Abdullah Dar

21L-7512

BSCS-6A

Cyber Security

Assignment no 3

Data Set Used

Cyber Security Attacks

Dataset Description

The cybersecurity attacks dataset, meticulously compiled by **Aashray Agur** and **Uma Venugopal**, stands as a comprehensive repository of **25 varied metrics**, meticulously cataloguing a staggering **40,000 records**. This treasure trove of information serves a noble purpose: to unravel the intricacies of cyber security incidents and unveil the cryptic patterns underlying **network traffic behaviours**.

Each entry in this expansive dataset unveils a myriad of attributes, painting a vivid picture of the cyber realm's intricate dance:

Timestamps serve as the temporal anchors, pinpointing the precise moments when network activities unfolded. **Source IP Addresses** and **Destination IP Addresses** reveal the digital footprints left by both senders and receivers, while **Source Ports** and **Destination Ports** expose the gateways through which data traverses. The Protocol employed — whether TCP, UDP, or others — sheds light on the language of communication.

Packet Length and **Packet Type** delineate the nature of the transmitted data, distinguishing between mere data packets and potent control packets. **Traffic Type** categorizes network activities into benign normalcy or the suspicious shadows where threats lurk. Meanwhile, **Payload Data** lays bare the content of the communication, potentially harbouring insights, or concealed dangers.

But the dataset's utility transcends mere observation; it ventures into the realm of proactive defence. **Malware Indicators**, **Anomaly Scores**, and **Alerts/Warnings** act as sentinels, flagging malicious intent or aberrant behaviour. **Attack Types** and their accompanying Signatures detail the modus operandi of cyber assailants, empowering defenders with the knowledge to thwart their advances. **Actions Taken** and **Severity Levels** chart the responses to these threats, guiding the escalation of defence measures.

Yet, the human element is not overlooked. **User Information** and **Device Information** cast a spotlight on the individuals and hardware embroiled in these digital skirmishes. **Network Segments** delineate the battlegrounds, while **Geo-location Data** offers geographical context, transcending virtual boundaries. **Proxy Information**, **Firewall Logs**, and **IDS/IPS Alerts** furnish additional layers of defence, fortifying the network's ramparts against incursions.

Behind every data point lies a narrative, a chronicle of the ceaseless struggle between security and subterfuge. And within this dataset lies the potential to decipher these narratives, to glean insights that safeguard digital sanctity.

Assembled with care in **2023**, this cybersecurity attacks dataset stands as a testament to vigilance, a beacon illuminating the path towards a safer digital frontier.

Description of Machine Learning Techniques

A brief description of each machine learning technique used in the code, along with their benefits and how they can help you analyse the cybersecurity attacks dataset:

1. **Decision Trees:**
    a. Decision trees are versatile and easy to interpret.
    b. They can handle both numerical and categorical data.
    c. Useful for understanding which features are most important for classification.
2. **Random Forest:**
    a. Random forests improve upon decision trees by reducing overfitting.
    b. They work well with large datasets and handle high-dimensional spaces.
    c. Provide an ensemble method for improved accuracy by combining multiple decision trees.
3. **Support Vector Machines (SVM):**
    a. SVMs are effective in high-dimensional spaces and when the number of features exceeds the number of samples.
    b. They are versatile and can handle both linear and non-linear data.
    c. Effective in cases where the decision boundary is not clearly defined.
4. **K-Nearest Neighbours (KNN):**
    a. KNN is simple and easy to understand.
    b. It does not require training time as it memorizes the entire training dataset.
    c. Useful for cases where decision boundaries are irregular or difficult to define.
5. **Logistic Regression:**
    a. Logistic regression is a probabilistic model suitable for binary classification tasks.
    b. It provides interpretable coefficients indicating the impact of each feature on the classification.
    c. Robust to noise and can handle both linear and non-linear relationships between features and target variable.
6. **Gradient Boosting:**
    a. Gradient boosting combines multiple weak learners (usually decision trees) to create a strong predictive model.
    b. It sequentially improves the model's performance by focusing on the samples that were previously misclassified.
    c. Effective in reducing bias and variance, leading to high accuracy.
7. **Naive Bayes:**
    a. Naive Bayes is simple, fast, and scalable.
    b. It works well with high-dimensional datasets and requires less training data.
    c. Particularly useful for text classification tasks but can also be applied to other types of data.
8. **Neural Networks:**
    a. Neural networks are powerful models capable of

learning complex patterns in data.
   b. They can automatically extract features from raw data, eliminating the need for manual feature engineering.
   c. Effective for tasks such as image recognition, natural language processing, and sequential data analysis.

By employing these machine learning techniques, you can:

- Identify patterns and relationships within the cybersecurity attacks dataset.
- Build predictive models to classify and predict different types of attacks.
- Improve incident response strategies by automating the detection of potential threats.
- Enhance cybersecurity measures by understanding the characteristics and behaviour of different types of attacks.
- Optimize resources and prioritize responses based on the severity and likelihood of attacks.

Analysis of Machine Learning Algorithms for Cybersecurity Attack Detection

The following are the accuracies achieved by different machine learning algorithms:

- **Decision Trees: 0.33275**
- **Random Forest: 0.330875**
- **Support Vector Machines (SVM): 0.330125**
- **K-Nearest Neighbours (KNN): 0.34125**
- **Logistic Regression: 0.323375**
- **Gradient Boosting: 0.33325**
- **Naive Bayes: 0.329625**
- **Neural Networks: 0.337125**

Analysis of Machine Learning Algorithms Used:

1. **Decision Trees:** Decision trees offer simplicity and interpretability but achieved a relatively low accuracy, indicating potential struggles in capturing the complexity of the relationships within the data.
2. **Random Forest:** Despite being an ensemble method aimed at improving decision trees, random forests achieved similar accuracy,

suggesting possible issues with dataset diversity or hyperparameter tuning.

3. **Support Vector Machines (SVM):** SVMs, known for their effectiveness in high-dimensional spaces, showed relatively low accuracy, indicating challenges in defining the decision boundary or noise in the dataset.
4. **K-Nearest Neighbours (KNN):** KNN performed slightly better than other algorithms, capturing local patterns, but still faced challenges indicating unclear clusters or noise.
5. **Logistic Regression:** Achieving results like other algorithms, logistic regression suggests that the relationships between features and the target variable might not be strictly linear.
6. **Gradient Boosting:** While typically providing better accuracy, gradient boosting's performance was like decision trees and random forests, indicating potential issues with weak learners or boosting.
7. **Naive Bayes:** Assuming independence between features, naive Bayes achieved similar accuracy to other algorithms, suggesting that the naive assumption might not significantly impact performance.
8. **Neural Networks:** Neural networks performed the best among tested algorithms, indicating capability in capturing complex patterns, but the marginal improvement suggests the dataset might not benefit significantly from added complexity.

Comparison with Related Works

Related Work no: 1

In this analysis, we have integrated findings from a related work conducted by **MEHMOOD BHUTTA in December 2023.** His study focused on Cyberattack exploration, utilizing both Exploratory Data Analysis (EDA) and Machine Learning (ML) techniques to investigate Cyber Security Attacks. By considering their methodology and results, we aim to compare our own findings with theirs to gain a comprehensive understanding of the dataset and the effectiveness of various ML algorithms in detecting cybersecurity threats. Additionally, we have compiled a comparison of Machine Learning Algorithm Results.

Calculated Results

Algorithm	Accuracy
Decision Trees	0.33275
Random Forest	0.330875
SVM	0.330125
KNN	0.34125
Logistic Regression	0.323375
Gradient Boosting	0.33325
Naive Bayes	0.329625
Neural Networks	0.337125

Related Work Results (Mehmood Bhutta):

Metric	Accuracy
Accuracy	0.33656
Precision	0.3365
Recall	0.3365
F1-score	0.3365
Macro Avg	0.33407
Weighted Avg	0.33653

Analysis:

Our results display varying accuracies across different algorithms, with KNN performing the best at 0.34125 accuracy.

The related work achieves an accuracy of 0.33656, slightly higher than your average accuracy.

Precision, recall, and F1-score exhibit consistency across different metrics in the related work, indicating balanced performance.

Conversely, our results showcase fluctuations in accuracy and other metrics across different algorithms, suggesting inconsistencies in model performance.

It's crucial to consider factors such as dataset composition, preprocessing techniques, and model hyperparameters when comparing results with related work.

Conclusion:

While our calculated results provide valuable insights into the performance of various machine learning algorithms for cybersecurity attack detection, there is room for improvement in achieving consistent and higher accuracies.

Further experimentation with feature engineering, hyperparameter tuning, and ensemble methods may help improve model performance and align with the related work's results.

Related Work no: 2

In this analysis, we have incorporated findings from another related work conducted **by IKHWANANDA approximately 6 months ago**. The study focused on Cyber Security Attack exploration, utilizing Python for both Exploratory Data Analysis (EDA) and Cyber Security Attacks investigation. According to their results, the following metrics were obtained:

Related Work Results (IKHWANANDA):

Metric	Value
Accuracy	0.33485
Precision	0.3358

Our results show a range of accuracies across different machine learning algorithms, with KNN achieving the highest accuracy at 0.34125.

Overall, our accuracies are relatively consistent with none deviating significantly from the others.

The accuracies range from 0.323375 to 0.34125, indicating a moderate level of performance across the algorithms tested.

Related Work Results (IKHWANANDA):

The related work by IKHWANANDA achieved an accuracy of 0.33485, slightly higher than

the average accuracy obtained in your results.

Additionally, the precision value reported by IKHWANANDA is 0.3358, which is in line with their accuracy metric.

Comparison:

Our results generally align with the related work by IKHWANANDA, with both achieving accuracies in the range of 0.33 to 0.34.

KNN in your results outperforms the related work's accuracy, indicating that your KNN model might be more effective in detecting cybersecurity threats in this dataset.

However, the precision value reported by IKHWANANDA is slightly higher than the accuracies obtained in your results, suggesting that their model might have better precision in identifying true positives.

Conclusion:

Both Our results and the related work by IKHWANANDA demonstrate moderate performance in detecting cybersecurity attacks using machine learning algorithms.

Further analysis and experimentation may be necessary to improve accuracy and precision, potentially through fine-tuning of hyperparameters, feature engineering, or ensemble techniques.

Considering the similarities in results, it's essential to validate findings across multiple studies to ensure robustness and reliability in detecting cybersecurity threats.

Overall, our results provide valuable insights into the performance of various machine learning algorithms for cybersecurity attack detection and comparing them with related work enhances our understanding of the dataset and the effectiveness of different approaches

---

Related Work no: 3

In this analysis, we'll incorporate findings from another related work conducted by DIOGENES_VICTOR approximately 6 months ago. The study focused on Exploratory Data Analysis (EDA) on Cyber Security Attacks using Python. According to their results, the following metrics were obtained:

Related Work Results (DIOGENES_VICTOR):

Metric	Value
Accuracy	0.33485
Precision	0.3358

We'll compare these results with your findings to gain further insights into the performance of various machine learning algorithms for cybersecurity attack detection. Integrating this related work into our analysis will enrich our understanding of the dataset and the effectiveness of different approaches in identifying cybersecurity threats. Let's proceed with the analysis.

Our Results:

Across various machine learning algorithms, your accuracies range from 0.323375 to 0.34125.

KNN in your results achieves the highest accuracy at 0.34125, while the other algorithms exhibit accuracies in the range of 0.32 to 0.33.

The accuracies obtained from your work show moderate performance in detecting cybersecurity attacks.

Related Work Results (DIOGENES_VICTOR):

DIOGENES_VICTOR's related work reports an accuracy of 0.33485 and precision of 0.3358.

Their results are slightly higher than the average accuracy obtained in your results, indicating comparable performance in detecting cybersecurity attacks.

Comparison:

Both your results and the related work by DIOGENES_VICTOR demonstrate similar performance in detecting cybersecurity attacks, with accuracies around 0.33.

While your KNN model outperforms the related work's accuracy, the precision reported by DIOGENES_VICTOR is slightly higher than the accuracies obtained in your results.

Conclusion:

The comparison between your work and the related work by DIOGENES_VICTOR indicates consistent performance in detecting cybersecurity attacks.

Further analysis and refinement may be necessary to improve accuracy and precision, potentially through fine-tuning of hyperparameters, feature engineering, or ensemble techniques.

It's essential to validate findings across multiple studies to ensure robustness and reliability in detecting cybersecurity threats.

Overall, both your work and the related work by DIOGENES_VICTOR provide valuable insights into the performance of various machine learning algorithms for cybersecurity attack detection, contributing to a better understanding of the dataset and the effectiveness of different approaches.

References **[Its in APA 7 style]**

Reference of the Dataset:

Cyber security attacks. (2024, March 16). https://www.kaggle.com/datasets/teamincribo/cyber-security-attacks/data

References of the related Work Attached.

1. Mehmoodbhutta. (2023, December 26). CyberAttack(EDA & ML). https://www.kaggle.com/code/mehmoodbhutta/cyberattack-eda-ml/notebook
2. Xokent. (2023, October 15). Cyber Security Attack - EDA. https://www.kaggle.com/code/xokent/cyber-security-attack-eda
3. Diogenesvictor. (2023, October 16). EDA ON CYBER SECURITY. https://www.kaggle.com/code/diogenesvictor/eda-on-cyber-security

---

Python Code:

Code for Checking data types of data in dataset.

```
import pandas as pd

# Read the data from the CSV file
data = pd.read_csv("cybersecurity_attacks.csv")

# Replace non-numeric values with 0
numeric_columns = ['Source Port', 'Destination Port', 'Packet Length',
'Anomaly Scores', 'Severity Level']
for column in numeric_columns:
    data[column] = pd.to_numeric(data[column],
errors='coerce').fillna(0)

# Split the data into features (X) and target variable (y)
X = data.drop(columns=['Attack Type', 'Timestamp'])  # Excluding non-
numeric and timestamp columns
y = data['Attack Type']

# Print the data types of all columns to ensure only numeric values are
present
print(X.dtypes)
```

Code of Machine Learning Algorithms used:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier


# Read the data from the CSV file
data = pd.read_csv("cybersecurity_attacks.csv")

# Replace null values with a dummy value (e.g., -1)
```

```
data.fillna(-1, inplace=True)

# Select columns with numeric data types
numeric_columns = data.select_dtypes(include=['int64',
'float64']).columns

# Exclude 'Attack Type' from numeric columns
numeric_columns = numeric_columns.drop('Attack Type', errors='ignore')

# Split the data into features (X) and target variable (y)
X = data[numeric_columns]
y = data['Attack Type']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Preprocessing: Standardize features by removing the mean and scaling
to unit variance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Machine Learning Techniques
# 1. Decision Trees
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train_scaled, y_train)
dt_pred = dt_classifier.predict(X_test_scaled)
dt_accuracy = accuracy_score(y_test, dt_pred)
print("Decision Trees Accuracy:", dt_accuracy)

# 2. Random Forest
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train_scaled, y_train)
rf_pred = rf_classifier.predict(X_test_scaled)
rf_accuracy = accuracy_score(y_test, rf_pred)
print("Random Forest Accuracy:", rf_accuracy)

# 3. Support Vector Machines (SVM)
svm_classifier = SVC()
svm_classifier.fit(X_train_scaled, y_train)
svm_pred = svm_classifier.predict(X_test_scaled)
svm_accuracy = accuracy_score(y_test, svm_pred)
print("SVM Accuracy:", svm_accuracy)

# 4. K-Nearest Neighbors (KNN)
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train_scaled, y_train)
```

```
knn_pred = knn_classifier.predict(X_test_scaled)
knn_accuracy = accuracy_score(y_test, knn_pred)
print("KNN Accuracy:", knn_accuracy)

# 5. Logistic Regression
lr_classifier = LogisticRegression()
lr_classifier.fit(X_train_scaled, y_train)
lr_pred = lr_classifier.predict(X_test_scaled)
lr_accuracy = accuracy_score(y_test, lr_pred)
print("Logistic Regression Accuracy:", lr_accuracy)

# 6. Gradient Boosting
gb_classifier = GradientBoostingClassifier()
gb_classifier.fit(X_train_scaled, y_train)
gb_pred = gb_classifier.predict(X_test_scaled)
gb_accuracy = accuracy_score(y_test, gb_pred)
print("Gradient Boosting Accuracy:", gb_accuracy)

# 7. Naive Bayes
nb_classifier = GaussianNB()
nb_classifier.fit(X_train_scaled, y_train)
nb_pred = nb_classifier.predict(X_test_scaled)
nb_accuracy = accuracy_score(y_test, nb_pred)
print("Naive Bayes Accuracy:", nb_accuracy)

# 8. Neural Networks
nn_classifier = MLPClassifier()
nn_classifier.fit(X_train_scaled, y_train)
nn_pred = nn_classifier.predict(X_test_scaled)
nn_accuracy = accuracy_score(y_test, nn_pred)
print("Neural Networks Accuracy:", nn_accuracy)
```

Note: Code Files are also uploaded in classroom.