Microsoft: Classifying Cybersecurity Incidents with Machine Learning

Project Overview

This project focuses on using **Machine Learning** techniques to classify cybersecurity incidents, enhancing threat detection and response. By leveraging historical data, machine learning models can be trained to classify various types of cybersecurity incidents, including malware attacks, phishing attempts, and unauthorized access.

The project is built with **Python**, using libraries such as **scikit-learn**, **pandas**. It also involves model evaluation and feature engineering to ensure optimal performance.

Requirements

- **Python 3.x**
- **scikit-learn**
- **pandas**
- **NumPy**
- **Jupyter Notebook**

Installation

1. **Clone the repository:**

   ```
   bash
   Copy code
   git clone https://github.com/Shabanabacker/cybersecurity-incident-classification.git
   cd cybersecurity-incident-classification
   ```

2. **Project Structure:**

```
bash
Copy code
cybersecurity-incident-classification/
│
├── data/                          # Folder containing datasets
│   ├── GUIDE_Test.csv.zip         # Cybersecurity incident data
    ├── GUIDE_Train.csv.zip
│
├── notebooks/
│   ├── microsoft.ipynb            #Jupyter notebook for model building
├
│
│
├── README.md                      # Project overview and instructions
├
└── Cybersecurity_Classification_Documentation.pdf # Full project documentation
```

Machine Learning Models

Multiple machine learning models were trained to classify cybersecurity incidents. These models include:

- **Logistic Regression**
- **Decision Tree Classifier**
- **Random Forest Classifier**

The models were evaluated using common metrics such as **accuracy**, **precision**, **recall** and **F1 score**.

Model Training Process

1. **Data Preprocessing:**
    - **Handling Missing Values:** Impute missing data using techniques like **mean imputation** for numerical features and **mode imputation** for categorical features.
    - **Feature Encoding:** Encode categorical variables using **label encoding**.
    - **Data Splitting:** Split the dataset into training and testing sets using an 80/20 ratio.
2. **Model Training:**
    - Use **scikit-learn** to train machine learning models.

Best Model

The **Random Forest Classifier** achieved the highest accuracy on the test set. The model was saved as a `.pkl` file for future use.

Usage

Run the Jupyter Notebook: To view the data analysis and model training process, open and run the provided Jupyter notebook:

```
bash
Copy code
jupyter notebook notebooks/microsoft.ipynb
```

Results and Insights

Model Performance

Model	Train Accuracy	Test Accuracy	Precision (Test)	Recall (Test)	F1 Score (Test)
Logistic Regression	47.5%	47.5%	31.34%	38.89%	33.75%
Decision Tree Classifier	100%	99.68%	99.65%	99.66%	99.66%
Random Forest Classifier	100%	98.62%	98.58%	98.46%	98.52%

1. **Model Performance:**
   - The **Random Forest Classifier** achieved an accuracy of 92% on the test data, with high precision and recall for detecting malware incidents.

     Accuracy: 0.9141750996626802
     Precision: 0.9167605633444288
     Recall: 0.9038889796279505
     f1 score: 0.9095150559855404

2. **Class Imbalance:**
   - There was a noticeable class imbalance in the dataset.**Oversampling** was applied to address this.

Acknowledgements

- **scikit-learn:** Used for building machine learning models.
- **pandas:** Used for data manipulation and preprocessing.

Contact

For any questions or feedback, please reach out to:

[Shabana Backer P]
Email: shabana.backer@gmail.com