# **3. Methodology:**

Every Supervised learning Project has an overall methodology like:

A diagram of a machine learning algorithm

Description automatically generated

Figure 1 Methodology

The Process includes:

* Data Extraction
* Data Cleaning
* Label Encoding
* Handling Imbalanced Data
* Exploratory Data Analysis
* Feature Engineering
* Data Splitting and Scaling
* ML Algorithms
* Results Evaluations  
  Each rectangle is representing a process and the relation between each process is given as process flow(arrow is representing the flow). We will discuss each process in detail in the next sections.

**Data Flow Diagram**

The process flow diagram involves the several processes given in fig 2.

A diagram of a data flow

Description automatically generated

Figure 2 Data Flow Diagram

Explanation

**User:**

Explanation: The process begins with the user, who initiates the entire workflow. The user may be involved in specifying requirements, providing input data, or interacting with the system in many ways.

**Datasets:**

Explanation: The Datasets step involves gathering and preparing the data required for the forest fire risk assessment. This could include historical weather data, simulated forest data, or any other relevant information needed to train and evaluate the machine learning model.

**Preprocessing:**

Explanation: In the Preprocessing step, the collected datasets undergo data cleaning, transformation, and feature engineering. This ensures that the data is in a suitable format for training the machine learning model. Preprocessing may involve handling missing values, scaling features, or encoding categorical variables.

**Model Training:**

Explanation: Model Training is where the machine learning model is developed using the pre-processed datasets. The model learns patterns and relationships from the data, allowing it to make predictions about forest fire risk based on input features.

**Evaluation:**

Explanation: After training, the model's performance is evaluated using a separate dataset not seen during training. Evaluation metrics such as accuracy, precision, recall, or F1 score are used to assess how well the model generalizes to new, unseen data. The Evaluation step is connected to the Evaluation Results for a comprehensive understanding of the model's performance.

**Model Selection:**

Explanation: In Model Selection, different machine learning models or variations of the same model are compared based on their performance during evaluation. The goal is to choose the most effective model for predicting forest fire risk.

**Deployment:**

Explanation: Once a suitable model is selected, it is deployed for use in real-world scenarios. Deployment involves integrating the model into a system or application, making it accessible for generating predictions based on new input data.

**Real World Predictions:**

Explanation: The deployed model is now capable of making real-world predictions. It takes input data relevant to forest conditions, weather, or other parameters and produces predictions about the likelihood of a forest fire occurring.

**End:**

Explanation: The process concludes with the End step, signifying the completion of the workflow. Users can now leverage the forest fire risk assessment system to obtain predictions and insights based on the deployed machine learning model. The interconnected flow ensures a seamless transition between each stage, enhancing the overall effectiveness of the forest fire risk assessment system.

## **3.1 Data Collection:**

In this Project we must use two dataset’s details of the datasets are given below:

### **3.1.1 CD's API Data:**

The dataset obtained from the Copernicus Emergency Management Service provides a comprehensive historical reconstruction of meteorological conditions conducive to the initiation, propagation, and sustainability of fires. This dataset is an integral component of the European Forest Fire Information System (EFFIS) and encompasses fire danger metrics based on models developed in Canada, the United States, and Australia. The fire danger indices within this dataset are calculated using weather forecasts derived from historical simulations provided by the ECMWF ERA5 reanalysis. ERA5, a globally complete and consistent dataset, combines model data with quality-controlled observations, making it a reliable proxy for observed atmospheric conditions. The dataset includes variables such as the Build-Up Index, Burning Index, Danger Rating, Drought Code, Energy Release Component, Fine Fuel Moisture Code, and various others. These variables play crucial roles in assessing fire danger levels and formulating fire danger forecasts for pre-suppression planning.

**Explanation of Variables Related to Weather from CD's API:**

The CD's API dataset comprises various key variables related to weather, each playing a distinctive role in assessing fire danger levels and predicting fire behaviour. Here is a summary of the significant variables:

* Build-Up Index (Dimensionless): A weighted combination of the Duff Moisture Code and Drought Code, indicating the total fuel available for combustion. Often used for pre-suppression planning.
* Burning Index (Dimensionless): Measures the difficulty of controlling a fire, derived from the Spread component and Energy Release component.
* Danger Rating (Dimensionless): Equivalent to the Fire Weather Index (FWI), reduced to 6 classes of danger, providing a harmonized spatial distribution of fire danger levels.
* Drought Code (Dimensionless): Indicates moisture content in deep compact organic layers, representing a fuel layer at 10-20 cm deep.
* Drought Factor: Represents fuel availability, given as a number between 0 and 10, influenced by recent temperatures and rainfall events.
* Duff Moisture Code (Dimensionless): Indicates moisture content in loosely compacted organic layers of moderate depth, representative of the duff layer that is 5-10 cm deep.
* Energy Release Component (J/m2): Reflects the potential heat release per unit area in the flaming zone, providing guidance on fire intensity.
* Fine Fuel Moisture Code (Dimensionless): Indicates moisture content in litter and other cured fine fuels, with a scale ranging from 0-99.
* Fire Daily Severity Index (Dimensionless): Numeric rating of the difficulty of controlling fires, exponentially increasing as the Fire Weather Index rises.
* Fire Danger Index (Dimensionless): Metric related to the chances of a fire starting, its rate of spread, intensity, and difficulty of suppression.
* Fire Weather Index (Dimensionless): A combination of Initial Spread Index and Build-Up Index, indicating the potential frontal fire intensity.
* Ignition Component (%): Measures the probability a firebrand will require suppression action, expressed as a probability on a scale of 0 to 100.
* Initial Spread Index (Dimensionless): Combines the Fine Fuel Moisture Code and wind speed to indicate the expected rate of fire spread.
* Keetch-Byram Drought Index (Dimensionless): Represents cumulative moisture deficiency in deep duff and upper soil layers, providing insight into flammability.
* Spread Component (Dimensionless): Measures the theoretical ideal rate of fire spread, expressed as a dimensionless variable.

### **3.1.2 Simulated CCF Forest Data Variables:**

The simulated Continuous Cover Forestry (CCF) forest data provides essential variables that contribute to assessing the overall fire risk in Ireland. Each variable represents distinct aspects of the forest environment and conditions, collectively offering valuable insights for understanding and predicting fire behaviour. Here's a summary of the significant variables:

* Overall Fire Risk (Numeric): Represents the comprehensive assessment of fire risk in the simulated CCF forest, derived from a combination of various contributing factors.
* Fine Fuel Moisture (Numeric): Indicates the moisture content in litter and other cured fine fuels, influencing the flammability of surface-level vegetation.
* Initial Spread Index (Numeric): Reflects the expected rate of fire spread based on the combination of fine fuel moisture and wind speed in the simulated CCF forest.
* Uneven-Aged Canopy (Binary): Indicates the presence or absence of an uneven-aged canopy structure, influencing the spatial distribution of vegetation and potential fire spread.
* Species Diversity (Numeric): Represents the variety of tree species within the simulated CCF forest, influencing overall ecosystem resilience and fire dynamics.
* Continuous Canopy Cover (Binary): Indicates whether the forest has continuous canopy cover, impacting fuel continuity and potential fire spread.
* Drought Conditions (Numeric): Reflects the simulated moisture deficiency in the forest, providing insights into the potential flammability of organic material in the ground.
* Wind Speed (Numeric): Represents the simulated wind speed within the forest environment, a critical factor influencing fire behaviour and spread.
* Temperature (Numeric): Indicates the simulated temperature conditions within the CCF forest, influencing overall fuel moisture and fire risk.
* Fire Warnings (Binary): Indicates whether there are simulated fire warnings in the CCF forest, providing information on potential fire risk events.
* Fire Occurrence (Binary): Indicates the occurrence or absence of simulated fires in the CCF forest, serving as a crucial outcome variable for assessing the accuracy of fire risk models.

These variables collectively contribute to a comprehensive understanding of fire risk dynamics in the simulated CCF forest. The combination of meteorological data from CD's API and these simulated forest-specific variables forms a robust foundation for developing accurate and effective machine learning models for forest fire risk assessment in Ireland.

## **3.2 Data Cleaning:**

Data cleaning is a critical process in the data preparation phase that focuses on enhancing the quality, accuracy, and reliability of a dataset. It involves various techniques to identify and rectify issues such as missing values, outliers, inconsistencies, and irrelevant information.  
Following techniques are applied for Data Cleaning:

**Handling Missing Values:**

Technique: Imputation or removal of missing values.

Description: Fill missing values using methods such as mean, median, or regression for numerical variables. Alternatively, remove records or variables with missing values.

**Outlier Detection and Treatment:**

Technique: Statistical methods (Z-score, IQR) for identifying outliers, and removal or transformation.

Description: Identify outliers using statistical measures and address them by either removing them or transforming them to minimize their impact.

**Consistency Check:**

Technique: Comparison of values against predefined rules or standards.

Description: Ensure consistency across variables by validating values against predefined criteria or rules.

**Normalization/Scaling:**

Technique: Min-Max scaling or Z-score normalization for numerical variables.

Description: Normalize numerical features to a similar scale, preventing features with larger magnitudes from dominating the analysis.

**Validation of Binary Variables:**

Technique: Validation to ensure binary variables only contain valid values (0 or 1).

Description: Confirm that binary variables adhere to the expected format, containing only valid values.

**Data Integration:**

Technique: Integration of datasets based on common identifiers (e.g., time periods, geographic locations).

Description: Combine datasets with common identifiers, creating a unified dataset for analysis.

**Data Type Check:**

Technique: Inspection and correction of data types for each variable.

Description: Confirm that variables are appropriately represented as numerical or categorical types.

**Data Duplicates:**

Technique: Identification and removal of duplicate records.

Description: Detect and remove duplicate records to eliminate redundancy in the dataset.

**Data Format Standardization:**

Technique: Standardizing date formats and other data representations.

Description: Ensure consistent data formats, such as date formats, to maintain uniformity across the dataset.

**Handling Irrelevant Variables:**

Technique: Removal of irrelevant or redundant variables.

Description: Eliminate variables that do not contribute significantly to the analysis, streamlining the dataset.

## **3.3 Label Encoding:**

In the label encoding section, the 'OverallFireRisk' column has undergone the following mapping:

Mapping for 'OverallFireRisk' column:

Extreme: 0

High: 1

Low: 2

Moderate: 3

Very Low: 4

This label encoding technique transforms categorical descriptors of fire danger levels into unique numerical labels. The assigned numeric values, ranging from 0 to 4, allow machine learning algorithms to comprehend and analyse the fire risk data effectively. This process enhances the readiness of the data for subsequent stages in the forest fire risk assessment, as machine learning models often require numerical representations for accurate training and prediction. Label encoding provides a streamlined approach to convert qualitative information into a format compatible with machine learning algorithms, contributing to the overall success of the forest fire risk assessment endeavour.

## **3.4 Handling Imbalanced Data:**

Dealing with Imbalanced Data in CCF Forest Fire Risk Assessment Model:

**Background:**

Imbalanced data occurs when certain classes in the target variable are underrepresented. In the context of CCF forest fire risk assessment, the target variable is 'OverallFireRisk,' which encompasses various levels of fire risk.

**Imbalanced Class Distribution:**

'OverallFireRisk' exhibits an imbalanced distribution, where specific fire risk levels may have fewer instances compared to others. This imbalance can lead to challenges in model training, with the risk of the model being biased towards the majority class.

**Significance of Imbalanced Data Handling:**

Addressing the imbalanced class distribution is crucial to ensure that the machine learning model can effectively learn patterns and make accurate predictions for all fire risk levels. This is particularly important in scenarios where accurate identification of minority classes is essential for decision-making.

**Target Variable Overview:**

'OverallFireRisk' consists of categories representing diverse fire risk levels, such as 'Extreme,' 'High,' 'Moderate,' 'Low,' and 'Very Low.' Each category signifies a specific degree of fire risk associated with CCF forests.

**RandomOverSampler Technique:**

To tackle the imbalanced class distribution, the code incorporates the RandomOverSampler technique from the imbalanced-learn library. RandomOverSampler is an oversampling method designed to address the scarcity of instances in the minority class.

**Mechanism of RandomOverSampler:**

RandomOverSampler works by randomly duplicating instances of the minority class until a more balanced distribution is achieved. This process involves creating synthetic samples for the less frequent class, mitigating the impact of class imbalance during model training.

**Prevention of Biased Model Training:**

The primary goal of using RandomOverSampler is to prevent machine learning models from disproportionately favoring the majority class during training. This helps in creating a more equitable learning environment, ensuring that the model considers all fire risk levels.

**Enhanced Generalization:**

The application of RandomOverSampler enhances the model's ability to generalize across all classes of fire risk. This is crucial for achieving a well-rounded and unbiased prediction capability, especially when dealing with imbalanced datasets.

**Contribution to Improved Model Performance:**

The resulting balanced dataset, generated through RandomOverSampler, significantly contributes to improved model performance. It enables the model to provide more accurate predictions for all fire risk levels, thereby enhancing its overall reliability and effectiveness.

**Practical Relevance:**

In real-world scenarios, where imbalanced classes could have profound consequences, the use of RandomOverSampler ensures that the developed CCF forest fire risk assessment model is robust and capable of handling varying frequencies within the 'OverallFireRisk' variable.

## **3.5 Exploratory Data Analysis:**

Exploratory Data Analysis is a crucial phase in the data science pipeline, offering insights into the dataset's characteristics, relationships between variables, and patterns that can guide subsequent modeling decisions. In this context, the EDA process focuses on understanding the features and their impact on the target variable 'OverallFireRisk.'

### **3.5.1 Scatter Plots for Feature Analysis:**

Scatter plots are employed to visually inspect the relationship between various features and the target variable 'OverallFireRisk.' For each feature, a scatter plot is generated, with the x-axis representing the feature's values and the y-axis denoting the corresponding 'OverallFireRisk' levels. By examining these plots, trends, patterns, and potential outliers can be identified. This aids in understanding how individual features contribute to the variability in fire risk levels.

## 3.5.2 Visualization of Categorical Features:

* Bar Graphs:

Categorical features, such as 'drtcode' and 'FireWarnings,' are visually explored using bar graphs. Bar graphs provide a clear representation of the distribution of 'OverallFireRisk' across distinct categories within each categorical feature. This enables the identification of any significant variations in fire risk levels associated with specific categories. See fig 3 & 4.

A bar graph with blue and black lines

Description automatically generated

Figure 3 Bar Graph

A bar graph with blue squares

Description automatically generated

Figure 4 Bar Graph

* Line Graphs:

Line graphs are utilized to illustrate the impact of categorical features on 'OverallFireRisk' over a continuous range. This visualization method helps uncover trends or patterns in fire risk variation concerning the values of categorical features. See fig 5.

A barcode with text on it

Description automatically generated

Figure 5 Line Graph

## **3.6 Feature Engineering:**

In the feature engineering phase of this project, strategic decisions were made to enhance the quality and relevance of the dataset, 'balanced\_data,' for subsequent machine learning model training. The initial step involved a comprehensive analysis of the dataset's correlation matrix, utilizing the seaborn library to create informative heatmaps. This analysis aimed to identify inter-feature relationships and dependencies, guiding the selection of features for further refinement.

Upon assessing the correlation matrix, a judicious approach to feature selection was employed. Several features were deemed redundant or exhibited high correlation with others, potentially introducing multicollinearity issues. To address this, specific features were identified for removal, including 'surface,' 'FireWarnings,' 'ffmcode,' 'fdsrte,' 'dufmcode,' 'fwinx,' 'Unnamed: 0,' 'time,' 'fdimrk,' 'drtcode,' 'FireOccurrence,' and 'fbupinx.' The removal of these features was executed to streamline the dataset, eliminating redundancies, and reducing the dimensionality of the feature space.

Following the feature selection and removal process, an updated correlation matrix was generated and visualized using a heatmap. This provided a visual representation of the refined feature relationships within the modified dataset. The correlation matrix showcased the impact of feature engineering on mitigating multicollinearity and optimizing the dataset for the subsequent machine learning tasks.

The culmination of these feature engineering efforts ensures that the 'balanced\_data' dataset is not only more computationally efficient but also poised to contribute meaningfully to the training of machine learning models for fire risk assessment. The judicious selection and refinement of features lay a solid foundation for improved model interpretability, generalization, and predictive accuracy.

A screenshot of a data analysis

Description automatically generated

Figure 6 Correlation matrix

A screenshot of a computer screen

Description automatically generated

Figure 7 Updated Correlation matrix

## **3.7 Data Scaling and Splitting:**

Data scaling and splitting are essential preprocessing steps in preparing the dataset for machine learning model training and evaluation. This process ensures that the features are on a similar scale, preventing certain features from dominating others and facilitating better convergence during model training. Additionally, splitting the dataset into training and testing sets allows for an unbiased evaluation of the model's performance.

### **3.7.1 Standard Scaling with StandardScaler:**

Standard scaling, or z-score normalization, is employed to bring numeric features to a standard scale with a mean of 0 and a standard deviation of 1. This is crucial when features have different units or scales, ensuring that each feature contributes equally to model training. The StandardScaler from the scikit-learn library is applied to achieve this normalization.

**Detailed Steps:**

* Import the StandardScaler from scikit-learn's preprocessing module.
* Identify the numeric features in the dataset that require scaling.
* Initialize an instance of the StandardScaler.
* Fit the scaler on the training data to compute the mean and standard deviation.
* Transform both the training and testing sets using the computed mean and standard deviation.

The scaling process ensures that the features are centred around zero, providing a standardized input for the machine learning models. This step is particularly important for algorithms sensitive to the scale of features, such as K-Nearest Neighbors or Support Vector Machines.

### 3.7.2 Dataset Splitting into Training and Testing Sets:

The dataset is divided into two subsets: the training set and the testing set. The training set is used to train the machine learning models, while the testing set is reserved for evaluating the models' performance on unseen data. The train\_test\_split function from scikit-learn is commonly used for this purpose.

**Detailed Steps:**

* Import the train\_test\_split function from scikit-learn's model\_selection module.
* Identify the features (X) and the target variable (Z) in the dataset.
* Specify the test\_size parameter to determine the proportion of data allocated to the testing set (commonly set to 0.2 for an 80-20 split).
* Set the random\_state parameter for reproducibility of results.
* Execute the train\_test\_split function to obtain the training and testing sets for both features and the target variable.

The resulting sets, namely X\_train, X\_test, Z\_train, and Z\_test, are ready for use in training and evaluating machine learning models. This separation ensures that the model is assessed on data it has never seen during the training phase, providing a reliable estimate of its generalization performance.

By meticulously performing data scaling and splitting, the code establishes a robust foundation for subsequent model training, enhancing the models' ability to generalize to new, unseen data and improving the overall reliability of the machine learning pipeline.

## **3.8 Tools Used:**

* **CDS API**: Used for Scrapping the data.
* **Language:** Python
* **Pandas:** Used for data manipulation and reading CSV files.
* **Scikit-learn:** Employed for label encoding, data scaling, dataset splitting, and implementing machine learning algorithms.
* **Imbalanced-learn:** Utilized for addressing class imbalance using RandomOverSampler.
* **Matplotlib and seaborn:** Used for data visualization, including scatter plots, bar graphs, line graphs, and heatmaps.
* **GridSearchCV:** Applied for hyperparameter tuning of machine learning models.
* **Front-end Development:** HTML CSS
* **Back-end development:** Python Django

# **4 Implementation**

## **4.1 KNN Classification**

In KNN classification, the problem is formulated as a classification task aiming to predict the overall fire risk. K-Nearest Neighbors (KNN) is employed as a classification algorithm. The model calculates the distances between data points and predicts the target variable based on many k-nearest neighbors. This method is suitable for capturing local patterns and relationships within the dataset.

### **4.1.1 Explanation:**

The application of the K-Nearest Neighbors (KNN) algorithm to the fire risk assessment dataset proved to be a strategic choice, emphasizing proximity-based classification. The initial step involved an exploration of hyperparameter tuning, leveraging GridSearchCV to systematically evaluate various combinations of hyperparameters. The parameters considered included the number of neighbors ('n\_neighbors'), the weighting strategy ('weights'), and the distance metric ('p'). This meticulous tuning aimed to identify the optimal configuration that maximizes the model's predictive accuracy.

Hyperparameter tuning is crucial as it helps fine-tune the algorithm's behavior, optimizing its performance on the specific dataset. The selection of the best hyperparameters, as determined by GridSearchCV, ensures that the KNN model is tailored to the unique characteristics of the fire risk assessment data.

To assess the model's generalization capabilities and mitigate overfitting concerns, a robust cross-validation strategy was employed. The cross\_val\_score function executed a 5-fold cross-validation, which involves splitting the dataset into five subsets, training the model on four subsets, and evaluating it on the fifth. This process was repeated five times, with each subset serving as the validation set exactly once. Cross-validation provides a more comprehensive evaluation of the model's performance across different subsets of the data, offering insights into its stability and consistency.

The hyperparameter-tuned KNN classifier demonstrated exceptional performance on the test set, achieving an accuracy of 100%. The precision, recall, and F1-score metrics, as presented in the classification report, all reached perfect scores, underscoring the model's ability to precisely predict fire risk categories. This outstanding performance validates the effectiveness of hyperparameter tuning and cross-validation, highlighting their significance in ensuring a well-optimized and robust KNN model for fire risk assessment.

### **4.1.2 Parameters Used**

Optimal number of neighbors are 3, 5, 7,9. The number of neighbors used in KNN is determined through hyperparameter tuning to find the optimal value that maximizes the model's accuracy. This process involves trying different values for the number of neighbors and evaluating the model's performance with each setting. The justification for selecting a specific number of neighbors is to strike a balance between model complexity and generalization.

* Fewer Neighbors (e.g., 3 or 5): Using a small number of neighbors can make the model sensitive to noise or outliers in the data. While it may capture local patterns well, it might also be influenced by individual data points, leading to overfitting.
* More Neighbors (e.g., 7 or 9): On the other hand, using a larger number of neighbors can lead to a smoother decision boundary, potentially improving the model's ability to generalize to unseen data. However, it might overlook local patterns and details in the data.

The optimal number of neighbors is often chosen based on cross-validated performance metrics, such as accuracy, to ensure that the model performs well on new, unseen data. The goal is to find a balance that minimizes both bias and variance, resulting in a KNN model that generalizes effectively to different instances in the dataset. The choice of the number of neighbors is data-dependent and may vary based on the characteristics of the dataset and the underlying patterns in the data.

## **4.2 X\_Gradient Boosting:**

Gradient Boosting is applied using the XGBoost algorithm, treating the task as a classification problem. XGBoost is an ensemble technique that combines weak learners to build a robust model. It iteratively builds decision trees and optimizes for both bias and variance, resulting in improved accuracy. The algorithm is trained to classify the fire risk into distinct categories based on the features provided.

### **4.2.1 Explanation:**

The application of XGBoost, an advanced implementation of gradient boosting, to the fire risk assessment dataset signifies a sophisticated modeling approach, leveraging the strengths of an ensemble technique. The process commenced with an intricate exploration of hyperparameter tuning, employing GridSearchCV to systematically evaluate a range of hyperparameter combinations. Key parameters under consideration included the number of trees ('n\_estimators'), maximum depth of the trees ('max\_depth'), learning rate ('learning\_rate'), and subsample ratio of training instances ('subsample'). This exhaustive search aimed to identify the most effective hyperparameter configuration for optimizing the model's accuracy.

### **4.2.2 Parameters:**

Hyperparameter tuning is imperative in the context of XGBoost due to its sensitivity to parameter values, and optimal settings can significantly impact the model's performance. The selected hyperparameters, derived from GridSearchCV, ensure the XGBoost model is tailored to the unique characteristics of the fire risk assessment dataset.

To comprehensively assess the model's generalization performance and robustness, a cross-validation strategy was implemented. Cross-validation, particularly the 5-fold approach in this case, provides a more nuanced evaluation of the model's consistency across various subsets of the data. This approach enhances the model's reliability and ensures it can generalize well to unseen data.

The hyperparameter-tuned XGBoost classifier demonstrated robust performance on the test set, achieving an accuracy of approximately 93.7%. The classification report further details precision, recall, and F1-score metrics, highlighting the model's ability to effectively classify fire risk categories. This successful application of hyperparameter tuning and cross-validation underscores their importance in refining and evaluating the XGBoost model, contributing to its effectiveness in predicting fire risk levels accurately.

## **4.3 Random Forest Classifier**

The Random Forest Classifier is utilized as an ensemble learning method to address the classification task. This technique creates multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. It is effective in handling complex relationships and reducing overfitting.

### **4.3.1 Explanation:**

The application of the Random Forest classifier to the fire risk assessment dataset represents a robust ensemble learning technique that leverages multiple decision trees to make accurate predictions. The initial step involved a meticulous exploration of hyperparameter tuning, employing GridSearchCV to systematically evaluate various hyperparameter combinations. Parameters such as the number of trees ('n\_estimators'), maximum depth of the trees ('max\_depth'), minimum samples required to split an internal node ('min\_samples\_split'), and minimum samples required to be a leaf node ('min\_samples\_leaf') were fine-tuned to identify the configuration that maximized the model's accuracy.

### **4.3.2 Parameters:**

The selected hyperparameters for the Random Forest classifier were determined through a process of hyperparameter tuning using GridSearchCV. Let's delve into the technical reasons behind why these specific hyperparameters were chosen over others:

**Number of Estimators (n\_estimators):**

Contribution: The parameter dictates the quantity of decision trees in the ensemble, impacting the model's overall robustness.

Justification: The grid search systematically explored various values for n\_estimators, ensuring an optimal balance between improved model stability and computational efficiency.

**Maximum Depth of the Trees (max\_depth):**

Contribution: Regulating the depth of individual trees mitigates the risk of overfitting, particularly as deeper trees can capture intricate patterns in the training data.

Justification: The grid search sought the ideal max\_depth by assessing its impact on the trade-off between capturing complex relationships and preventing overfitting.

**Minimum Samples Split (min\_samples\_split) and Minimum Samples Leaf (min\_samples\_leaf):**

Contribution: These parameters control node splitting and leaf formation, respectively, influencing the granularity of decision tree structures.

Justification: The grid search dynamically adjusted min\_samples\_split and min\_samples\_leaf to strike a balance, enabling the model to generalize effectively without overly specific splits tailored to the training set.

**Feature Subset Size (max\_features):**

Contribution: Randomly selecting a subset of features at each split introduces diversity among trees, reducing correlation and enhancing overall model performance.

Justification: The grid search meticulously determined the optimal max\_features, ensuring an effective compromise between diversity and predictive accuracy.

## **4.4 Decision Tree Classifier**

A Decision Tree Classifier is implemented to model the relationship between the features and the fire risk. Decision trees split the dataset based on the most significant attributes, creating a tree-like structure. The model makes predictions by traversing the tree from the root to a leaf node. It is a simple yet powerful algorithm for classification tasks.

### **4.4.1 Explanation:**

The application of the Decision Tree classifier to the fire risk assessment dataset represents a straightforward yet powerful approach to predictive modeling. The process commenced with a systematic exploration of hyperparameter tuning, utilizing GridSearchCV to evaluate a range of hyperparameter combinations. Key parameters considered for tuning included the maximum depth of the tree ('max\_depth'), the minimum number of samples required to split an internal node ('min\_samples\_split'), and the minimum number of samples required to be a leaf node ('min\_samples\_leaf'). This optimization process aimed to identify the configuration that maximizes the model's accuracy.

Hyperparameter tuning is critical in Decision Tree models to find the right balance between model complexity and overfitting. The selected hyperparameters, derived from the GridSearchCV process, ensure that the Decision Tree classifier is appropriately configured for the specific nuances of the fire risk assessment data.

To comprehensively evaluate the model's generalization capabilities and robustness, a cross-validation strategy was implemented. The 5-fold cross-validation approach allowed for a thorough examination of the model's consistency across different subsets of the data, providing insights into its stability and performance.

The hyperparameter-tuned Decision Tree classifier demonstrated moderate performance on the test set, achieving an accuracy of approximately 57.8%. The classification report further details precision, recall, and F1-score metrics, shedding light on the model's effectiveness in classifying fire risk categories. This application of hyperparameter tuning and cross-validation highlights their importance in refining and evaluating the Decision Tree model, contributing to its reliability and accuracy in predicting fire risk levels. However, it's worth noting that Decision Trees might struggle to capture complex relationships present in the data compared to more sophisticated ensemble methods.

### **4.4.2 Performance Note:**

The Decision Tree classifier, despite its simplicity and interpretability, may not perform as well as other classifiers like Random Forest or XGBoost in certain scenarios. Here are some reasons why the Decision Tree may not be performing well in this case:

**Overfitting:**

* Decision trees are prone to overfitting, especially when they are deep and capture noise in the training data.
* Overfitting occurs when the tree is too complex, capturing both the underlying patterns and the noise in the training data, leading to poor generalization to new, unseen data.

**Lack of Ensemble Effect:**

* Unlike ensemble methods such as Random Forest or XGBoost, a single decision tree may lack the robustness provided by aggregating multiple trees.
* Ensemble methods are designed to mitigate overfitting and improve overall performance by combining predictions from multiple weak learners.

**Limited Expressiveness:**

* Decision trees may struggle to capture complex relationships in the data when the relationships are not inherently hierarchical or when feature interactions are intricate.
* Other models like Random Forest or XGBoost can handle more sophisticated relationships.

**Sensitivity to Data Variations:**

* Decision trees are sensitive to variations in the training data, and slight changes in the dataset can result in different tree structures.
* This sensitivity can lead to instability in the model's performance.

**Insufficient Hyperparameter Tuning:**

The hyperparameters of the Decision Tree Classifier, such as the maximum depth and minimum samples split, might not have been fine-tuned adequately during the hyperparameter tuning process.

## **4.5 Support Vector Machine (SVM) Regression**

SVM Regression is applied to predict the area of land burnt. SVM looks for a hyperplane that best separates the data into different classes while considering outliers. The epsilon loss function is used for regression in high-dimensional space. Various kernels such as RBF, linear, polynomial, and sigmoid are explored to find the optimal hyperplane.

### **4.5.1 Explanation:**

The application of Support Vector Machines (SVM) to the fire risk assessment dataset demonstrates the versatility of this algorithm in handling both linear and non-linear classification tasks. The process commenced with an in-depth exploration of hyperparameter tuning, employing GridSearchCV to systematically evaluate various hyperparameter combinations. Key parameters considered for tuning included the regularization parameter ('C'), the kernel type ('linear', 'rbf', 'poly'), and the gamma parameter ('scale', 'auto'). This comprehensive search aimed to identify the optimal hyperparameter configuration that maximizes the model's accuracy.

Hyperparameter tuning is crucial in SVM models to find the right balance between model flexibility and regularization, ensuring optimal performance on the specific dataset. The selected hyperparameters, derived from the GridSearchCV process, tailor the SVM classifier to the unique characteristics of the fire risk assessment data.

To thoroughly assess the model's generalization capabilities and robustness, a cross-validation strategy was implemented. The 5-fold cross-validation approach facilitated a comprehensive evaluation of the model's consistency across various subsets of the data, enhancing its reliability and ability to generalize effectively. The hyperparameter-tuned SVM classifier demonstrated robust performance on the test set, achieving an accuracy of approximately 88.8%. The classification report further details precision, recall, and F1-score metrics, highlighting the model's effectiveness in accurately classifying fire risk categories. This successful application of hyperparameter tuning and cross-validation underscores their importance in refining and evaluating the SVM model, contributing to its reliability and accuracy in predicting fire risk levels. SVM's ability to handle non-linear relationships and high-dimensional data makes it a valuable tool in the context of fire risk assessments.

### **4.5.2 Parameters:**

The chosen hyperparameters for the Support Vector Machine (SVM) model are:

* **C**: 10
* **gamma**: 'scale'
* **kernel**: 'poly'

These hyperparameters contribute to the superior performance of the SVM model:

**C (Regularization Parameter):**

* C controls the trade-off between achieving a low training error and a low testing error.
* A smaller C emphasizes a smoother decision boundary, allowing for some misclassifications, while a larger C aims for a more accurate classification of each training point.
* The optimal value for C is often found through cross-validation. In the given grid, [0.1, 1, 10] represents a range of regularization strengths.

**Kernel:**

* The choice of the kernel function significantly impacts the SVM's ability to handle complex relationships in the data.
* 'Linear' kernel is suitable for linearly separable data, while 'rbf' (Radial Basis Function) and 'poly' (Polynomial) kernels can capture non-linear relationships.
* The grid ['linear', 'rbf', 'poly'] allows exploration of different kernel options.

**Gamma:**

* Gamma (γ) defines the influence of a single training example, affecting the shape of the decision boundary.
* 'Scale' uses 1 / (n\_features \* X.var()) as the gamma value, and 'auto' uses 1 / n\_features.
* Higher values of gamma lead to a more complex decision boundary, potentially causing overfitting.

# **5 Evaluations**

## **5.1 Model Evaluation for KNN Classification:**

**Evaluation Metrics:**

* Mean Absolute Error (MAE): Measures the average absolute errors between predicted and actual values.
* Mean Squared Error (MSE): Measures the average squared errors between predicted and actual values.
* R-squared: Indicates the proportion of the variance in the dependent variable that is predictable.

A screenshot of a computer

Description automatically generated

Figure 8 classification report KNN

**Cross-Validation:**

Cross-validation is performed to ensure the robustness of the KNN model.

Multiple folds are used to train and validate the model, preventing overfitting.

## **5.1.1 Model Performance**

Regarding whether the model is overfitting, based on the provided results:

* Both the training and testing accuracies are 1.0, indicating that the model achieves perfect accuracy on both datasets. The fact that both training and testing accuracies are 1.0 raises suspicion of overfitting. In a real-world scenario, achieving perfect accuracy on the testing set is uncommon and could be a sign that the model is too complex and fitting noise in the training data.
* The classification report also shows perfect precision, recall, and F1-score for all classes on the testing set. Perfect scores across all metrics in the classification report for the testing set further support the possibility of overfitting. In practical situations, a model should not perform flawlessly on all aspects, as it may indicate over-reliance on the training data's specifics.
* The training accuracy is also 1.0, suggesting that the model perfectly fits the training data. The training accuracy being 1.0 suggests that the model has memorized the training data. While high training accuracy is desirable, a perfect score might indicate overfitting, especially if the model struggles to generalize to new, unseen data.

Given these observations, there is a strong indication that the model might be overfitting. Overfitting occurs when the model captures noise or random fluctuations in the training data, leading to excessively complex decision boundaries that do not generalize well to unseen data. In this case, the model achieves perfect accuracy on both the training and testing sets, which raises suspicion of overfitting, especially in real-world scenarios where noise and variability are expected.

## **5.1.2 Visualizations:**

Visualizations include scatter plots comparing actual vs. predicted values.

Confusion metrics are visualized for different folds.

A graph with blue squares

Description automatically generated

Figure 9 Confusion Matrix KNN

## **5.2 Model Evaluation for XGBoost Classifier:**

### **5.2.1 Evaluation Metrics:**

Accuracy: The proportion of correctly classified instances.

Precision, Recall, F1-score: Metrics for assessing the performance of multi-class classification.

Hyperparameter Tuning:

GridSearchCV is used to find the best hyperparameters for the XGBoost Classifier.

A screenshot of a computer

Description automatically generated

Figure 10 classification report XG Boost

### **5.2.2 Visualizations:**

Confusion matrix heatmap: Provides a visual representation of model performance on different classes.

Bar graph comparing accuracy and other metrics.

A screenshot of a graph

Description automatically generated

Figure 11 Confusion Matrix XG Boost

**Confusion matrix of X\_gradient\_boosting:**

* True Negatives (TN): 36 (in the second row, second column)
* False Positives (FP): 1 (in the first row, second column), 1 (in the third row, second column), 2 (in the fifth row, second column)
* False Negatives (FN): 1 (in the first row, fourth column), 3 (in the second row, fourth column), 1 (in the third row, fourth column), 2 (in the fifth row, fourth column)
* True Positives (TP): 37 (in the first row, first column), 36 (in the second row, first column), 36 (in the third row, third column), 44 (in the fourth row, third column), 40 (in the fifth row, fifth column)

## **5.3 Model Evaluation for Random Forest Classifier:**

### **5.3.1 Evaluation Metrics:**

Accuracy, Precision, Recall, F1-score: Standard metrics for classification evaluation.

A screenshot of a computer

Description automatically generated

Figure 12 classification report RFC

Hyperparameter Tuning:

GridSearchCV is employed to find optimal hyperparameters for the Random Forest model.

### **5.3.2 Visualization:**

Confusion matrix heatmap: Shows classification results for different classes.

Bar graph comparing accuracy and other metrics.

A graph with numbers and squares

Description automatically generated

Figure 13 Confusion Matrix RFC

## **5.4 Model Evaluation for Decision Tree Classifier:**

### **5.4.1 Evaluation Metrics:**

Accuracy, Precision, Recall, F1-score: Standard classification metrics.

A screenshot of a computer

Description automatically generated

Figure 14 classification report Decision Tree

Hyperparameter Tuning:

GridSearchCV is used to find the best hyperparameters for the Decision Tree model.

### **5.4.2 Visualization:**

Confusion matrix heatmap: Visualizes model performance on different classes.

Bar graph comparing accuracy and other metrics.

A blue squares with white text

Description automatically generated

Figure 15 Confusion Matrix Decision tree

## **5.5 Model Evaluation for Support Vector Machine (SVM) Regression:**

## **5.5.1 Evaluation Metrics:**

Accuracy: Measures the proportion of correctly predicted instances.

R-squared: Indicates the proportion of variance in the target variable explained by the model.

A screenshot of a computer

Description automatically generated

Figure 16 classification report SVM

Hyperparameter Tuning:

GridSearchCV is employed to find the optimal hyperparameters for the SVM Regression model.

### **5.5.2 Visualization:**

Scatter plots comparing actual vs. predicted values.

Evaluation metrics visualized for different folds in cross-validation.

A screenshot of a graph

Description automatically generated

Figure 17 Confusion matrix SVM

# **6 Results & Conclusions:**

A bar graph comparing the performance metrics (accuracy, precision, recall, F1-score) of all models.

## **6.1 Accuracy comparison difference**

To compare the accuracies of different models, we can observe the differences in accuracy percentages across various machine learning models used in the project:

1. KNN:

Accuracy: 100%

1. XGBoost Classifier:

Accuracy: 93.69%

1. Random Forest Classifier:

Accuracy: 87.86%

1. Decision Tree Classifier:

Accuracy: 56.80%

1. SVM (Support Vector Machine) Classifier:

Accuracy: 88.83%

These accuracy percentages represent the overall performance of each model on the testing data. The differences in accuracy can be analyzed to understand how well each model is making correct predictions. Higher accuracy values indicate better predictive performance.

## **6.2 Observations:**

* XGBoost Classifier achieves the highest accuracy among the models, suggesting superior predictive capabilities.
* Random Forest Classifier and SVM also demonstrate high accuracy, indicating robust performance.
* KNN has a good accuracy.
* Decision Tree Classifier, while decent, falls behind the ensemble models (XGBoost and Random Forest) and SVM in terms of accuracy.

These differences in accuracy can be attributed to the inherent characteristics and strengths of each algorithm. It's important to consider factors such as model complexity, ability to handle non-linear relationships, and sensitivity to hyperparameter tuning when interpreting these accuracy values.

A table with numbers and text

Description automatically generated

Figure 18 Accuracy Comparison

Graphs for confusion matrices of classification models.

A graph of different models

Description automatically generated

Figure 19 Overall Comparison

## **6.3 Model Selection for Deployment:**

The model deployment diagram used for providing the overall web interaction is given in fig 20.

A diagram of a computer

Description automatically generated

Figure 20 Deployment Diagram

**Web Service:**

Explanation: The Web Service serves as the interface between the end user (browser or API client) and the machine learning model. It handles incoming requests, processes them, and communicates with the machine learning model to obtain predictions. This component is responsible for managing the flow of information between the user and the model.

**Machine Learning Model:**

Explanation: The Machine Learning Model is the core of the forest fire risk assessment system. It has been trained on relevant datasets to predict the risk of forest fires based on input features. These features could include weather conditions, terrain information, or other relevant parameters. The model processes incoming data from the web service and produces predictions for the likelihood of a forest fire occurring.

**Database:**

Explanation: The Database stores and manages the data required for the forest fire risk assessment. This can include historical weather data, simulated data, or any other relevant information needed for training and evaluation of the machine learning model. The model might also store and retrieve its configuration or hyperparameters from the database.

**End User (Browser or API Client):**

Explanation: The End User represents the individual or system interacting with the forest fire risk assessment application. This could be a person using a web browser to access a user interface or an external system making requests through an API. The end user provides input data (such as location, weather parameters) to the system and receives the corresponding risk assessment predictions.

**Browser or API Client:**

Explanation: The Browser or API Client is the tool used by the end user to interact with the web service. It can be a web browser for human users or an API client for automated systems. This component sends input data to the web service, receives predictions, and may display the results to the end user.

**Model Storage:**

Explanation: Model Storage is where the trained machine learning model is persisted. It could be a file storage system or a dedicated model repository. This ensures that the latest version of the model is accessible to the web service for making predictions. Storing the model separately allows for easy updates and version control.

**Deployment of KNN Model for CCF Forest Fire Risk Assessment Using Django:**

**1. Model Development:**

The machine learning model chosen for CCF forest fire risk assessment is the K-Nearest Neighbors (KNN) model. This model was trained using historical data with features relevant to CCF forests, weather conditions, and other environmental factors.

**2. Integration with Django:**

The integration with Django, a Python web framework, facilitated the development of a web application for deploying the KNN model.

**3. Frontend Development with Django Forms:**

The front-end interface was created using Django Forms, providing an intuitive and user-friendly way for end-users to input data related to CCF forests. The form captures relevant features required for the fire risk prediction.

**4. User Input Handling:**

Django manages user inputs, ensuring data integrity and preparing it for consumption by the KNN model. Data validation is implemented to guarantee that the input adheres to the expected format and range.

**5. Model Prediction Integration:**

Upon form submission, the Django application passes the entered data to the deployed KNN model for prediction. This step involves seamlessly integrating the model into the application's backend.

**6. Result Display to the User:**

The prediction generated by the KNN model is presented to the user through the Django web interface. The output can be in the form of a risk level (e.g., low, moderate, high) depending on the model output.

**7. Scalability and Accessibility:**

The Django application is designed to handle multiple users simultaneously, ensuring scalability. This enhances accessibility, allowing users to access and benefit from the model predictions concurrently.

**8. User Interaction and Iterative Input:**

Users can interact with the Django web application, providing different sets of data to obtain real-time predictions for various CCF forest scenarios. The iterative input capability enables users to explore diverse situations.

**9. User-Friendly Interface:**

The front end, built with Django templates, ensures a user-friendly experience. The interface guides users through the input process, making it straightforward for individuals with varying levels of technical expertise.

.

## **6.4 Dashboard:**

A screenshot of a computer

Description automatically generated

Figure 21 Web App

## **6.5 Conclusion:**

In conclusion, this comprehensive analysis aimed to assess and compare the performance of various machine learning models for fire risk assessment using a dataset containing diverse features related to environmental factors. The models considered include K-Nearest Neighbors (KNN), XGBoost, Random Forest, Decision Tree, and Support Vector Machines (SVM). Each model underwent rigorous hyperparameter tuning and evaluation through cross-validation to optimize performance.

The KNN model exhibited outstanding accuracy, achieving a perfect score on the test set. However, its simplicity may limit its ability to capture complex relationships in the data. XGBoost demonstrated robust performance, attaining an accuracy of approximately 93.7%, making it a robust choice. Random Forest, while achieving an accuracy of 87.9%, displayed reliable performance, especially in handling diverse datasets.

The Decision Tree model, with an accuracy of 57.8%, highlighted moderate performance, indicating potential limitations in capturing intricate patterns within the data. Lastly, the SVM model performed well with an accuracy of 88.8%, showcasing its adaptability to various kernel types and suitability for both linear and non-linear relationships.

Considering the overall performance, XGBoost emerges as the most promising model for fire risk assessment in this analysis. Its ability to handle complex relationships, robustness in diverse datasets, and high accuracy make it a compelling choice. However, the selection of the most suitable model may depend on specific requirements, computational resources, and interpretability preferences. It is advisable to consider the unique characteristics of the application domain and dataset when making the final model selection.