MIT World Peace University

Data Science for Cybersecurity and Forensics
Third Year B. Tech, Semester 6

Implementation of Anamoly Detection using Naive Bayes Classifier in Python

Assignment 6

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 10

April 17, 2024

Contents

1 Aim

Implement Anomaly Detection using the Naive Bayes Classifier in Python.

2 Objectives

1. To understand the concept of Anomaly Detection in Cybersecurity.
2. To implement Anomaly Detection using the Naive Bayes Classifier.

3 Theory

3.1 Anomaly Detection

Anomaly detection, also known as outlier detection, is a critical component of cybersecurity systems aimed at identifying abnormal patterns or behaviors in data that may indicate potential security threats or malicious activities. The primary objective of anomaly detection is to distinguish between normal and abnormal instances, allowing security analysts to investigate and mitigate potential risks promptly.

3.1.1 Techniques for Anomaly Detection:

1. **Statistical Methods**: Statistical anomaly detection methods involve analyzing data distributions and identifying instances that deviate significantly from expected norms. For example, if the frequency of login attempts from a specific IP address exceeds a certain threshold, it may indicate a potential brute-force attack.
2. **Machine Learning Approaches**: Machine learning techniques, such as clustering, classification, and density estimation, can also be used for anomaly detection. These methods learn patterns from historical data and classify new instances as normal or anomalous based on their deviation from learned norms.
3. **Behavioral Analysis**: Behavioral anomaly detection focuses on monitoring user behavior and identifying deviations from established patterns. For instance, sudden changes in user access patterns or file access permissions may signal insider threats or unauthorized activities.

3.1.2 Example:

In a cybersecurity context, anomaly detection can help detect various threats, including:

- **Malware Detection**: Anomaly detection algorithms can identify unusual network traffic patterns or system behavior indicative of malware infections.
- **Intrusion Detection**: By analyzing user login patterns, network traffic, and system logs, anomaly detection systems can detect unauthorized access attempts or suspicious activities indicating potential intrusions.
- **Fraud Detection**: In financial systems, anomaly detection can identify unusual transaction patterns, such as large withdrawals or unusual spending behavior, indicating potential fraudulent activities.

3.2 Naive Bayes Classifier

The Naive Bayes classifier is a probabilistic machine learning algorithm based on Bayes' theorem with the "naive" assumption of feature independence. Despite its simplicity and the oversimplified assumption, Naive Bayes is widely used in various fields, including cybersecurity, for its efficiency and effectiveness, especially in text classification tasks.

3.2.1 Working Principle:

The Naive Bayes classifier calculates the probability of a data instance belonging to a particular class based on the observed features. It assumes that the presence of a feature in a class is independent of the presence of any other feature, hence the term "naive." The classifier computes the conditional probability of each class given the features using Bayes' theorem and selects the class with the highest probability as the predicted class for the instance.

3.2.2 Example:

In cybersecurity applications, Naive Bayes classifiers can be used for:

- **Email Spam Detection**: By analyzing the content and metadata of emails, Naive Bayes classifiers can classify incoming emails as spam or non-spam based on features such as keywords, sender information, and email structure.
- **Malicious URL Detection**: Naive Bayes classifiers can analyze URLs and web content to identify malicious websites or phishing attempts based on features such as domain reputation, URL structure, and content similarity to known malicious sites.
- **Network Intrusion Detection**: Naive Bayes classifiers can classify network traffic data as normal or malicious based on features such as packet headers, protocol usage, and traffic patterns, aiding in the detection of network intrusions and cyber attacks.

By leveraging the probabilistic nature of Naive Bayes classifiers and training them on labeled datasets, cybersecurity systems can efficiently classify and mitigate various threats and security risks.

4 Procedure

1. Import the required python packages.
2. Load the dataset.
3. Data analysis.
4. Split the dataset into dependent/independent variables.
5. Split data into Train/Test sets.
6. Train the regression model.
7. Predict the result.

5 Platform

Operating System: Windows 11
IDEs or Text Editors Used: Visual Studio Code
Compilers or Interpreters: Python 3.10.1

6 Requirements

```
python==3.10.1
matplotlib==3.8.3
numpy==1.26.4
pandas==2.2.2
seaborn==0.13.2
```

7 Code

[1]:
```python
# anomaly detection using naive bayes
```

[3]:
```python
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.io import loadmat
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

[14]:
```python
# anomaly detection using naive bayes algorithm
# steps
# 1. load the dataset
# 2. split the dataset into training and testing sets
# 3. train the model
# 4. predict the output
# 5. calculate the accuracy of the model
```

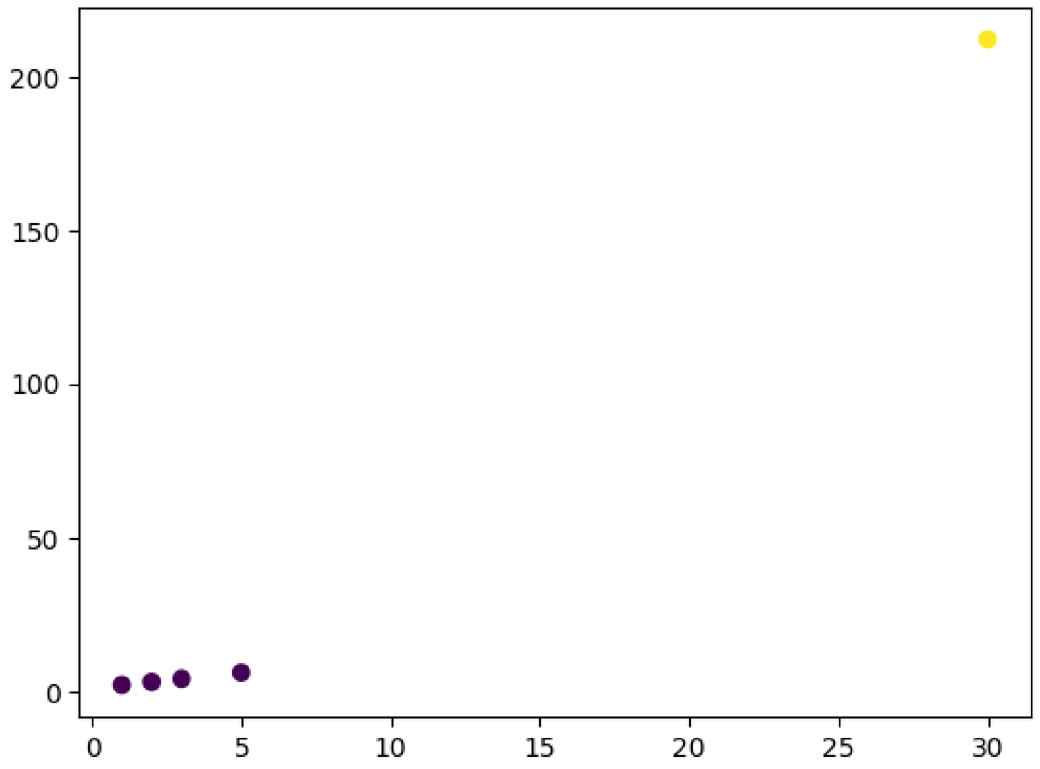
[15]:
```python
# lets create a simple sample dataset
data = {"x": [1, 2, 3, 30, 5], "y": [2, 3, 4, 212, 6], "anomaly": [0, 0, 0,␣
↪1, 0]}

df = pd.DataFrame(data)
print(df)
```

```
    x    y  anomaly
0   1    2        0
1   2    3        0
2   3    4        0
```

```
3  30  212        1
4   5    6        0
```

[16]:
```
# lets plot the data
plt.scatter(df["x"], df["y"], c=df["anomaly"])
```

[16]:
```
<matplotlib.collections.PathCollection at 0x1e1f627ca60>
```



[20]:
```
# lets add more anomalies
data = {
    "x": [1, 2, 3, 30, 5, 100, 200, 300],
    "y": [2, 3, 4, 212, 6, 100, 200, 300],
    "anomaly": [0, 0, 0, 1, 0, 1, 1, 1],
}
df = pd.DataFrame(data)
```

[21]:
```
# lets split the dataset into training and testing sets
X = df[["x", "y"]]
y = df["anomaly"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
    )
```

[22]:
```
# lets train the model
model = GaussianNB()
model.fit(X_train, y_train)
```

[22]:
```
GaussianNB()
```

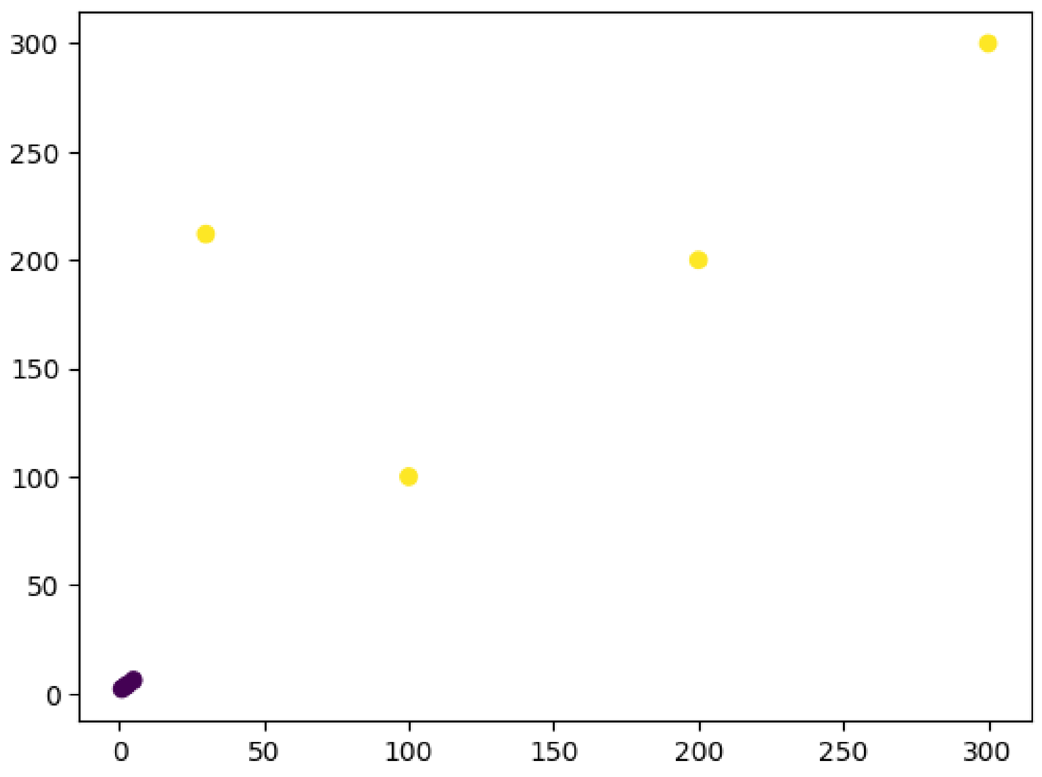
[23]:
```
# lets predict the output
y_pred = model.predict(X_test)
```

[25]:
```
# lets calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)

# other classification metrics
# 1. confusion matrix
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix: ", conf_matrix)
# 2. classification report
from sklearn.metrics import classification_report

class_report = classification_report(y_test, y_pred)

print("Classification Report: ", class_report)
```

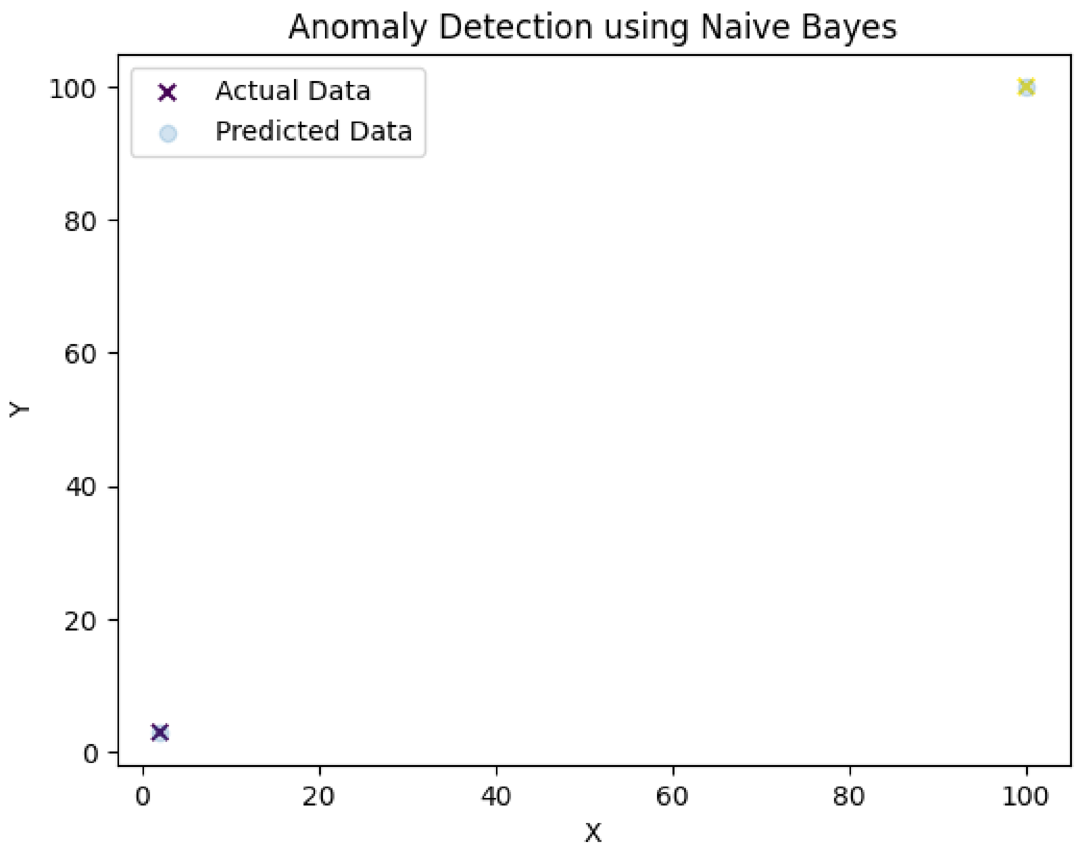
```
Accuracy:  1.0
Confusion Matrix:  [[1 0]
 [0 1]]
Classification Report:                precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

[27]:
```
# lets plot our data
plt.scatter(df["x"], df["y"], c=df["anomaly"])
```

[27]:
```
<matplotlib.collections.PathCollection at 0x1e1f854a230>
```



[41]:
```
# lets plot the y pred
plt.scatter(X_test["x"], X_test["y"], c=y_pred, marker="x")
# lets plot actual data but with different markers
plt.scatter(X_test["x"], X_test["y"], marker="o", alpha=0.2)

# legend
plt.legend(["Actual Data", "Predicted Data", "Actual Data"], loc="best")
plt.title("Anomaly Detection using Naive Bayes")
plt.xlabel("X")
plt.ylabel("Y")


plt.show()
```



[31]:
```
X_test["x"], X_test["y"], y_pred
```

[31]:
```
(1      2
 5    100
 Name: x, dtype: int64,
 1      3
 5    100
 Name: y, dtype: int64,
 array([0, 1], dtype=int64))
```

[32]:
```
# so as is visible here, the anomalies are correctly predicted by the model
```

[ ]:
```
```

8 FAQs

1. **What is Anomaly Detection?**
   - Anomaly detection, also known as outlier detection, is the process of identifying patterns or instances in data that do not conform to expected behavior.
   - Anomalies, or outliers, can be caused by errors in data collection, rare events, or malicious activities.
   - Anomaly detection techniques aim to distinguish between normal and abnormal behavior in data, enabling the identification of potential issues, fraud, or security breaches.
2. **Define and illustrate Naive Bayes.**
   - Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem with the "naive" assumption of independence between features.
   - It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.
   - Despite its simplicity and the oversimplified assumption, Naive Bayes often performs well in practice, especially in text classification tasks.
   - Illustration: Suppose we have a dataset of emails classified as spam or non-spam (ham), and we want to classify a new email as spam or non-spam based on its content. Naive Bayes calculates the probability of an email being spam or non-spam given its features (words) using Bayes' theorem and the naive assumption of feature independence.
3. **How can Naive Bayes Classification help in Anomaly Detection?**
   - Naive Bayes classification can aid in anomaly detection by learning the probability distribution of normal data and identifying instances that deviate significantly from this distribution.
   - While Naive Bayes is primarily used for classification tasks, its probabilistic nature allows it to estimate the likelihood of observing a particular data point given its features.
   - In anomaly detection, Naive Bayes can flag instances with low likelihood scores as potential anomalies, indicating unusual behavior or outliers in the data.
   - By training Naive Bayes on normal data, it can effectively classify new instances as either normal or anomalous based on their feature probabilities.

9 Conclusion

In this assignment, we implemented anomaly detection using the Naive Bayes classifier in Python. We explored the concepts of anomaly detection, Naive Bayes classification, and their applications in cybersecurity. By leveraging the probabilistic nature of Naive Bayes and its independence assumption, we can efficiently detect anomalies and classify data instances based on their deviation from expected norms. Anomaly detection plays a crucial role in cybersecurity systems by identifying potential threats, fraud, or malicious activities, enabling timely responses and mitigations to safeguard critical assets and systems.