ML PROJECT

FOREST COVER TYPE PREDICTION

by

Bhavya Bhalla – 102103345
Shivansh Gupta - 102103321

Submitted to

Dr. Jyoti Maggu

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, (A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

INDIA

July – December 2023

Introduction to the project – Forest Cover Type Prediction

The "Forest Cover Type Prediction" project is an exciting endeavor in the field of machine learning and environmental research. This project involves tackling a multi-class classification problem, where the goal is to determine the specific forest cover type of a given area based on a variety of environmental and geographical features. These cover types could include various forest categories, such as spruce, Douglas, lodgepole, and more, which are crucial for understanding the composition of forested regions.

Literature

The significance of this project lies in its potential applications for land management, conservation, and ecological studies. Accurate forest cover type predictions can aid in making informed decisions about forest management and resource allocation. For instance, it can help identify areas where specific tree species are thriving or struggling, allowing for targeted interventions to preserve biodiversity and maintain healthy forests. To address this challenging problem, machine learning techniques are employed, leveraging algorithms that can effectively analyze and learn from the provided dataset. The dataset comprises various environmental attributes like soil types, elevation, aspect, and more, which play a crucial role in determining the forest cover type of a given region. By training machine learning models on this dataset, researchers and conservationists can gain insights into the relationships between environmental factors and forest cover types, leading to more sustainable and data-driven forest management strategies. In summary, the "Forest Cover Type Prediction" project offers an opportunity to harness the power of machine learning to better understand and predict forest cover types, with far-reaching implications for environmental conservation and sustainable land management. It demonstrates the potential of data science in making a positive impact on our natural ecosystems and biodiversity.

Methodology

The methodology for the Forest Cover Type Prediction project involves a series of steps to preprocess and analyze the dataset, balance the class distribution, and train machine learning models to predict forest cover types. Here is a summary of the key steps in the process:

1) **Importing Libraries**: The project begins by importing essential libraries like pandas, NumPy, scikit-learn(sklearn), matplotlib, imbalanced-learn (imblearn), and seaborn. These libraries provide tools for data manipulation, visualization, and machine learning.
2) **Data Exploration:** The first step in understanding the dataset is to read the CSV file and explore its contents. This includes identifying the various features and checking for any missing (null) values. Additionally, the distribution of different classes in the target column, representing different forest cover types, is examined.
3) **Data Visualization:** The project includes data visualization steps, such as plotting curves to show the distribution of different features. Visualizing the data is crucial for gaining insights into its characteristics.
4) **Data Scaling:** As the dataset may contain features with varying scales, data scaling is applied to standardize the data, ensuring that all features have a similar impact on the machine learning models.
5) **Class Imbalance Handling:** To address the class imbalance issue, the Synthetic Minority Over-sampling Technique (SMOTE) is used to balance the dataset by generating synthetic examples of minority classes.
6) **Data Splitting:** The dataset is divided into two parts: a training dataset and a testing dataset. This separation allows for model training and evaluation.
7) **Model Selection and Training:** Several machine learning models such as Logistic Regression (56.3%), Decision Tree Classifier (81.02%) and Random Forest Classifier (90.81%) are evaluated for their predictive accuracy. The project mentions trying logistic regression, decision tree, and random forest classifier models. The accuracy of each model on the testing dataset is recorded.
8) **Model Selection:** Based on the accuracy results, the Random Forest Classifier is chosen as the final model for forest cover type prediction due to its superior performance.
9) **Model Export:** The trained Random Forest Classifier model is exported using the 'pickle' library, making it available for future predictions without the need to retrain the model.

In summary, the Forest Cover Type Prediction project employs a systematic approach to data preprocessing, class imbalance handling, and model selection to achieve accurate predictions of forest cover types. The chosen Random Forest Classifier is exported for practical use in real-world applications.

Dataset Description

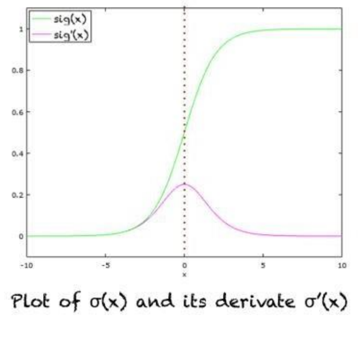
The dataset which we are using is a tabular dataset used for forest cover type prediction. Each column represents a specific feature, and the last column ("class") indicates the target variable/ column, which is the forest cover type. Here is a description of the features:
1) **Elevation:** This feature typically represents the elevation above sea level at a specific location in the forest. It is an important environmental factor that can influence the type of vegetation and trees found in thatarea.
2) **Aspect:** Aspect refers to the compass direction in which a slope faces. It can affect temperature, sunlight exposure, and soil moisture, all of which play a role in determining forest cover types.
3) **Slope:** Slope represents the steepness of the terrain. Steeper slopes can influence water drainage and affect the types of trees and vegetation that can grow in an area.
4) **Horizontal Distance to Hydrology:** This feature measures the horizontal distance from the location to the nearest water body, such as a river or stream. It's important for understanding the proximity to water sources.
5) **Vertical Distance to Hydrology:** This represents the vertical distance from the location to the nearest water body. It can affect soil moisture levels and vegetation types.
6) **Horizontal Distance to Roadways:** This feature measures the horizontal distance from the location to the nearest road or path, which can impact accessibility and land use.
7) **Horizontal Distance to Fire Points:** This feature indicates the horizontal distance to the nearest fire ignition point. It is relevant for understanding fire risk and forest management.
8) **Wilderness Area 1-4:** These binary features likely represent categorical variables indicating the presence or absence of certain wilderness areas. Each wilderness area might have unique characteristics.
9) **Soil Type 1-40:** These binary features represent categorical variables indicating the presence or absence of specific soil types. Different soil types can support different vegetation and tree species.
10) **Class:** This is the target variable, representing the forest cover type that the model aims to predict. It likely includes several distinct classes or categories (e.g., different types of trees or forest cover).

This dataset is well-suited for machine learning project where we can use the various environmental and geographical features to predict the forest cover type, which is a multi-class classification task. The dataset provides a comprehensive set of attributes that can influence the distribution of forest types in a given area, making it valuable for ecological research and land management applications.

Models Used

1) **Logistic Regression:** Logistic Regression is a straightforward yet powerful model for binary and multi-class classification. It's interpretable, computationally efficient, and serves as a foundational building block in machine learning. Its output is a probability that an input belongs to a specific class, making it a versatile tool for a wide range of classification tasks. Here's a detailed explanation of how the Logistic Regression model works:

- Sigmoid Function: At the heart of Logistic Regression is the sigmoid (logistic) function, which transforms any real-valued number into a value between 0 and 1. The sigmoid function is defined as:



Plot of $\sigma(x)$ and its derivate $\sigma'(x)$

Domain: $(-\infty, +\infty)$
Range: $(0, +1)$
$\sigma(0) = 0.5$

Other properties

$$\sigma(x) = 1 - \sigma(-x)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

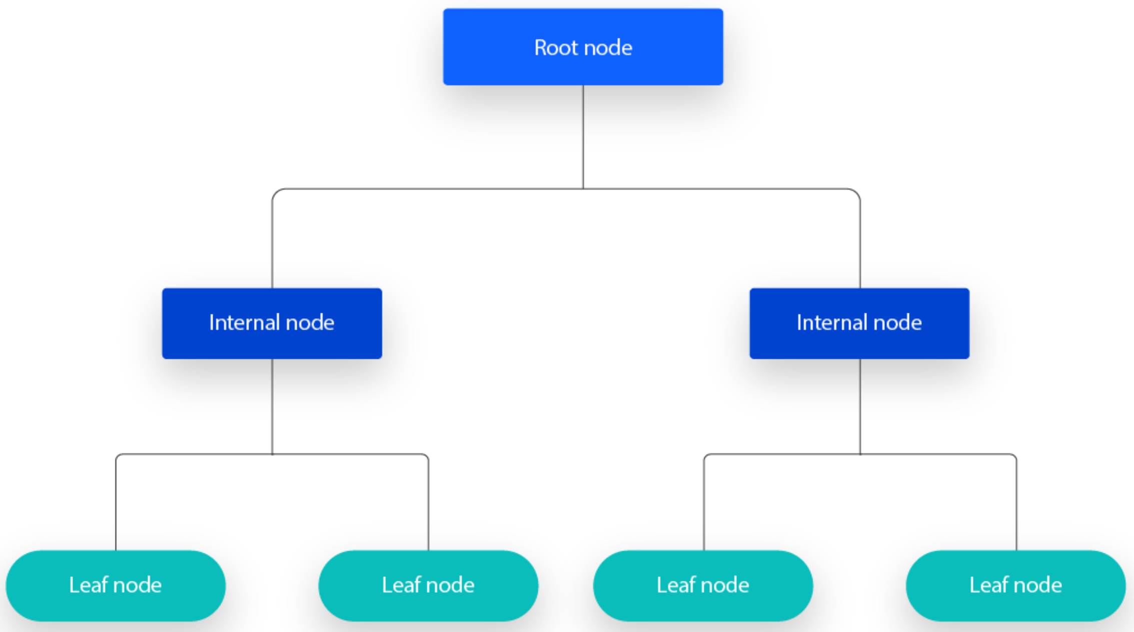
Here, "z" represents the linear combination of the input features and their associated weights, plus a bias term. In logistic regression, the goal is to find the optimal weights and bias to best fit the data.

- Thresholding: To make a binary classification decision, you can set a threshold (usually 0.5) on the predicted probabilities. If the predicted value is greater than the threshold, the input is classified as the positive class (e.g., 1), and if it's less than the threshold, it's classified as the negative class (e.g., 0).
- Regularization: Logistic Regression can be regularized to prevent overfitting. Common regularization techniques include L1 regularization (Lasso) and L2 regularization (Ridge), which add penalty terms to the cost function.
- Multi-Class Classification: Logistic Regression can be extended to handle multi-class classification problems

2) **Decision Tree Classifier:** A Decision Tree Classifier is a popular machine learning algorithm used for both classification and regression tasks. It's a tree-like structure where each internal node represents a feature (or attribute), each branch represents a decision rule, and each leaf node represents a class label. Decision trees are particularly useful for their simplicity, interpretability, and the ability to handle both categorical and numerical data. Here's a detailed explanation of how a Decision Tree Classifier works:

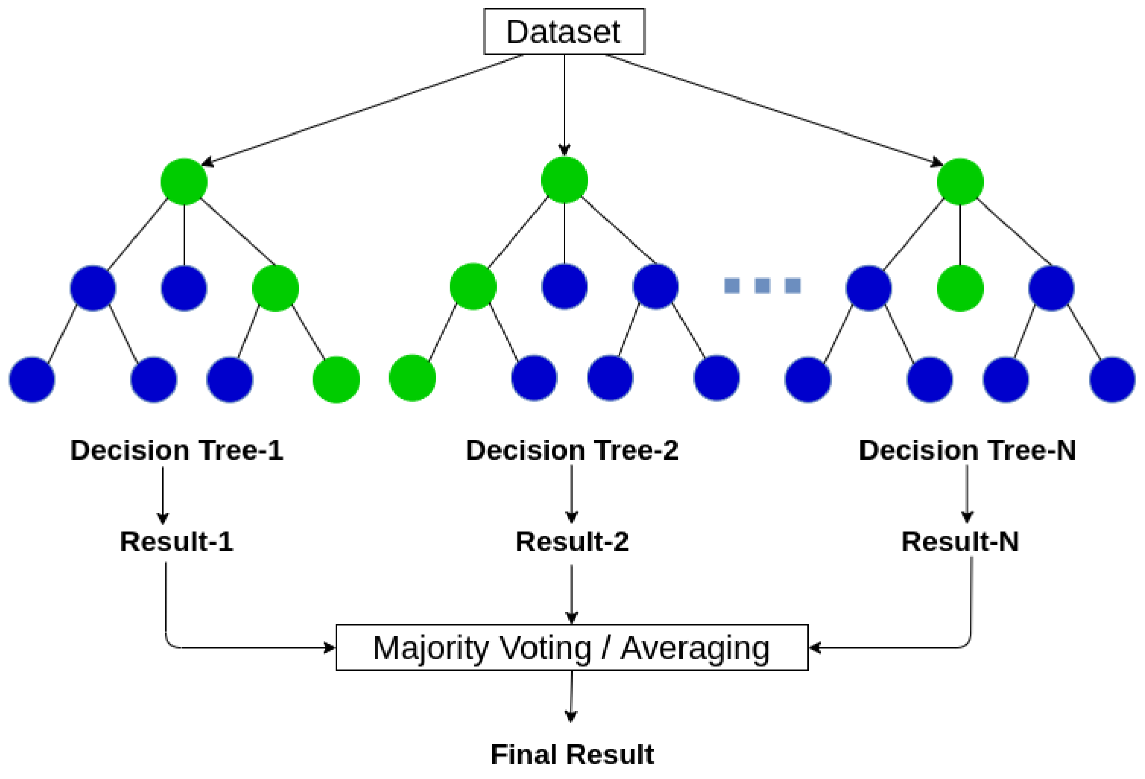
- Tree Structure: A Decision Tree starts with a single root node that represents the entire dataset. At each internal node, the algorithm selects a feature to split the data into subsets based on a specific decision rule (e.g., if-else conditions). The data is divided into subsets and assigned to child nodes based on the decision rule.
- Splitting Criteria: The algorithm determines the best feature and decision rule for each split. This is done by evaluating different splitting criteria, such as Gini impurity, information gain, or entropy. The goal is to create subsets that are as pure as possible, meaning they contain data points predominantly from a single class.
- Predictions: To make predictions for a new data point, you start at the root node and follow the decision rules down the tree until you reach a leaf node. The class label associated with the leaf node is the predicted class for the input data point.
- Handling Categorical and Numerical Data: Decision Trees can handle both categorical and numerical features. For categorical features, the algorithm can split the data into branches based on distinct categories. For numerical features, it finds the best threshold to split the data.
- Prone to Overfitting: Decision Trees are prone to overfitting, especially when they are deep and complex. Overfitting occurs when the model captures noise in the training data, resulting in poor generalization to new, unseen data. Techniques like pruning (reducing the size of the tree) and setting maximum tree depth or minimum samples per leaf can help mitigate overfitting.

Decision Tree Classifiers are useful in a wide range of applications, from medical diagnosis to customer churn prediction. They are particularly valuable when you need a transparent and interpretable model that provides insights into the decision-making process. However, it's essential to be cautious about overfitting, and in practice, ensemble methods are often preferred for better generalization.



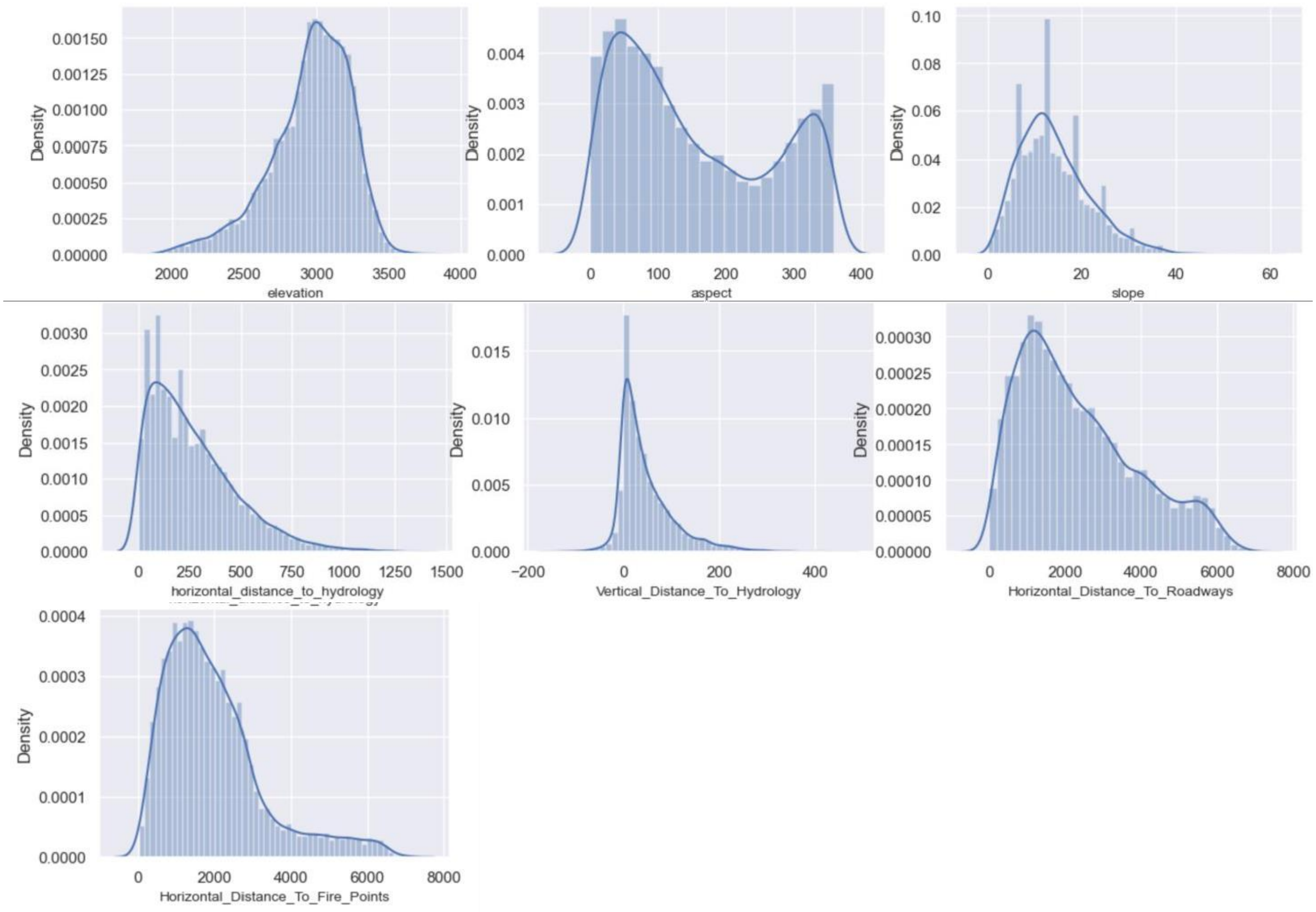
3) **Random Forest Classifier:** The Random Forest Classifier is an ensemble learning method that builds multiple decision trees during the training process and combines their outputs to make predictions. It's a powerful and widely used algorithm for both classification and regression tasks. The key idea behind Random Forest is to reduce overfitting and improve the predictive accuracy by aggregating the results of multiple individual decision trees. Here's a detailed explanation of how the Random Forest Classifier works:

- Ensemble of Decision Trees: Random Forest consists of an ensemble of decision trees, where each tree is built independently from the others.
- Parallel Processing: Random Forests are well-suited for parallel processing since each decision tree can be built independently, making them computationally efficient.
- Bootstrapping (Sampling with Replacement): For each decision tree in the ensemble, a random subset of the training data is selected with replacement. This means that some data points will appear multiple times in the training set, while others may not be included at all. This process is known as bootstrapping and helps introduce diversity among the trees.
- Feature Randomization: In addition to sampling data, Random Forest introduces randomness in the feature selection process. At each node of a decision tree, a random subset of features is considered for splitting. The number of features to consider at each split is a hyperparameter that can be adjusted, typically set to the square root of the total number of features.
- Voting or Averaging: Once all decision trees in the ensemble are built, they can collectively make predictions. For classification tasks, each tree "votes" for a class, and the class with the most votes becomes the final prediction (the mode of the class predictions).

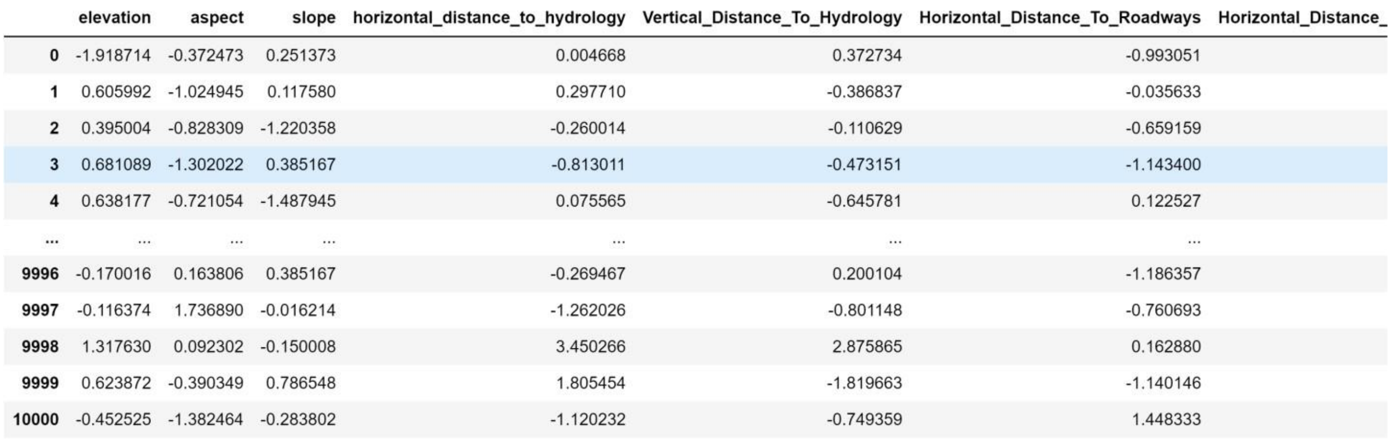


Experimental Analysis and Results

1) **Distribution of features:** Here we wrote a code which is creating distribution plots for the numerical data in the dataset. Each subplot in the grid displays the distribution of data in a specific numerical column. These distributions show how the data is distributed across different values, making it easy to visualize patterns and characteristics of the data, such as its central tendency, spread, and potential outliers. It's a common way to explore and analyze the distribution of numerical data in data analysis and data visualization tasks.

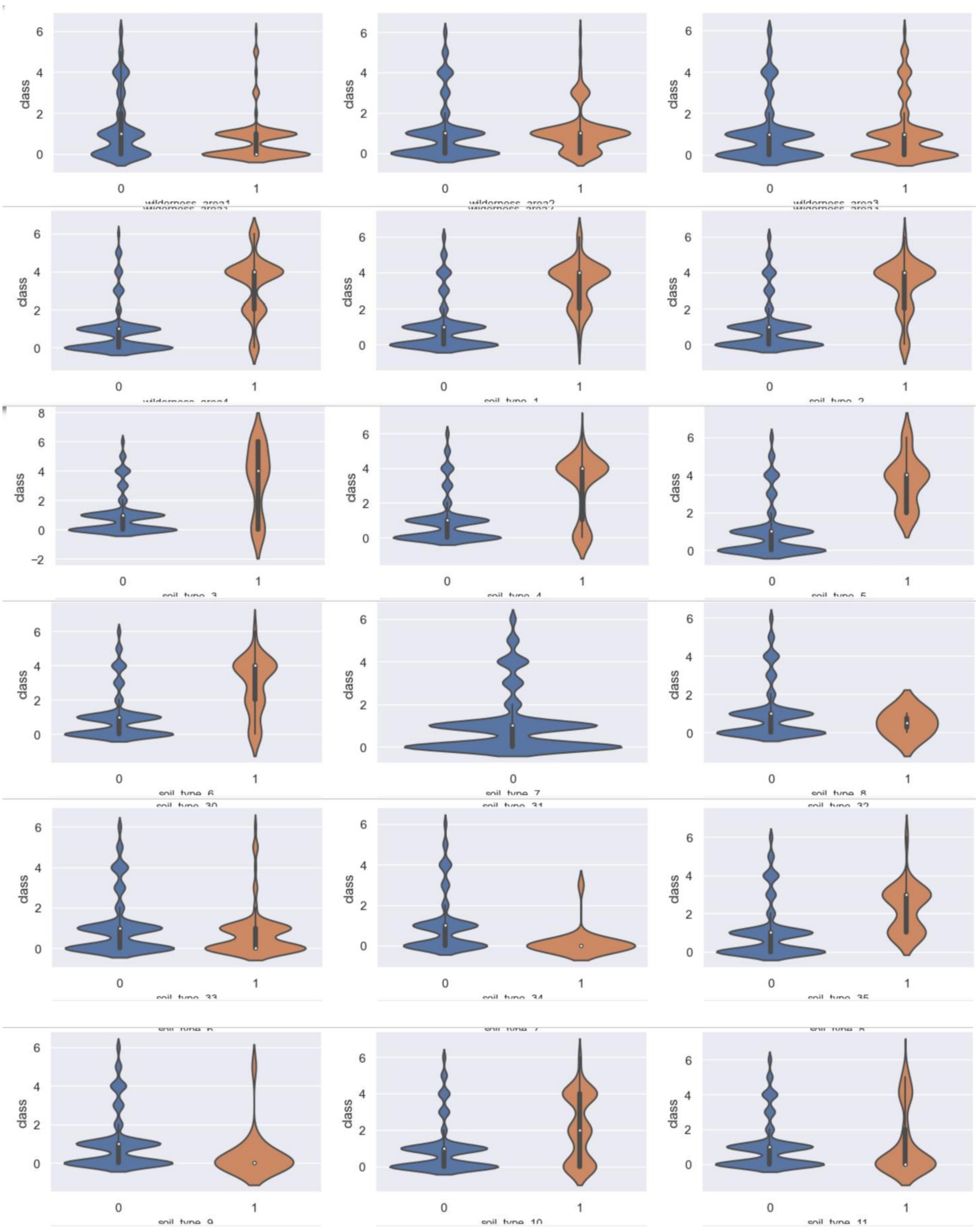


2) **Scaling of dataset:** Scaling is an essential preprocessing step to ensure that the features in your dataset are on a similar scale, which can improve the performance and stability of various machine learning algorithms and make the analysis of the data more meaningful and interpretable. The choice of scaling method (e.g., standardization or min-max scaling) depends on the specific characteristics of your data and the requirements of the algorithm you are using.

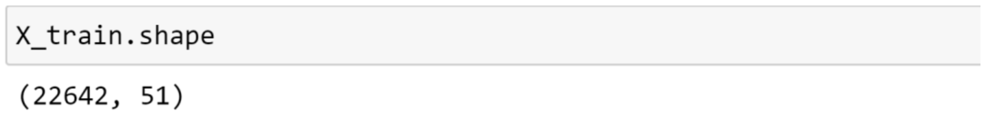


	elevation	aspect	slope	horizontal_distance_to_hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Horizontal_Distance_
0	-1.918714	-0.372473	0.251373	0.004668	0.372734	-0.993051	
1	0.605992	-1.024945	0.117580	0.297710	-0.386837	-0.035633	
2	0.395004	-0.828309	-1.220358	-0.260014	-0.110629	-0.659159	
3	0.681089	-1.302022	0.385167	-0.813011	-0.473151	-1.143400	
4	0.638177	-0.721054	-1.487945	0.075565	-0.645781	0.122527	
...	...	...	...	...	...	...	
9996	-0.170016	0.163806	0.385167	-0.269467	0.200104	-1.186357	
9997	-0.116374	1.736890	-0.016214	-1.262026	-0.801148	-0.760693	
9998	1.317630	0.092302	-0.150008	3.450266	2.875865	0.162880	
9999	0.623872	-0.390349	0.786548	1.805454	-1.819663	-1.140146	
10000	-0.452525	-1.382464	-0.283802	-1.120232	-0.749359	1.448333	

3) **Violin Plots:** Violin plots are a versatile and informative visualization tool that can help you understand and compare data distributions, identify patterns, and communicate insights effectively. They are particularly useful when you need to explore and convey the nuances of your data beyond what basic summary statistics or traditional box plots can offer. Here with the help of violin plots we got to know that majority of values in the class column is of Lodgepole_Pine and Spruce_Fir.

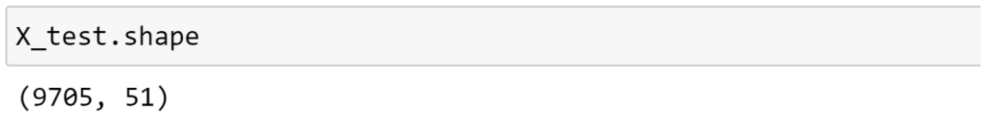


4) **Balancing of the dataset:** Balancing a dataset with Synthetic Minority Over-sampling Technique (SMOTE) is a common approach to address class imbalance in machine learning. Class imbalance occurs when one class in the dataset has significantly fewer instances than another, which can lead to biased model performance. SMOTE is used to generate synthetic examples of the minority class, making it more balanced with the majority class.



```
X_train.shape
```

```
(22642, 51)
```



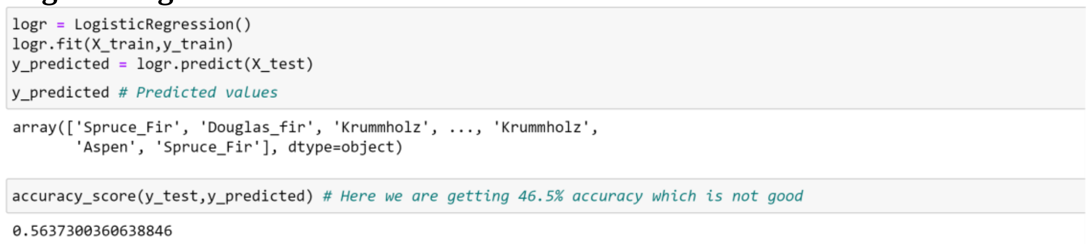
```
X_test.shape
```

```
(9705, 51)
```

As we can see that now the number of rows have increased from 10001 to 32347 rows thus making the dataset balanced.

5) **Accuracy of different models used:** We have used three models and then compared the accuracies of these three models. At the end we have selected the best out of the three models.

- **Logistic Regression**



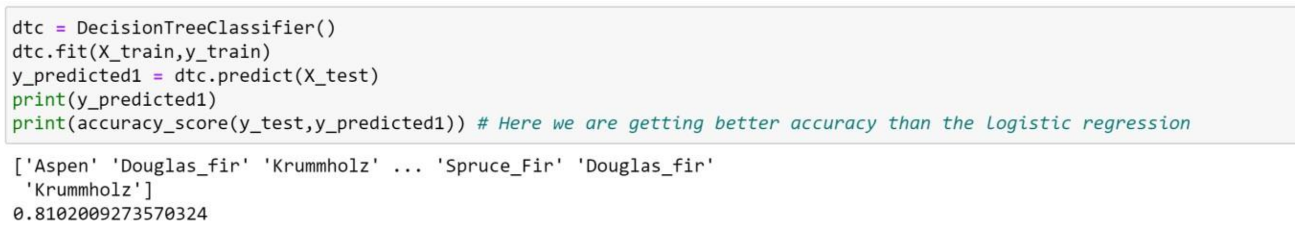
```
logr = LogisticRegression()
logr.fit(X_train,y_train)
y_predicted = logr.predict(X_test)
y_predicted # Predicted values
```

```
array(['Spruce_Fir', 'Douglas_fir', 'Krummholz', ..., 'Krummholz',
       'Aspen', 'Spruce_Fir'], dtype=object)
```

```
accuracy_score(y_test,y_predicted) # Here we are getting 46.5% accuracy which is not good
```

```
0.5637300360638846
```

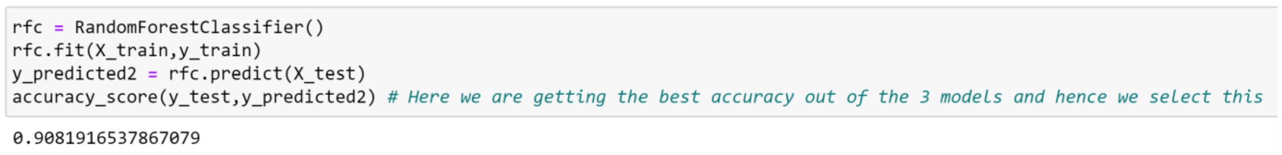
- **Decision Tree Classifier**



```
dtc = DecisionTreeClassifier()
dtc.fit(X_train,y_train)
y_predicted1 = dtc.predict(X_test)
print(y_predicted1)
print(accuracy_score(y_test,y_predicted1)) # Here we are getting better accuracy than the logistic regression
```

```
['Aspen' 'Douglas_fir' 'Krummholz' ... 'Spruce_Fir' 'Douglas_fir'
 'Krummholz']
0.8102009273570324
```

- **Random Forest Classifier**

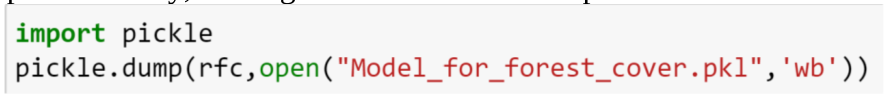


```
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
y_predicted2 = rfc.predict(X_test)
accuracy_score(y_test,y_predicted2) # Here we are getting the best accuracy out of the 3 models and hence we select this
```

```
0.9081916537867079
```

As we can see from the above code snippets, Random Forest Classifier is having the best accuracy and hence is selected for predictions.

6) **Exporting the model using pickle:** The trained Random Forest Classifier model is exported using the 'pickle' library, making it available for future predictions without the need to retrain the model.



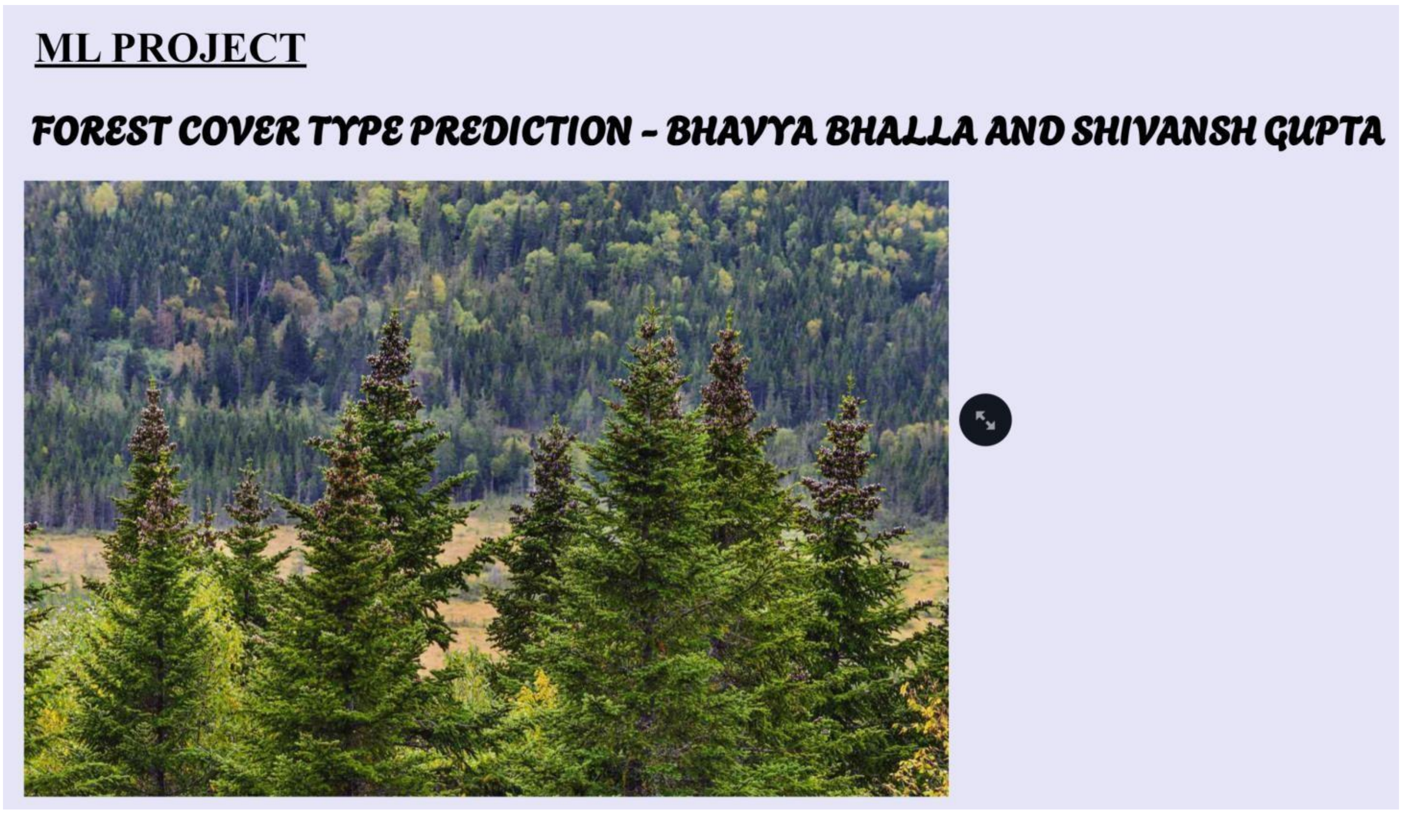
```
import pickle
pickle.dump(rfc,open("Model_for_forest_cover.pkl",'wb'))
```

Model_for_forest_cover.pkl   2 hours ago   93.4 MB

As we can see from the last photo, we have exported the model and now it is ready for use.

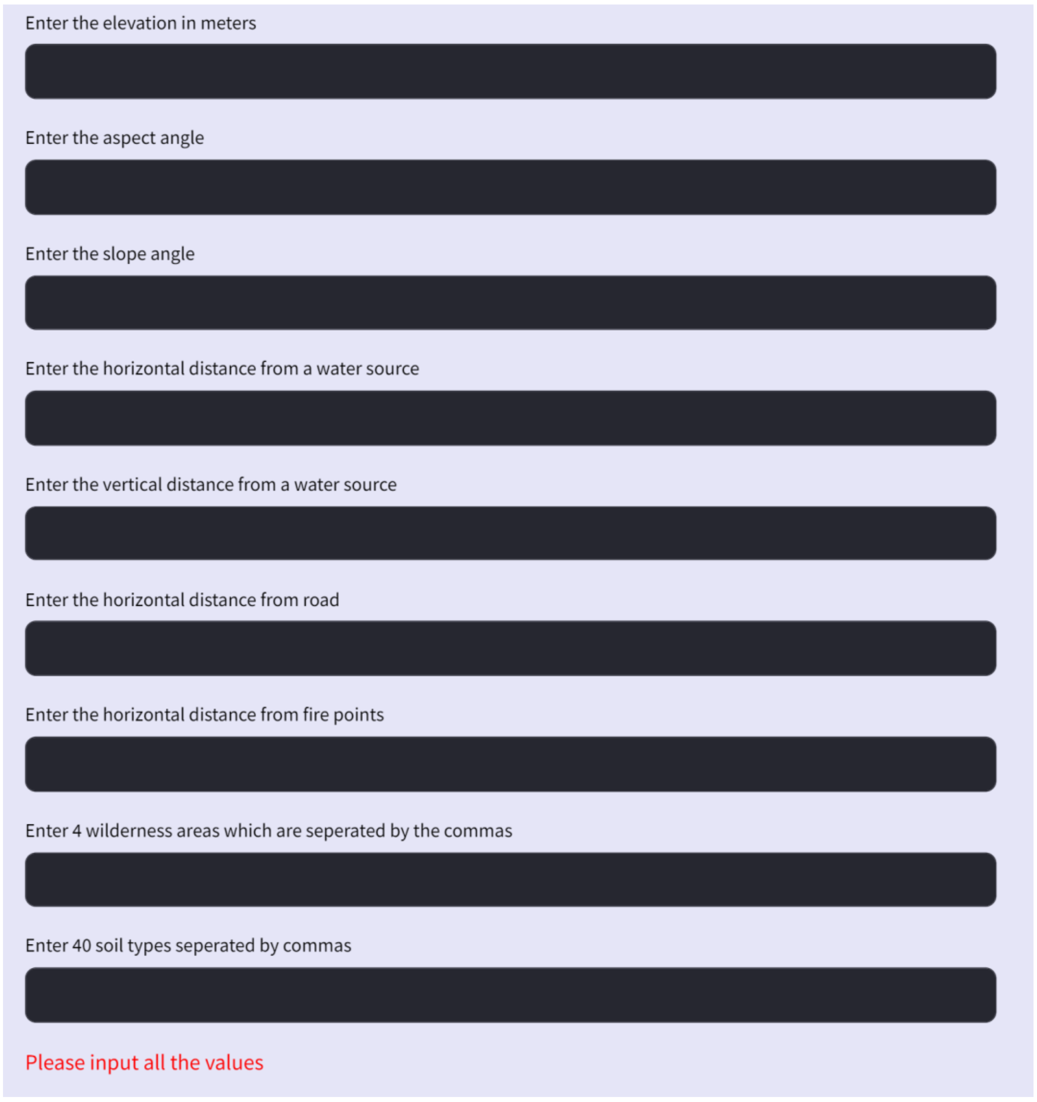
7) **Use of Streamlit for making the UI:** In our project, we leveraged Streamlit to create an intuitive and user-friendly user interface (UI). Streamlit is a powerful Python framework that allowed us to rapidly develop and deploy a web application with minimal effort. With Streamlit, we were able to transform our data and analysis into an interactive and visually appealing interface, making it easy for users to explore and

interact with our project's features and functionalities. Streamlit's simplicity and versatility were key factors in streamlining the UI development process and enhancing the overall user experience.



ML PROJECT

FOREST COVER TYPE PREDICTION - BHAVYA BHALLA AND SHIVANSH GUPTA



Enter the elevation in meters

Enter the aspect angle

Enter the slope angle

Enter the horizontal distance from a water source

Enter the vertical distance from a water source

Enter the horizontal distance from road

Enter the horizontal distance from fire points

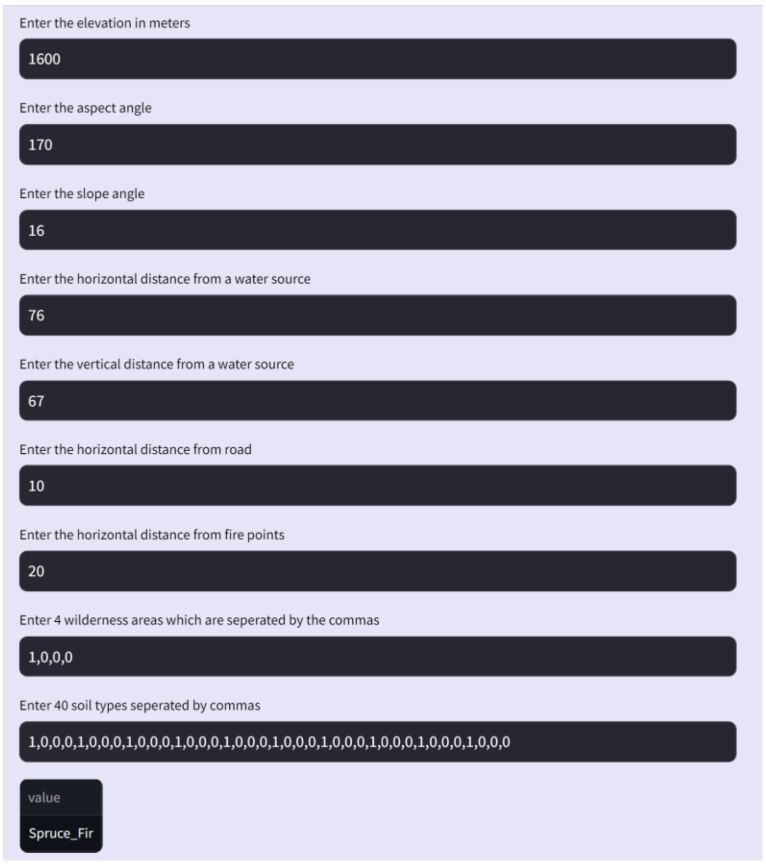
Enter 4 wilderness areas which are seperated by the commas

Enter 40 soil types seperated by commas

Please input all the values

Conclusion

In our project, we employed the Random Forest Classifier to make predictions on our dataset, and the results were quite promising, achieving an accuracy of 90.8%. This high accuracy is a strong indication that the model performed well in classifying and making predictions on the data.



Enter the elevation in meters

1600

Enter the aspect angle

170

Enter the slope angle

16

Enter the horizontal distance from a water source

76

Enter the vertical distance from a water source

67

Enter the horizontal distance from road

10

Enter the horizontal distance from fire points

20

Enter 4 wilderness areas which are seperated by the commas

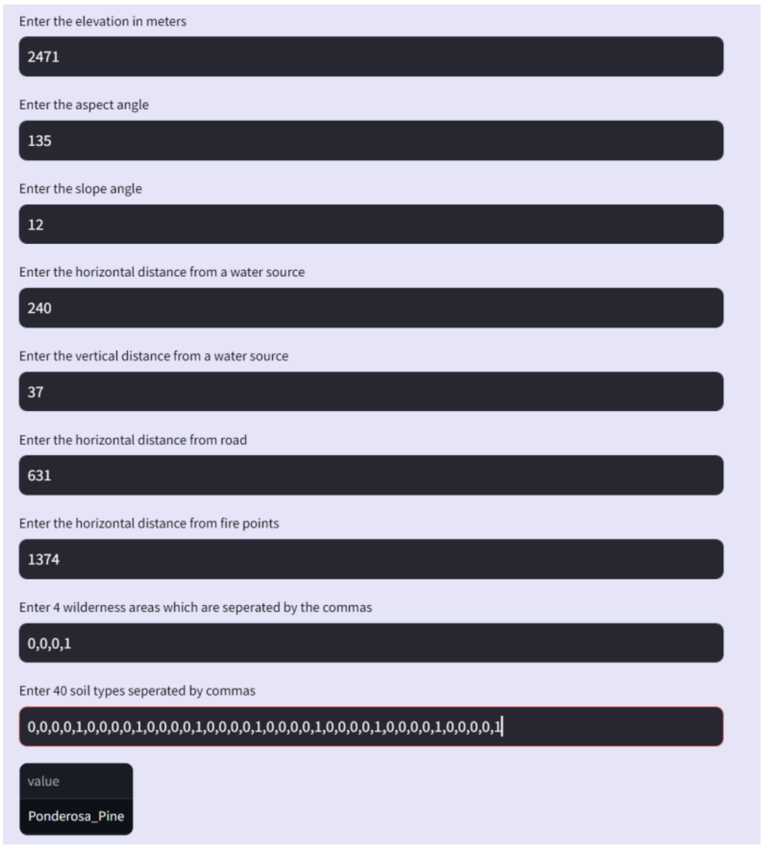
1,0,0,0

Enter 40 soil types seperated by commas

1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0

value

Spruce_Fir



Enter the elevation in meters

2471

Enter the aspect angle

135

Enter the slope angle

12

Enter the horizontal distance from a water source

240

Enter the vertical distance from a water source

37

Enter the horizontal distance from road

631

Enter the horizontal distance from fire points

1374

Enter 4 wilderness areas which are seperated by the commas

0,0,0,1

Enter 40 soil types seperated by commas

0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1

value

Ponderosa_Pine