PROJECT REPORT

Executive summary

This code is designed for detecting cyber-attacks in a network using data analysis and modeling. The code imports necessary libraries, reads data from text files, explores the data through correlation analysis and data preprocessing, and encodes categorical data using Label Encoder. The dataset is then scaled using the Standard Scaler method and is run on models such as KNN, Decision Tree, and ANN to measure their accuracy, precision, recall, and F1 score. The use of PCA is also explored for reducing dimensions to improve the time taken to predict outcomes. The code also utilizes the K-means clustering algorithm to cluster data without labels and visualize it in scatter plots. The performance of the models is evaluated through comparison using accuracy scores, precision, recall, F1 score, and time taken to predict outcomes. The Decision Tree algorithm performs slightly better than the K-Nearest Neighbors algorithm in terms of accuracy, while both algorithms have good precision scores. The ANN model's performance is dependent on the number of hidden layers used. Overall, this code provides a comprehensive approach to detecting cyber-attacks in a network through data analysis and modeling, with the potential to improve the efficiency of cybersecurity measures.

Introduction

This code is designed for data analysis and modeling to detect cyber-attacks in a network. It imports necessary libraries such as NumPy, pandas, matplotlib, seaborn, and sklearn to facilitate data processing, analysis, visualization, and modeling. It reads data from two text files named Dataset.txt and Attack_types.txt and assigns attack types to each attack category in the data. It then explores the data through correlation analysis, data visualization, and data preprocessing. Finally, it encodes categorical data using Label Encoder and prepares the data for modeling After which the data is run on some models that help us the users to better understand why how and which cyber attacks are

related to which conditions thus helping us avoid them and detect which attack a user may be facing with the help of AI bots which can be fed the data from the models.

Data-preprocessing (extraction and cleaning)

And

Feature Engineering

The dataset used in this code is in a txt format and contains information about network attacks, such as the type of attack, protocol used, and number of packets. The first step in the code is to import the necessary libraries for data manipulation, visualization, and modeling. Then, the dataset is loaded into a Pandas DataFrame object using the read_csv method. After loading the dataset, the next step is to clean the data by dropping columns with missing values and removing columns with low variability. There are 23 classes in the Dataset and we need to convert them into 5 classes which is achieved by Mapping the Attack.txt files column attack type to a new column we make in the Data.txt with the same name. Correlation analysis is performed to identify columns that have strong correlations with each other. These columns are then dropped to avoid redundancy in the dataset. The data is then encoded

PROJECT REPORT

using the Label Encoder class from the sklearn.preprocessing library, which converts categorical data to numerical data. The dataset is then scaled using the Standard Scaler method to ensure all features have equal weight in the modeling process.

We also used a model named **Principal Component Analysis (PCA)** for reducing our dimensions in order to reduce time in running our Project. Although this model was useful in some Models such as KNN where the drop-in time was a lot a notable example would be that when PCA was not used an KNN was run we saw that it took approx. 35 seconds to predict the out come where as it took only 2 seconds or less to predict the label of the data given when PCA was used to reduce 15 dimensions. The accuracy has negligible difference as without PCA Accuracy was 0.9997221670966462 whereas with PCA It was Accuracy was 0.9969438380631077. However other models weren't as forgiving but they did see some drop in Accuracy and time. More description is given in the code of the project.

Use of the given classification and clustering algorithms

classification

Models used:

PROJECT REPORT

- KNN
- Decision Tree
- ANN

How did we use them and what did we measure?

We measured their accuracy, Precision, Recall and F1 score. These readings are taken after use of PCA and without the use of PCA.

- KNN

We separated the data into test and train sections and simply fit the train data. After which we send the test data to the predict section and get predictions and accuracy.

We reduce the dimensions and then split the data when we use PCA and measure the results separately for it.

- Decision Tree

In our approach, we first divided the available data into two sets, namely the train and test sets. Next, we trained our model

using the train set and subsequently evaluated its performance on the test set by predicting the outcomes and measuring accuracy.

To further improve the model's performance, we also utilized PCA to reduce the dimensions of the data. This involved dividing the data into train and test sets separately for PCA and measuring the results individually for each set.

- ANN
  - What model did we use?
    - We used an MLP
  - What activation function did we use?
    - We used tanh
  - Why did we use it?
    - Our data after standardization mostly ranged from -1 to 1 thus using tanh seemed the best fit which was proven more with tests and such to see the result.

Clustering

K-means

We first dropped the column containing the labels after which we fitted the rest of the data

into the model giving it 5 clusters to make and assign labels too the labels were then converted into the attacks given by datasets according to the number of times they appeared and made to represent in scatter plots to visualize it.

Comparison and Performance Evaluation:

a1. Classification of Cyber Attacks Using Decision Tree Algorithm without reducing dimensions

```
1.0
Accuracy: 0.9997221670966462
Precision: 0.9997221670966462
Recall: 0.9997221670966462
F1 score:  0.9997221670966462
```

a2. Classification of Cyber Attacks Using Decision Tree Algorithm with reducing dimensions

PROJECT REPORT

```
print( Training time:  ,end_time-start_time
```

```
1.0
Training time:  4.929955005645752
```

```
Accuracy: 0.9971422901369319
Precision: 0.9971422901369319
Recall: 0.9971422901369319
F1 score:  0.9971422901369319
```

b1. Classification of Cyber Attacks Using K-Nearest Neighbors Algorithm without reducing dimensions¶

```
Training time:  89.1923291683197
```

```
Accuracy: 0.9991268108751736
Precision: 0.9706175159277184
Recall: 0.9539293416530648
F1 score:  0.9620368086276592
```

PROJECT REPORT

b2. Classification of Cyber Attacks Using K-Nearest Neighbors Algorithm with reducing dimensions

```
Training time:  3.2213778495788574
```

```
Accuracy: 0.998809287557055
Precision: 0.998809287557055
Recall: 0.998809287557055
F1 score:  0.998809287557055
```

c1. Classification of Cyber Attacks Using Artificial Neural Networks (ANN) without reducing dimensions

```
Training time:  38.34300875663757
```

```
Accuracy: 0.9993649533637626
Precision: 0.992256432120197
Recall: 0.890383874717943
F1 score:  0.9219003071592882
```

Two hidden layer:

```
Training time:  101.85595178604126
```

```
Accuracy: 0.9976979559436396
Precision: 0.7920811764233556
Recall: 0.7894392553858587
F1 score:  0.7907424641285815
```

Three hidden layers:

```
Accuracy: 0.9976979559436396
Precision: 0.7920811764233556
Recall: 0.7894392553858587
F1 score:  0.7907424641285815
```

c2. Classification of Cyber Attacks Using Artificial Neural Networks (ANN) with reducing dimensions

1 hidden layer

```
Training time:  35.7399742603302
```

```
Accuracy: 0.9982536217503473
Precision: 0.9734799882799823
Recall: 0.8956036100320655
F1 score:  0.9255613221256465
```

2 hidden layers

```
Training time:  119.88437414169312
```

```
Accuracy: 0.9882516372296091
Precision: 0.7853715000213832
Recall: 0.709288959267983
F1 score:  0.7380560974338001
```

3 hidden layers

```
Accuracy: 0.9882516372296091
Precision: 0.7853715000213832
Recall: 0.709288959267983
F1 score:  0.7380560974338001
```

PROJECT REPORT

Comparison between Decision Tree algorithm and K-Nearest Neighbors algorithm:

The Decision Tree algorithm has a slightly better performance compared to the K-Nearest Neighbors algorithm. The Decision Tree algorithm achieved a higher accuracy score of 0.9997 compared to the K-Nearest Neighbors algorithm which had an accuracy score of 0. 9991.In terms of precision, both algorithms performed well with scores of 0.9997 for the Decision Tree algorithm and 0.9706 for the K-Nearest Neighbors algorithm. However, the Decision Tree algorithm performed better in terms of recall with a score of 0.9997 compared to 0.9539 for the K-Nearest Neighbors algorithm. The F1 score takes into account both precision and recall, and here too, the Decision Tree algorithm outperforms the K-Nearest Neighbors algorithm with a score of 0.9997 compared to 0.9620.

Comparison between Decision Tree algorithm and Artificial Neural Networks:

The Decision Tree algorithm outperforms Artificial Neural Networks (ANN) in terms of accuracy, precision, recall, and F1 score. The Decision Tree algorithm achieved a higher accuracy, precision, recall, and F1 score compared to the ANN. The Decision Tree algorithm achieved an accuracy of 0.9997, while the ANN achieved an accuracy of 0.9993. The Decision Tree algorithm also achieved a higher precision, recall, and F1 score compared to the ANN.

Comparison between KNN and Artificial Neural Networks:

The Artificial Neural Networks (ANN) outperforms K-Nearest Neighbors (KNN) algorithm in terms of precision, recall, and F1 score. However, KNN algorithm achieves a slightly lower accuracy compared to ANN. KNN algorithm achieved an accuracy of 0.9991, while ANN achieved an accuracy of 0.9993. In terms of precision, ANN achieved a higher precision of 0.992, compared to KNN's precision

of 0.971. ANN also achieved a higher recall of 0.890, compared to KNN's recall of 0.953, and a higher F1 score of 0.922 compared to KNN's F1 score of 0.962. Therefore, based on the given evaluation metrics, the ANN algorithm is a better choice for classifying cyber-attacks than KNN, considering the higher precision, recall, and F1 score achieved by the ANN algorithm.

Conclusion

In conclusion, this code provides a comprehensive approach to data analysis and modeling for detecting cyber-attacks in a network. We have learned that data preprocessing, feature engineering, and model selection are crucial for achieving accurate and efficient results. Through the use of libraries such as NumPy, pandas, matplotlib, seaborn, and sklearn, we can perform tasks such as data processing, analysis, visualization, and modeling. Additionally, we have seen that reducing dimensions through PCA can lead to significant improvements in run time without sacrificing too much accuracy. The models used for classification and clustering, such as KNN, Decision Tree, and ANN, all have their strengths and weaknesses, and selecting the appropriate model is essential for achieving the desired outcome. Finally, we have seen that visualizing the results of clustering algorithms can provide insight into the nature of the attacks and aid in further analysis. Overall, this code provides a valuable resource for detecting and preventing cyber-attacks in a network.