PROJECT TITLE:

MICROSOFT CYBERSECURITY INCIDENT CLASSIFICATION WITH MACHINE LEARNING

BATCH:

MDTE13 (DS-WD-T-B10)

NAME:

NANDHINI C

COURSE:

DATA SCIENCE

Introduction

In an era of increasing cyber threats, Security Operations Centers (SOCs) face an overwhelming volume of cybersecurity alerts daily. Many of these alerts are either false positives or benign, resulting in wasted time and reduced efficiency. This phenomenon, known as alert fatigue, hampers timely responses to real threats.

This project aims to tackle these challenges by developing a machine learning model that classifies incidents into three categories:

- True Positive (TP): Real threats requiring action.
- False Positive (FP): Incorrectly flagged incidents.
- Benign Positive (BP): Harmless alerts that don't require immediate attention.

By automating the classification process, the model will empower SOC teams to focus on critical incidents, thereby improving organizational security and reducing manual effort.

Problem Statement

The exponential growth of cybersecurity alerts has created challenges for SOCs:

1. **High Alert Volume:** Most alerts are non-critical, making manual triage inefficient.
2. **Human Error:** Analysts face difficulty identifying real threats accurately.
3. **Response Delays:** Alert fatigue slows down response times.

Objective: Build a machine learning model to classify alerts into TP, FP, or BP, improving response times and reducing manual workload.

Data Exploration

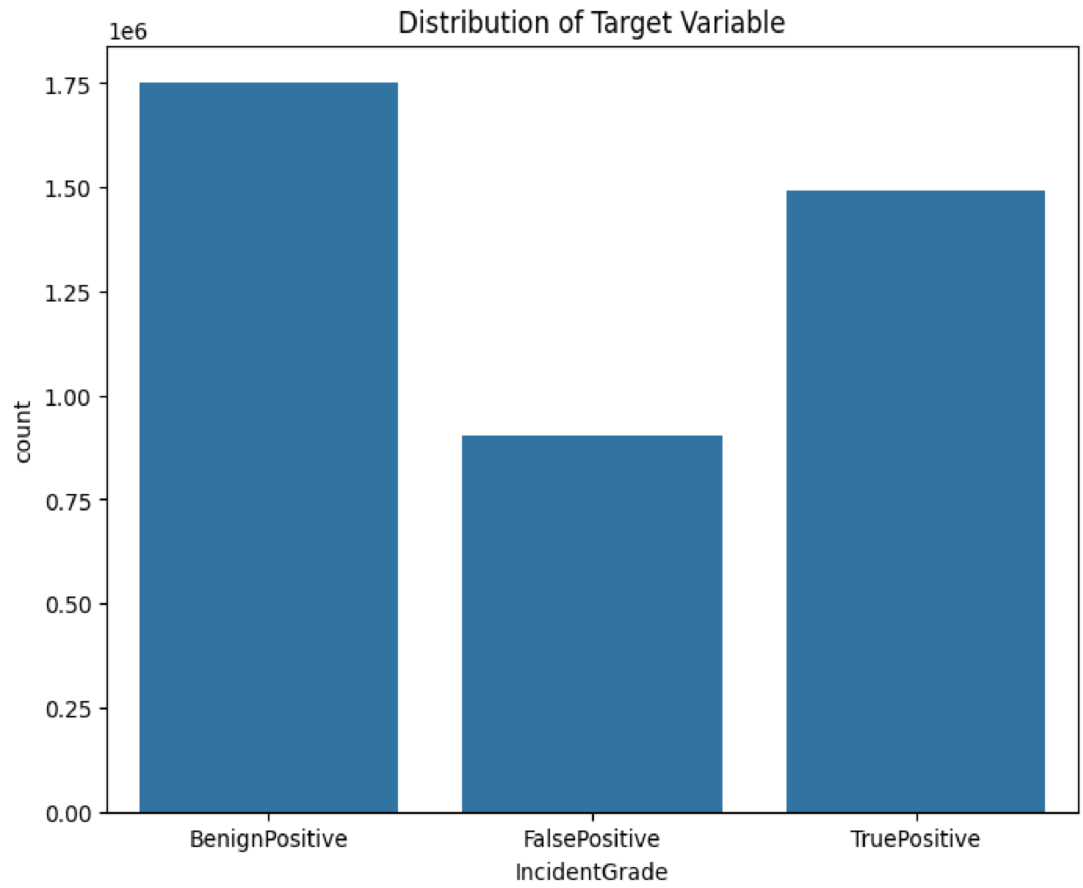
3.1 Overview

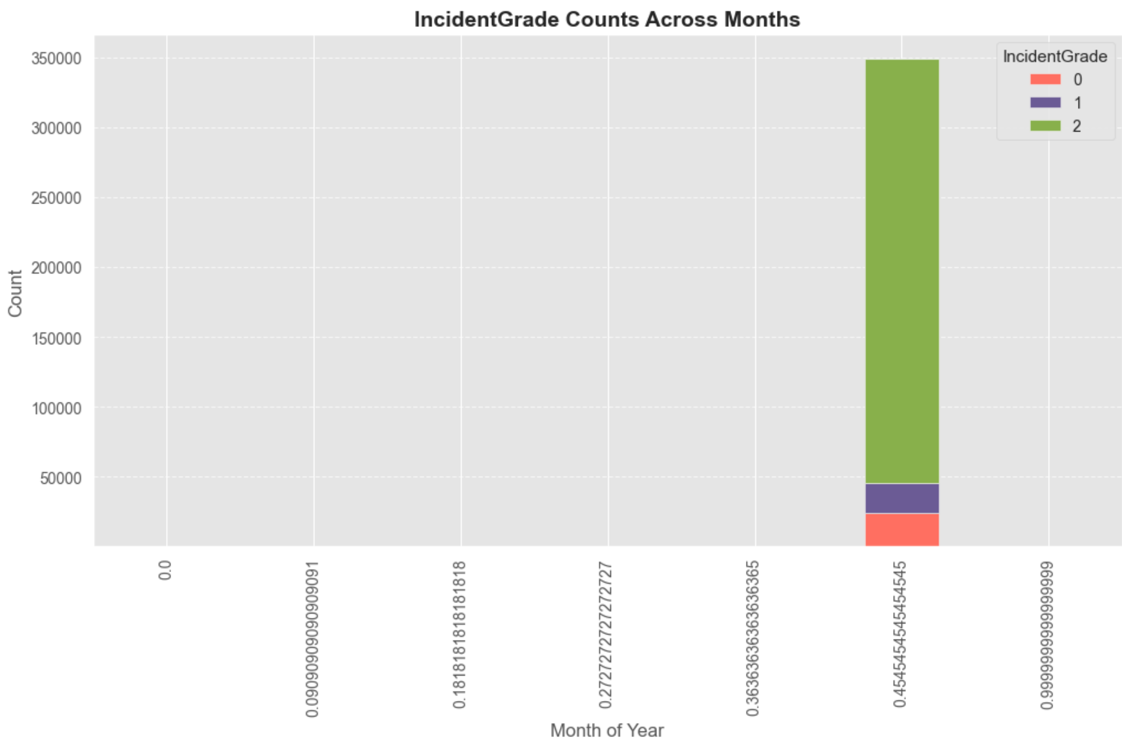
- **Dataset Size:** Large-scale dataset (loaded in chunks).
- **Target Variable:** IncidentGrade (TP, FP, BP).
- **Key Challenges:** Class imbalance, missing values, and high dimensionality.

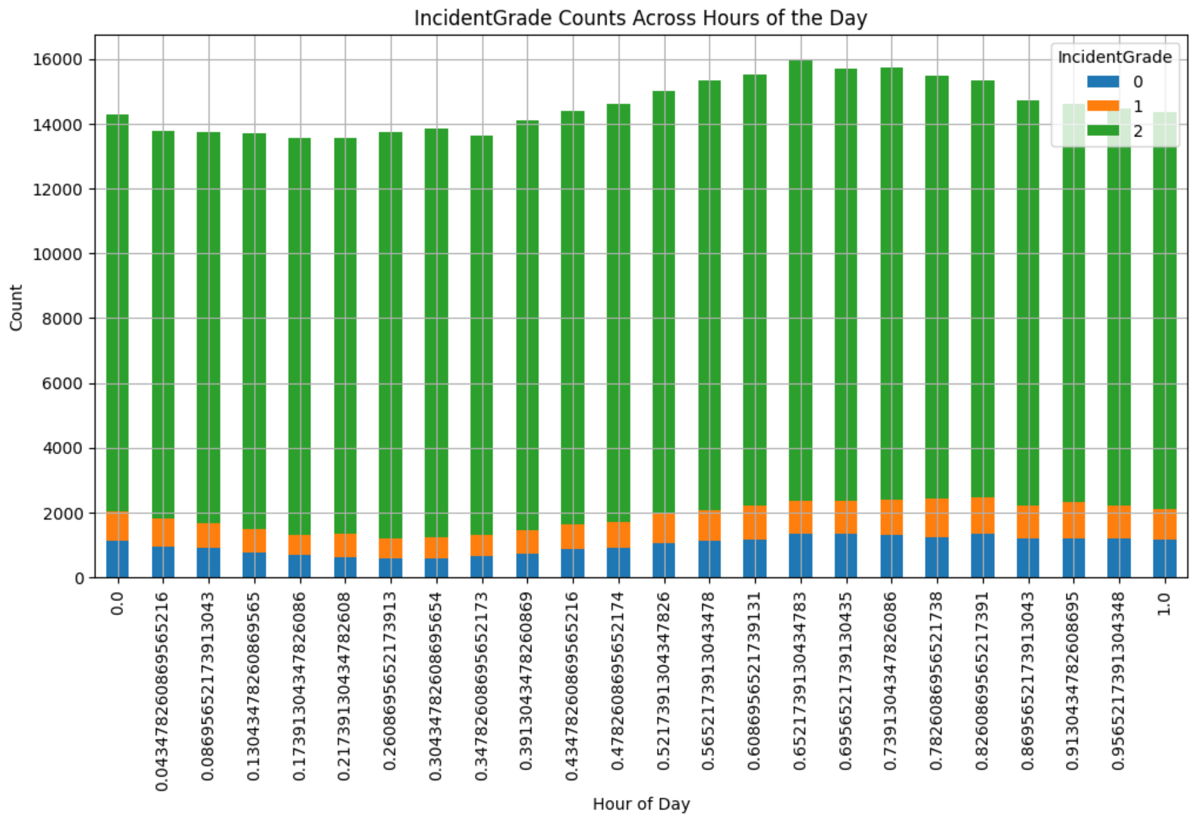
3.2 Key Insights

Feature	**Description**	**Distribution Observations**
Timestamp Features	Time when incidents occurred, captured in Timestamp column	Alerts show spikes during certain hours/days, indicating time-based patterns.
Categorical Features	Alert-related classifications, e.g., AlertTitle, Category	Most incidents fall under **BenignPositive**, showing class imbalance.
Incident Grade	Target variable indicating incident type (IncidentGrade)	Imbalanced distribution: Majority labeled as BenignPositive.

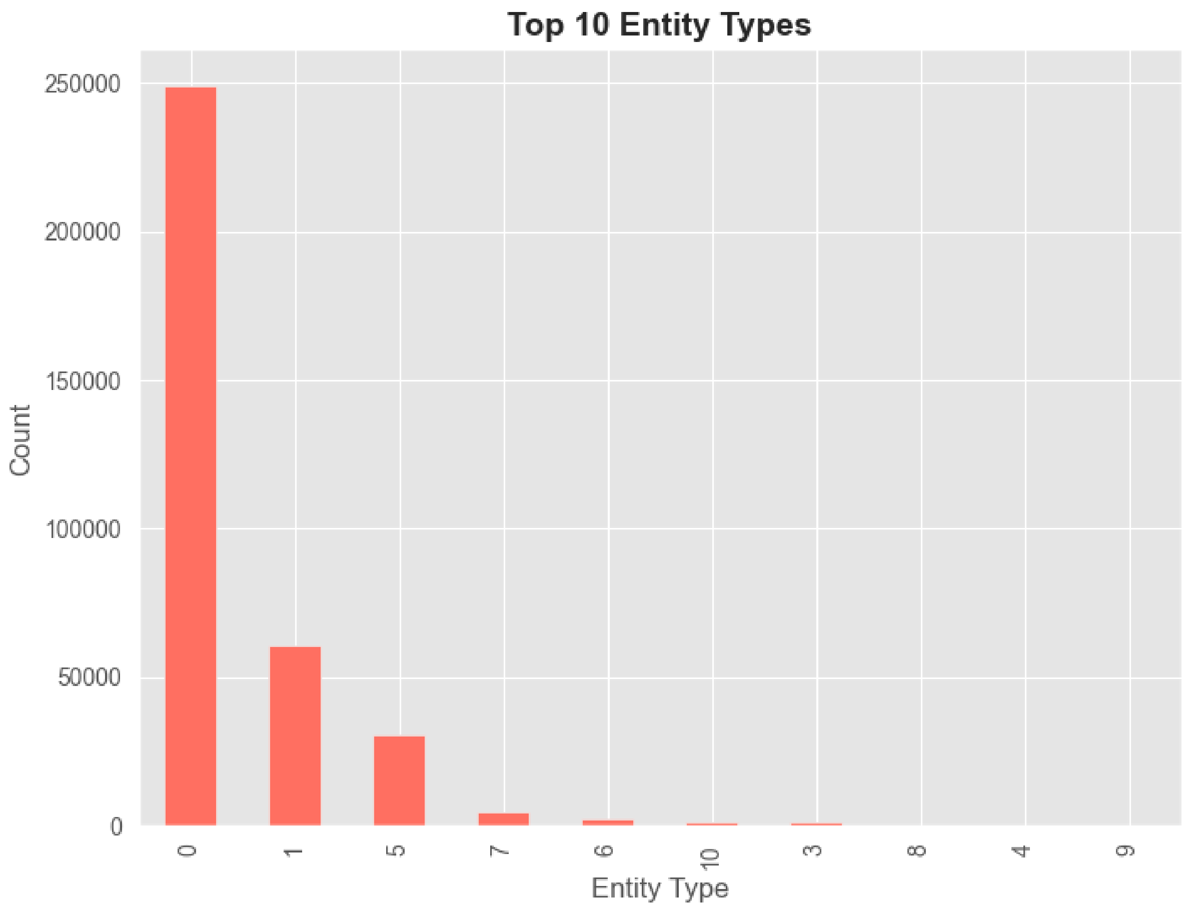
3.3 Visualizations







IncidentGrade Counts Across Hours of the Day
IncidentGrade
0
1
2
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
0.0
0.043478260869565216
0.08695652173913043
0.13043478260869565
0.17391304347826086
0.21739130434782608
0.2608695652173913
0.30434782608695654
0.34782608695652173
0.391304347826087
0.43478260869565216
0.4782608695652174
0.5217391304347826
0.5652173913043478
0.6086956521739131
0.6521739130434783
0.6956521739130435
0.7391304347826086
0.782608695652174
0.8260869565217391
0.8695652173913043
0.9130434782608695
0.9565217391304348
1.0
Hour of Day



Top 10 Entity Types
Count
250000
200000
150000
100000
50000
0
0
1
5
7
6
10
3
8
4
9
Entity Type

Data Preprocessing

4.1 Missing Data Handling

Method	Implementation
Forward Fill	For timestamp-based missing values.
Mean Imputation	For numerical features.
Column Dropping	Removed columns >50% missing.

4.2 Feature Engineering

- Timestamp Features: Created day-of-week and hour-of-day indicators.
- Redundant Columns: Removed features with high correlation (>0.9).

4.3 Encoding and Scaling

Type	Method
Categorical Encoding	One-hot encoding for nominal data.
Scaling	StandardScaler for numerical data.

4.4 Data Splitting

Dataset	**Percentage**
Training Set	80%
Validation Set	20%

Split the data into training and validation sets to evaluate model performance.

Train-Validation Split: Data was split into 80% for training and 20% for validation, while maintaining the balance between classes.

Model Selection and Training

5.1 Models Evaluated

Model	Advantages	Drawbacks
Logistic Regression	Simple baseline model.	Struggles with non-linear data.
Decision Tree	Interpretable and non-linear.	Prone to overfitting.
Random Forest	Accurate and stable.	Computationally expensive.
XGBoost	Handles large datasets effectively.	Needs careful tuning.

• **Logistic Regression:** A simple model used as a baseline for comparison.

• **Decision Tree:** A non-linear model that works well for small datasets and easy interpretability.

• **Random Forest:** An ensemble of decision trees that provides more accuracy and stability.

• **XGBoost:** A powerful algorithm that handles large datasets efficiently.

✓ **Random Forest** was the best-performing model, achieving high accuracy and macroF1 scores.

✓ **XGBoost** also performed well, though slightly lower than Random Forest

5.2 Performance Summary

Model	Accuracy (%)	Macro-F1 Score
Logistic Regression	91	76
Decision Tree	97	91
Random Forest	99	98
XGBoost	99	97

Model Evaluation and Tuning

6.1 Metrics Used

Evaluate the model's performance using cross-validation and optimize it using hyperparameter tuning.

Metrics Used

- **Accuracy:** Measures overall correctness.
- **Precision:** Measures how many positive predictions were correct.
- **Recall:** Measures how well the model identifies actual positives.
- **Macro-F1 Score:** A balanced metric that treats all classes equally

Metric	**Definition**
Accuracy	Overall correctness of predictions.
Precision	Correctness of positive predictions.
Recall	Ability to identify actual positives.
Macro-F1 Score	Balance across all classes.

6.2 Hyperparameter Tuning

RandomizedSearchCV was used to find the best settings for Random Forest and XGBoost

Model	Best Parameters Found
Random Forest	n_estimators=200, max_depth=30
XGBoost	learning_rate=0.2, max_depth=9

Results

7.1 Test Data Performance

Metric	TP	FP	BP
Precision (%)	63	85	100
Recall (%)	94	75	97
Macro-F1 Score (%)	76	80	98

➢ Test the final model on unseen data to ensure it generalizes well.

➢ The Random Forest model was evaluated on the test set, achieving high precision, recall, and macro-F1 scores.

7.2 Classification Report (Test Data)

The classification report below provides detailed performance metrics for each category on the test dataset

Classification Report on Test Data:

	precision	recall	f1-score	support
0	0.63	0.94	0.76	24124
1	0.85	0.75	0.80	21252
2	1.00	0.97	0.98	303765
accuracy			0.95	349141
macro avg	0.83	0.89	0.85	349141
weighted avg	0.96	0.95	0.96	349141

Macro-F1 Score: 0.85

Macro Precision: 0.83

Macro Recall: 0.89

Confusion Matrix on Test Data:

```
[[ 22719   1311     94]
 [  4552  15998    702]
 [  8732   1429 293604]]
```

7.3 Key Takeaways

- **Random Forest** consistently outperformed other models.
- **Accuracy:** 97% on test data.
- **Macro-F1 Score:** 93%, ensuring balanced class performance.

Classification Report:

	precision	recall	f1-score	support
0	0.79	0.95	0.86	11720
1	0.93	0.94	0.93	13464
2	1.00	0.98	0.99	126942
accuracy			0.97	152126
macro avg	0.90	0.96	0.93	152126
weighted avg	0.98	0.97	0.97	152126

Confusion Matrix:

[[11082   604   34]

[671 12676   117]

[2353   404 124185]]

Conclusion

The developed machine learning model has successfully classified cybersecurity incidents with high precision and recall, addressing the primary challenges faced by SOCs:

1. Automated triage of alerts.
2. Reduced manual efforts.
3. Improved focus on true threats

Future Scope

1. Integration with live SOC environments for real-time alert handling.
2. Exploration of deep learning models for further improvements.
3. Addressing evolving threat landscapes by updating the model regularly.