1. ITNPBD6 Assignment 1. Student Number 3309061

2. Project Methodology

The Methodology used for this project is the CRISP-DM Methodology

CRIPS-DM	Business Understanding	Data Understanding	Data Preparation
	The task is to develop a predictive model for the national forestry commission to accurately classify forests as either at risk of experiencing a fire or not, based on environmental and forest condition data. By employing logistic regression, decision trees, and neural networks, the aim is to identify the most accurate model for predicting fire or not fire likelihood to enhance forest management and preventive measures against potential wildfires.	· Loaded the Data · Checked for datatypes and Non-Null Values · Checked Statistical Summary · Checked for Missing Values · Checked for Unique Values · Checked Variance of Numeric features · Checked the correlation of features using a heatmap. · Plotted a histogram and boxplot of features. · Identified outliers. · Identified Misspelled Feature · Checked for imbalanced features. · Feature selection - Removed Collector ID	· Split Data into Train, Validate and Test - 60%, 20%, 20%. · Handled Missing Values using Imputation fitted on the training data and transforming validate and test data using the same imputer. · Handled outliers using clipping. · Corrected Misspellings in dataset. · Performed Normalization of numerical features. · Applied One hot encoding to categorical Features. · Used Smote to balance target Variable.
CRIPS-DM	**Modelling**	**Evaluation**	**Deployment**
	· Trained the models · Predicted using the Validation data. · Calculated evaluation metrics (Accuracy, precision, Recall and F1) · Plotted the Confusion Matrix · Performed Cross Validation and Hyperparameter Tuning · Performed Error Analysis · Performed Feature Importance Analysis · Selected the best model based on Recall , F1, Stability and Simplicity	. Combined the Training and Validation Set. . Checked for Missing Values and Handled Outliers . Applied smote to the combined Set. . Trained the best model on the training set. . Evaluated on the test set by optimizing Recall and F1 . Performed Feature Importance Analysis. . Calculated the Confusion Matrix	. Made a poster of my findings . Will Copy Codes and share with the developer - for integration into the existing technology stack. . Detailed the best model, tuning and reason for choice

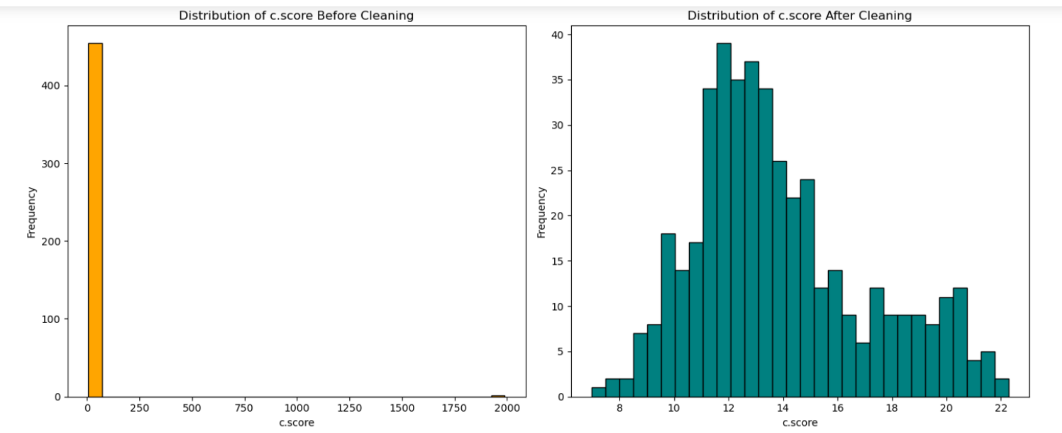
3. Variables

Variables/features, how they were treated and their impact

Features	Data Types	Roles	Treated As	Variable Type	Impact on Model
collector.id	int64	Discrete	Numeric	Input Variable	Irrelevant to the prediction target, it could introduce noise
c.score	float64	Continuous	Numeric	Input Variable	Higher carbohydrate makeup might influence fire likelihood due to fuel quality.
l.score	float64	Continuous	Numeric	Input Variable	The wood-to-leaves mass ratio could affect flammability and, thus, fire risk.
rain	float64	Continuous	Numeric	Input Variable	More rain could reduce fire risk due to increased moisture
tree.age	float64	Continuous	Numeric	Input Variable	Older trees might influence fire risk due to differences in flammability.
surface.litter	float64	Continuous	Numeric	Input Variable	More litter could indicate a higher fire risk due to more potential fuel.
wind.intensity	float64	Continuous	Numeric	Input Variable	Higher wind speeds could indicate a higher risk of fire spread
humidity	float64	Continuous	Numeric	Input Variable	Lower humidity typically increases fire risk due to dryer conditions.
tree. density	float64	Continuous	Numeric	Input Variable	Higher density could suggest a higher fire risk if a fire starts.
month	int64	Discrete	Categorical	Input Variable	Different months may have varying fire risks due to seasonal changes.
time.of.day	object	Categorical	Categorical	Input Variable	The time of day could affect the likelihood of fires due to temperature and human activity.
fire	int64	Discrete	Categorical	Target Variable	The outcome variable the model is trying to predict.

4. Data Preparation

Histogram showing c.score before and after cleaning



5. Model Training and Hyperparameters

Results on the validation set and Hyperparameter and CV best Score

Logistic Regression	Recall	Precision	F1 Score	Accuracy
Validation Set	0.954	0.933	0.943	0.945
Hyperparameter Tuning and CV – Best Score	0.972	0.955	0.962	0.962
Decision Tree	**Recall**	**Precision**	**F1 Score**	**Accuracy**
Validation Set	0.931	0.931	0.931	0.934
Hyperparameter Tuning and CV – Best Score	0.929	0.912	0.911	0.910
Random Forest	**Recall**	**Precision**	**F1 Score**	**Accuracy**
Validation Set	0.977	0.955	0.966	0.967
Hyperparameter Tuning and CV – Best Score	0.967	0.918	0.939	0.937
Neural Network	**Recall**	**Precision**	**F1 Score**	**Accuracy**
Validation Set	0.931	0.953	0.942	0.945
Hyperparameter Tuning and CV – Best Score	0.978	0.939	0.957	0.956

The hyperparameter values were chosen based on commonly used ranges that provide a balance between model complexity, regularisation and computation time. The aim was to explore a comprehensive grid of options to determine the best configuration for each model.

Model	Hyper Parameters Explored	Selected Hyperparameters	Optimised Metric	Metric Score
Logistic Regression	'C': uniform(0.001, 10), 'max_iter': [100, 1000, 2000] , 'solver': ['newton-cg', 'lbfgs' , 'liblinear']	'C': 6.119528947223795, 'max_iter': 1000, 'solver': 'liblinear'	Accuracy, Recall and F1	96%, 97% and 96%
		'C': 0.09297051616629648, 'max_iter': 1000, 'solver': 'newton-cg'	Precision	96%
Decision Tree	'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]	'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10	Accuracy, Recall and F1	91%,93% and 91%
		'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2	Precision	91%
Random Forest	'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]	'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200	Accuracy, Recall and F1	94%, 97% and 94%
		'bootstrap': False, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200	Precision	92%
Neural Network	'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)], 'alpha': [0.0001, 0.001, 0.01], 'learning_rate_init': [0.001, 0.01],	'learning_rate_init': 0.001, 'hidden_layer_sizes': (50,), 'alpha': 0.001	Accuracy, F1 and Recall	96%, 96% and 98%
		'learning_rate_init': 0.001, 'hidden_layer_sizes': (100,), 'alpha': 0.01	Precision	94%

The hyperparameter selection process was strategic, drawing on expert insight and research to explore a broad spectrum of values, ensuring a balanced exploration between conservative and aggressive settings.

I utilized GridSearchCV for thorough searches and RandomizedSearchCV for efficient exploration across extensive parameter spaces, with cross-validation ensuring consistent performance across the training data.

The tuning focused on recall and F1 score to mitigate false negatives and prioritizing a balanced prediction of fire occurrences. This approach aimed to minimize the risk of overlooking fire likelihood, a critical consideration for forest management and fire prevention strategies. Despite this focus, all metrics were optimized to ensure a comprehensive understanding of the model's performance across different evaluation criteria, balancing the need for precision and accuracy (Winkler et al, 2019)

6. Final Model and Results

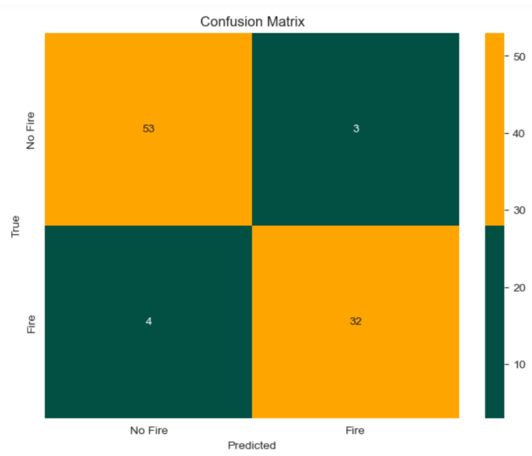
Choosing the Final Model and How it was trained and Tested

Choosing the final model involved evaluating the performance metrics of logistic regression, decision trees, and neural networks. Logistic regression was selected as the final model because it demonstrated a balanced performance across recall, f1 Score- and precision,, which is crucial for accurately predicting forest fires. Specifically, logistic regression provided a satisfactory balance between minimizing false negatives (missing potential fires) and maintaining overall accuracy. Despite neural networks having close scores, logistic regression was preferred due to its simplicity, interpretability, ease of implementation and more balanced performance (Hosmer et al, 2013).

The dataset was first split into training, validation, and test sets to train the logistic regression model. Preparation steps included imputing missing values, handling outliers, normalisation, one-hot encoding and applying SMOTE (Synthetic Minority Over-sampling Technique) to address class imbalance. The logistic regression model was then optimized using RandomizedSearchCV, which efficiently explored the hyperparameter space to enhance recall without significantly compromising other metrics.

After optimization, the logistic regression model underwent further training on a merged dataset comprising the training and validation sets. Outliers were handled, and balancing was done using SMOTE. The final Logistic Regression model, optimised for recall, achieved a recall of 0.89, precision of 0.91, accuracy of 0.92, and F1 score of 0.90 on the test set. This effectively minimized false negatives in predicting fire occurrences while striking a balance across evaluation metrics. This approach aimed to prepare the chosen model to offer reliable predictions in real-world scenarios.

Explanation of the Confusion Matrix



	Predicted No Fire	Predicted Fire
Actual No Fire	53	3
Actual Fire	4	32

True Negatives (TN): The model correctly predicted “No Fire” 53 times.

False Positives (FP): The model incorrectly predicted “Fire” 3 times when there was actually no fire. These are type I errors.

False Negatives (FN): The model incorrectly predicted “No Fire” 4 times when there was actually a fire. These are type II errors, and they are particularly critical in this context because failing to predict a fire could have serious consequences.

True Positives (TP): The model correctly predicted “Fire” 32 times. These are cases where there was actually a fire, and the model successfully identified it.

7. References

IBM. (2024) Data Science Professional Certificate [Online Course]. Coursera. Available at: https://www.coursera.org/professional-certificates/ibm-data-science (Accessed: 15 March 2024).

Hosmer Jr, D.W., Lemeshow, S. and Sturdivant, R.X., (2013) Applied Logistic Regression. 3rd ed. John Wiley & Sons.

Winkler, J.P., Grönberg, J. and Vogelsang, A. (2019) ‘Optimizing for Recall in Automatic Requirements Classification: An Empirical Study’, *2019 IEEE 27th International Requirements Engineering Conference (RE)*, Jeju, Korea (South), pp. 40-50. doi: 10.1109/RE.2019.00016.