Configuration Manual

MSc Research Project

Cyber Security

Krishna Reddy Karri

Student ID: X23224215

School of Computing

National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Krishna Reddy karri ……. ……………………………………………………………………………………………………
Student ID:	X23224215 ……………………………………………………………………………………………………..……
Programme:	……CyberSecurity……………………………………………… ………
Year:	………2024……………… …..
Module:	……Msc Cyber Security Practicum-2 …………………………………………………………………………………………….………
Lecturer:	…………Khadija Hafeez ………………………………………………………………………………………….………
Submission Due Date:	………12-12-24 …………………………………………………………………………………………….………
Project Title:	……………………………Insider Threats in Cyber Security ……………………………………………………………………….………
Word Count:	…………………1133……………………
Page Count:	…………………………………….……..………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	krishnareddykarri ……………………………………………………………………………………………………
Date:	…12-12-2024

……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Krishna Reddy Karri

X23224215

1 Project Overview

This project addresses the detection of insider threats through the analysis of security incident data using machine learning techniques. It combines behavioral analytics, anomaly detection, and supervised learning to identify suspicious patterns and assign risk levels.

Key Objectives:

1. Detect anomalies in user behavior through Isolation Forest and One-Class SVM.
2. Use Random Forest classification to predict the incident grades.
3. Present insights in the form of graphs and charts for easier understanding.

Technologies and Tools:

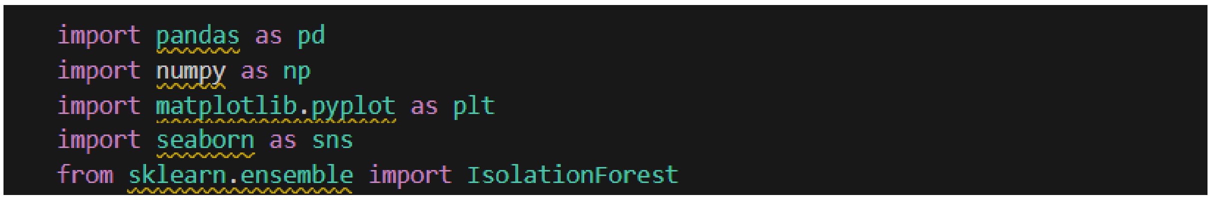
- Programming Language: Python
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn
- Dataset: Microsoft security incidents (GUIDE_Train.csv)

Code Setup and Execution

This section will make it clear to the reader how to set up and run the code.

1. **Environment Setup:**

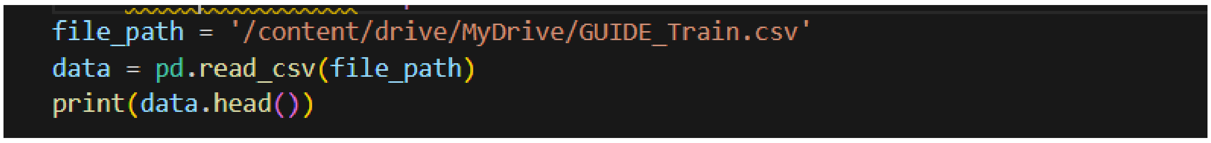
- Install Python (version 3.7 or higher).
- Use the following command to install dependencies:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
```

Dataset:

Ensure that the GUIDE_Train.csv is placed in the same working directory as the code.

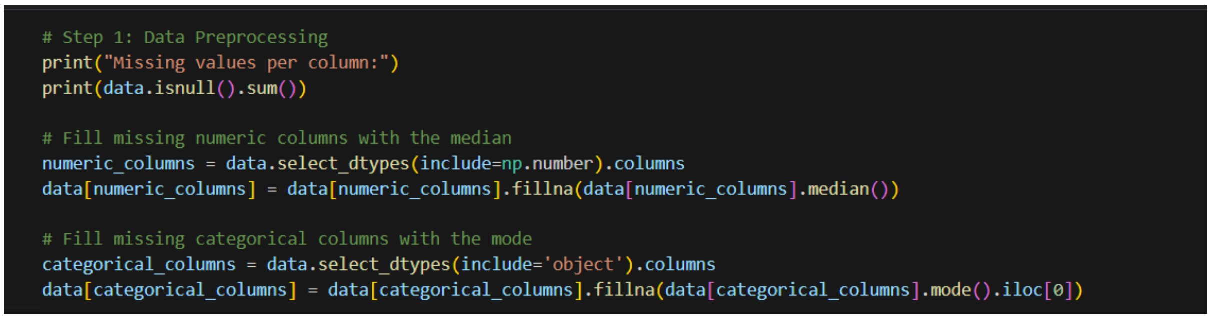


```python
file_path = '/content/drive/MyDrive/GUIDE_Train.csv'
data = pd.read_csv(file_path)
print(data.head())
```

Execution Steps:

a. Data Preprocessing:

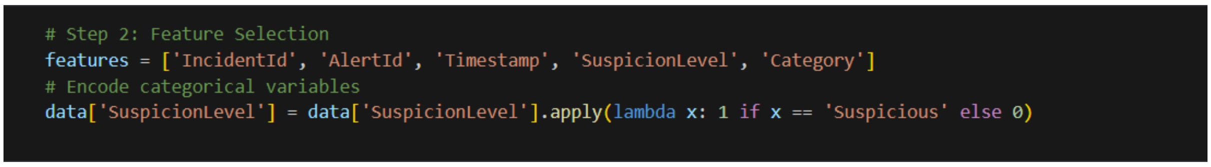
Impute the missing data and perform the encoding of the categorical variables.



```python
# Step 1: Data Preprocessing
print("Missing values per column:")
print(data.isnull().sum())

# Fill missing numeric columns with the median
numeric_columns = data.select_dtypes(include=np.number).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].median())

# Fill missing categorical columns with the mode
categorical_columns = data.select_dtypes(include='object').columns
data[categorical_columns] = data[categorical_columns].fillna(data[categorical_columns].mode().iloc[0])
```

Feature Selection:

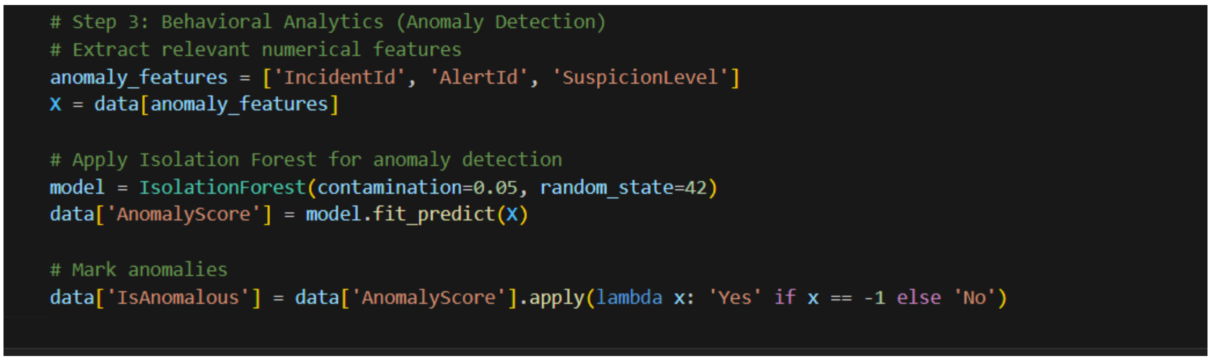
- Confine the analysis to fields such as IncidentId, AlertId and SuspicionLevel.



```python
# Step 2: Feature Selection
features = ['IncidentId', 'AlertId', 'Timestamp', 'SuspicionLevel', 'Category']
# Encode categorical variables
data['SuspicionLevel'] = data['SuspicionLevel'].apply(lambda x: 1 if x == 'Suspicious' else 0)
```

Anomaly Detection:

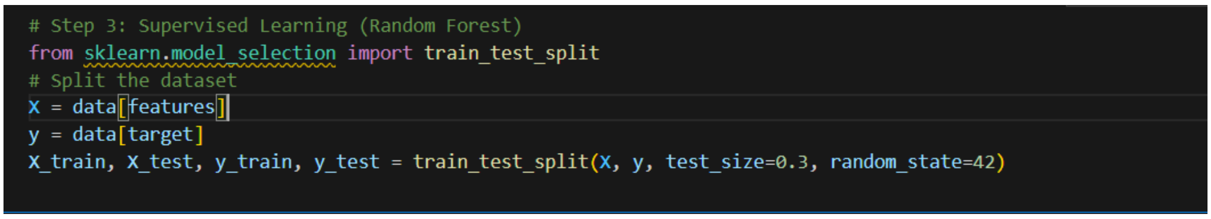
Apply Isolation Forest for the purpose of outlier identification.



```python
# Step 3: Behavioral Analytics (Anomaly Detection)
# Extract relevant numerical features
anomaly_features = ['IncidentId', 'AlertId', 'SuspicionLevel']
X = data[anomaly_features]

# Apply Isolation Forest for anomaly detection
model = IsolationForest(contamination=0.05, random_state=42)
data['AnomalyScore'] = model.fit_predict(X)

# Mark anomalies
data['IsAnomalous'] = data['AnomalyScore'].apply(lambda x: 'Yes' if x == -1 else 'No')
```

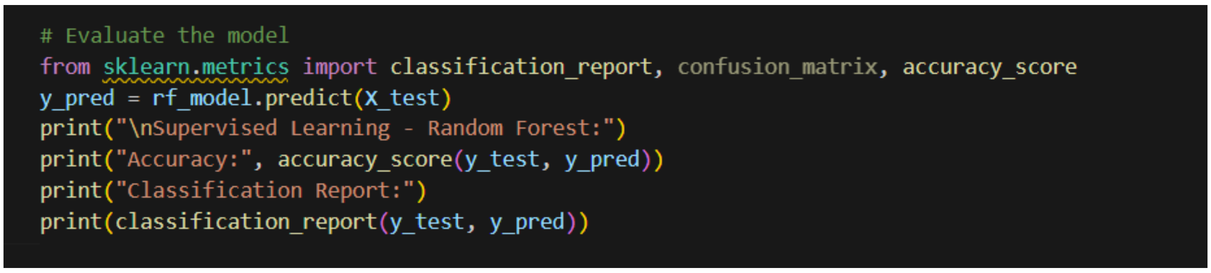
Supervised Learning

Train and Test a Random Forest model for incident classification.



```python
# Step 3: Supervised Learning (Random Forest)
from sklearn.model_selection import train_test_split
# Split the dataset
X = data[features]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Assess the performance of the model that has been built.

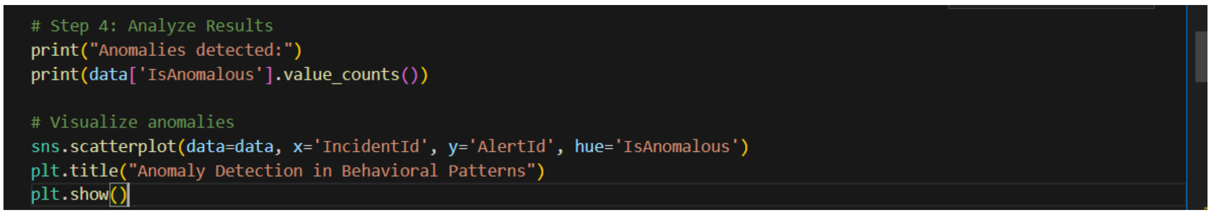


```python
# Evaluate the model
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
y_pred = rf_model.predict(X_test)
print("\nSupervised Learning - Random Forest:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Visualization

- Create scatter plots, bar graphs and pie charts for data analysis.

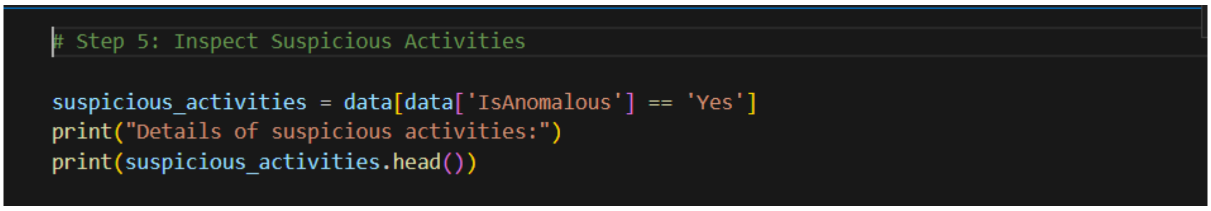
.



```python
# Step 4: Analyze Results
print("Anomalies detected:")
print(data['IsAnomalous'].value_counts())

# Visualize anomalies
sns.scatterplot(data=data, x='IncidentId', y='AlertId', hue='IsAnomalous')
plt.title("Anomaly Detection in Behavioral Patterns")
plt.show()
```

Inspect Suspicious Activities

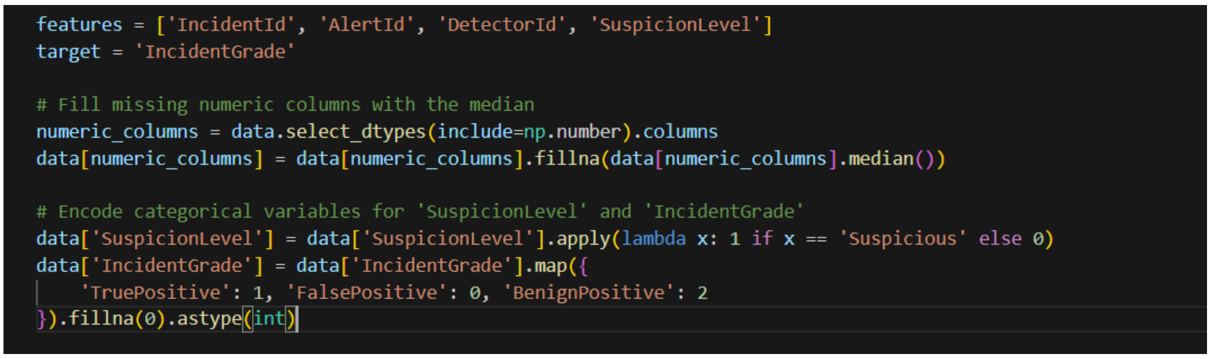
This step selects and presents information on the activities that have been flagged as anomalies.



```python
# Step 5: Inspect Suspicious Activities

suspicious_activities = data[data['IsAnomalous'] == 'Yes']
print("Details of suspicious activities:")
print(suspicious_activities.head())
```

Feature Preparation and Encoding

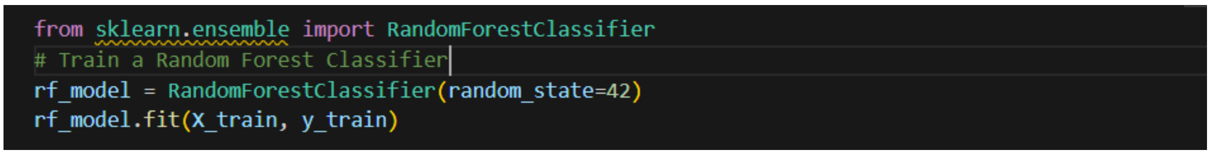
This step preprocesses the data by dealing with missing values and converting categorical data into numerical data for analysis.



```python
features = ['IncidentId', 'AlertId', 'DetectorId', 'SuspicionLevel']
target = 'IncidentGrade'

# Fill missing numeric columns with the median
numeric_columns = data.select_dtypes(include=np.number).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].median())

# Encode categorical variables for 'SuspicionLevel' and 'IncidentGrade'
data['SuspicionLevel'] = data['SuspicionLevel'].apply(lambda x: 1 if x == 'Suspicious' else 0)
data['IncidentGrade'] = data['IncidentGrade'].map({
    'TruePositive': 1, 'FalsePositive': 0, 'BenignPositive': 2
}).fillna(0).astype(int)
```

Train the Random Forest Classifier

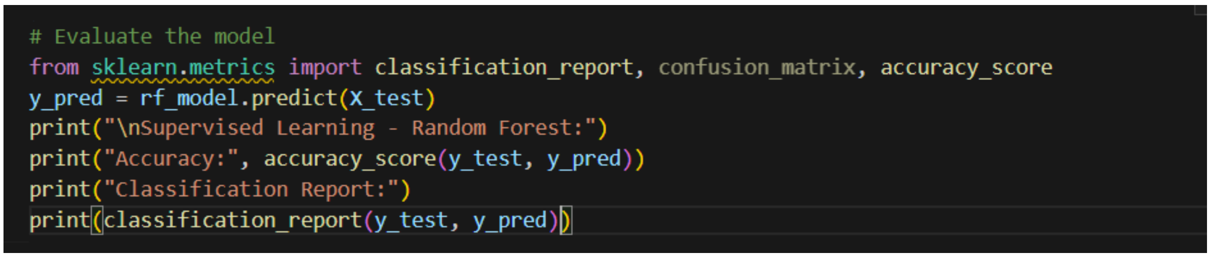
This step involves training a Random Forest Classifier to predict the IncidentGrade using selected features.



```python
from sklearn.ensemble import RandomForestClassifier
# Train a Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

Evaluate the Random Forest Classifier

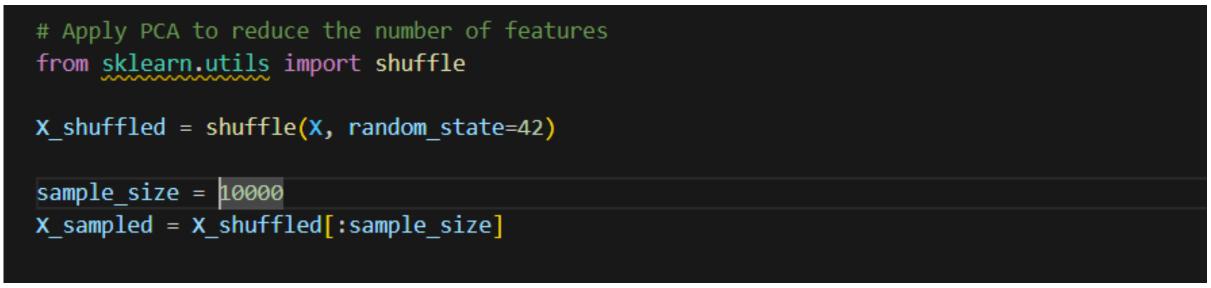
This step preprocesses the data by dealing with missing values and converting categorical data into numerical data for analysis.



```python
# Evaluate the model
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
y_pred = rf_model.predict(X_test)
print("\nSupervised Learning - Random Forest:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Feature Reduction with PCA Preparation

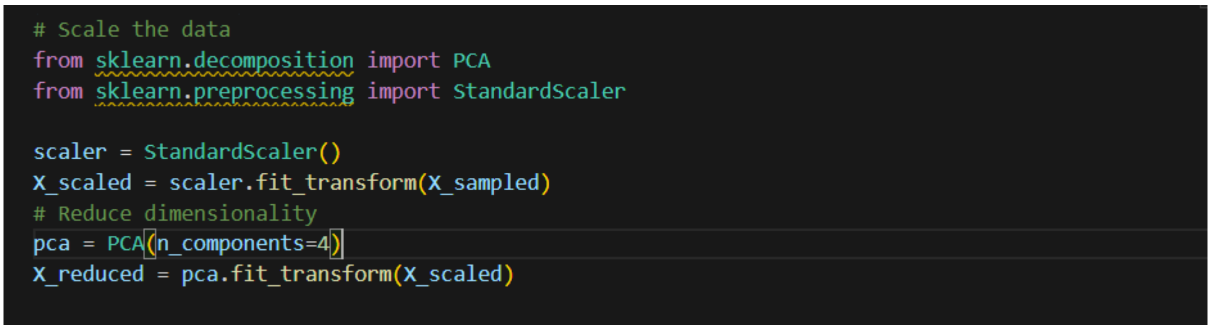
This step is to arrange the dataset and select the features randomly for the purpose of feature selection and dimensionality reduction for Principal Component Analysis (PCA).



```python
# Apply PCA to reduce the number of features
from sklearn.utils import shuffle

X_shuffled = shuffle(X, random_state=42)

sample_size = 10000
X_sampled = X_shuffled[:sample_size]
```

Dimensionality Reduction with PCA

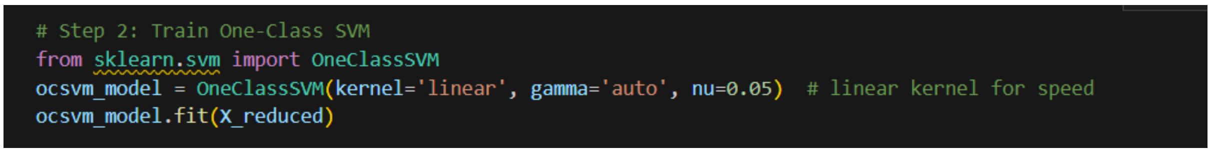
This step involves standardizing the data and then applying PCA to down select the feature set to a manageable set of principal components.



```
# Scale the data
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_sampled)
# Reduce dimensionality
pca = PCA(n_components=4)
X_reduced = pca.fit_transform(X_scaled)
```

Train One-Class SVM

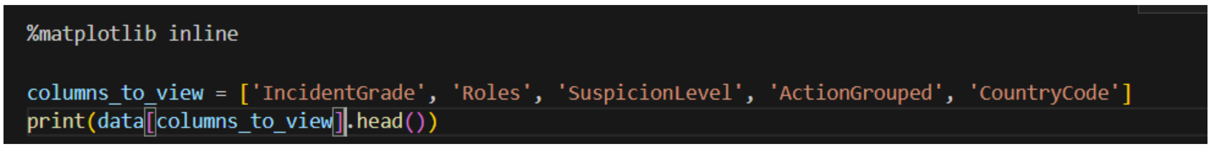
This step uses a One-Class Self learning model based on the reduced feature set to identify anomalies.



```
# Step 2: Train One-Class SVM
from sklearn.svm import OneClassSVM
ocsvm_model = OneClassSVM(kernel='linear', gamma='auto', nu=0.05)  # linear kernel for speed
ocsvm_model.fit(X_reduced)
```

Data Inspection and Visualization Preparation

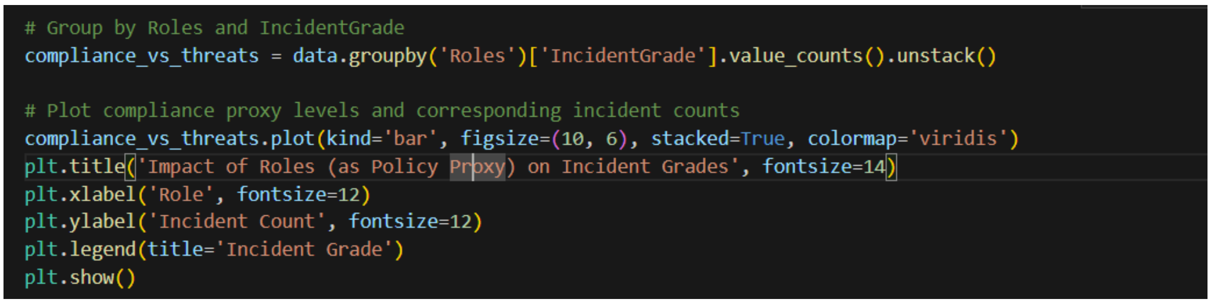
This step focuses on checking some critical columns of the dataset to get an insight of the structure and format for visualization.



```
%matplotlib inline

columns_to_view = ['IncidentGrade', 'Roles', 'SuspicionLevel', 'ActionGrouped', 'CountryCode']
print(data[columns_to_view].head())
```

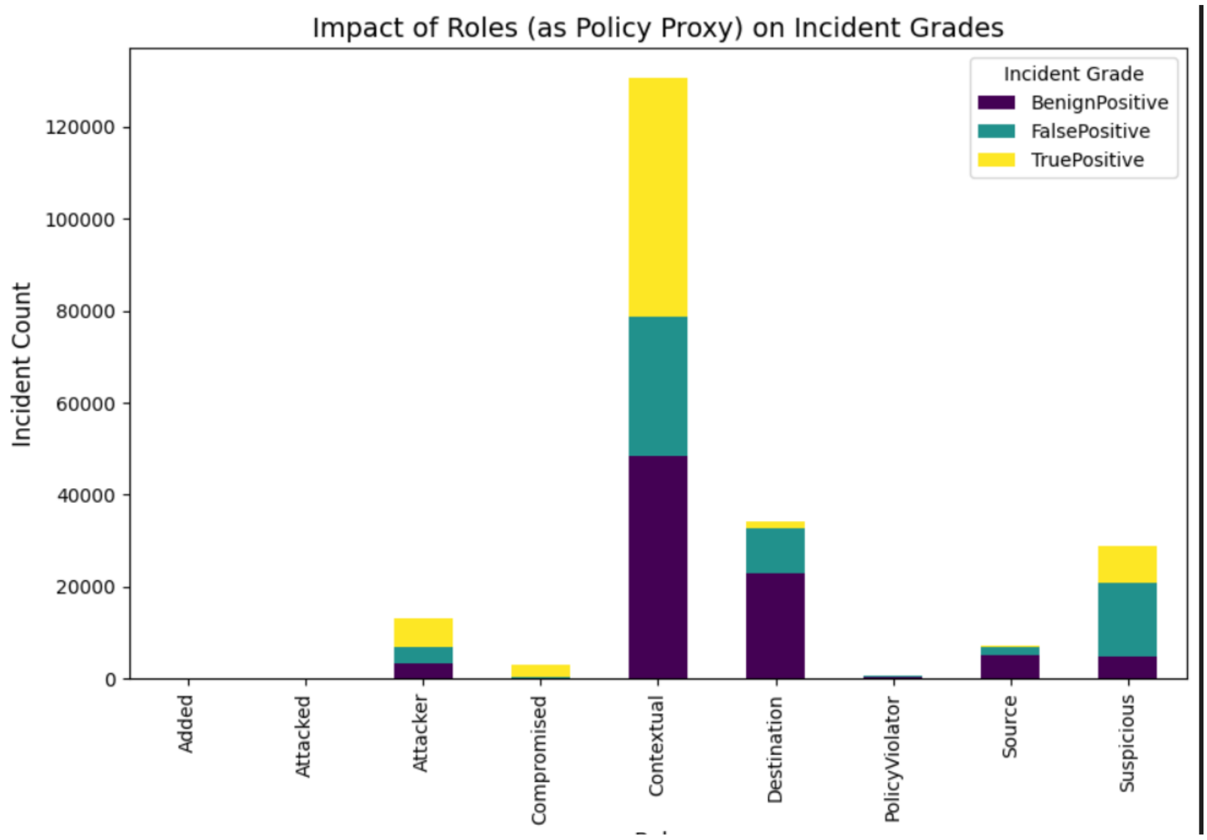
Analyze Impact of Roles on Incident Grades

Here, we focus at grouping the data in accordance to their roles and incident grades to assess compliance levels and make it possible to visualize the results.



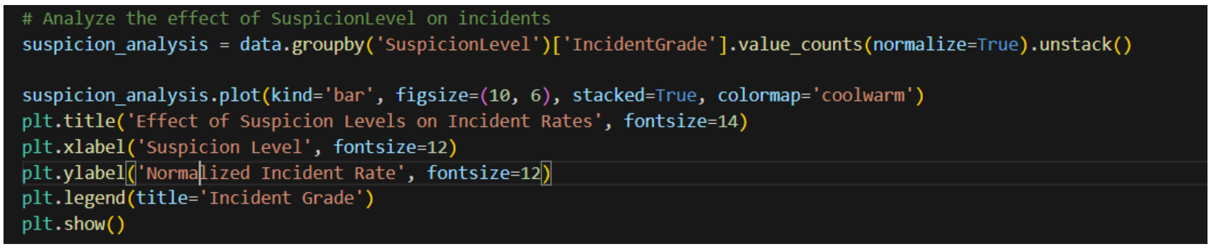
```python
# Group by Roles and IncidentGrade
compliance_vs_threats = data.groupby('Roles')['IncidentGrade'].value_counts().unstack()

# Plot compliance proxy levels and corresponding incident counts
compliance_vs_threats.plot(kind='bar', figsize=(10, 6), stacked=True, colormap='viridis')
plt.title('Impact of Roles (as Policy Proxy) on Incident Grades', fontsize=14)
plt.xlabel('Role', fontsize=12)
plt.ylabel('Incident Count', fontsize=12)
plt.legend(title='Incident Grade')
plt.show()
```

The code snippet above is used to create a stacked bar chart to show the relationship between roles and incident grades.



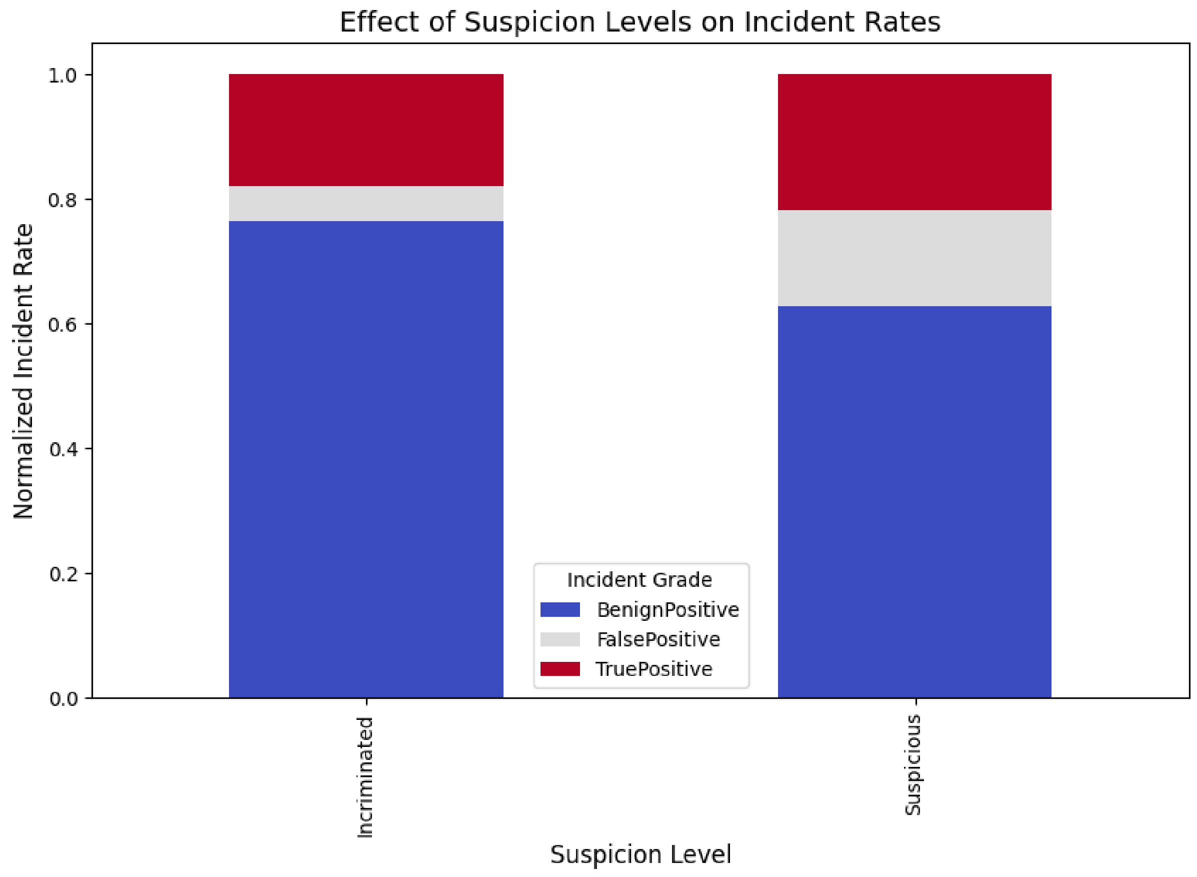
Analyze the Effect of Suspicion Level on Incident Rates

This step examines how different suspicion levels impact the rates of incident grades, visualized using a bar chart.



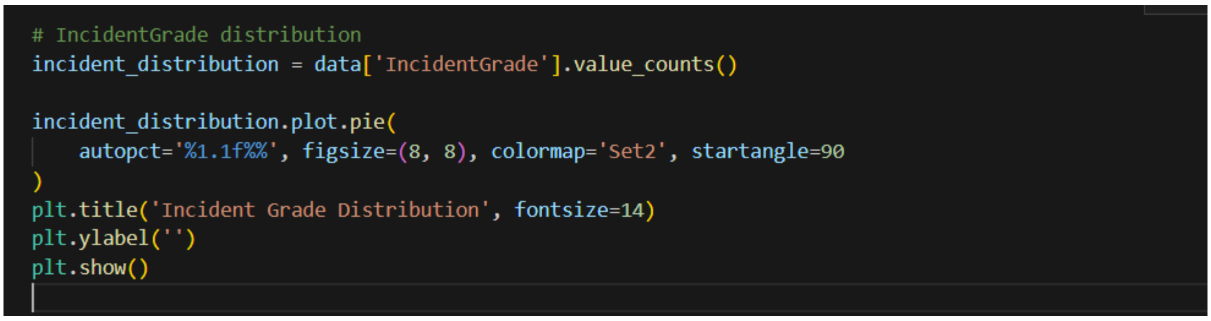
```python
# Analyze the effect of SuspicionLevel on incidents
suspicion_analysis = data.groupby('SuspicionLevel')['IncidentGrade'].value_counts(normalize=True).unstack()

suspicion_analysis.plot(kind='bar', figsize=(10, 6), stacked=True, colormap='coolwarm')
plt.title('Effect of Suspicion Levels on Incident Rates', fontsize=14)
plt.xlabel('Suspicion Level', fontsize=12)
plt.ylabel('Normalized Incident Rate', fontsize=12)
plt.legend(title='Incident Grade')
plt.show()
```

Groups data by SuspicionLevel and calculates the normalized counts for IncidentGrade for comparative analysis.Displays a stacked bar chart to show how different suspicion levels correlate with incident grades.



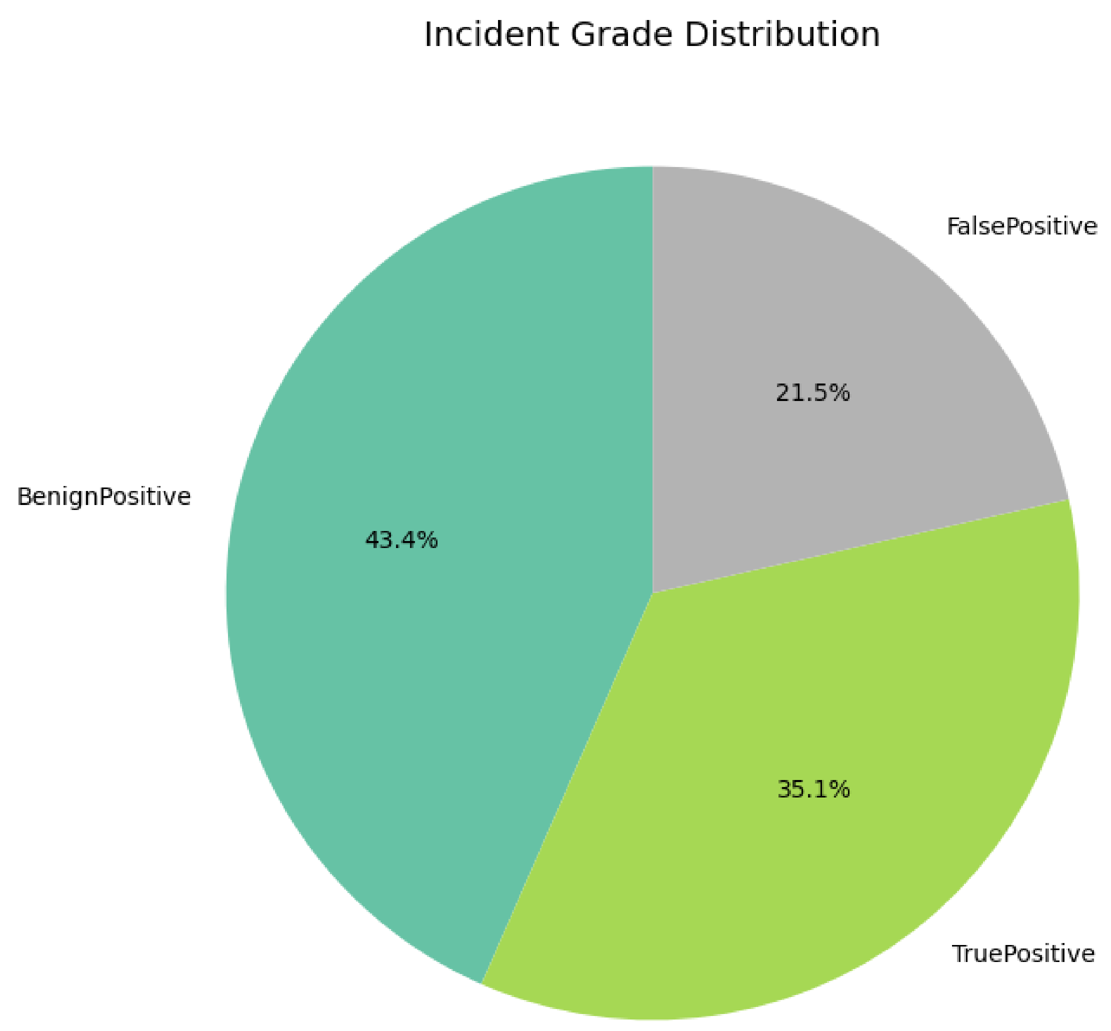
Visualize Incident Grade Distribution

This step shows the distribution of IncidentGrade across the dataset using a pie chart for a clear representation.



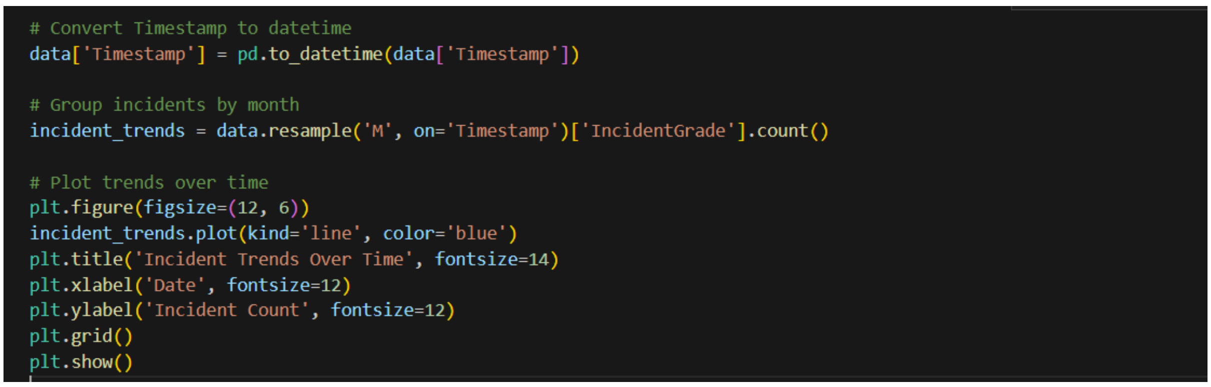
```python
# IncidentGrade distribution
incident_distribution = data['IncidentGrade'].value_counts()

incident_distribution.plot.pie(
    autopct='%1.1f%%', figsize=(8, 8), colormap='Set2', startangle=90
)
plt.title('Incident Grade Distribution', fontsize=14)
plt.ylabel('')
plt.show()
```

The distribution of IncidentGrade values is calculated using value_counts().A pie chart is generated to show the proportion of each IncidentGrade category.



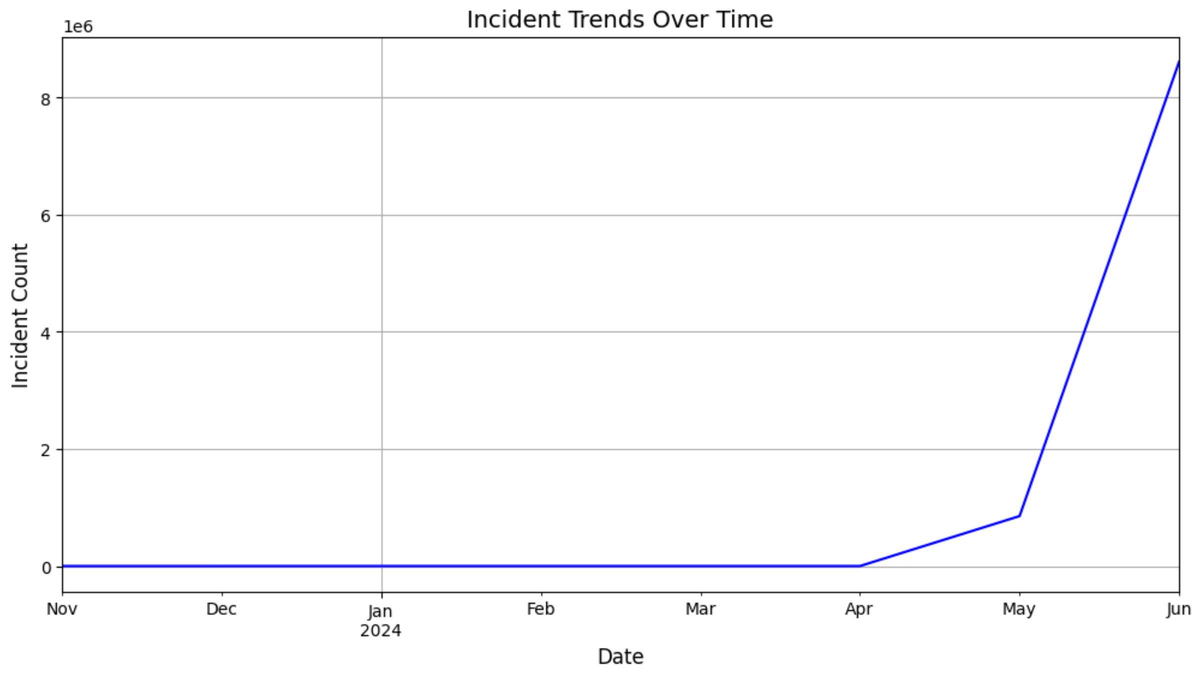
Analyze Incident Trends Over Time

This step converts timestamps to datetime format, groups incidents by month, and plots trends to analyze patterns over time.



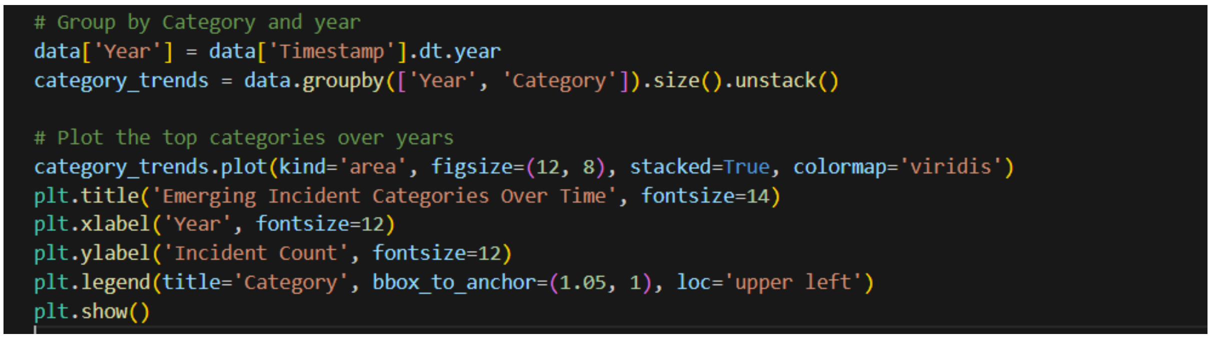
```python
# Convert Timestamp to datetime
data['Timestamp'] = pd.to_datetime(data['Timestamp'])

# Group incidents by month
incident_trends = data.resample('M', on='Timestamp')['IncidentGrade'].count()

# Plot trends over time
plt.figure(figsize=(12, 6))
incident_trends.plot(kind='line', color='blue')
plt.title('Incident Trends Over Time', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Incident Count', fontsize=12)
plt.grid()
plt.show()
```

Groups and counts incidents monthly to detect trends over time.A line plot illustrates changes in incident counts.



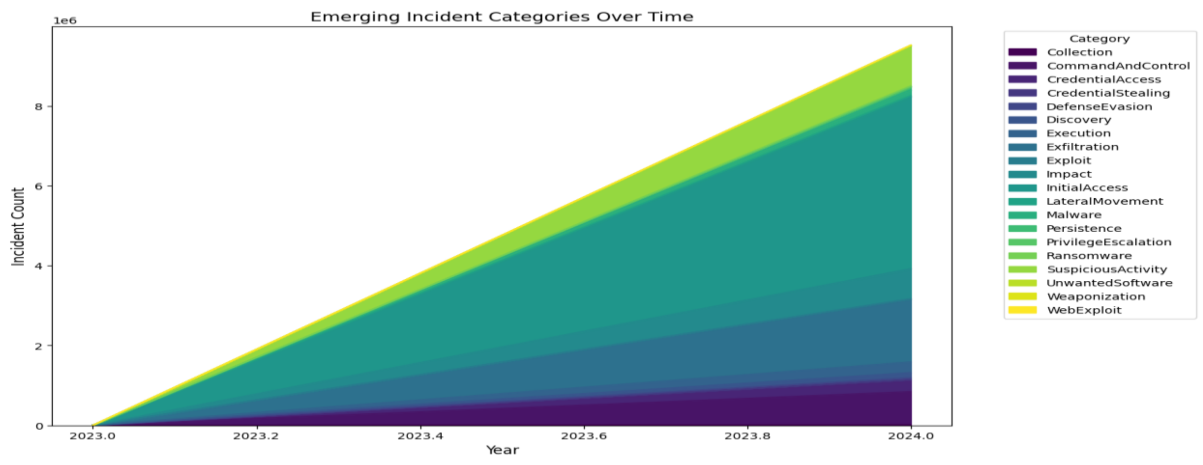
Analyze Emerging Incident Categories Over Time

This step groups the data by year and category, then visualizes the top incident categories over the years using an area plot.



```
# Group by Category and year
data['Year'] = data['Timestamp'].dt.year
category_trends = data.groupby(['Year', 'Category']).size().unstack()

# Plot the top categories over years
category_trends.plot(kind='area', figsize=(12, 8), stacked=True, colormap='viridis')
plt.title('Emerging Incident Categories Over Time', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Incident Count', fontsize=12)
plt.legend(title='Category', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

Groups data by Year and Category and counts the incidents.An area plot illustrates the distribution and trends of incident categories over time.



2 Section 3

Results and Interpretation

Analyzing the Results

The exploration of the dataset through various methods provides important findings about security incidents and their trends.

Anomaly Detection Analysis: The Isolation Forest algorithm was used to identify anomalies and a subset of data was flagged as unusual and investigated for possible insider threats. The scatterplot visualization evidenced these anomalies since it gave a clear indication of the data points that do not follow the usual pattern.

Inspection of Suspicious Activities: filtering and displaying the details of the suspicious activities have given the first records that contain some anomalies. This was important to establish the type of events that had been flagged as suspicious and the extent of the impact.

Feature Preparation and Encoding: The missing values were imputed and the categorical variables such as SuspicionLevel and IncidentGrade were encoded for the purpose of analysis. This step made it easier for the data to be fed to the machine learning models, thus improving the accuracy of the models.

Training the Random Forest Classifier: To predict IncidentGrade, a Random Forest model was built with features like IncidentId, AlertId, DetectorId, and SuspicionLevel. The model was evaluated to determine the accuracy of the classification of the incidents and the classification report provided the precision, recall, and F1 scores for all the grades.

Dimensionality Reduction with PCA: In order to control the computational resources and to facilitate the analysis, PCA was used to decrease the dimensionality of the data. This step ensured that only the important information was kept while at the same time enabling the One-Class SVM model to process data quicker.

One-Class SVM Analysis: The One-Class SVM model was applied to the PCA-reduced data in order to detect anomalies in the data set. This approach was useful in separating the normal data from the abnormal ones, supporting the findings from the Isolation Forest.

Incident Trends Over Time: A frequency distribution of incidents by month showed trends and variation in the number of incidents. This analysis is useful for identifying the intervals of high security activity and the reasons behind them.

Category Analysis Over Years: This is because, the area plot which presented the incident categories over time helped explain trends regarding some of the incidents. This visual analysis can help in policy change as it shows which categories have gained prominence and when.

Interpretation

With the integration of anomaly detection, supervised learning, and trend analysis, there is a holistic approach to security incidents. The model has potential in helping to classify and detect anomalous behaviour that can be used to identify potential insider threats and enhance security measures. These include bar graphs and area plots, which can easily show how incidents have evolved over a certain period and aid in the formulation of strategies to prevent similar occurrences.

References

Xu, H., Pang, G., Wang, Y., & Wang, Y. (2023). Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, *35*(12), 12591-12604.

Xiang, H., Zhang, X., Dras, M., Beheshti, A., Dou, W., & Xu, X. (2023, December). Deep optimal isolation forest with genetic algorithm for anomaly detection. In *2023 IEEE International Conference on Data Mining (ICDM)* (pp. 678-687). IEEE.

Bin Sarhan, B., &Altwaijry, N. (2022). Insider threat detection using machine learning approach. *Applied Sciences*, *13*(1), 259.

Lukito, K., & Ihsan, A. F. (2023, December). Comparison of Isolation Forest and One Class SVM in Anomaly Detection of Gas Pipeline Operation. In *2023 3rd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)* (pp. 118-123). IEEE.