

ASSESSMENT
COMP3009 Data mining
SEMESTER 2,2024

ASSIGNMENT

BY
SUPAWIT PRADITKUL
STUDENT ID: 21657005

LAB CLASS
THURSDAY 3:00 PM CLASSROOM 220
BUILDING 314

Bachelor of Science (Data Science)

Curtin University

Content

1. Summary	3
2. Data Preparation	4
2.1. Missing entries	4
2.2. Duplicate entries	6
2.3. Data type	7
2.4. Feature Selection	7
2.5. Scaling and Standardisation	8
2.6. Data imbalance	8
2.7. Feature Engineering	8
3. Data Classification	8
3.1. Splitting training, validation, and test data	8
3.2. KNN	9
3.3. Naive Bayes	10
3.4. Decision tree	10
3.5. Comparison	11
4. Prediction	11
4.1. The models used	11
4.2. Data Test1	12
5. Conclusion	13
6. Reference	13

1. Summary

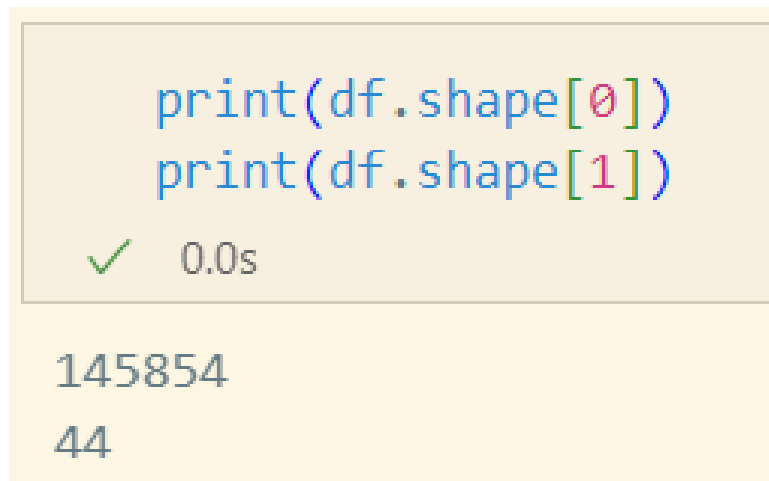
The project is trying to develop classification models using data from log activity to detect attack activity. The 3 classification methods used are K-nearest neighbors, Naive Bayes, and decision tree. After looking at the data and doing data preprocessing, the data was split into training data, testing data, and validation data to develop those 3 models. The method that creates the best performance model is the decision tree, but the K-nearest neighbors method also performs very similarly. The Naive Bayes is not able to build a model that has high performance in classifying attack activity from log activity.

2. Data Preparation

2.1. Missing entries

Missing entries can make a huge impact on the final classification models if they are not handled correctly. In this study, two types of missing entries are found. Those are missing attributes and missing instances.

The initial data “Assignment-2024-training-data-set.xlsx” has a total attribute of 44 attributes including the attribute label which is the target value of the study. On the other hand, the file name “Assignment-2024-feature-description” which is supposed to have the description of all the attributes has a total of 49 attributes which means that there are at least 6 attributes missing from in the initial data of the study.



```
print(df.shape[0])
print(df.shape[1])
```

✓ 0.0s

145854

44

Figure 1: The output of the code that shows the amount of rows and columns of the data

Upon inspection, those attributes are *srcip*, *sport*, *dstip*, *dsport*, *Stime*, and *Ltime* (all of those names come from the data description file). Moreover, the data attribute *is_fip_login* is also

missing, the discussion of that will be in the part of duplicate entries. Since all of the above attributes is missing the decision to remove all 7 of them from the training models was made

The missing instances are less obvious than the missing attributes. Append the function in Pandas library *Dataframe.isnull()* cannot find any *null* object that is identified as missing. Finding the missing values requires looking at the data and some field knowledge. There are 3 places where the missing instances were found during the studies, in the attributes *dur*, *sload*, and *dload*.

For missing instances found at the attribute *dur* this is from the assumption that since data are being transferred the duration should be more than 0 to transfer such data

```
temp = ["dur", "sbytes", "dbytes"]
df[temp][df.dur == 0][(df.sbytes != 0) | (df.dbytes != 0)].count()
```

✓ 0.0s

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_23120\3797832464.py:2: UserWarning:
df[temp][df.dur == 0][(df.sbytes != 0) | (df.dbytes != 0)].count()
```

```
dur      1333
sbytes   1333
dbytes   1333
dtype: int64
```

Figure 2: The output of the code that there are duration that has 0 values when there are some data transfer

For missing instances found at the attribute *sload*, and *dload*. this is from the assumption that since data are being transferred the rate of transfer should be more than 0 if there are data transfers in that direction

```
df.sload[(df.sload==0)&(df.sbytes!=0)].count()
# Since there always a data transfer from the variable
# low amount of data missing replace with mean
✓ 0.0s
1572

#there should be 7 missing value
df.dload[(df.dload == 0) & (df.dbytes!=0)].count()
✓ 0.0s
7
```

Figure 3&4: The output of the code that there are duration that has 0 values when there are some data transfer

The data that missing in this situation is replaced with mean instead of removing them since there are a small amount of them and they are

2.2. Duplicate entries

There are two types of duplicated entries found in the data, duplicate attributes and duplicate data points.

The duplicated attributes are found by checking for the data type and found that the data in attribute *is_ftp_login* is supposed to contain binary according to the data description but contains a set of unique values (0,1,2,4) which is the same as the next attributes in the data. After further inspection, the values match one-to-one with the data in the attribute *ct_ftp_cmd*, from this point the decision to delete the attributes *is_ftp_login* is made and labeled the attributed as missing entries.

```
df.is_ftp_login.equals(df.ct_ftp_cmd)
# this confirm it
# is_ftp_login should be drop since from
✓ 0.7s
True
```

Figure 5: The output of the code that the data in *is_ftp_login* is the same as the data in *ct_ftp_cmd*

The duplicated data points are found by using the function `Dataframe.duplicated()` from Pandas library there are a total of 65753 that are copies of each other, if we removed and living one as a replicant of that duplicates group the data will shrink down to about 61 percent of the initial data. From the field knowledge and some inspection, the assumption is that the replication can be from the same user assessing the same data in the system multiple times or the attacker attacking the system multiple times either way there is no need to remove this kind of duplicated data. The best to check if they are a copy of each other or not is the variable times which is missing as stated in section 2.1 from this data, for the conclusion to not remove them from the data is made.

```
print(df.shape[0])
print(df.duplicated(keep=False).sum())
```

✓ 0.2s

145854
65753

Figure 6: The output of the code that there are duplicated data points

2.3. Data type

Besides the attribute `is_ftp_login` mentions in duplicate entries that are supposed to be binary but instead contain a set of unique values (0,1,2,4), there are no other attributes that have miss labeled data types or data types that don't suit the data that they have.

```
pd.unique(df.is_ftp_login)
```

✓ 0.0s

array([0, 1, 4, 2], dtype=int64)

Figure 6: The output of the code that there are unique values (0,1,2,4) in the attribute `is_ftp_login`

2.4. Feature Selection

Besides the 7 attributes that were removed due to missing entries, 4 more attributes consisting of `rate`, `attack_cat`, `sload`, and `dload` are removed during this part.

The *rate* is the attribute not mentioned in the description file, so the assumption that the data is a random attribute that makes its way into the data making it irrelevant data, so the decision to remove this attribute is made.

The *attack_cat* attribute on the other hand after some inspection recognizes that the values “*normal*” in this attribute mean that the datapoints are not an attack which is especially the same as the *label* data which is the project target values of the project. Leaving this values meant that the model might develop the model base stuff that it should be predicted. Therefore, the decision to remove this attribute is made.

```
print(df.label.loc[df.attack_cat == "Normal"].unique())  
#yep  
print(df.label.loc[df.attack_cat != "Normal"].unique())  
✓ 0.0s  
[0]  
[1]
```

Figure 7: The output of the code that there the values *normal* in *attack_cat* equal 1 in the *label*

On the other hand, the data *sload* and *dload* are highly correlated to the data *sbytes* and *dbytes*. Since the *sload* and *dload* are the rates of the data being transferred which can be represented with the attributes *dur* in combination with *sbytes* and *dbytes* in order. They got removed.

2.5. Scaling and Standardisation

Data standardization is are required for this since the range of maximum values of each attribute in this data is large. Some data have maximum values as large as 1181164.235, some is a binary data of 0 and 1. In the above case, the large data will have more weight than the binary, so data standardization is needed.

2.6. Data imbalance

The initial data is slightly imbalanced, there are more datapoints labels as 1 more than 0 account for 73.41 percent of the data. Thus, resampling the data points labeled as 1 to have the same amount of data points labeled 0 is done to serve data balance and make the model train more effective.

```
# check the amount of normal and attack data
print(df.label.loc[df.label == 0].count())
print(df.label.loc[df.label == 1].count())
# there are data imbalance in the data

] ✓ 0.0s

38777
107077
```

Figure 8: The number of data points labeled as 1 and 0

2.7. Feature Engineering

Since the data have parameters that contain object data types such as proto, service, and state to use them in training the data in Sklearn API One-hot encoding has to be done.

3. Data Classification

3.1. Splitting training, validation, and test data

Splitting the data that got processed into training data, validation data, and test data, the ratio that got split are 60% training data, 20% validation data, and 20% test data. Training data will be used in training the machine learning model to predict the target value *label*. Validation data was used in the cross-validation process to find the general effectiveness of the model before training the model with training data. Test data is used to evaluate the model produced by the machine learning

3.2. KNN

In the K nearest neighbors model, the training was done using the *Sklearn* API after cross-validation to find the amount of K that the models perform the best here is the result.

	n_neighbors	Mean CV Score	Std CV Score
0	3	0.988653	0.002052
1	5	0.989040	0.001966
2	10	0.988137	0.002450
3	20	0.987493	0.002739
4	50	0.986397	0.002700
5	100	0.984720	0.002312
6	200	0.972213	0.001746
7	500	0.967507	0.007513
8	1000	0.960286	0.002156

Table 1: Cross-validation result of KNN models

The results show that the number of neighbors K that perform the best for this model is 5 followed by 3 and 10 using this result to train the K nearest neighbors as $K = 5$ and check how it performs using the confusion matrix and compute accuracy, precision, and F1 -measure using test data.

KNN(K=5)	Predict 0	Predict 1
Label 0	7657	130
Label 1	49	7675

Table 2: Confusion Matrix of KNN(K = 5) model

KNN(K=5)	Values
Accuracy	0.988460
Precision	0.983344
F1-measure	0.988461

Table 3: Performance tables of KNN(K = 5) model

From the tables, the models perform very well only 0.6 % of the data points labeled 1 (attack data) are undetected and have 1.6 % false positives. The evaluation score tables also confirm that interpretation.

3.3. Naive Bayes

For the Naive Bayes model, the training was done using the *Sklearn* API. and check how it performs using the confusion matrix and compute accuracy, precision, and F1 -measure using test data.

Naive Bayes	Predict 0	Predict 1
Label 0	7787	0
Label 1	6139	1585

Table 4: Confusion Matrix of Naive Bayes model

Naive Bayes	Values
Accuracy	0.604216
Precision	1.000000
F1-measure	0.340531

Table 5: Performance tables of Naive Bayes model

The model performs very poorly in detecting the data points labeled 1 (attack data). only 20.5 % of the data points labeled 1 (attack data) are undetected but do not have false positives. The evaluation matrix Accuracy and F1-measure stated that the model performed poorly, but performed very well in Precision evaluation

3.4. Decision tree

Decision tree models are built by performing cross-validation on 3 hyperparameters; maximum depth, minimum sample leaf, and minimum sample split. Using the hyperparameters that perform the best in building the decision tree and check how it performs using the confusion matrix and compute accuracy, precision, and F1 -measure using test data.

Decision Tree	Predict 0	Predict 1
Label 0	7678	109
Label 1	35	7689

Table 6: Confusion Matrix of the best perform decision tree model

Decision Tree	Values
Accuracy	0.990394
Precision	0.990545
F1-measure	0.990353

Table 7: Performance tables of the best perform decision tree model

The model performs very well in all evaluation methods, and from the confusion matrix, the model only has 1.4% false positives and failed to detect the attack 0.4% of the time.

3.5. Comparison

	KNN(K=5)	Naive Bayes	Decision Tree
Accuracy	0.988460	0.604216	0.990394
Precision	0.983344	1.000000	0.990545
F1-measure	0.988461	0.340531	0.990353

Table 7: Performance tables of the best-performing model of each model type

From the table above, the best-performing models in classifying attack data points are the Decision Tree models and the K-nearest neighbors model ($K = 5$). Although the Naive Bayes models are not wrongly accused that normal datapoints to attack datapoints and perform better in the Precision method but, overall the others model perform better.

4. Prediction

4.1. The models used

- K-nearest neighbors model ($K = 5$)
- Decision Tree models(max_depth=10, min_samples_split=20, random_state = 5601)

4.2. Data Test1

KNN(K=5)	Predict 0	Predict 1
Label 0	9516	12769
Label 1	246	37276

Table 7: Performance tables of the best-performing model of each model type

Decision Tree	Predict 0	Predict 1
Label 0	9051	13234
Label 1	380	37142

Table 7: Performance tables of the best-performing model of each model type

Both the K-nearest neighbors mode and decision tree model have a similar problem in classifying this dataset, they are likely to wrongly accuse the model that is labeled as a normal activity to attack data points. But almost all of the

4.3. Data Test2

Decision Tree	Predict 0	Predict 1
Label 0	12642	6079
Label 1	222	39078

Table 7: Performance tables of the best-performing model of each model type

Decision Tree	Predict 0	Predict 1
Label 0	12389	6332
Label 1	358	38942

Table 7: Performance tables of the best-performing model of each model type

Both the K-nearest neighbors mode and decision tree Perform very well in classifying the dataset test giving that less than 10 percent of the attack activity is un detect and onlu wrongly labeled about half of the normal activity to be attack activity

5. Conclusion

Of all the three models used to classify the attacking activity from the log data. The models that perform very well at detecting the attacking activity in this study are the decision tree and K-nearest neighbors. Both models performed very well with the testing data after fitting to the training data, but their performance dropped from the testing when introduced to the new data that they hadn't seen before. There are 2 speculations of why that might happen. First, the testing data given might have lots of outliers that are hard for the models to classify, this one is not likely since there needed lots of outliers data to see the significant drop in performance as seen in the result. The second speculation that is more likely to happen to this project is that the model is overfitting to the training data so when the model got introduced to the new information the performance of them drops. In the future, might try some method that aspect to reduced overfitting to see that the speculation is true or not.

6. Reference

[1]

scikit-learn, "DecisionTreeClassifier," *scikit-learn*, 2024. Available:

<https://scikit-learn.org/1.5/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>. [Accessed: Oct. 18, 2024]

[2]

scikit-learn, "sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.22.1 documentation," *Scikit-learn.org*, 2019. Available:

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

[3]

scikit-learn, "sklearn.naive_bayes.GaussianNB — scikit-learn 0.22.1 documentation," *scikit-learn.org*. Available:

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html