**Microsoft: Classifying Cybersecurity Incidents with Machine Learning**

**Project Report**

**Submitted By:**

**Yamini Devi**

**Index**

1. Introduction
2. Data Preprocessing  
   2.1 Loading the Data  
   2.2 Inspecting Missing Values  
   2.3 Handling Missing Data  
   2.4 Cleaning and Formatting  
   2.5 Correlation Analysis
3. Data Splitting  
   3.1 Splitting the Dataset  
   3.2 Scaling the features
4. Machine Learning Algorithms
5. Results and Comparisons  
   5.1 Model comparison   
   5.2 Metric Summary
6. Results visualizations  
   6.1 Confusion matrices  
   6.2 Feature Importances
7. Discussion
8. Conclusion

**1. Introduction**

**Objective:**

The objective of this report is to explore the data processing and machine learning pipeline to predict IncidentGrade, a target variable that determines the severity or category of incidents. This involves two main phases:

1. Preprocessing the dataset to clean, prepare, and transform it for modeling.
2. Training and evaluating machine learning models to derive insights and assess their effectiveness.

**Dataset:**

The dataset contained numerous features, each representing attributes related to the incidents. The target variable, IncidentGrade, had a classification nature (e.g., binary or multi-class). Understanding the dataset’s structure, addressing missing values, and preprocessing steps were key components before moving to model development.

**Scope:**

This report is aimed at:

1. Describing the preprocessing workflow for cleaning and transforming raw data.
2. Presenting two machine learning algorithms: Logistic Regression and Decision Trees.
3. Analyzing the results, strengths, and limitations of the models.

**2. Data Preprocessing**

The preprocessing phase ensures that the dataset is ready for machine learning. This section describes each step with detailed explanations.

**2.1 Loading the Data**

* The dataset was loaded using Python’s Pandas library, a powerful tool for data manipulation and analysis.
* Key steps included:
  + Reading the dataset using pd.read\_csv() to load it into a DataFrame.
  + Inspecting the data structure using functions like df.info() and df.head().
  + Observations:
    - The dataset contained several numerical and categorical features, each requiring specific handling.
    - Missing values were identified in numerous columns.

**2.2 Inspecting Missing Values**

* Missing data can lead to biases or model errors.
* The percentage of missing values for each column was calculated using:

**missing\_values = df.isnull().sum()**

**missing\_percentage = (missing\_values / len(df)) \* 100**

* **Observations**:
  + Columns with more than 50% missing data were deemed non-informative and candidates for removal.
  + Other columns with missing values were flagged for imputation strategies.

**2.3 Handling Missing Data**

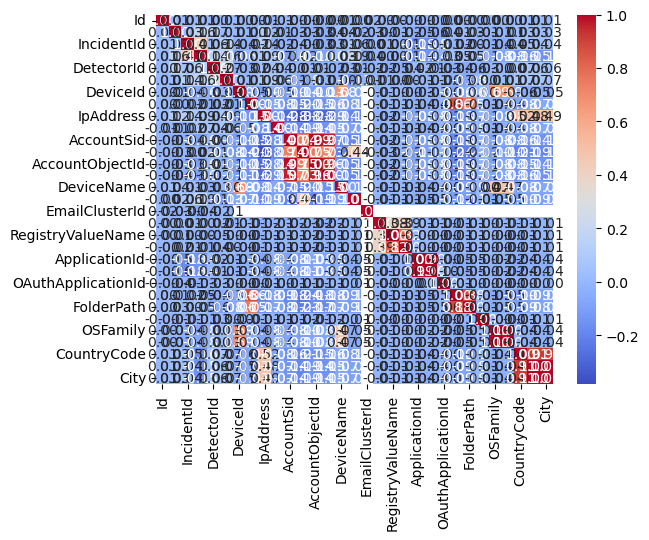
1. **Removing Columns**:
   * Irrelevant columns, including IDs or those with excessive missing values, were removed to reduce noise.
2. **Imputation**:
   * Numerical features with missing values were replaced using the mean or median.
   * Categorical features with missing values were replaced with the most frequent category or encoded with special tags like Unknown.

**2.4 Cleaning and Formatting**

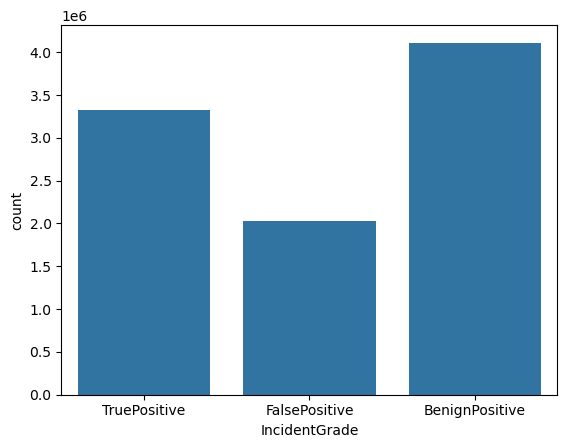
* Duplicate rows were identified and removed to prevent redundant information.
* Inconsistent data types were corrected (e.g., ensuring numerical values were properly formatted).
* A final dataset summary was generated to confirm cleanliness.

**2.5 Correlation Analysis**

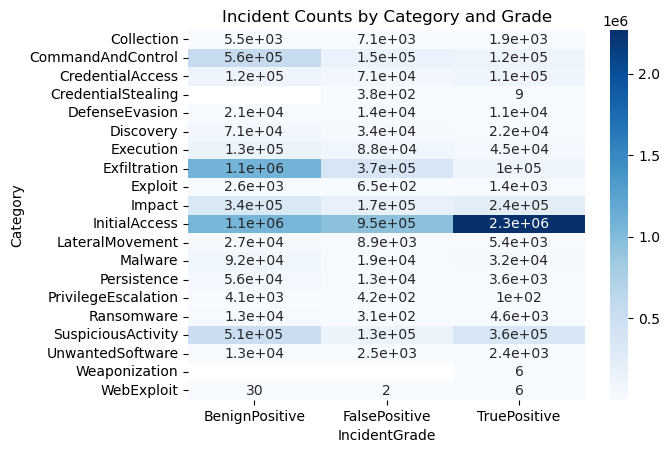
* **Objective**: Correlation analysis helps in understanding the relationships between numerical features in the dataset. It also identifies potential multicollinearity (when two or more features are highly correlated) which can negatively affect model performance.
* **Steps**:
  + A correlation matrix was calculated using the corr() function in Pandas to measure pairwise correlations between features.
  + The correlation coefficients range from -1 to 1:
    - **-1**: Perfect negative correlation (as one increases, the other decreases).
    - **0**: No correlation.
    - **1**: Perfect positive correlation (as one increases, so does the other).
  + Features with high correlation were flagged for potential removal to avoid redundancy.
* **Visualization**:
  + A heatmap was created using the seaborn library to visualize the correlation matrix.
  + Features highly correlated with the target variable (IncidentGrade) were highlighted for further exploration.
* **Insights**:
  + Features with high correlation to IncidentGrade were prioritized for model development.



We can see that the correlation in the data is less since the diagonal is prominent. The distribution of the IncidentGrade (output) is shown here. We can see that this is almost uniformly distributed indicating that the dataset is balanced.



The following plot shows the incident count by category and grade. This shows that the dataset is balanced.



**3. Data Splitting**

Proper data splitting ensures unbiased model evaluation.

**3.1 Splitting the Dataset**

* **Procedure**:
  + The dataset was divided into training (80%) and validation (20%) sets using the train\_test\_split function.
  + The target variable, IncidentGrade, was stratified to maintain its distribution in both splits.
* **Why Split?**:
  + Training data is used to fit the model, while validation data is used to evaluate the model’s ability to generalize.

**3.2 Scaling the Features**

* Feature scaling was performed to normalize numerical data between 0 and 1 using the MinMaxScaler.
* Why scaling is important:
  + Models like Logistic Regression are sensitive to the magnitude of input features.
  + Scaling ensures features contribute equally to the model’s performance.

**4. Machine Learning Algorithms**

This section describes the machine learning models used, their training process, and evaluation results.

**4.1 Logistic Regression**

1. **Overview**:
   * Logistic Regression is a linear model often used for binary classification tasks.
   * The sigmoid function maps predictions to probabilities, helping classify inputs into distinct categories.
2. **Training Process**:
   * Features were scaled and the model was fitted using the fit() method.
   * Hyperparameters such as the regularization parameter C were set to default.
3. **Evaluation**:
   * Predictions on the validation set were evaluated using:
     + **Precision**: Ratio of true positives to total predicted positives.
     + **Recall**: Ratio of true positives to total actual positives.
     + **F1-score**: Harmonic mean of precision and recall.
   * Confusion matrix analysis:
     + **True Positives (TP)**: Correctly classified positive instances.
     + **False Positives (FP)**: Instances wrongly classified as positive.
     + **True Negatives (TN)**: Correctly classified negative instances.
     + **False Negatives (FN)**: Instances wrongly classified as negative.

**4.2 Decision Tree Classifier**

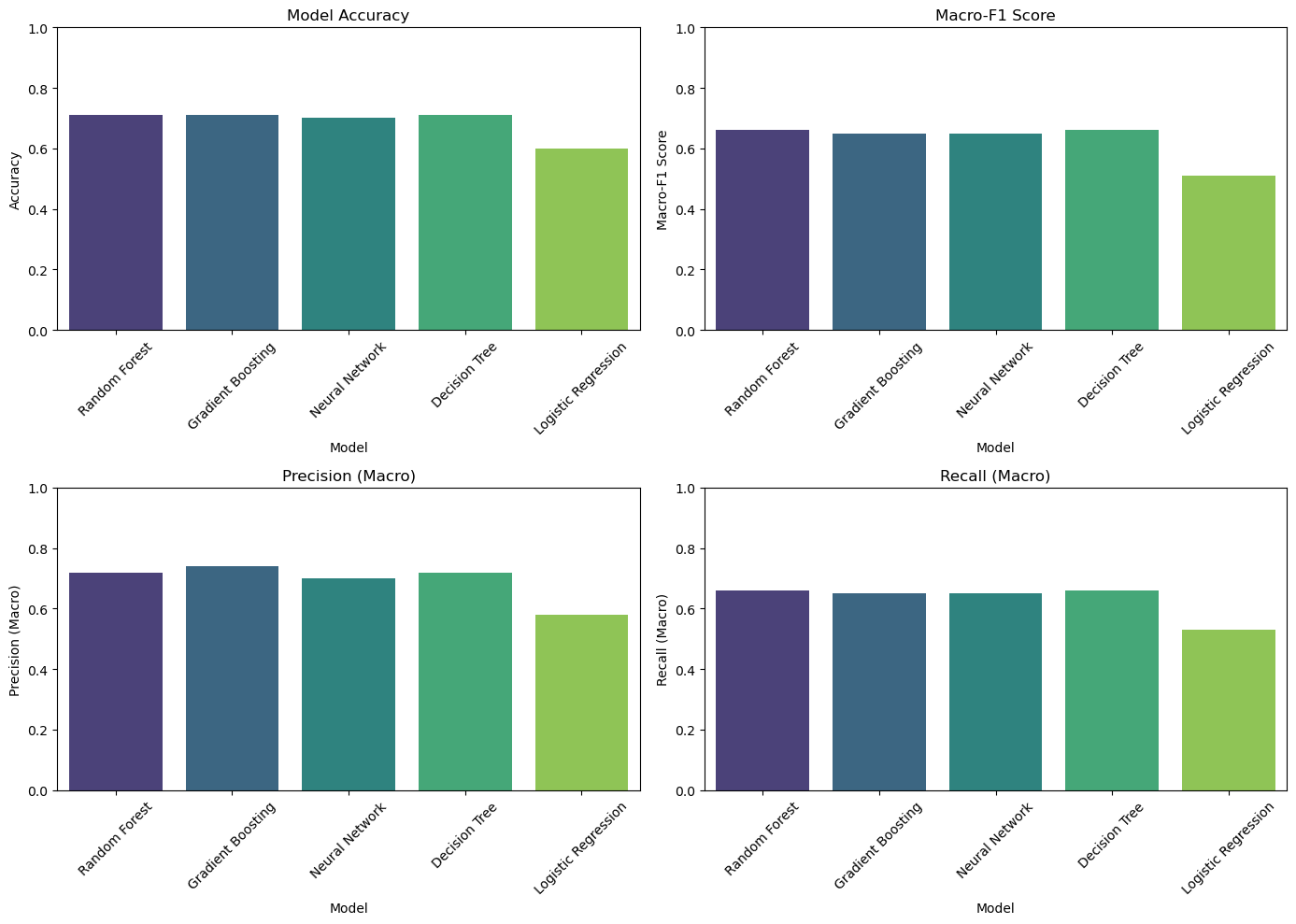
1. **Overview**:
   * Decision Trees are non-linear models that partition the dataset into subsets based on feature conditions.
   * They handle categorical and numerical data effectively without requiring scaling.
2. **Training Process**:
   * The model recursively split data into smaller subsets, maximizing information gain at each step.
3. **Evaluation**:
   * Similar evaluation metrics (precision, recall, F1-score) were used.
   * Observations:
     + Decision Trees captured complex patterns but were prone to overfitting.
     + Regularization techniques like pruning were considered to enhance generalization.

Other ML algorithms used are Logistic regression, Random Forest, fully connected Neural network, Decision Tree, Gradient boosting, Balanced Random Forest. The results comparison in terms of the metrics and the confusion matrix are shown below.

**5. Results and Comparisons**

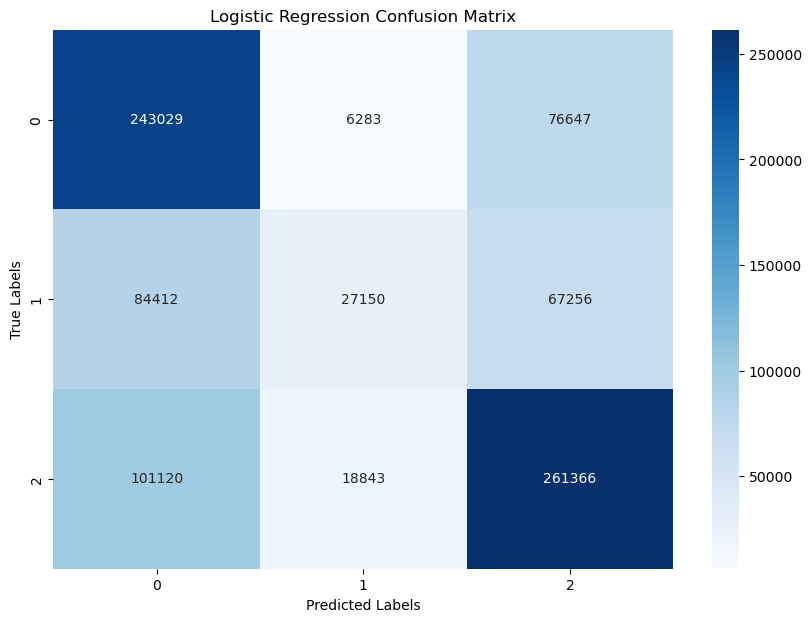
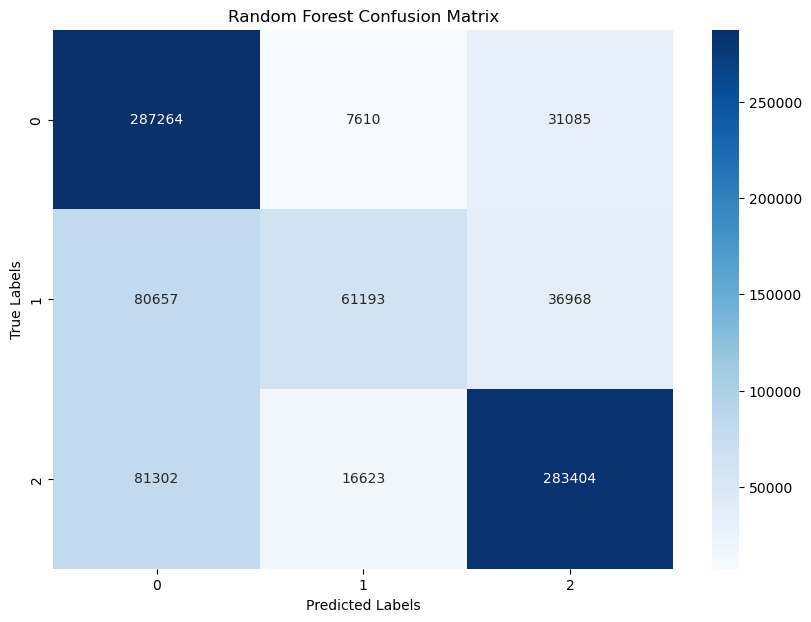
1. **Model Comparison**:
   * Logistic Regression:
     + Strength: Simple and interpretable.
     + Weakness: Limited in capturing non-linear relationships.
   * Decision Tree:
     + Strength: Effective at capturing non-linear patterns.
     + Weakness: Susceptible to overfitting.
2. **Metric Summary**:
   * Logistic Regression achieved higher precision, making it suitable for applications prioritizing accuracy in positive predictions.
   * Decision Tree achieved better recall, excelling at identifying all positives but at the cost of more false positives.

Comparison of various evaluation metrics is shown below.

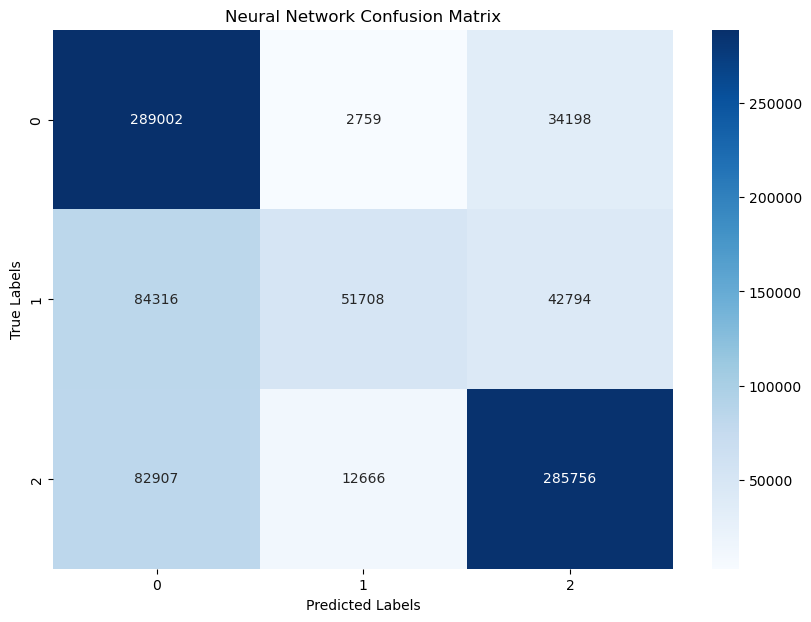
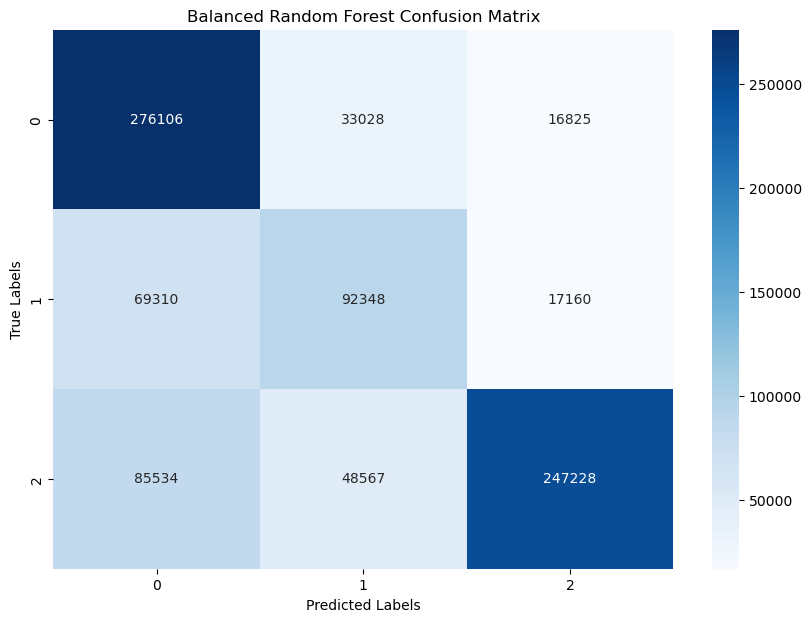
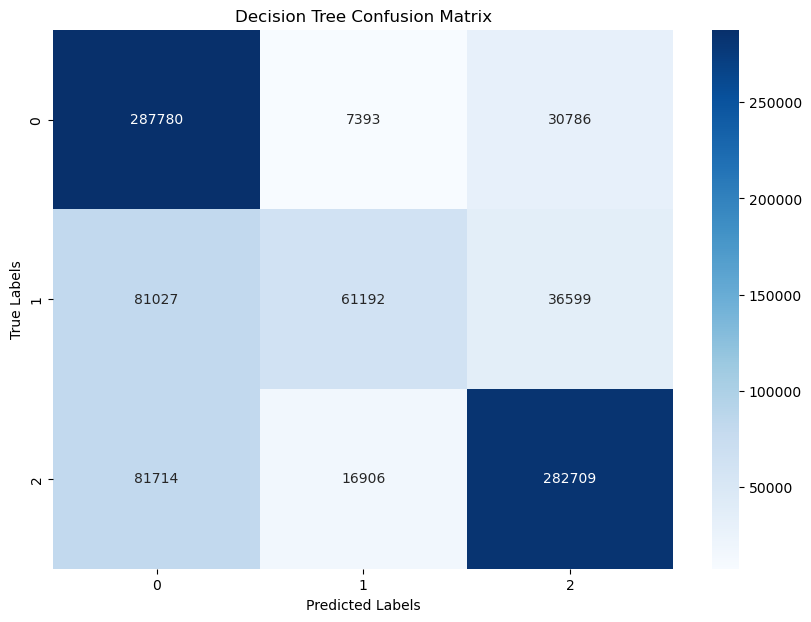


**6. Results visualizations**

1. **Confusion Matrices**:
   * Indicates the model performance



A screenshot of a graph

Description automatically generated

1. **Feature Importances**
   * Decision Tree visualizations ranked the importance of features, helping to identify key drivers for IncidentGrade.

A graph with blue bars

Description automatically generated

**7. Discussion**

* Logistic Regression proved to be a robust baseline model with fewer tuning requirements.
* The Decision Tree model highlighted complex feature interactions but required regularization to avoid overfitting.

**8. Conclusion**

1. **Summary**:
   * Data preprocessing ensured a clean, well-structured dataset.
   * Various models demonstrated strengths and trade-offs, catering to different needs.