ANSWER SHEET
DAC 2023
PRELIMINARY ROUND

DATA ANALYSIS COMPETITION 2023

TEAM NAME	Doa Ibu
TEAM ID	DAC-01-0002
UNIVERSITY	SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY

CHAPTER I: Introduction

Information Technology is advancing rapidly, leading to a surge in the transmission of global data and information. The telecommunications sector, particularly in Indonesia, has experienced substantial transformations over the past few decades. Telecommunications has emerged as a pivotal sector in Indonesia's progress, profoundly influencing enhanced connectivity, information access, and community empowerment.

The Indonesian government has undertaken diverse initiatives to propel the telecommunications sector forward in alignment with the vision of "Indonesia Emas 2045." This involves augmenting internet connectivity and data transfer speeds and capacities, epitomizing telecommunications development. Additionally, the evolution of wireless networks has revolutionized how we communicate and connect globally. Telecommunications encompasses the transmission of information over long distances using technology, and its development continues with a focus on enhancing speed and connectivity across numerous networks.

Nonetheless, concomitant with telecommunications advancement, a significant network security challenge has emerged, namely network attacks. Network attacks entail unauthorized or malicious endeavors to manipulate, disrupt, or gain access to computer systems, networks, or communication infrastructure without authorization. These network attacks carry the potential to precipitate various other threats, including the compromise of sensitive and confidential information. Inadequately protected, network attacks pose a risk to the integrity of the owner's information assets. Consequently, to mitigate these risks, it is imperative to employ software development for the detection of network attacks, enabling effective mitigation measures. In this context, we already possess data pertaining to network traffic and historical web attack types.

Thus, research is needed, where this research is about identifying types of computer or telecommunications network attacks so that problems that can be used as a reference in this analysis include how to select the best model using several models and how variables influence the identification of the type of network attack.

CHAPTER II: Theoretical Framework

A. Logistic Regression

Logistic Regression is a statistical method used to model the relationship between one or more independent variables (predictors) and a binary dependent variable (two categories or classes, such as 0 and 1, Yes and No, Success and Failure). [1]

Logistic regression is a useful tool for a variety of applications, including in risk analysis, probability prediction, data classification, epidemiological analysis, and more. It allows researchers and data analysts to understand and model how certain factors influence the probability of an event occurring in the context of two possible categories.

B. Naive Bayes

The Naive Bayes method is one of the popular classification and probability modeling methods in machine learning. It is based on Bayes' theorem and a "naive" or simple assumption, i.e. all features or predictors in the model are assumed to be mutually independent, although in the real world, this assumption is often not fully justified. Despite its simplicity, Naive Bayes is often very effective in many applications. [2]

C. Support Vector Machine

Support Vector Machine (SVM) is one of the powerful and versatile machine learning methods used for classification and regression problems. SVM focuses on finding the optimal hyperplane (separating plane) to separate two classes in high-dimensional space. In classification problems, SVM searches for the hyperplane that has the largest margin between two classes. The margin is the distance between the hyperplane and the closest sample from each class. [3]

D. Random Forest Classifier

The Random Forest ensemble method, implemented in scikit-learn through the RandomForestClassifier and RandomForestRegressor classes, builds each tree within the ensemble using a bootstrap sample, which is a random sample drawn with replacement from the training dataset. [4]

In contrast to the original publication, scikit-learn's implementation of Random Forest combines classifiers by averaging their probabilistic predictions rather than having each classifier vote for a single class.An alternative to Random Forest is Histogram-Based Gradient Boosting (HGBT) models

CHAPTER III: Analytical Steps

The data used is data regarding network traffic which has a total of 42 variables. The steps in conducting this research are as follows.

1. Understand the problems that occur, understand the data, and identify variables in the data
2. Cleaning the data by replacing data that is strange and inappropriate and very different from other data values in the variables, changing several data types, handling missing values, and deleting duplicate data
3. Prepare data by carrying out one-hot coding for categorical data and normalizing numerical data
4. Analyze export data by visualizing boxplots to check for outliers then using the Z-Score method to overcome outliers
5. Visualize the correlation matrix and delete predictor variables that have high correlation to avoid multicollinearity
6. Check data imbalance in the response variable and overcome it by resampling the data using the SMOTE method

7. Carry out a train-test split where the response/target variable is the type of attack on the internet network
8. Carry out data modeling using Naive Bayes models, Random Forest Classifier, Logistic Regression, and Support Vector Machine
9. Selection of the best model with the highest accuracy selection from several models
10. Confusion Matrix visualization so you can find out the predicted and actual results of the best model
11. Perform feature importance to find out the most influential variables/columns
12. Make predictions on predicted data using the best model.

CHAPTER IV: Analysis of Results

A. Modeling and Selection Best Model

The results of the analysis after data cleaning, data preparation, then modeling were carried out on the data train using 4 models, namely Naive Bayes, Logistic Regression, Random Forest Classifier, and Support Vector Machine. These four models are popular models used in machine learning. The results of the modeling are obtained with accuracy values and F1 score values in **Table 1** below.

Tabel 1. Naive Bayes Modeling Results

Model	Accuration	F1 Score
Naive Bayes	66.25%	69.67%
Logistic Regression	90.45%	90.53%
Random Forest Classifier	99.26%	99.26%
Support Vector Machine	91.22%	91.53%

In **Table 1**, it can be seen that the highest accuracy of the four models is in the Random Forest Classifier model with an accuracy value and F1 score value of 99.26%. This accuracy result is a high result so that the accuracy of the model in predicting is very good, namely 99.26% of the predictions are correct for real events. Apart from that, the F1 score value is also high, which shows that the model has a good level of precision in identifying types of internet attacks.

B. Feature Importance

The feature importance results in this analysis are displayed based on the top 5 rankings of influential variables. These results are presented in **Table 2** below.

Tabel 2. Feature Importance

Variabel	Feature Importance
service_ecr_i	0.140
wrong_fragment	0.135
dst_host_count	0.131
flag_S0	0.113
dst_host_srv_rerror_rate_1	0.108

Based on **Table 2**, the highest feature importance results are in the service_ecr_i variable of 0.14, so it can be said that the type of network service running on the target host in the context of network analysis or scanning which influences the type of attack is the ECR service. Apart from that, the wrong_fragment variable also has a big influence after the ECR service. This variable is the number of incorrect or inappropriate fragments in a network connection so that it can influence the type of network attack that occurs.

CHAPTER V : Conclusion and Recommendation

A. Conclusion

The conclusions of this research are as follows.

1. The best model for predicting the type of network attack is the Random Forest Classifier model with an accuracy of 99.26%. This model is very good at predicting and identifying types of attacks on telecommunications or computer networks.
2. In terms of feature importance, the variables that influence determining the type of telecommunication or computer network attack include service ECR and wrong fragment.

B. Recommendation

The recommendations we can give based on the analysis we obtained are as follows.

1. In order to minimize exposure to telecommunications network attacks, you can pay attention to ECR services in network analysis or scanning. Apart from that, also pay attention to monitoring the number of incorrect fragments in the network connection including interruptions in sending data over the network, damage to network devices, or problems with the software that manages these fragments so that it can help in diagnosing network problems and ensuring correct data delivery and safe.
2. Research using this model can be used as a reference in subsequent research.

REFERENCES

[1] Kleinbaum, David G and Klein, Mitchel, "Logistic Regression: A Self-Learning Text", 2010.

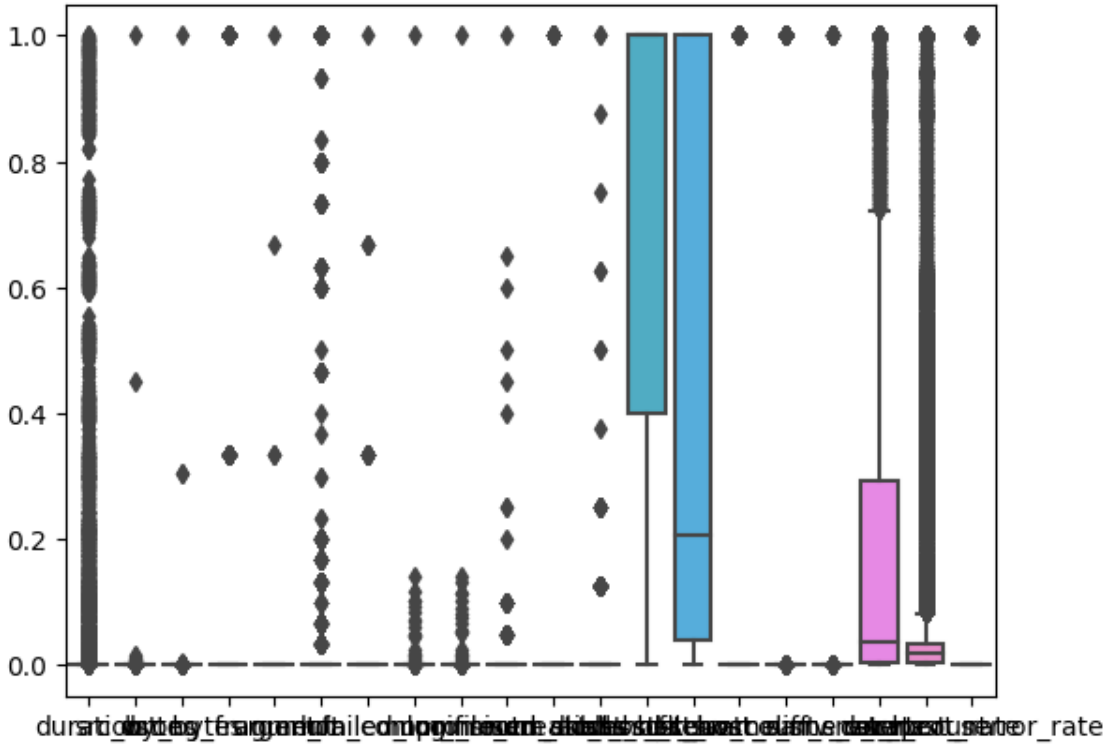
[2] Jurafsky, D., & Martin, J. H, "Speech and Language Processing." Naive Bayes Classification. Pearson., 2020.

[3] Cortes, C., & Vapnik, V. "Support-vector networks." Machine learning, 20(3), 273-297, 1995.

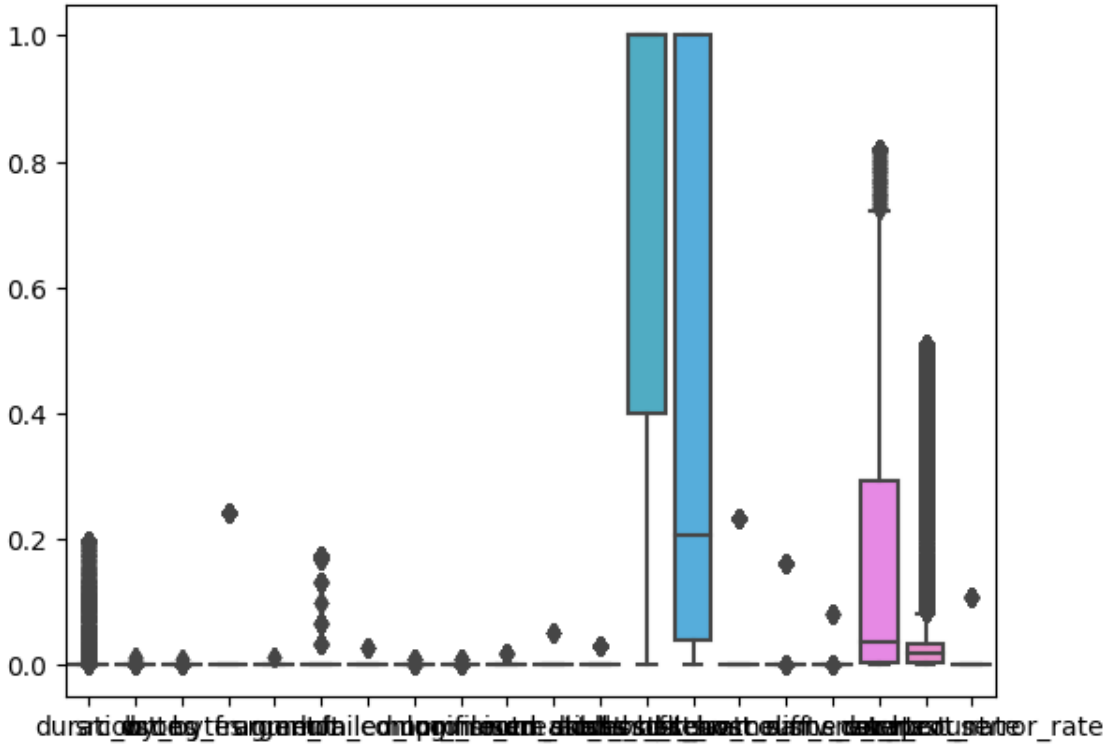
[4] *https://scikit-learn.org/stable/modules/ensemble.html#random-forests*

ATTACHMENT

Attachment 1: Boxplot Outlier Detection

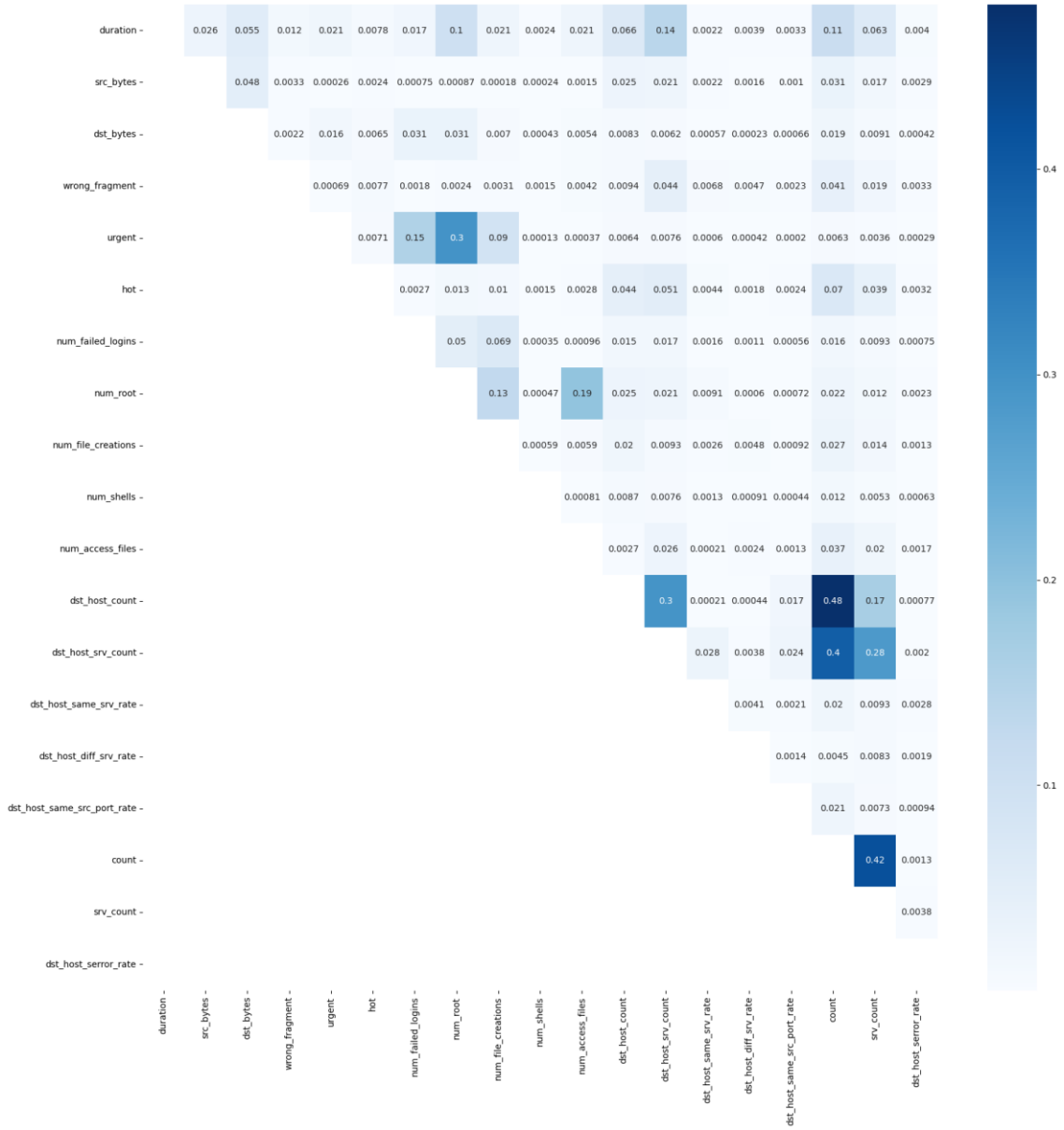


Attachment 2: Boxplot Outlier Detection After Z-Score

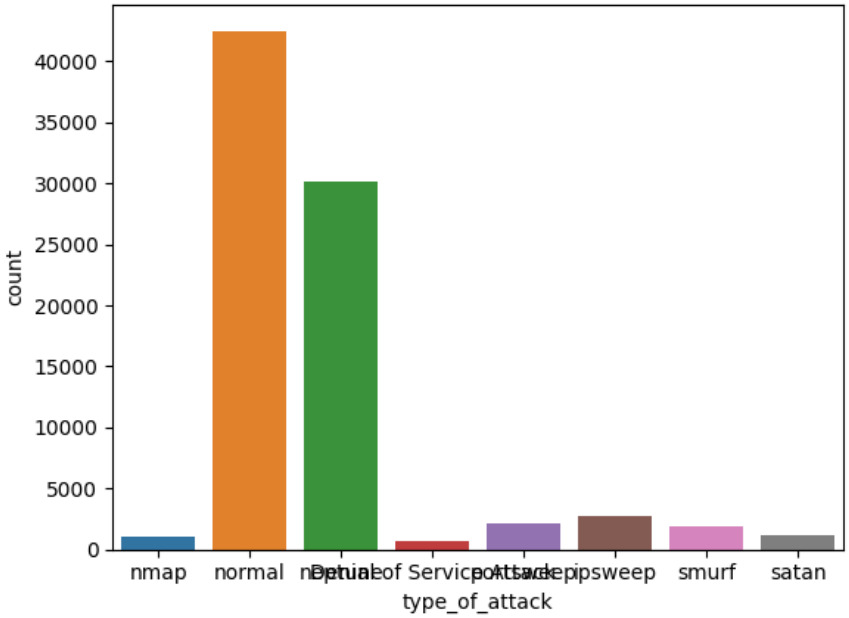


Attachment 3: Correlation Matrix

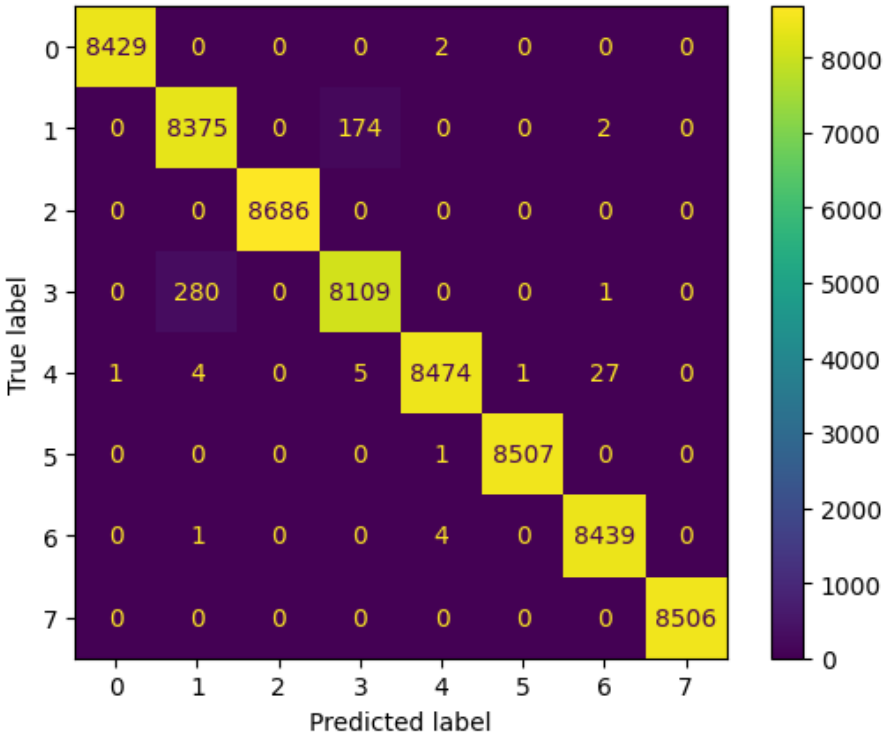
Attachment 4: Correlation Matrix Deleted High Correlation



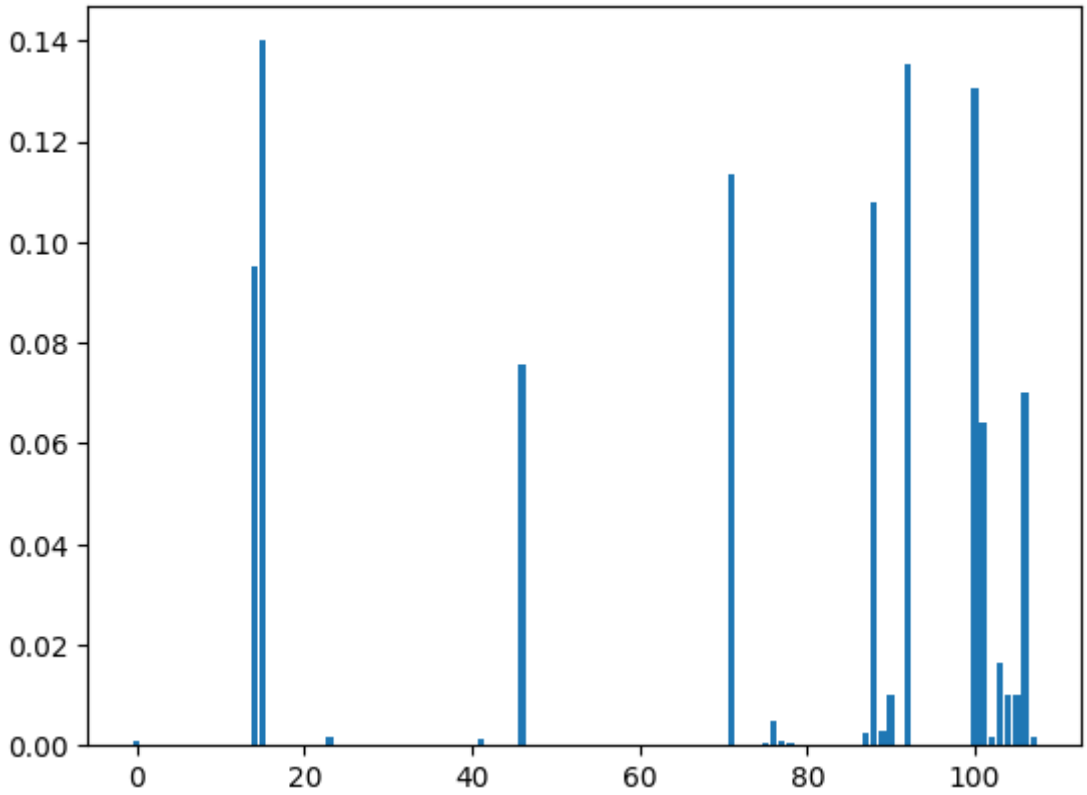
Attachment 5: Imbalance Data



Attachment 6: Confusion Matrix Random Forest



Attachment 6: Feature Importance Random Forest



SYNTAX

```
# Data Analyst Competition 2023

Team : **Doa Ibu**

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
    classification_report,
)
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

## Dataset

from google.colab import drive
drive.mount('/content/drive')

df=pd.read_csv("/content/drive/MyDrive/DataTrain_Preliminary.csv", sep=';')

pd.set_option('display.max_columns', 200)
df.head(10)

df.info()

df.type_of_attack.value_counts()
```

Identifikasi Type of Attack

1. **Normal**: "Normal" dalam konteks keamanan jaringan tidak merujuk kepada serangan, melainkan aktivitas jaringan yang tidak mencurigakan atau aktivitas yang normal dari pengguna jaringan yang sah.
2. **Neptune**: Neptune adalah nama umum untuk berbagai jenis serangan yang menghasilkan penggunaan yang berlebihan dari sumber daya jaringan dengan mengirimkan permintaan yang tidak sah secara berulang-ulang. Ini dapat menyebabkan penurunan kinerja atau penolakan layanan.
3. **Ipsweep**: Ipsweep adalah serangan di mana penyerang mencoba untuk menemukan alamat IP yang valid dalam jaringan. Ini dapat menjadi langkah awal untuk serangan yang lebih luas.
4. **Satan**: SATAN (Security Administrator Tool for Analyzing Networks) adalah alat pemindaian jaringan yang dirancang untuk mengidentifikasi kerentanannya dalam sistem dan perangkat di jaringan. Tujuan dari pemindaian ini adalah membantu administrator jaringan untuk meningkatkan keamanan mereka dengan mengidentifikasi dan memperbaiki kerentanannya.
5. **Portscan (Portsweep)**: Portsweep adalah jenis serangan di mana penyerang mencoba untuk memindai atau menguji beberapa port pada sejumlah besar perangkat dalam jaringan untuk menemukan port yang terbuka. Ini bisa menjadi tindakan persiapan untuk serangan berikutnya.
6. **Smurf**: Serangan Smurf adalah serangan amplifikasi yang memanfaatkan protokol Internet Control Message Protocol (ICMP). Penyerang mengirimkan permintaan ICMP dengan alamat IP sumber palsu ke jaringan yang memiliki banyak host, dan host-host tersebut membalas permintaan tersebut kepada alamat IP sumber yang sebenarnya. Ini dapat membanjiri jaringan dengan lalu lintas ICMP yang tidak diinginkan.
7. **Denial of Service (DoS) Attack**: Serangan Denial of Service (DoS) bertujuan untuk membuat sumber daya komputer atau jaringan tidak tersedia untuk pengguna yang sah dengan mengalirkan lalu lintas yang sangat besar atau mengeksploitasi kerentanannya pada sistem yang diserang.
8. **Nmap** (Network Mapper): Nmap adalah sebuah alat pemindaian jaringan yang digunakan untuk mengekplorasi dan mengidentifikasi perangkat yang terhubung ke jaringan. Ini dapat digunakan untuk pemetaan jaringan, penemuan port terbuka, sistem operasi, dan lebih banyak informasi yang berguna.

Data Cleaning

Replace Data

```
for kolom in df.columns:
    nilai_unik = df[kolom].unique()
    print(f'=====[kolom]=====:')
    print(nilai_unik)
```

Terdapat beberapa nilai yang aneh sehingga harus mengubah nilai tersebut menjadi missing value, diantaranya :

1. #NA
2. '*'
3. 99999
4. (Blank)

```
# replace menjadi missing value
df.replace('*', np.nan, inplace=True)
df.replace('99999', np.nan, inplace=True)
```

Change Data Type

```
# data numerik
df_num=df[['duration', 'src_bytes', 'dst_bytes', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins',
        'num_compromised', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files',
        'num_outbound_cmds', 'count', 'srv_count','dst_host_count', 'dst_host_srv_count',
'dst_host_same_srv_rate',
        'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
'dst_host_serror_rate']].astype(float)

# data kategorik
df_cat=df.drop(df_num, axis=1)

# gabung data
df=pd.concat([df_cat, df_num], axis=1)
df.info()
```

Handle Missing Value

```
df.isnull().sum()

# menghapus kolom yang bernilai hanya 0 dan memiliki missing value banyak
```

```
df=df.drop(columns=['num_outbound_cmds', 'same_srv_rate', 'diff_srv_rate',
'srv_diff_host_rate',
                   'dst_host_srv_diff_host_rate', 'dst_host_rerror_rate'], axis=1)
df.shape

df=df.dropna(axis=0)

df=df.reset_index()

### Data Duplicated

df.duplicated().sum()

## Data Preparation

df_num=df[['duration', 'src_bytes', 'dst_bytes', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins',
          'num_compromised', 'num_root', 'num_file_creations', 'num_shells',
'num_access_files',
          'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate',
          'dst_host_same_src_port_rate', 'count', 'srv_count', 'dst_host_serror_rate']]
df_cat=df.drop(df_num, axis=1)
df_cat=df_cat.drop(columns=['index'], axis=1)

### One-Hot Encoding

df_cat

df_onehot=df_cat.drop(columns=['type_of_attack'], axis=1)
df_onehot=pd.get_dummies(df_onehot, drop_first=True)
df_onehot=df_onehot.replace({False:0, True:1})
df_onehot

df_onehot.info()

# gabung data
df_model=pd.concat([df_onehot, df_num], axis=1)
df_model.shape

### Normalized
```

```
scaler = MinMaxScaler()
df_norm=scaler.fit_transform(df_num)
df_norm=pd.DataFrame(df_norm, columns=df_num.columns)

df_norm

## EDA

### Outlier

sns.boxplot(df_norm)

# Z-Score
numeric_cols = [col for col in df_num.columns.tolist()]
for i in numeric_cols:
    upper_limit = df_norm[i].mean() + 3*df_norm[i].std()
    lower_limit = df_norm[i].mean() - 3*df_norm[i].std()

    df_norm[i] = np.where(
        df_norm[i]>upper_limit,
        upper_limit,
        np.where(
            df_norm[i]<lower_limit,
            lower_limit,
            df_norm[i]
        )
    )

sns.boxplot(data=df_norm)

### Correlation Matrix

corr_matrix = df_norm.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
plt.figure(figsize=(20,20))
sns.heatmap(upper, annot=True, cmap='Blues')

# menghapus kolom num_compromised karena memiliki korelasi yang tinggi karena
jika tidak dihapus dapat menyebabkan multikolinearitas
df_norm.drop(columns=['num_compromised'], inplace=True)
```

```
corr_matrix = df_norm.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
plt.figure(figsize=(20,20))
sns.heatmap(upper, annot=True, cmap='Blues')

## Transformed Data

df_model=pd.concat([df_onehot, df_norm, df_cat['type_of_attack']], axis=1)
df_model.head()

## Modeling Data Train

X = df_model.drop('type_of_attack', axis=1)
y = df_model['type_of_attack']

### Imbalance Data

sns.countplot(x='type_of_attack', data=df_model)

X_res, y_res = SMOTE().fit_resample (X, y)
y_res.value_counts()

X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.2,
random_state=123)
print('Jumlah baris dan kolom dari x_train adalah:', X_train.shape,', sedangkan Jumlah
baris dan kolom dari y_train adalah:', y_train.shape)
print('Prosentase unique di data Training adalah:')
print(y_train.value_counts(normalize=True))
print('Jumlah baris dan kolom dari x_test adalah:', X_test.shape,', sedangkan Jumlah
baris dan kolom dari y_test adalah:', y_test.shape)
print('Prosentase unique di data Testing adalah:')
print(y_test.value_counts(normalize=True))

### Naive Bayes

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)

### Decision Tree

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

f1 = f1_score(y_pred, y_test, average="weighted")

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("F1 Score:", f1)

### Logistic Regression

lr = LogisticRegression()

lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)

### Random Forest

rfc = RandomForestClassifier()
```

```
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)

### SVM

svm_model = SVC()
# svm_model = SVC()

svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)

accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)

## Confusion Matrix of Best Model

Pemilihan Model terbaik berada pada Model **Random Forest** karena memiliki
akurasi yang lebih tinggi dari model lainnya

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")

print("Accuracy:", accuray)
print("F1 Score:", f1)

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
## Feature Importance

importance = clf.feature_importances_

# Membuat DataFrame dengan nama kolom dari X_train
df_importance = pd.DataFrame({'Feature Importance': importance},
index=X_train.columns)

# Menyortir DataFrame berdasarkan Feature Importance
df_importance_sorted = df_importance.sort_values(by='Feature Importance',
ascending=False)

# Menampilkan DataFrame dengan nama fitur
pd.set_option('display.max_rows', None)
print(df_importance_sorted)

# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()

## Data Prediction

df_test=pd.read_csv("/content/drive/MyDrive/Data_Prediction.csv", sep=';')

df_test.info()

df_test=df_test.drop(columns=['id', 'num_outbound_cmds', 'same_srv_rate',
'diff_srv_rate', 'srv_diff_host_rate',
                             'dst_host_srv_diff_host_rate', 'dst_host_rerror_rate',
'num_compromised'], axis=1)
df_test.shape

df_num_test=df_test[['duration', 'src_bytes', 'dst_bytes', 'wrong_fragment', 'urgent', 'hot',
'num_failed_logins',
                     'num_root', 'num_file_creations', 'num_shells', 'num_access_files',
                     'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_diff_srv_rate',
                     'dst_host_same_src_port_rate', 'count', 'srv_count', 'dst_host_serror_rate']]
df_cat_test=df_test.drop(df_num_test, axis=1)
```

One-Hot Encoding Data Test

```
df_cat_test

# memisahkan beberapa kolom yang akan dijadikan tipe int lalu string
df_cat_test1=df_cat_test[['serror_rate', 'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate',
                          'dst_host_srv_serror_rate', 'dst_host_srv_rerror_rate']].astype(int)
df_cat_test2=df_cat_test.drop(df_cat_test1, axis=1)

# mengubah tipe data ke string semua
df_cat_test=pd.concat([df_cat_test2, df_cat_test1], axis=1)
df_cat_test=df_cat_test.astype(str)
df_cat_test

df_onehot_test=pd.get_dummies(df_cat_test, drop_first=True)
df_onehot_test=df_onehot_test.replace({False:0, True:1})
df_onehot_test

# menambahkan kolom yang sama dengan data train
df_onehot_test.insert(43, 'service_pm_dump', 0)

df_onehot_test.insert(59, 'service_tftp_u', 0)

df_onehot_test.insert(60, 'service_tim_i', 0)

df_onehot_test.insert(62, 'service_urh_i', 0)

df_onehot_test
```

Normalized Data Test

```
scaler = MinMaxScaler()
df_norm_test=scaler.fit_transform(df_num_test)
df_norm_test=pd.DataFrame(df_norm_test, columns=df_num_test.columns)

df_norm_test
```

Transformed Data Test

```
df_model_test=pd.concat([df_onehot_test, df_norm_test], axis=1)
df_model_test
```

Data Train

df_train=df_model

df_train.insert(22, 'service_harvest', 0)

df_train

Modeling Data Train and Data Test

# menentukan X_train, y_train, dan X_test
X_train=df_train.drop('type_of_attack', axis=1)
y_train=df_train['type_of_attack']
X_test=df_model_test

Dengan menggunakan model terbaik yang telah dilakukan pada data train sebelumnya, yaitu model **Random Forest** maka untuk memprediksi data test sebagai berikut

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

df_model_test['type_of_attack']=y_pred

df_model_test

# export
df_model_test.to_csv('/content/drive/My Drive/df_model_test.csv', index=False)