusion matrix, and a detailed classification report. These metrics provide a comprehensive understanding of the model's performance on the test dataset. The accuracy of the model, which measures the proportion of correct predictions among the total number of cases evaluated, was found to be **96.75%**. This high accuracy indicates that the model is highly effective in classifying the data correctly.

Precision and recall are critical metrics, especially in scenarios where the cost of false positives and false negatives varies significantly. Precision, which measures the accuracy of positive predictions, was **96.75%**. Recall, which measures the ability of the model to find all the positive samples, was also **96.75%**. The F1 score, a weighted average of precision and recall, was **96.74%**, indicating a balanced performance between precision and recall.

The confusion matrix provides a detailed breakdown of the model's predictions, showing the number of correct and incorrect predictions for each class. The matrix for our model is as follows:

	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**Class 6**	**Class 7**
Class 1	39821	1492	0	0	9	4	80
Class 2	962	54520	89	0	73	45	11
Class 3	1	80	6687	30	8	121	0
Class 4	0	0	59	464	0	22	0
Class 5	21	239	19	0	1551	5	2
Class 6	1	58	168	3	3	3190	0
Class 7	129	14	0	0	0	0	4025

Table 6: Confusion matrix of the RandomForestClassifier

The classification report provides a detailed analysis of the performance for each class. It includes metrics such as precision, recall, and F1 score for each class. The report for our model is as follows:

Class	Precision	Recall	F1-Score	Support
1	0.97	0.96	0.97	41406
2	0.97	0.98	0.97	55700
3	0.95	0.97	0.96	6927
4	0.93	0.85	0.89	545
5	0.94	0.84	0.89	1837
6	0.94	0.93	0.94	3423
7	0.98	0.97	0.97	4168
Accuracy	0.97 (114006)			
Macro Avg	0.96	0.93	0.94	114006
Weighted Avg	0.97	0.97	0.97	114006

Table 7: Classification report of the RandomForestClassifier

7 Conclusion

In this project, This project focuses not only on machine learning algorithms but also on the crucial steps of feature extraction and preprocessing. By comparing multiple algorithms and trying various preprocessing techniques, you will gain valuable insights into the interplay between data preparation and model performance. This approach aims to provide a deeper understanding of the nuances involved in real-world machine learning tasks.

We tackled a multiclass classification problem by comparing various machine learning models, including SVM, Decision Trees, KNN, and Random Forest. After thorough evaluation, the RandomForestClassifier was identified as the most effective model.

The RandomForestClassifier exhibited outstanding performance with an accuracy of 96.75%, precision and recall of 96.74%, and a F1 score of 96.74%. Its success is attributed to its robust handling of feature interactions and resilience to overfitting, enhanced by rigorous preprocessing and intelligent feature selection. Extensive hyperparameter tuning further optimized its performance.

While other models showed promise, the RandomForestClassifier outperformed them in terms of accuracy, stability, and handling complex data structures. Its superior performance metrics and robustness against overfitting solidified its selection as the model of choice.

In conclusion, the RandomForestClassifier stands out as a reliable and effective tool for multiclass classification tasks. Future work may involve exploring advanced ensemble methods, deep learning models, or applying the model to real-world scenarios to assess its practical utility.