Forest Fire Risk Classification

CSC 859 AI Ethics and Explainability
San Francisco State University

Fall 2024
Team project: Development and Ethics and Trustworthiness Audit of AI Application

Team 4

<Rosaclaire Baisinger> <rbaisinger@sfsu.edu>
<Zeyu Long > <zlong@sfsu.edu>
<Anushka Mondal> <amondal@sfsu>

December 17, 2024

Table of Contents

1. Executive Summary

This report will use the reported factors of temperature, relative humidity, wind speed, and rainfall to predict two outcomes, on a hectare-by-hectare basis: (1) the likelihood of a fire, and (2) where a fire is more likely than zero, whether the hectare in question has a low, medium, or high fire risk. The goal is to determine whether machine learning is a good way to determine fire risk.

The economic impact of escalating wildfires is also dramatic, as discussed more fully in the next section, both in immediate cost of suppression and in economic losses. To mitigate these harms, communities and private companies must be able to predict the likelihood of fires in their regions.

The rapid increase in forest fires over the last two decades, however, means that past fires are no longer an accurate measure of an area's future risk from wildfires. Monitoring for fire risk can be very expensive, whether the monitoring is done by satellites or human inspection. Weather data, on the other hand, is widely available and can be aggregated in many cases without new infrastructure. This project is important because it allows us to predict fire likelihood based on environmental factors. If we can approximate fire risk as an output of temperature, relative humidity, and wind speed, we will be able to quickly change forecasts when those environmental factors change in the future.

To model this approach, we used a dataset of temperature, relative humidity, wind speed, and rainfall in a region of Portugal, based on a dataset [3] and a data mining model set out by Cortez and Morais in 2017. [1 at pp. 3-4]. First, we audited the database according to the Audit Checklist and industry best practices. The authors used analyzed the data using a regression model. We optimized the data to create a classification model, classifying each hectare as at risk or not at risk for fire. Then, we applied Random Forest to estimate the likelihood of fires in each hectare. Finally, we followed best practices for engineering to assess the results.

Our model, like the regression modeling in the article that established our ground truth, did not produce very high accuracy or confidence. We also uncovered some issues with the underlying data, discussed in more detail in Section 3. We would not recommend our model as a substitute for satellite surveillance or other, more expensive, ways to predict fire risk.

We would, however, recommend this approach for customers that either cannot afford more expensive ways to monitor fire risk, or as a supplement to existing surveillance, especially in remote areas like natural parks. We would also recommend further research to determine whether accuracy is improved when applying our model to a more robust dataset.

2. Motivation, Problem, Description, and Case Study Goals

Forest fires have increased dramatically in the past two decades. In 2023, fires caused nearly 6 million more hectares of tree cover loss worldwide than they did in 2001. [7 at p. 2]. The economic impact of increasing fires is severe. In the United States alone, the national cost of fire suppression rose from $809,499 to $3,166,300,000 during that time period. [10]. Similarly, from 2017 to 2022, the European Union increased its firefighting budget about 35% to €37.8 billion ($42 billion) in 2022. [2, p. 1].

As staggering as these direct costs of fire suppression are, they only tell part of the story of forest fires' economic impact. Fires also, of course, impact agriculture, tourism, and construction in affected areas. [Meier et al., 1.] In Southern Europe from 2011 to 2018, researchers found a consistent negative contemporary effect of wildfires on the annual regional GDP growth rate ranging from 0.11 to 0.18%. [8, p. 17].

For the most severe wildfire years, GDP growth rate decreased approximately 3.3–4.8% using fire numbers and burned area, respectively. [8.] In the United States, the U.S Congress Joint Economic Committee found that climate-exacerbated wildfires cost the United States between $394 to $893 billion each year in economic costs and damages. [5, p. 1].

Environmental Data collection is widespread. It tends to be automatically, continuously collected – and perhaps most important, because it is already being collected, is cost-effective. Cortez and Morais investigated whether

Environmental Data could cheaply predict forest fire risk, using regression models. [1, p.2.] We employed a classification model, using the same data set, to determine whether Environmental Data could be a cost-effective way to monitor fire risk.

3. Data

3.1. How were the features (variables) obtained?

We used a data set collected by and Morais in the article that is our ground truth. The authors reviewed data collected between January 2000 and December 200C3 ("**Study Period**") in Montesinho Natural Park. [1, pp. 3-4].

Montesinho Natural Park covers 74225 hectares, or 286.58 square miles, in the Trás-os-Montes northeast region of Portugal. [9]. Its elevation ranges from 438 meters to a maximum of 1,486 meters. [9]. Montesinho Natural Park has a Mediterranean climate with an average annual temperature range from 8 to 12◦C, supporting a diverse range of flora and fauna. [1, p.3.] It has diverse flora and fauna. [1, p.3].

Cortez and Morais's data came from two sources: (1) the park's fire occurrence data ("**Fire Data**"); and (2) the park's Environmental Data ("**Environmental Data**"). Environmental Data was automatically collected from a weather station in the center of the park every thirty minutes. Fire Data was collected by an inspector on every day that a fire occurred within the park. We can summarize provenance and meaning for Fire Data and Environmental Data as follows:

Data Collected	**Method**	**Collection point**	**Frequency**	**Features**
Environmental Data	Automated collection by a Bragança Polytechnic Institute weather station	One station in the center of the park	Every 30 min	Temperature, wind speed, relative humidity, rain.
Fire Data	Manually collected by fire inspector	Location of 517 separate fires during the Study Period	Per fire, per day	Time, date, spatial location within a 9×9 grid, the type of vegetation involved, the total burned area, FFMC, DMC, DC, and ISI (each acronym defined in 3.1.2 below).

Table 1: How Feature Data Was Obtained

3.2. Meaning of features/variables

Feature	**Meaning**
Temperature	Temperature at weather station in Celsius on day of fire
Wind speed	Wind speed at weather station on day of fire
Relative humidity	Relative humidity at weather station on day of fire
Rain	Rainfall at weather station on the day of fire
X	X axis of spatial location of fire within a 9x9 grid
Y	Y axis of spatial location of fire within a 9x9 grid
Month	Month of fire
Day	Day of fire
Area	Total burned area in hectares
FFMC	FFMC stands for Fine Fuel Moisture Code. It expresses the moisture content at the burned area's surface level, which affects how a fire a starts and spreads. FFMC is a code used in the to calculate the Canadian Fire Weather Index or "FWI", a commonly used system to predict fires based on environmental data.
DMC	DMC stands for Duff Moisture Code. It expresses the moisture content of shallow layers of ground at the fire site. The DMC affects fire intensity.

Feature	Meaning
DC	DC stands for Drought Code. It expresses the moisture content of deep layers of ground at the fire site. Like the DMC, it affects fire intensity.
ISI	ISI stands for Initial Spread Index, a score correlated with fire intensity spread.

Table 2: Meaning of Features

3.2.1. Type of features (numerical, categorical nominal, or ordinal)

The data is ordinal.

3.2.2. List and description of features, formats well-organized

The dataset consists of 12 features and 517 instances, with no missing values. Below is a detailed overview of the features.

Feature Category	Feature Name	Description
Environmental Data	temp	temperature (from 2.2 to 33.3, in Celsius degree)
	wind	wind speed (from 0.4 to 9.4, in km/h)
	RH (Relative humidity)	moisture levels in the air (from 15 to 100, in%)
	rain	precipitation (from 0.0 to 6.4) in mm/m
Fire Weather Index Elements	FFMC (Fine Fuel Moisture Code)	moisture content surface litter and influences ignition and fire spread (from 18.7 to 96.2)
	DMC (Duff Moisture Code)	moisture content of shallow (from 1.1 to 291.3)
	DC (Drought Code)	moisture content of deep organic layer (from 7.9 to 860.6)
	ISI (Initial Spread Index)	a score that correlates with fire velocity spread (from 0.0 to 56.10)
Geospatial Coordinates	X	x-axis spatial coordinate within the Montesinho park map (from 1 to 9)
	Y	y-axis spatial coordinate within the Montesinho park map (from 2 to 9)
Time-based Features	Month	month of the year (from 'jan' to 'dec')

Feature Category	Feature Name	Description
	Day	day of the week (from 'mon' to 'sun')

Table 3: Description of features

3.2.3. Missing values (do they need to be imputed)

Cortez and Morais cleaned the data for us. First, they manually integrated the Environmental Data and the Fire Data into one data set containing 517 entries, one for each day of with a fire. There were 247 samples with a 0 data value for the Area feature, meaning that an area smaller than 1ha/100 = 100m2 was burned. To reduce the skew and improve symmetry, the authors applied the logarithm function y = ln(x + 1) to the Area feature. The transformed variable was the output target of this work. (Cortez and Morais, 5).

3.3. Class labels

Team 4 used Environmental Data and Fire Data to create a classification system.

Original Fire Severity Class (EFFIS Standards)	Area (ha)	Simplified Class
Unburned or Negligible Impact	area= 0	NoRisk
Low Severity	0 < area <= 10	Risk
Moderate Severity	10 < area <= 100	Risk
High Severity	100 < area <= 500	Risk
Extreme Severity	area > 500	Risk

Table 4: Original and Simplified Classes

First, the original five fire severity classes were derived based on the European Forest Fire Information System (EFFIS) standards, given the dataset's origin in the Montesinho Natural Park, Portugal. Next, for improved machine learning model performance, the classes were simplified into two broader categories: (i) **Risk**, which combined Low Severity, Moderate Severity, High Severity, and Extreme Severity classes; and (ii) **NoRisk**, which includes only the Unburned or Negligible Impact class.

3.3.1. Number of samples in each original and simplified class

Class	Number of Samples
Unburned or negligible area, in hectares	244
Low Severity area, in hectares	175
Moderate Severity area, in hectares	83
High Severity area, in hectares	9
Extreme Severity, area in hectares	2
Total	**513**

Table 5: Number of samples in Original Fire Severity Class (EFFIS Standards)

Class	Number of Samples
Risk area, in hectares	269
NoRisk area, in hectares	244
Total	**513**

Table 6: Number of samples in simplified classes

3.3.2. Verification of class labels via ground truth

The ground truth in this study was the total hectares in Montesinho Natural Park that actually burned during the Study Period. This data was collected and recorded by the inspector responsible for the Montesinho fire occurrences. [2, p.3].

3.4. Is the data unbalanced?

Data is considered unbalanced when one class has less than 10 percent of the overall data. Like Cortez and Morais, we had a problem with unbalanced data. As one would expect, and as is certainly good news for the inhabitants of this park, nearly half of the hectares of the park did not burn during the Study Period. This made for an extremely unbalanced dataset when we initially separated the data into five classes.

We solved this problem by simplifying five classes into two classes as explained in the prior section. Our simplified Risk class had approximately 52.4% samples, while No Risk class had approximately 47.6% samples. Accordingly, our classes were not unbalanced.

3.5. Is demography covered in a fair and accurate way?

Neither the Environmental Data nor the Fire Data contain any direct demographic data. There are, however, approximately 9,000 inhabitants living in 92 remote villages within the park. [9]. Villages include Vinhais (the northern half of which is in the park), Chacim, and Freixo de Espada à Cinta. [13]. To the best of our knowledge, the data collected was not affected by the demographics of these villagers.

Geography, on the other hand, was not covered in a fair and accurate way. Environmental Data was only collected from one station in the center of the park, whereas Fire Data was collected in 512 separate locations around the park. Given that the park covers more than 700 square kilometers of varying altitudes, it is unlikely that the Environmental Data was accurate for each location of fire.

3.6. Privacy

Privacy is important to consider when data sets include personal data that can be traced to individuals. [14]. The best practice is to anonymize personal data for machine learning data sets, even though such anonymization is not perfect. [4]. Our data set, however, does not include personal data or private information. Accordingly, our data audit did not require us to anonymize the data.

3.7. Goals of Machine Learning Decisions; risk and impact

As forest fires become more prevalent across the world, it is crucial to monitor risk in a cost-effective way. Wealthy regions monitor fire risk via satellite, but this is prohibitive for much of the world. Weather monitoring, however, is something that is widely collected and would not require additional infrastructure or technology.

The goal of our project was to determine whether we can use existing weather data to cheaply determine where to focus resources on fire prevention. Of course, the risk of forest fires is huge. If a government decided to allocate scarce fire-prevention resources to the wrong place, houses could burn, and people could die. It would not be appropriate to substitute a machine learning decision for human judgment and oversight.

Like most technology, the risk and impact of our model depends on its end use. For example, many regions do not have access to satellite imagery and other expensive means to predict fire risk. Many regions – like Montesinho Natural Park – are sparsely inhabited, which means that a fire may not be noticed immediately. Human judgment is required to decide where to focus prevention efforts. If our Random Forest model can provide an additional data point for humans to use in this risk assessment, it will reduce overall risk to sparsely inhabited or under-resourced regions and have a positive impact.

4. Machine Learning Methods

4.1. Tools and Libraries

In this section, we outline the key libraries and tools utilized in the machine learning pipeline. The following table provides an overview of each library and its specific purpose in the project:

Library	**Purpose**
Pandas	Helps with organizing, cleaning, and manipulating data.
Numpy	Provides tools for numerical calculations and working with arrays.
Matplotlib.pyplot	Used to create basic charts and graphs to visualize the data and results.
Seaborn	Helps create more advanced and attractive visualization to explore data patterns.
RandomForestClassifier	Implements the Random Forest algorithm from scikit for classification tasks
StratifiedKFold	Ensures balanced class distribution during cross-validation by splitting the data into folds with proportional class representation.
train_test_split	Splits the dataset into training and testing subsets, ensuring that the model is evaluated on unseen data.
confusion_matrix, classification_report, accuracy_score	Standard evaluation metrics for assessing classification model performance, including accuracy, precision, recall, and F1-score.
LabelEncoder	Converts categorical variables into numerical labels to prepare the data for machine learning algorithms.

Table 7: Libraries used

4.2. Data Preparation

The data preparation process consists of two main steps: exploration and processing. First, we explored the dataset to understand its structure and distribution. Using panda's describe function, we obtained summary statistics, such as the mean, standard deviation, and range. The info function was also applied to check the data types of each column and to identify any missing values. From this exploration, we found that the distribution of "area" followed a long-tail distribution, which was also consistent with the description in the original paper. This indicates that most of the samples are concentrated in a small range, while a few have significantly larger values.

Next, we examined the dataset for any duplicate entries and removed them to ensure the model's performance was not affected by redundant data. This step is essential for maintaining the quality of the dataset.

In terms of feature processing, we converted categorical variables into numerical format using one-hot encoding. This transformation was applied to the "month" and "day" columns. One-hot encoding creates binary columns for each category, enabling the model to work with these features effectively.

Finally, we transformed the target variable. Originally in a continuous format, the target values were categorized into two classes: "Risk" and "No Risk," based on the severity of the fire. A larger burned area was classified as "Risk," while smaller areas were classified as "No Risk." This binary classification allows the model to predict whether a forest fire is likely to cause significant damage or not.

Our data exploration and preprocessing summary is in the table below:

Method	Exploration	Processing
Describe ()	The original target variable "area" follows a long-tail distribution, which can cause data imbalance when categorized.	We applied binary classification, resulting in 269 samples for "Risk" and 244 samples for "No Risk."
Info ()	Two categorical features were identified, and there were no missing values in the dataset.	One-hot encoding was applied to the categorical features.
Duplicated ()	Four duplicate samples were found in the dataset.	Duplicates were removed to maintain data integrity.

Table 8: Data Exploration and Preprocessing Summary

4.3. Random Forest Methodology

4.3.1. Decision Tree: Foundation of Random Forest

A Decision Tree is a supervised machine learning model that makes predictions by asking a series of questions, based on the features of the data. It splits the dataset into smaller subsets, creating a tree-like structure. Each internal node represents a question, and each leaf node represents an outcome or prediction.

Gini Impurity is commonly used to evaluate the quality of a split. It measures the level of disorder or impurity in a dataset. A split that reduces Gini Impurity is preferred, as it creates purer groups that are more homogeneous in terms of the target variable. The goal is to minimize Gini Impurity with each split, improving the model's decision-making process.

4.3.2. Limitations of Decision Trees

While decision trees are easy to interpret and visualize, they tend to suffer from overfitting. A tree may become too complex by learning patterns that are specific to the training data, leading to poor generalization to new, unseen data. This is where Random Forest provides a significant improvement.

4.3.3. Random Forest: An Ensemble of Decision Trees

Random Forest is an ensemble learning method that combines multiple decision trees to improve model performance. Unlike a single decision tree, Random Forest generates multiple trees that are trained on different random subsets of the data and features, which helps increase the model's robustness.

Each decision tree in the forest is built independently using a random sample of the data (known as bootstrap sampling) and a random subset of features. This randomness ensures that each tree is slightly different from the others, contributing to a diverse set of models that collectively improve the final prediction.

4.3.4. Key Parameters in Random Forest

- ntree (Number of Trees): This parameter determines how many trees the Random Forest will contain. Increasing the number of trees generally improves performance by reducing variance, but after a certain point, the improvement becomes marginal.
- mtry (Number of Features for Each Split): This parameter controls how many features are considered for each split in a decision tree. Randomly selecting a subset of features for each split reduces the correlation between individual trees and increases model diversity, making the overall model more robust.
- OOB (Out-of-Bag) Error: During training, some data points are not selected in the bootstrap sample and are used to test the model. This is called the Out-of-Bag (OOB) sample. The OOB error rate is calculated

by averaging the error across all OOB samples and provides a useful estimate of model performance without needing a separate test set.

- Cutoff threshold: The final prediction in Random Forest can be adjusted by setting a cutoff value. Instead of relying on the majority vote, the cutoff can be set to a higher threshold, such as 70%, for example. This means that for a prediction to be classified as "Risk," at least 70% of the trees must vote "Risk." This gives the model greater control over its sensitivity to different classes.

4.3.5. How Random Forest Combines Trees

Each decision tree in a Random Forest makes its own prediction based on the data it was trained on. To obtain the final prediction, the Random Forest uses majority voting for classification problems. If most trees predict "Risk," for example, then the final prediction is "Risk."

This voting process ensures that the model is less sensitive to noise and outliers, as individual errors from individual trees are likely to be corrected by the majority vote.

4.4. Training and Evaluation

4.4.1. Parameter Selection

The grid search parameters are set for two key factors in Random Forest: n_estimators(ntree) and max_features(mtry).

For n_estimators, we are testing values of 100, 500, 1000, and 1500. Increasing the number of trees generally improves model stability and reduces variance.

For max_features, we are considering three values based on the square root of the total number of features: half of sqrt_features, sqrt_features, and double sqrt_features (up to the total number of features in the dataset). Using a smaller number of features at each split helps reduce overfitting and encourages more diversity among the trees, making the model more robust.

This grid search setup helps find the right combination of trees and feature selection for improved model performance while considering training time.

Parameter	Values Considered
n_estimators	[100, 500, 1000, 1500]
max_features	[max(1, int(0.5 *(sqrt_features)), sqrt_features, min(X_train.shape[1], int(2 * sqrt_features))]

Table 9: Data Exploration and Preprocessing Summary

4.4.2. Cross-Validation and Grid Search

We conducted a hyperparameter search to optimize the performance of our Random Forest model using grid search and Stratified K-Fold Cross-Validation. Stratified K-Fold allowed us to split the data into three folds while maintaining the same class distribution as the original dataset.

For each combination of hyperparameters, we trained the Random Forest model on all the folds. During training, we utilized the Out-of-Bag (OOB) score, a built-in metric in Random Forest, to evaluate the model's performance. The OOB score provided an efficient way to assess accuracy without the need for a separate validation set, allowing us to fully utilize the data.

After calculating the OOB scores for all folds, we averaged them to measure the overall performance of each parameter configuration. By focusing on the mean OOB score, we ensured that the selected hyperparameters offered strong generalization across the data. This approach not only streamlined the process but also provided us with a robust method to identify the best combination of parameters for our model.

4.5. Code Flow

The overall code flow is summarized in the following table:

Step	Description
1	Data audit and preprocessing. Remove duplicate rows to ensure data quality. Apply classification functions to area to create Fire_Risk_Class. One-hot encode the month and day. Encode Fire_Risk_Class into numeric labels.
2	Define classification functions/ classify_severity: Categorizes fire severity based on the burned area. map_classes: Maps severity levels to "Risk" and "NoRisk" (low/no risk).
3	Building Random Forest Model. Define features and target variables. Split data into training and test sets. Set up cross-validation and hyperparameter grid.
4	Grid searching. Perform grid search over hyperparameters to find the best hyperparameter.
5	Train final model on full training data.
6	Generate classification report and calculate accuracy.
7	Model confidence levels for each class.
8	Feature importance analysis.

Table 10: Code Flow

5. Machine Learning Accuracy Results

5.1. Final Model and Result

Using the best parameters identified through our grid search (n_estimators=1000 and max_features=2), we trained the final Random Forest model. The model achieved an accuracy of 63.11% on the test set, with a slightly higher performance for the "No Risk" class (precision: 64%, recall: 69%) compared to the "Risk" class (precision: 62%, recall: 57%).

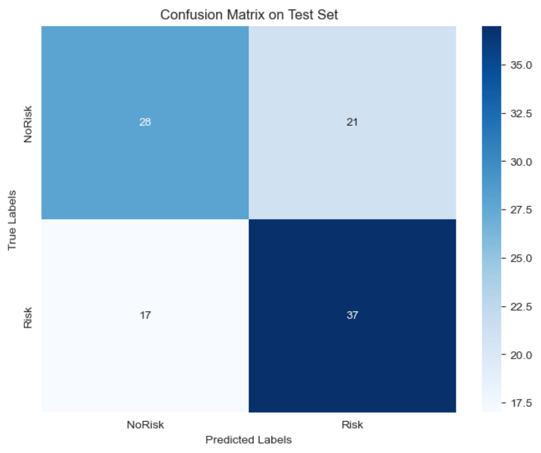


Figure 1: Confusion Matrix on Test Set

The Out-of-Bag (OOB) score for the final model was 54.88%, reflecting its performance on unseen data during training. Additionally, class confidence levels were 75.18% for "No Risk" and 77.94% for "Risk".

Feature importance analysis revealed that factors like temperature, relative humidity, and wind speed were the top 3 influential predictors, while some categorical variables, such as certain months, contributed minimally.

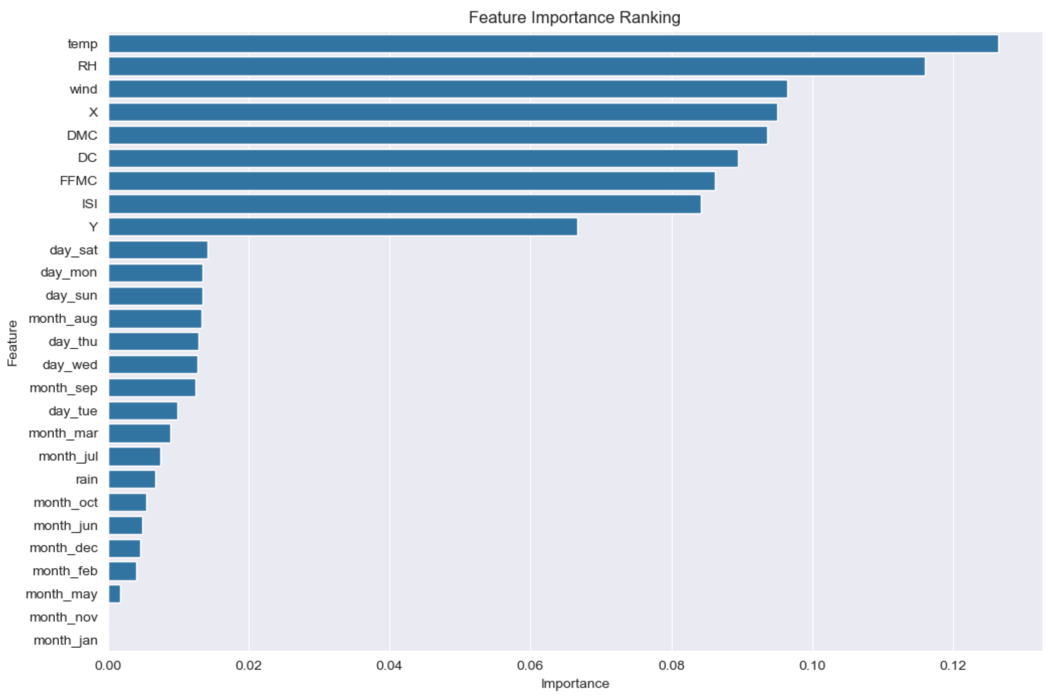


Figure 2: Feature Importance Ranking

5.2. Explainability and Analysis

5.2.1. Confidence Level

The confidence levels for each class were calculated using the average probability scores assigned by the model. In a Random Forest, each tree provides a probability for its prediction. By aggregating these probabilities across all trees, we obtain robust confidence estimates for each prediction. For this model, the average confidence for "Risk" predictions was 52%, while it was 59% for "No Risk." This low confidence level means that our model is not particularly useful. Additionally, since our model has low accuracy, it might be overconfident in borderline cases. We recommend further investigation through an analysis of misclassified samples.

5.2.2. Feature Importance Ranking

Feature importance rankings provide insights into how the model makes decisions. The most significant feature was Temperature, contributing 12.6% to the model's decision-making, which aligns with its strong real-world correlation to fire risk. Relative Humidity (RH) and Wind Speed followed closely, contributing 11.6% and 9.6%, respectively. These factors are critical in fire spread and drying conditions. Spatial and moisture-related variables like the X-coordinate and Duff Moisture Code (DMC) also played substantial roles, indicating that the model integrates both environmental and geographic information effectively.

In contrast, some features like categorical variables for specific months contributed very little or nothing to the model. For example, month_nov and month_jan had no importance, suggesting that seasonal effects were minimal or overshadowed by environmental variables.

6. <u>Audit for Trustworthiness and Ethics</u>

6.1. <u>Trustworthiness</u>

Team 4 audited trustworthiness by evaluating the explainability and reliability of our model. We audited ethics by evaluating the privacy, demographic, risks and impact of the model as discussed in Section 3.

6.1.1. <u>Explainability</u>

There are three issues affecting explainability: (i) applicability of Fire Data to Environmental Data, (i) provenance of Fire Data, and (iii) provenance of combined dataset.

Applicability of Fire Data to Environmental Data. Because Environmental Data was collected by machine, every 30 minutes, for the duration of the Study Period, we consider its provenance to be very trustworthy. Unfortunately, we have no way to verify that the rainfall, humidity, and wind speed in each of 269 burned hectares was identical to conditions recorded at one weather station in the center of this very large park. From what we know about the varying altitude and flora, it seems unlikely. Until we can convince the authorities of Portugal to build more weather stations for us, however, we can assume that the conditions at the weather station are at least comparable to the conditions in the rest of the park.

Provenance of Fire Data. The provenance of the Fire Data is less clear. We know that it was collected by a human "fire inspector." We do not know about fire inspector staffing in the park or anything about his or her collection process, workload, or personal reliability. The FWI and ISI were recorded by this fire inspector, but we do not have a way to calculation of either index.

Provenance of Combined Dataset. Finally, we should note that Cortez and Morais manually aggregated the Fire Data and Environmental Data into the dataset that we used for our model. Cortez and Morais are human and could have made errors in data entry; we are relying on the accuracy of their data entry, as well as the accuracy of their data cleanup discussed above.

6.1.2. <u>Reliability</u>

Our model had low accuracy (approximately 63%), and our confidence level is so low that the benefit of our model does not outweigh the low cost of implementing our model. We do not consider this to be a reliable model.

6.1.3. <u>Ethics</u>

Because the dataset does not identify individual inhabitants or fire inspectors, there are no privacy concerns. We have no information about the demographics of the small population living in Montesinho Natural Park, but we have no reason to believe that these anonymous and neutral facts would be demographically skewed. Accordingly, we conclude that neither privacy nor demography pose an ethical problem with our model.

The impact and risk of our machine learning model, however, must be reviewed carefully because our model is so unreliable. Forest fires pose grave risks to life and property. Any region with a choice between our model and satellite imagine should choose to use satellite imaging. Any region with a choice between our model and constant human surveillance should choose constant human surveillance. It would be very risky to rely solely on our model, and the impact could kill people.

In the real world, however, resources are not infinite. Many regions cannot afford satellite or human surveillance. Many regions cannot afford constant human surveillance. For a cash-strapped government with no existing fire monitoring infrastructure, our model is better than nothing. For regions that <u>do</u> have fire monitoring infrastructure, our model provides another data point. For example, they may benefit from devoting 60% of the budget to Risk Areas. Even a small increase in efficiency could mitigate fire risks for an under-resourced region. Our model can be ethically used in connection with good human judgment.

7. ChatGPT and Related GenAI Usage

7.1. Tool Version Used

We used ChatGPT, GPT-4.0 (OpenAI, Version: December 2023).

7.2. Tasks Where ChatGPT Was Used

7.2.1. Debugging and Fixing Errors

How it Helped. ChatGPT helped resolve a TypeError in the classification_report by identifying the mismatch between numeric and string labels in the target variable. It suggested that we use LabelEncoder to encode both the target variable and model predictions into numeric values, which would be compatible with the classification_report function.

***Example**:* The error was due to y_test containing string labels, while the classification_report function expected numeric labels (0, 1, 2). ChatGPT recommended using LabelEncoder to encode both the true labels and predicted labels.

Helpfulness Rank: HIGH. The guidance provided by ChatGPT directly fixed the error, allowing the classification report to generate correctly.

7.2.2. Code Optimization and Structuring

How it Helped. ChatGPT suggested improvements for code organization by adding headers to each section (e.g., preprocessing, training, evaluation). This improved the clarity of the code and made it easier to follow and debug.

***Example**:* ChatGPT recommended adding headers like:

```
"Add a header to preprocessing: # ===========================================
Purpose: Preprocess Dataset ..."

Resulting Code:

# ===========================================

# Purpose: Preprocess the Dataset

# Developer: [Name]

# Date: [Insert Date]

# ===========================================
```

Helpfulness Rank: MEDIUM. The code became more organized and easier to understand for everyone on the team. It added structure but didn't directly impact functionality.

7.2.3. Clarifying Concepts

***How it Helped*:** ChatGPT provided clarification on Out-of-Bag ("OOB") scoring versus cross-validation, explaining their differences and how to properly use them for model evaluation. This helped the team make better decisions regarding model evaluation techniques.

Example: ChatGPT explained: "OOB score evaluates the model using out-of-bag samples during training, whereas cross-validation tests the model on unseen folds. Both should give similar performance estimates when the model is robust."

***Helpfulness Rank*:** HIGH. ChatGPT's explanation of OOB scoring helped the team understand how to properly use it in the model evaluation phase.

7.3. Main Prompts Used

Here are the main prompts we used for assistance with debugging errors and code structure.

7.3.1. Debugging Errors:

For debugging errors, we used these prompts:

"How do I fix the TypeError in classification_report when using string labels?"

"How can I encode the target variable and predictions to ensure they are compatible with classification_report?"

7.3.2. Code Structuring:

For code structuring, we used the following prompts:

"How can I organize my code to include proper section headers for preprocessing, training, and evaluation?"

Conceptual Clarifications:

"What is the difference between OOB score and cross-validation accuracy?"

"How can I calculate class confidence levels for a Random Forest model?"

7.4. Verification of Tool Output

7.4.1. Testing Tool Recommendations:

Every suggestion was tested directly in the Google Colab environment. For instance, after implementing the encoding suggestion, we ran the model and checked if the classification report was generated correctly without errors.

7.4.2. Cross-Verification:

We cross-checked ChatGPT's suggestions with official documentation from scikit-learn, particularly regarding the use of LabelEncoder and the proper handling of classification_report.

7.4.3. Collaboration with Team:

We discussed the tool's recommendations with the team, ensuring that everyone understood and agreed on the changes made to the code.

7.4.4. Reflection on Tool Usefulness

Overall Helpfulness: HIGH. ChatGPT acted as a valuable assistant throughout the process, helping with debugging, organizing code, and clarifying key concepts. It significantly improved the efficiency of the project.

Lessons Learned:

- Verification is Key: Even though ChatGPT provides useful suggestions, we made sure to verify the output through testing and checking against documentation.
- Collaborative Use: The tool was most effective when used collaboratively. We discussed suggestions with the team before implementing them.

8. Summary and recommendations

We converted categorical variables into numerical format using one-hot encoding and classified each hectare in the park as "Risk" or "No Risk." We conducted a hyperparameter search to optimize the performance of our Random Forest model using grid search and Stratified K-Fold Cross-Validation. Stratified K-Fold allowed us to split the data into three folds while maintaining the same class distribution as the original dataset.

For each combination of hyperparameters, we trained the Random Forest model on all the folds. During training, we utilized the Out-of-Bag (OOB) score, a built-in metric in Random Forest, to evaluate the model's performance. The OOB score provided an efficient way to assess accuracy without the need for a separate validation set, allowing us to fully utilize the data.

We determined that the most significant Environmental Data predictor of fire risk was Temperature, contributing 12.6% to the model's decision-making, which aligns with its strong real-world correlation to fire risk. Relative Humidity (RH) and Wind Speed followed closely, contributing 11.6% and 9.6%, respectively. These factors are critical in fire spread and drying conditions. Spatial and moisture-related variables like the X-coordinate and Duff Moisture Code (DMC) also played substantial roles, indicating that the model integrates both environmental and geographic information effectively.

Our model had low accuracy (approximately 63%), and Team 4 has low confidence (52% for Risk, 59% for No Risk) in the predictions generated by our model. We do not consider this to be a reliable model. Our model should never be used as a substitute or replacement for more expensive and comprehensive ways to monitor fire risk.

Given that the confidence level is so close to 50%, our suspicion is that human judgment outperforms the Random Forest model we developed, even as a starting point for cash-strapped customers with no existing fire monitoring infrastructure.

Finally, we recommend further research to evaluate the effectiveness of our model on a more robust data set, given the issues discussed in our audit of the underlying data.

9. **References**

1) Cortez, Paolo; Morals, Anibel. "A Data Mining Approach to Predict Forest Fires using Meteorological Data." December 1, 2007. Department of Information Systems/R&D Algorithmi Centre, University of Minho, Portugal, http://www3.dsi.uminho.pt/pcortez/fires.pdf

2) Eamon, Farhat and Tugwell, Paul. “Europe’s $42 Billion Effort to Fight Wildfires is an Uphill Battle.” Insurance Summary of data training statistics (# of samples, # of classes, # of features) Journal, August 19, 2024, https://www.insurancejournal.com/news/international/2024/08/19/788859.htm#:~:text=As%20the%20climate%20crisis%20has,of%20an%20intense%20wildfire%20season.

3) Forest Fire Database, https://archive.ics.uci.edu/dataset/162/forest+fires

4) Gadotti, Andrea; Rocher, Luc; Houssiau, Florimond; Creţu Ana-Maria; de Montjoye, Yves-Alexandre (2024). Anonymization: The imperfect science of using data while preserving privacy. Science Advances, 10 (29). DOI:10.1126/sciadv.adn7053.

5) Hailstone, Jamie. “Wildfires Costing The U.S. $89 Billion In Lost Output, Study Finds.” Forbes, August 21, 2024.

6) Institute of Electrical and Electronics Engineers. (2014). Code of Ethics. https://www.ieee.org/about/corporate/governance/p7-8.html.

7) MacCarthy, James, et al. “The Latest Data Confirms: Forest Fires Are Getting Worse.” Insights, World Resources Institute, August 13, 2024, https://www.wri.org/insights/global-trends-forest-fires#:~:text=Using%20data%20from%20researchers%20at,year%20over%20that%20time%20period.

8) Meier, Sarah, et al. “The regional economic impact of wildfires: Evidence from Southern Europe.” Journal of Environmental Economics and Management, Volume 119, May 2023, Pages 102823

9) Monteshino National Park. (2024, November 17. In Wikipedia. https://en.wikipedia.org/w/index.php?title=Montesinho_Natural_Park&oldid=1257870543

10) National Interagency Fire Center, Federal Firefighting Costs (Suppression Only) 1985-2023, https://www.nifc.gov/fire-information/statistics/suppression-costs

11) National Oceanic and Atmospheric Administration, U.S. Department of Commerce, “Wildfire climate connection.” July 24, 2023, https://www.noaa.gov/noaa-wildfire/wildfire-climate-connection.

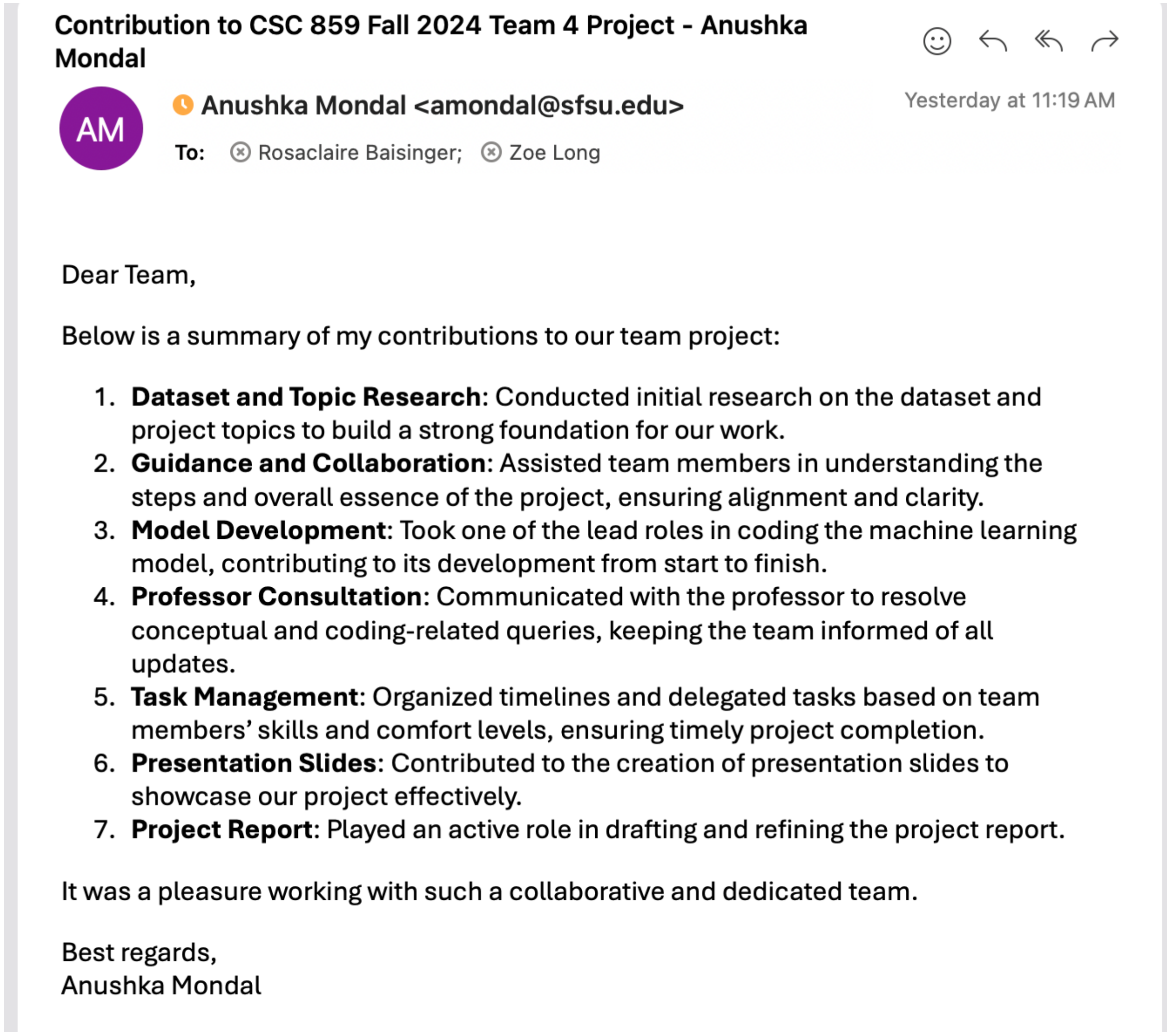
12) United States Environmental Protection Agency, “Climate Change Indicators: Wildfires.” June 2024, https://www.epa.gov/climate-indicators/climate-change-indicators-wildfires

13) Visit Portugal. (n.d.) Braganca. https://www.travel-in-portugal.com/braganca.

14) Yeolib Kim, Seung Hyun Kim, Robert A. Peterson & Jeonghye Choi. (November 2023). Privacy concern and its consequences: A meta-analysis. Technological Forecasting and Social Change, 196 (2023). https://doi.org/10.1016/j.techfore.2023.122789

10. Appendix I: Summary of Each Team Member’s Contributions

10.1. Anushka Mondal



Contribution to CSC 859 Fall 2024 Team 4 Project - Anushka Mondal

AM **Anushka Mondal <amondal@sfsu.edu>** Yesterday at 11:19 AM

To: Rosaclaire Baisinger; Zoe Long

Dear Team,

Below is a summary of my contributions to our team project:

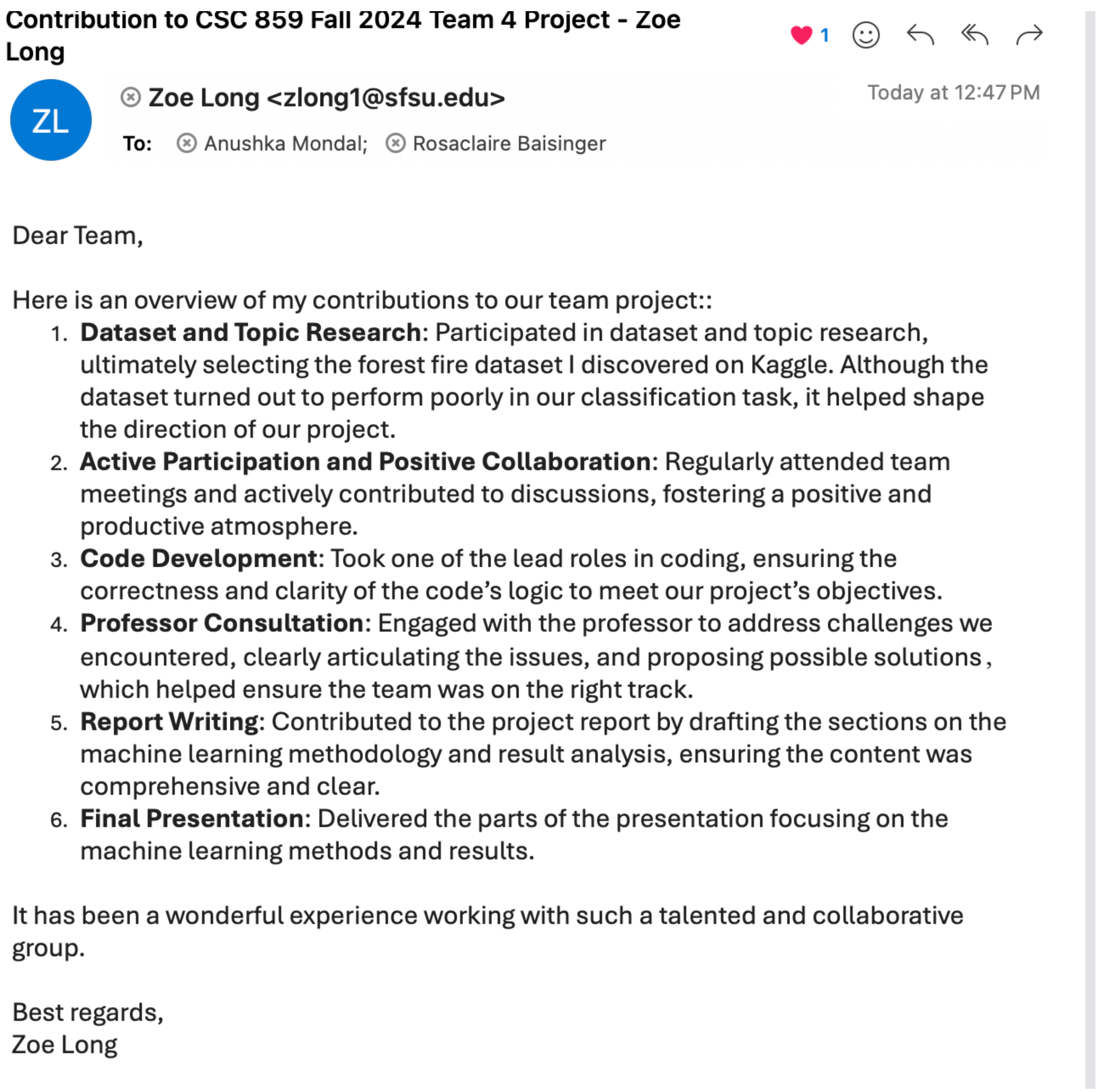
1. **Dataset and Topic Research**: Conducted initial research on the dataset and project topics to build a strong foundation for our work.
2. **Guidance and Collaboration**: Assisted team members in understanding the steps and overall essence of the project, ensuring alignment and clarity.
3. **Model Development**: Took one of the lead roles in coding the machine learning model, contributing to its development from start to finish.
4. **Professor Consultation**: Communicated with the professor to resolve conceptual and coding-related queries, keeping the team informed of all updates.
5. **Task Management**: Organized timelines and delegated tasks based on team members' skills and comfort levels, ensuring timely project completion.
6. **Presentation Slides**: Contributed to the creation of presentation slides to showcase our project effectively.
7. **Project Report**: Played an active role in drafting and refining the project report.

It was a pleasure working with such a collaborative and dedicated team.

Best regards,
Anushka Mondal

Figure 3: Summary of Anushka's contributions

10.2. Zoe Long



Contribution to CSC 859 Fall 2024 Team 4 Project - Zoe Long

1

Zoe Long <zlong1@sfsu.edu>

Today at 12:47 PM

To: Anushka Mondal; Rosaclaire Baisinger

Dear Team,

Here is an overview of my contributions to our team project::

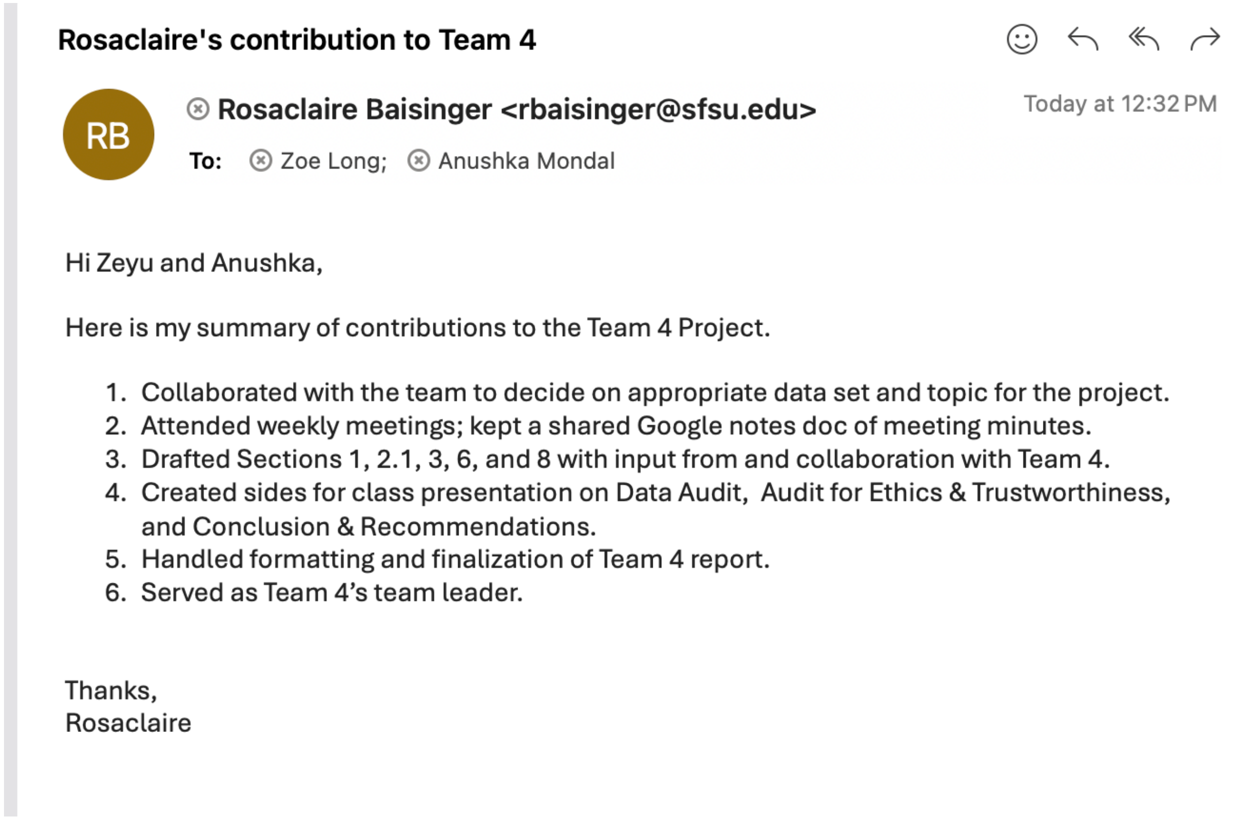
1. **Dataset and Topic Research**: Participated in dataset and topic research, ultimately selecting the forest fire dataset I discovered on Kaggle. Although the dataset turned out to perform poorly in our classification task, it helped shape the direction of our project.
2. **Active Participation and Positive Collaboration**: Regularly attended team meetings and actively contributed to discussions, fostering a positive and productive atmosphere.
3. **Code Development**: Took one of the lead roles in coding, ensuring the correctness and clarity of the code's logic to meet our project's objectives.
4. **Professor Consultation**: Engaged with the professor to address challenges we encountered, clearly articulating the issues, and proposing possible solutions , which helped ensure the team was on the right track.
5. **Report Writing**: Contributed to the project report by drafting the sections on the machine learning methodology and result analysis, ensuring the content was comprehensive and clear.
6. **Final Presentation**: Delivered the parts of the presentation focusing on the machine learning methods and results.

It has been a wonderful experience working with such a talented and collaborative group.

Best regards,
Zoe Long

Figure 4: Summary of Zoe's Contributions

10.3. Rosaclaire Baisinger



Rosaclaire's contribution to Team 4

Rosaclaire Baisinger <rbaisinger@sfsu.edu>
Today at 12:32 PM
To: Zoe Long; Anushka Mondal

Hi Zeyu and Anushka,

Here is my summary of contributions to the Team 4 Project.

1. Collaborated with the team to decide on appropriate data set and topic for the project.
2. Attended weekly meetings; kept a shared Google notes doc of meeting minutes.
3. Drafted Sections 1, 2.1, 3, 6, and 8 with input from and collaboration with Team 4.
4. Created sides for class presentation on Data Audit, Audit for Ethics & Trustworthiness, and Conclusion & Recommendations.
5. Handled formatting and finalization of Team 4 report.
6. Served as Team 4's team leader.

Thanks,
Rosaclaire

Figure 5: Summary of Rosaclaire's Contribution

10.4. Team Lead Summary

10.4.1. Organizational and Teamwork Challenges

Our team worked well together. Zoe and Anushka led the coding efforts, working tirelessly through flawed data and coordinating with our professor and teaching assistant to build the model. Anushka also helped immensely with organization, organizing Zoom calls and keeping us on track assigned tasks and instructions. Rosaclaire helped with research and writing, along with keeping track of ongoing discussions and processes. Rosaclaire, who disappeared during an overwhelming period at work, was the only person to miss a team meeting. Aside from Rosaclaire's delinquency while fighting tooth and nail with Adobe customers over end-of-year contracts, the team faced no organizational or teamwork challenges.

10.4.1. Technical Challenges

First, we had a problem with unbalanced data skewing our results. Most hectares in the Montesinho Natural Park did not burn or had only small fires. The number of large fires was tiny in comparison to small fires. Our original plan was to classify the hectares into ascending categories of risk from no risk to very high risk. Because our data was so skewed, however, this approach did not work. After consulting with each other, with our professor, and with our teaching assistant, we addressed the unbalanced data by reconfiguring our classes into a binary: Risk vs No Risk.

Second, we struggled with the fact that our overall confidence and accuracy was low. After a very helpful few consultations with the professor, we submitted Zoe's code to the teaching assistant for review. There were no errors in her code. Rather, we concluded that the model's accuracy was truly low.

10.4.2. <u>What would we do better next time?</u>

Next time, we would undertake a more rigorous data audit prior to selecting the problem to be solved and the dataset to use. We did review the Forest Fire Data for completeness and common-sense data hygiene, but it was not until digging into the more detailed audit that we realized the original regression model we were starting with also had poor accuracy.

10.4.1. <u>Technical Challenges and Teamwork – Addendum</u>

The Team Leader commends Zoe for additional due diligence, re-checking all calculations after we had worked extensively on the project. She discovered that we had inadvertently calculated confidence based on the entire dataset, rather than on the test set as we had intended. This meant that rather than a 75% confidence rate, we had an average confidence rate of approximately 55%.

At 75% confidence, we had concluded that our model would be a helpful starting point for under-resourced customers developing a fire monitoring strategy from scratch. This would have been a costly mistake, especially since the customers in question would have such limited resources.

Zoe's diligence allowed our Team to quickly recalibrate our recommendations to ensure that we did not recommend a costly and ineffective course of action for our customers.

11. Appendix II: Code used for ML experiments

Import Required Libraries

Developer: Anushka Mondal, Rosaclaire Baisinger, Zeyu Long

Date: 6 November

```
# Import essential libraries for data manipulation, visualization, and modeling
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

Load and Inspect the Dataset

Developer: Anushka Mondal, Rosaclaire Baisinger, Zeyu Long

Date: 6 November

```
# Load the dataset
data_raw = pd.read_csv('forestfires.csv')
```

```
# Inspect the first few rows of the dataset
print("Dataset Head:")
print(data_raw.head())
```

```
Dataset Head:
   X  Y month  day  FFMC   DMC     DC  ISI  temp  RH  wind  rain  area
0  7  5   mar  fri  86.2  26.2   94.3  5.1   8.2  51   6.7   0.0   0.0
1  7  4   oct  tue  90.6  35.4  669.1  6.7  18.0  33   0.9   0.0   0.0
2  7  4   oct  sat  90.6  43.7  686.9  6.7  14.6  33   1.3   0.0   0.0
3  8  6   mar  fri  91.7  33.3   77.5  9.0   8.3  97   4.0   0.2   0.0
4  8  6   mar  sun  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0   0.0
```

```
# Display summary statistics for numerical columns in the dataset
# Includes count, mean, standard deviation, min, and max values for each column
data_raw.describe()
```

	X	Y	FFMC	DMC	DC	ISI	temp	RH
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000 5
mean	4.669246	4.299807	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201
std	2.313778	1.229900	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469
min	1.000000	2.000000	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000
25%	3.000000	4.000000	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000
50%	4.000000	4.000000	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000
75%	7.000000	5.000000	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000
max	9.000000	9.000000	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000

```
# Display a concise summary of the dataset, including column names, non-null counts, and data t
data_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X       517 non-null    int64  
 1   Y       517 non-null    int64  
 2   month   517 non-null    object 
 3   day     517 non-null    object 
 4   FFMC    517 non-null    float64
 5   DMC     517 non-null    float64
 6   DC      517 non-null    float64
 7   ISI     517 non-null    float64
 8   temp    517 non-null    float64
 9   RH      517 non-null    int64  
 10  wind    517 non-null    float64
 11  rain    517 non-null    float64
 12  area    517 non-null    float64
dtypes: float64(8), int64(3), object(2)
memory usage: 52.6+ KB
```

This shows that there are no missing values and all data types are correct

```
# Find duplicate rows based on all columns
duplicate_rows = data_raw[data_raw.duplicated()]


# Print duplicate rows
print(duplicate_rows)
```

```
     X  Y month  day  FFMC    DMC     DC   ISI  temp  RH  wind  rain   area
53   4  3   aug  wed  92.1  111.2  654.1   9.6  20.4  42   4.9   0.0   0.00
100  3  4   aug  sun  91.4  142.4  601.4  10.6  19.8  39   5.4   0.0   0.00
215  4  4   mar  sat  91.7   35.8   80.8   7.8  17.0  27   4.9   0.0  28.66
303  3  6   jun  fri  91.1   94.1  232.1   7.1  19.2  38   4.5   0.0   0.00
```

```
# Count duplicate rows
duplicate_count = duplicate_rows.shape[0]
print(duplicate_count)
```

```
4
```

```
# Drop duplicates permanently
data_raw.drop_duplicates(inplace=True)

print(data_raw)
```

```
     X  Y month  day  FFMC    DMC     DC   ISI  temp  RH  wind  rain   area
0    7  5   mar  fri  86.2   26.2   94.3   5.1   8.2  51   6.7   0.0   0.00
1    7  4   oct  tue  90.6   35.4  669.1   6.7  18.0  33   0.9   0.0   0.00
2    7  4   oct  sat  90.6   43.7  686.9   6.7  14.6  33   1.3   0.0   0.00
3    8  6   mar  fri  91.7   33.3   77.5   9.0   8.3  97   4.0   0.2   0.00
4    8  6   mar  sun  89.3   51.3  102.2   9.6  11.4  99   1.8   0.0   0.00
..  .. ..   ...  ...   ...    ...    ...   ...   ...  ..   ...   ...    ...
512  4  3   aug  sun  81.6   56.7  665.6   1.9  27.8  32   2.7   0.0   6.44
513  2  4   aug  sun  81.6   56.7  665.6   1.9  21.9  71   5.8   0.0  54.29
514  7  4   aug  sun  81.6   56.7  665.6   1.9  21.2  70   6.7   0.0  11.16
515  1  4   aug  sat  94.4  146.0  614.7  11.3  25.6  42   4.0   0.0   0.00
516  6  3   nov  tue  79.5    3.0  106.7   1.1  11.8  31   4.5   0.0   0.00

[513 rows x 13 columns]
```

Preprocess the Dataset

Developer: Anushka Mondal, Rosaclaire Baisinger, Zeyu Long

Date: 27 November

```
# Define severity categories based on burned area
# This step helps in understanding the target variable better
def classify_severity(area):
    if area == 0:
        return 'Unburned or Negligible Impact'
    elif 0 < area <= 10:
        return 'Low Severity (Small Fire)'
    elif 10 < area <= 100:
        return 'Moderate Severity (Moderate Fire)'
    elif 100< area <= 500:
        return 'High Severity (Large Fire)'
    elif area > 500:
        return 'Extreme Severity (Very Large Fire)'
    else:
        return 'Unknown'

# Apply classification
data_raw['Fire_Severity_Class'] = data_raw['area'].apply(classify_severity)


# View the updated data with fire severity classes
data_raw[['area', 'Fire_Severity_Class']].head()
```

	area	Fire_Severity_Class
0	0.0	Unburned or Negligible Impact
1	0.0	Unburned or Negligible Impact
2	0.0	Unburned or Negligible Impact
3	0.0	Unburned or Negligible Impact
4	0.0	Unburned or Negligible Impact

```
# checking the changes in the dastaset columns
data_raw.head()
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	Fire_Severity_Class
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	Unburned or Negligible Impact
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	Unburned or Negligible Impact
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	Unburned or Negligible Impact

Next steps: Generate code with data_raw View recommended plots New interactive sheet

```
# Display the count of samples in each class after classification
counts = data_raw['Fire_Severity_Class'].value_counts()
print(counts)
```

```
Fire_Severity_Class
Unburned or Negligible Impact          244
Low Severity (Small Fire)              175
Moderate Severity (Moderate Fire)       83
High Severity (Large Fire)               9
Extreme Severity (Very Large Fire)       2
Name: count, dtype: int64
```

```
# Reclassify into 2 broader classes (risk, no risk) for modeling
# Consolidating into 2 classes makes it simpler to work with during modeling

# Map the original classes to Risk and NoRisk
def map_classes(area):
    if area == 'Unburned or Negligible Impact':
        return 'NoRisk'
    else:
        return 'Risk'


# combine as risk and norisk
data_raw['Fire_Severity_Class'] = data_raw['Fire_Severity_Class'].apply(map_classes)


# Display the count of samples in each class after classification
counts = data_raw['Fire_Severity_Class'].value_counts()
```

```
print(counts)
```

```
Fire_Severity_Class
Risk      269
NoRisk    244
Name: count, dtype: int64
```

```
# One-hot encode categorical features 'month' and 'day'
# This transforms categorical features into numerical ones for modeling
data_raw = pd.get_dummies(data_raw, columns=['month', 'day'], drop_first=True)

# Encode target variable ('Fire_Severity_Class') into numeric labels
# The LabelEncoder converts 'low', 'medium', 'high' into 0, 1, 2
label_encoder = LabelEncoder()
data_raw['Fire_Severity_Class'] = label_encoder.fit_transform(data_raw['Fire_Severity_Class'])

# Output the encoding mapping to understand how the labels were transformed
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_
print("Label Encoding Mapping:")
for label, encoded_label in label_mapping.items():
    print(f"{label}: {encoded_label}")
```

```
Label Encoding Mapping:
NoRisk: 0
Risk: 1
```

Building Random Forest Model

Developer: Anushka Mondal, Rosaclaire Baisinger, Zeyu Long

Date: 11 December

Define Features and Target Variables

```
# Define features (X) and target (y) for the model
# X is all columns except the target variable and 'area'
y = data_raw['Fire_Severity_Class']
X = data_raw.drop(columns=['Fire_Severity_Class', 'area'])
```

Split Data into Training and Test Sets

```
# Split the data into training (80%) and test (20%) sets
# The stratify option ensures the same distribution of target variable classes in both sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_sta
```

Set Up Cross-Validation and Hyperparameter Grid

```
# Set up Stratified K-Fold Cross-Validation with 3 folds
# Stratified ensures the same proportion of classes in each fold
skf = StratifiedKFold(n_splits=3, shuffle=True, random_state=98)

# Define a grid for hyperparameters - 'n_estimators' and 'max_features'
# We will explore different values for the number of trees and features considered at each spli
n_estimators_grid = [100, 500, 1000, 1500]
sqrt_features = int(np.sqrt(X_train.shape[1]))  # Calculate sqrt of feature count for max_featu
print("features", X_train.shape[1])
print(f"Square Root of Features: {sqrt_features}")
max_features_grid = [max(1, int(0.5 * sqrt_features)), sqrt_features, min(X_train.shape[1], int
print(f"Max Features Grid: {max_features_grid}")

# Track the best parameters and their corresponding OOB score
best_params = None
best_mean_score = 0
```

```
features 27
Square Root of Features: 5
Max Features Grid: [2, 5, 10]
```

Perform Grid Search Over Hyperparameters

```
# Grid search to test different hyperparameter combinations
# OOB score is used to evaluate the model's performance during training
for n_estimators in n_estimators_grid:
    for max_features in max_features_grid:
        oob_scores = []  # List to store OOB scores for this configuration

        # Cross-validation loop to evaluate the model's performance
        for train_index, val_index in skf.split(X_train, y_train):
            X_train_fold, X_val_fold = X_train.iloc[train_index], X_train.iloc[val_index]
            y_train_fold, y_val_fold = y_train.iloc[train_index], y_train.iloc[val_index]

            # Train Random Forest with current hyperparameters
            clf = RandomForestClassifier(
                n_estimators=n_estimators,
                max_features=max_features,
                oob_score=True,
                bootstrap=True,
                random_state=42
            )
            clf.fit(X_train_fold, y_train_fold)

            # Append OOB score for this fold
            oob_scores.append(clf.oob_score_)

        # Average OOB score across all folds
        mean_oob_score = np.mean(oob_scores)

        # Update the best parameters if the current configuration is better
        if mean_oob_score > best_mean_score:
            best_mean_score = mean_oob_score
            best_params = {'n_estimators': n_estimators, 'max_features': max_features}
```

```
# Output the best hyperparameters and their corresponding OOB score
print(f"Best Parameters: {best_params}")
print(f"Best Mean OOB Score: {best_mean_score:.4f}")
```

```
Best Parameters: {'n_estimators': 1000, 'max_features': 2}
Best Mean OOB Score: 0.5402
```

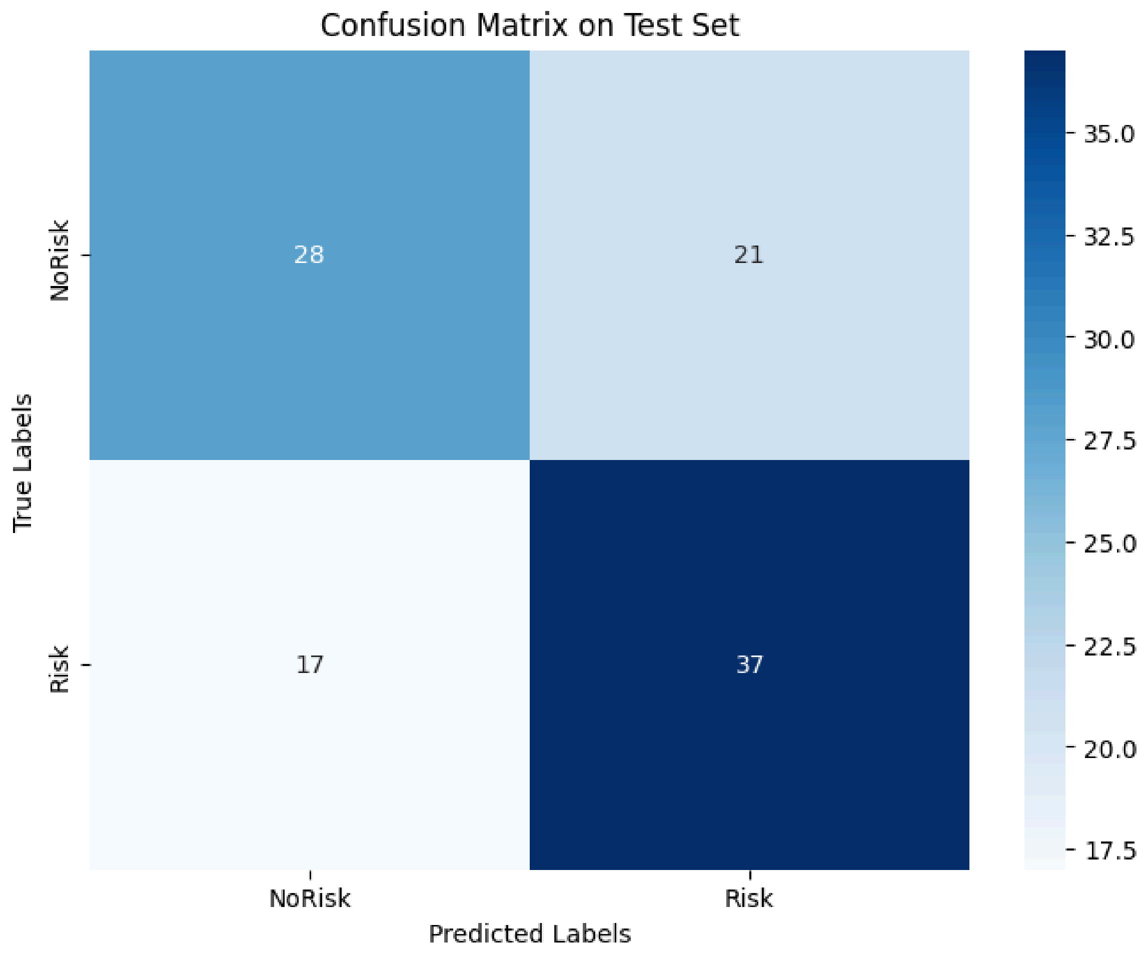
Train Final Model on Full Training Data

```
# Train the final model using the best hyperparameters
final_model = RandomForestClassifier(
    n_estimators=best_params['n_estimators'],
    max_features=best_params['max_features'],
    oob_score=True,
    bootstrap=True,
    random_state=42
)
final_model.fit(X_train, y_train)
```

```
RandomForestClassifier

RandomForestClassifier(max_features=2, n_estimators=1000, oob_score=True,
                       random_state=42)
```

Evaluate the Final Model on the Test Set

```
# Predict the target values for the test set
y_pred_test = final_model.predict(X_test)

# Generate confusion matrix to evaluate model performance
conf_matrix_test = confusion_matrix(y_test, y_pred_test, labels=[0, 1])

# Plot the confusion matrix for better visualization
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_test, annot=True, fmt='g', cmap='Blues', xticklabels=label_encoder.clas
plt.title('Confusion Matrix on Test Set')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



Generate Classification Report and Calculate Accuracy

```
# Define class names for interpretation
class_names = ['NoRisk','Risk']

# Generate and display the classification report for the test set
class_report_test = classification_report(y_test, y_pred_test, target_names=class_names)
print("Classification Report on Test Set:")
print(class_report_test)

# Compute accuracy on the test set
accuracy_test = accuracy_score(y_test, y_pred_test)
print(f"Accuracy on Test Set: {accuracy_test:.4f}")

# Output the final OOB score for the trained model
print(f"Final OOB Score: {final_model.oob_score_:.4f}")
```

```
Classification Report on Test Set:
              precision    recall  f1-score   support

      NoRisk       0.62      0.57      0.60        49
        Risk       0.64      0.69      0.66        54

    accuracy                           0.63       103
   macro avg       0.63      0.63      0.63       103
weighted avg       0.63      0.63      0.63       103
```

```
Accuracy on Test Set: 0.6311
Final OOB Score: 0.5488
```

Model Confidence Levels for Each Class

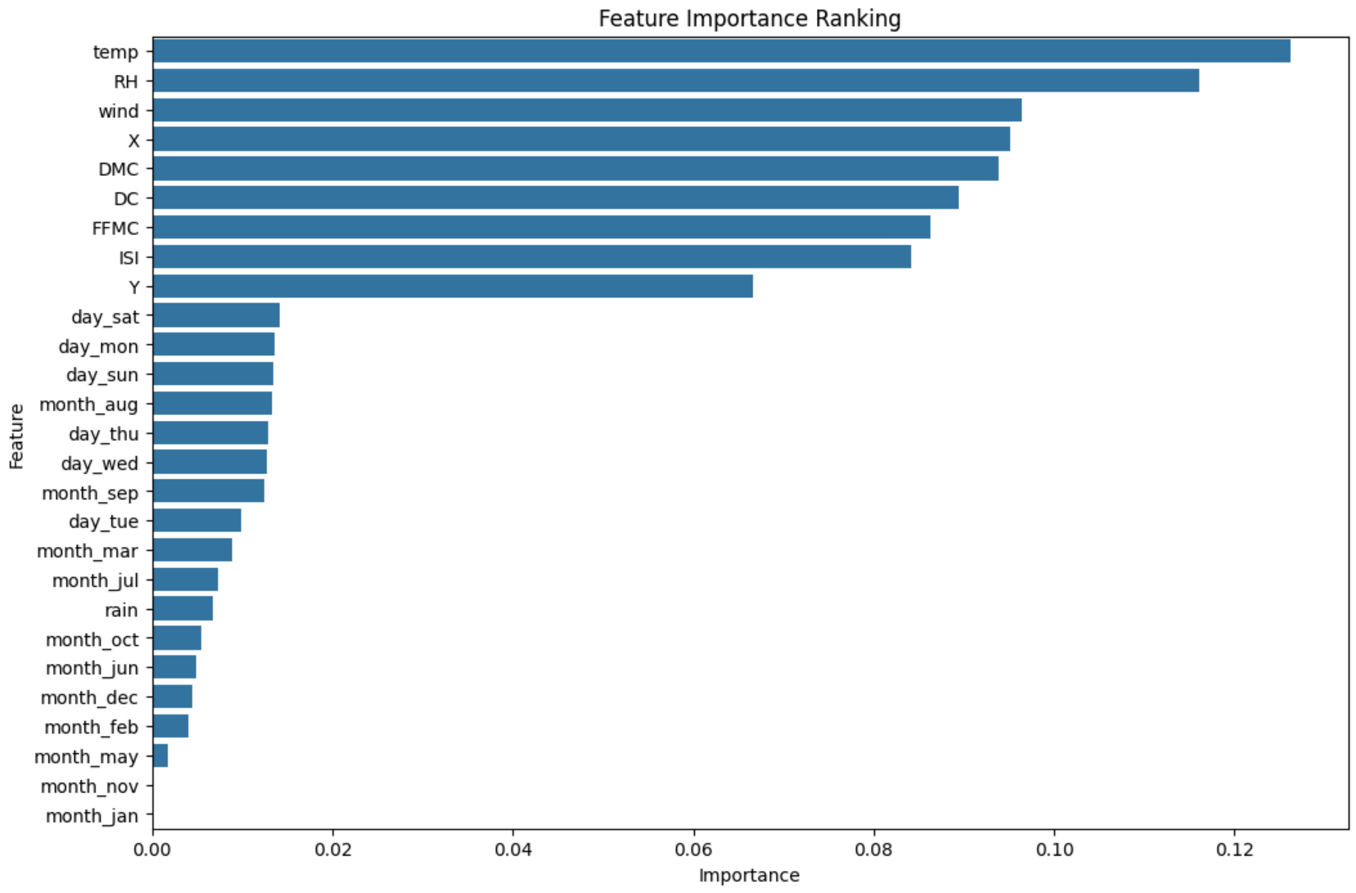
```
# Get predicted probabilities for each class
y_pred_prob = final_model.predict_proba(X_test)

# Initialize list to store class-wise confidence levels
class_confidences = []

# Calculate the class confidence level for each class
for class_label in range(2):  # NoRisk: 0, Risk: 1
    # Get the indices where the true class equals the current class label
    correct_class_indices = np.where(y_test == class_label)[0]

    # Get the probabilities for the correct class
    correct_class_probs = y_pred_prob[correct_class_indices, class_label]

    # Calculate the average of the correct class probabilities for those samples
    class_confidence = np.mean(correct_class_probs) if len(correct_class_probs) > 0 else 0
    class_confidences.append(class_confidence)

# Display the class confidence levels for each class
for i, class_label in enumerate(class_names):
    print(f"Class Confidence for '{class_label}': {class_confidences[i]:.4f}")
```

```
Class Confidence for 'NoRisk': 0.5269
Class Confidence for 'Risk': 0.5918
```

Feature Importance Analysis

```
# Rank features based on their importance according to the model
feature_importances = final_model.feature_importances_
features = X.columns
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})

# Sort features by importance
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Print the feature importance ranking
print("Feature Importance Ranking:")
print(feature_importance_df)

# Plot feature importance to visualize the most influential features
plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance Ranking')
plt.show()
```

```
Feature Importance Ranking:
       Feature  Importance
6         temp    0.126270
7           RH    0.116037
8         wind    0.096402
0            X    0.095188
3          DMC    0.093830
4           DC    0.089425
2         FFMC    0.086237
5          ISI    0.084182
1            Y    0.066632
22     day_sat    0.014154
21     day_mon    0.013539
23     day_sun    0.013392
10   month_aug    0.013251
24     day_thu    0.012896
26     day_wed    0.012673
20   month_sep    0.012426
25     day_tue    0.009859
16   month_mar    0.008848
14   month_jul    0.007371
9         rain    0.006715
19   month_oct    0.005427
15   month_jun    0.004893
11   month_dec    0.004524
12   month_feb    0.004089
17   month_may    0.001741
18   month_nov    0.000000
13   month_jan    0.000000
```



We are considering all the features that have a score greater than 0.9 to be the ones that highly affect the target