DATA MINING REPORT

NETWORK INTRUSION DETECTION SYSTEM

Module Coordinator
Dr. Antonio Nehme

Submitted By
Meera Mohan Mullamkuzhy (22185279)
MSc Big Data Analytics

Faculty of Computing, Engineering and the Built Environment

December 2022

Contents

List of Figures

List of Tables

1 INTRODUCTION

Data security on our computer systems is constantly in danger. Rapid expansion of the Internet has made intrusion detection and prevention an essential element of networked systems.Things have gotten worse due to the proliferation of tools and techniques for breaking into and attacking networks.The number of internet security events has increased over the last two decades, and high-profile data breaches and cyber attacks continue to make news on a daily basis. Hence,the presence of an intrusion detection system is crucial to identify and detect these situations.

1.1 Intrusion Detection System (IDS)

An intrusion detection system is a tool or software that monitors a network for unauthorized users and detects network intrusions. The system defends against internet attacks on the confidentiality, integrity, and availability of data and information systems. Figure 1 shows the overall architecture of IDS.

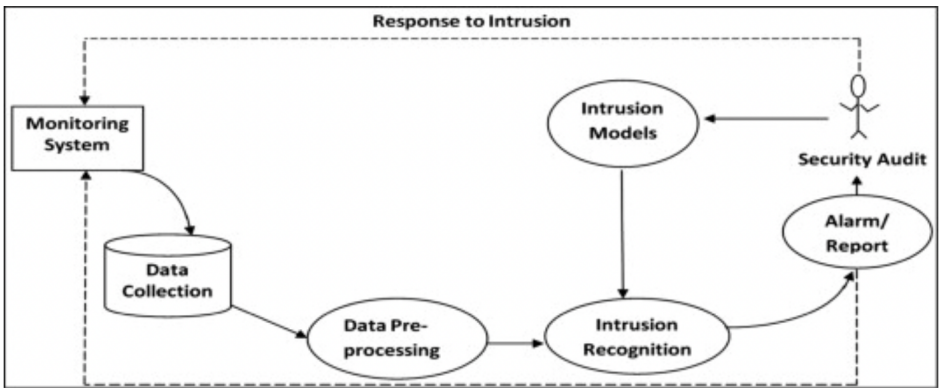

Figure 1: *Overall architecture of IDS (Nadiammai and Hemalatha, 2013)*

2 DOMAIN DESCRIPTION : DATA MINING IN CYBER SECURITY

Cyber security is the primary application of Intrusion Detection Systems. Data mining and cyber security can be combined to improve attack detection methods and identify characteristics of cyber attacks.It involves analyzing massive collections of data using sophisticated algorithms and techniques in order to find patterns, trends, and anomalies that can point to possible security risks.Additionally, data mining can be used to find potential flaws in systems or networks and to create predictive models for detecting and preventing future security risks. Overall architecture of a Network Intrusion Detection System is given in Figure 2.

Figure 2: *Overall architecture of IDS (Apriorit, 2022)*

2.1 IDS in Cyber Security

Every network security system includes an IDS.An IDS assists in finding, determining, and identifying the unauthorized use, duplication, alteration, and destruction of information systems.It plays a crucial role in identifying malicious intrusions in organizational databases, networks, servers, web clients and operating systems.

Any domain where data management is involved would benefit from this. E.g., Network Security Surveillance, Software security, Compliance And Investigations.

3 PROBLEM DEFINITION

With the explosion of Internet and networks into almost all fields of business and life, data security is constantly at risk. New and improved methods of Intrusion Softwares and other malicious malwares are available in the market today, which can bring even big organizations to their knees. Year on year Billions of dollars are lost by individuals and organizations alike to these Intrusions and the data that is being transferred from point to point is increasing by several folds each year.

The sectors where intrusion detection plays a crucial role includes National security, healthcare and the financial sector. All these sectors thrive on secrecy and privacy of data and the moment there is a breach, the losses could be huge. The recent data breach of First American Financial Corp in 2019 where 885 million financial & personal records were breached,(Dellinger, 2019) or the 2020 United States Federal Government data breach, or the One Touch Point breach where 2.6 million individuals were affected are all testimonies to the necessity of an improved intrusion detection and prevention system. (Steven, 2022)

3.1 Cyber Crime Statistics

1. 175 Zettabytes – of data will be generated and will be circulating by 2025 (Sean, 2022).

2. $ 10.5 trillion – is the predicted Cost of cyber crime by 2025 (Sausalito, 2020).

3. $ 56 Billion – is the tallied total of Identity fraud losses according to 2021 Identity Fraud study from Javellin Strategy Research (Javelin, 2021).

4. 287 days – is the average time it takes for security teams to identify and contain a data breach.(Sabrina, 2022).

5. $ 4.35 Million – is the average cost of a Data breach for Small and Medium Businesses. (IBM, 2022)

6. 72 Million – people fall victim to Cyber crimes each year with numbers increasing (Sean, 2022)

For preventing such a big data loss or breach, data intrusion systems combined with Cyber security is essential. Since the amount of traffic that has to be analyzed for achieving this feat is huge, Data mining is done to filter through the network traffic data.

4 LITERATURE REVIEW

Several research papers are available for developing models for intrusion detection system.Dikshant Gupta et al. (2016) suggests to perform data normalisation and data transformation to make the data ready for analysis. An accuracy of 80.14% and 67.53% are obtained by using the techniques Linear regression and K-means clustering respectively.

Solane Duque and Dr.Mohd. Nizam bin Omar (2015) verifies that K-means clustering shows a high accuracy of 81.61% when the correct number of clusters is used. Identifying the correct number of clusters will be a challenge as there is no standard available for deciding the number of clusters.

For Naive Bayes, some of the attributes are irrelevant, which lowers the model's performance.Gaurav Meena and Ravi Raj Choudhary (2017) studied the important features and achieved an accuracy of 99.435% and 92.715% for the models J48 Graft Decision Tree and Naive Bayes respectively.

An automated data mining tool WEKA is used to generate models with NSL-KDD data set. Correlation based feature selection and normalisation of data produced a model with better accuracy. J48 model with Correlation Feature Selection classified the data with an accuracy of 99.8%. Support Vector Machine and Naive Bayes models showed an accuracy of 98.8% and 74.9% respectively with Correlation based Feature Selection (L.Dhanabal and Dr. S.P. Shantharajah, 2015).

Arif Jamal Malik et al. (2015) suggests a hybrid PSO-RF model to select the best features for Intrusion Detection System. Particle Swarm Optimisation technique was used to select the features. All 23 types of attacks were grouped into four categories such as Denial-of-Service, PROBE, User-to-Root, and Remote-to-local. Random Forest model was used for classification on the

features selected with PSO. The performance of Random Forest was compared with PART, Naive Bayesian, CART, and Bagging models. PSO-RF proved to be a model with the highest efficiency.

Different data mining models were evaluated by Ajayi Adebowale et al. (2013) the performance of Decision Trees, Naive Bayes, Artificial Neural Network, Support Vector Machine, and K-Nearest Neighbor.
Ajayi Adebowale et al. (2013) compared the performance of five classification models such as Decision Trees, Naive Bayes, Artificial Neural Networks, K-Nearest Neighbor, and Support Vector Machine. The experiments were carried out on Weka 3.6.7. The algorithms were evaluated with 10-fold cross validation. Decision Tree J48 yielded the highest accuracy of 99.5594% and the best kappa statistics at 0.9911. K-Nearest Neighbour algorithm had the best build time at 0.03 seconds whereas Naive Bayes model had the second best build time at 1.6 seconds.

Enhanced Adaboost algorithm with Rough set theory based feature selection achieves a classification model with low false alarm rate and high attack detection rate. A multi stage filter that filters DoS attacks, Probe attacks, and R2L and U2R attacks is used. The detection rate of multi stage filter with Adaboost was better compared with the existing NSL-KDD'99 benchmark (Prof P Balasubramanie and P Natesan, 2012).

Hee-su Chae et al. (n.d) chose J48 classifier and 10-fold cross validation to evaluate the features selected using Attribute Ratio (AR) ranker. The accuracy of AR was compared with Correlation Feature Selection (CFS), Information Gain (IG), and Gain Ratio (GR). Models that used the features selected using GR and AR shared an accuracy of 99.794% whereas CFS showed an accuracy of 99.781% with 25 features. J48 classifier that used IG based feature selection had an accuracy of 99.781% with 23 features.

5 DATA SET DESCRIPTION

The application uses the NSL-KDD data set, which was downloaded from Kaggle.One of the few publicly available data sets for the network-based anomaly detection system, it was first utilized in the 3rd International Knowledge Discovery and Data Mining tools Competition.

MIT Lincoln Labs conducted an Intrusion detection Evaluation Program in 1998 in which a local-area network (LAN) modeled after a typical U.S. Air Force LAN, was set up in order to collect nine weeks' worth of raw TCP dump data.The data set includes multiple network intrusions simulated in a military network environment.The KDDCUP'99 data set and NSL-KDD data sets are modified versions of the DARPA'98 data set.

We chose the NSL-KDD data set over KDDCUP'99 since it does not contain any redundant or duplicate records.Also,the number of records in the train sets and test sets of NSL-KDD data set is reasonable.

The chosen data set consists of 42 attributes, 125,973 instances in the training set and 22,544 instances in the test set.
The details of the attributes in the dataset is given the Table 1.

Nature of data	
Type	**Attributes**
Nominal	protocol_type service flag
Binary	land logged_in root_shell su_attempted is_host_login is_guest_login
Numeric	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 serror_rate srv_serror_rate rerror_rate srv_rerror_rate same_srv_rate diff_srv_rate srv_diff_host_rate 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_srv_diff_host_rate dst_host_serror_rate dst_host_srv_serror_rate dst_host_rerror_rate dst_host_srv_rerror_rate

Table 1 : *Nature of data*

6 DATA SET PRE-PROCESSING AND EDA

6.1 Missing/Duplicate/Null values:

Data pre-processing was done on the data set and it was observed that the data is devoid of missing values, null values and duplicate values.

6.2 Data Imbalance

As, it is observed from the KDD Train data set, the attribute "label" is not balanced, we had balanced the data by grouping all the small malicious attacks in one category. After processing, all malicious attacks account for 46.54% of all attacks, while all other attacks account for 53.46% . Figure 3 shows a pie chart of normal and malicious attacks.

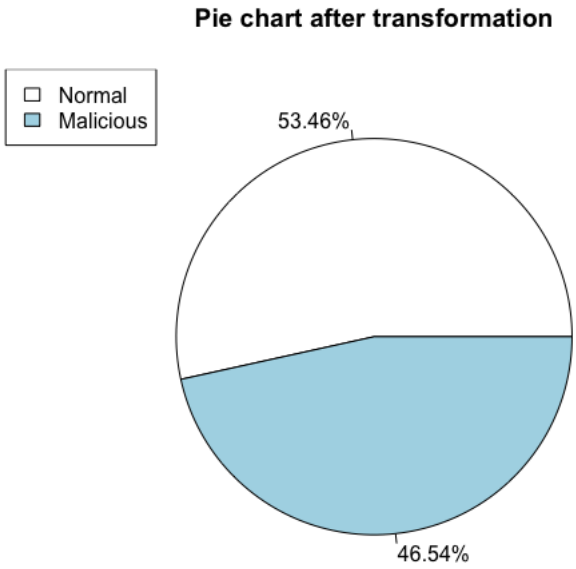

Figure 3: *Pie Chart after Data Transformation*

6.3 Feature Selection

Feature selection is performed before data mining because it helps in reducing the number of predictor variables to those that are regarded as the most beneficial to a model in order to predict the target variable.

6.3.1 Using Information Gain

Initially, we used the Information Gain approach to reduce the 41 attributes in the data set. Information Gain for each attribute is calculated and the average of all gains are computed.Features with gains more than the average would be selected as the shortlisted features. The overall average gain is 0.009753441.

The selected features after applying the Feature selection is as follows:

- service
- flag
- src_bytes
- dst_bytes
- hot
- is_guest_login
- count
- srv_count
- dst_host_srv_count
- dst_host_same_srv_rate
- dst_host_diff_srv_rate
- dst_host_same_src_port_rate
- labels

Figure 4 shows structure of the selected data.

```
> str(kdd_train)
'data.frame':   125973 obs. of  14 variables:
 $ service                  : chr  "ftp_data" "other" "private" "http" ...
 $ flag                     : chr  "SF" "SF" "S0" "SF" ...
 $ src_bytes                : int  491 146 0 232 199 0 0 0 0 0 ...
 $ dst_bytes                : int  0 0 0 8153 420 0 0 0 0 0 ...
 $ hot                      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ is_guest_login           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ count                    : int  2 13 123 5 30 121 166 117 270 133 ...
 $ srv_count                : int  2 1 6 5 32 19 9 16 23 8 ...
 $ diff_srv_rate            : num  0 0.15 0.07 0 0 0.06 0.06 0.06 0.05 0.06 ...
 $ dst_host_srv_count       : int  25 1 26 255 255 19 9 15 23 13 ...
 $ dst_host_same_srv_rate   : num  0.17 0 0.1 1 1 0.07 0.04 0.06 0.09 0.05 ...
 $ dst_host_diff_srv_rate   : num  0.03 0.6 0.05 0 0 0.07 0.05 0.07 0.05 0.06 ...
 $ dst_host_same_src_port_rate: num  0.17 0.88 0 0.03 0 0 0 0 0 0 ...
 $ labels                   : chr  "normal" "normal" "neptune" "normal" ...
```

Figure 4: *Structure of the Features selected through IG method*

6.3.2 Validation of selected features using Pair plot

A pair plot is plotted to compute the correlation between selected features. Out of the 14 features selected, after validating the relationship between selected features, it is observed that the Pearson coefficient value of the attributes src_bytes and dst_bytes are 0.01 and 0.00 respectively. Pair plot of the 14 selected features is given in the Figure 5.

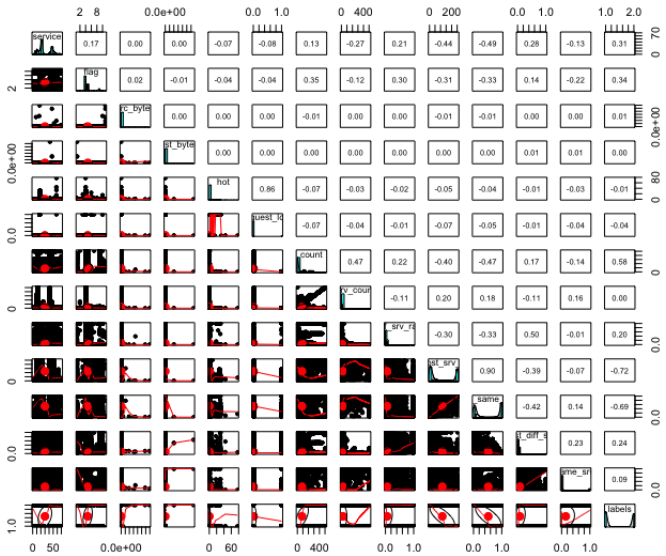

Figure 5: *Pair plot of the 14 selected features*

6.4 Data Transformation

We need to convert the nominal features from these labelled connection instances to numerical values in order to make them viable inputs for classification utilising data mining techniques.(Dikshant et al., 2016) Nominal values from features flag, service and labels were converted to numeric values. For example; In the column "labels", we have assigned a target class of zero for a normal connection and one for any attack.
Table 2 shows the values assigned to the attributes.

Nature of data		
Attribute	**Attribute Value**	**Assigned Value**
flag	OTH	2
	SF	3
	S0	4
	S1	5
	S2	6
	S3	7
	REJ	8
	RSTO	9
	RSTOSO	10
	RSTR	11
	SH	12
Binary	normal	0
	neptune	1
	warezclient	1
	ipsweep	1
	portsweep	1
	teardrop	1
	nmap	1
	satan	1
	smurf	1
	pod	1
	back	1
	guess_passwd	1
	multihop	1
	rootkit	1
	buffer_overflow	1
	imap	1
	warezmaster	1
	phf	1
	land	1
	loadmodule	1
	spy	1
	perl	1
	ftp_write	1

Numeric	aol	13
	auth	14
	bgp	15
	courier	16
	csnet_ns	17
	ctf	18
	daytime	19
	discard	20
	domain	21
	domain_u	22
	echo	23
	eco_i	24
	ecr_i	25
	efs	26
	exec	27
	finger	28
	ftp	29
	ftp_data	30
	gopher	31
	harvest	32
	hostnames	33
	http	34
	http_2784	35
	htp_443	36
	http_8001	37
	imap4	38
	IRC	39
	iso_tsap	40
	klogin	41
	kshell	42
	ldap	43
	link	44
	login	45
	mtp	46
	name	47
	netbios_dgm	48
	netbios_ns	49
	netbios_ssn	50
	netstat	51
	nnsp	52
	ntp_u	53
	other	54
	pm_dump	55
	pop_2	56
	pop_3	57
	printer	58
	private	59
	red_i	60
	remote_job	61
	rje	62
	shell	63
	smtp	64
	sql_net	65

Numeric	ssh sunrpc supdup systat telnet tftp_u tim_i time urh_i urp_i uucp uucp_path vmnet whois X11 Z39_50	66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81

Table 2 : *Details of the Values used for Data Transformation*

6.5 Data Normalization

Data set need to be normalized so that all the values are in the range of 0 to 1.(Bhupendra and Anamika, 2015) Normalised data is used as an input for the KNN algorithm.

7 EXPERIMENTS

7.1 Random Forest Algorithm

Random forest is a commonly-used machine learning algorithm which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems (IBM, 2020).

The decision tree's fundamental goal is classification or prediction. Each and every decision tree has a unique outcome.For classification and regression, the final result is based on the majority vote or average, respectively.

7.2 Naive Bayes

Naive Bayes is a supervised non-linear classification technique. A family of straightforward probabilistic classifiers called Naive Bayes classifiers works by applying the Bayes theorem to features or variables while making strong (Naive) assumptions about their independence. Because it assumes that the occurrence of one feature is unrelated to the occurrence of other features, the Naive Bayes method is known as "Naive."

7.3 K Nearest Neighbour

The k-nearest neighbours algorithm, or KNN for short, is a supervised learning classifier that uses proximity to produce classifications or predictions about how a particular data point will be grouped(IBM, n.d).

8 ANALYSIS AND RESULTS

8.1 Random Forest Algorithm

Table 3 shows accuracy of Random Forest model with different set of features.

Feature Selection Method and Accuracy of Random Forest	
Feature Selection Method	**Accuracy**
Information Gain	91.47%
IG Removal of features with 0 coefficient value	91.47%

Table 3 : *Feature Selection Method and Accuracy of Random Forest*

The accuracy of RF models is 91.47% in both cases where features are chosen either using the IG method alone or in conjunction with the correlation method. Figures 6 and 7 provide a summary of the RF model with features selected using the IG approach, as well as a combination of IG and the removal of attributes with zero correlation.

```
Overall Statistics

               Accuracy : 0.9147
                 95% CI : (0.911, 0.9184)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8368

 Mcnemar's Test P-Value : NA
```

Figure 6: *Summary of RF with features selected using IG method*

```
Overall Statistics

               Accuracy : 0.9147
                 95% CI : (0.911, 0.9184)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8368

 Mcnemar's Test P-Value : NA
```

Figure 7: *Summary of RF with features selected using IG method and zero related attributes removal*

The confusion matrix for both the models are given in the figures 8 and 9 respectively.

```
Call:
 randomForest(formula = labels ~ ., data = kdd_train, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 0.15%
Confusion matrix:
      0     1 class.error
0 67262    81 0.001202798
1   104 58526 0.001773836
```

Figure 8: *Summary of RF with features selected using IG method*

```
Call:
 randomForest(formula = labels ~ ., data = kdd_train_reduced,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 0.51%
Confusion matrix:
      0     1 class.error
0 66971   372 0.005523959
1   268 58362 0.004571039
```

Figure 9: *Summary of RF with features selected using IG method and elimination of features with zero relationship coefficient*

8.1.1 Cross Validation and Hyper-Parameter Tuning

The parameters mtry and ntree are considered for RF model parameter tuning. RF model gives the best accuracy for mtry value equal to square root of p or p/3 , where p is the number of predictors, and for large values of ntree(Simon Bernard et al., 2009;Ramón Díaz-Uriarte and Sara Alvarez de Andrés, 2006). Accuracy of RF models with different values of ntree are given in the Table 4.

Accuracy of Random Forest for different values of ntree	
ntree value	**Accuracy**
300	91.41%
400	91.47%
500	91.41%
600	91.43%
700	91.43%
800	91.43%

Table 4 : *Accuracy of Random Forest for different values of ntree*

ntree value of 400 was selected because of the better accuracy of the RF model. Table 5 shows the accuracies of RF models for ntree value equal to 400 and for different values of mtry.

Accuracy of Random Forest for different values of mtry	
mtry value	**Accuracy**
1	90.99%
3	91.35%
4	91.35%
5	91.34%

Table 5 : *Accuracy of Random Forest for different values of ntree*

The models with mtry value equal to 3 and 4 showed better accuracy as the value of p is 13.
K-cross validation was carried out on the Random Forest model with high accuracy. 10-fold cross validation yielded better accuracy than other values of k. 91.5% accuracy was observed for 10-fold cross validation model. Figures 10 shows the model's accuracy with 10-fold validation applied.

```
Overall Statistics

               Accuracy : 0.9153
                 95% CI : (0.9116, 0.9189)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8378

 Mcnemar's Test P-Value : NA
```

Figure 10: *Summary of RF model with 10-fold cross validation*

8.2 Naive Bayes Algorithm

Two Naive Bayes classification models were evaluated based on the two different feature selection methods mentioned in the section 6.3.

Table 6 shows accuracy of the model with different set of features.

Feature Selection Method and Accuracy of Naive Bayes Classifier	
Feature Selection Method	**Accuracy**
Information Gain	78.54%
IG Removal of features with 0 coefficient value	91.2%

Table 6 : *Feature Selection Method and Accuracy of Naive Bayes Classifier*

```
Overall Statistics

               Accuracy : 0.7854
                 95% CI : (0.7799, 0.7907)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5859
```

Figure 11: *Summary of Naive Bayes model with features selected using IG method*

```
Overall Statistics

               Accuracy : 0.912
                 95% CI : (0.9083, 0.9157)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8319
```

Figure 12: *Summary of Naive Bayes model with features selected using IG method and elimination of features with zero relationship coefficient*

The figures 11 and 12 shows the accuracy, p-value and Kappa statistic of both the models. Kappa value of 0.8319 represents an excellent agreement to the data selected in the study.

8.2.1 Cross Validation and Hyper-Parameter Tuning

K-cross validation was carried out on the Naive Bayes model with high accuracy. Figures 13 and 14 show, respectively, the model's accuracy when 3-fold validation and 10-fold validation are applied.

```
> nb_model$results
  usekernel fL adjust  Accuracy     Kappa  AccuracySD     KappaSD
1     FALSE  0      1 0.9860367 0.9718877 0.001200298 0.002420582
2      TRUE  0      1 0.9860367 0.9718877 0.001200298 0.002420582
```

Figure 13: *Accuracy of 3-fold validation on Naive Bayes model*

```
> nb_model$results
  usekernel fL adjust  Accuracy     Kappa   AccuracySD     KappaSD
1     FALSE  0      1 0.9864019 0.9726243 0.0009301562 0.001875856
2      TRUE  0      1 0.9864019 0.9726243 0.0009301562 0.001875856
```

Figure 14: *Accuracy of 10-fold validation on Naive Bayes model*

Accuracy after performing both 3-fold and 10-fold validations are approximately equal. The hyper-parameters used for validation are usekernal, fl and adjust.

8.3 K Nearest Neighbour

Table 7 shows accuracy of KNN model with different set of features.

Feature Selection Method and Accuracy of KNN	
Feature Selection Method	**Accuracy**
Information Gain	88.36%
IG Removal of features with 0 coefficient value	88.36%

Table 7 : *Feature Selection Method and Accuracy of Random Forest*

Figures 15 and 16 provide a summary of the KNN model with features selected using the IG approach, as well as a combination of IG and the removal of attributes with zero correlation.

```
Overall Statistics

               Accuracy : 0.8836
                 95% CI : (0.8794, 0.8878)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7773

 Mcnemar's Test P-Value : NA
```

Figure 15: *Summary of KNN model with features selected using IG method*

```
Overall Statistics

               Accuracy : 0.8836
                 95% CI : (0.8794, 0.8878)
    No Information Rate : 0.4988
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7773

 Mcnemar's Test P-Value : NA
```

Figure 16: *Summary of Naive Bayes model with features selected using IG method and elimination of features with zero relationship coefficient*

8.3.1 Hyper-Parameter tuning

The number of neighbours (K) for KNN is the most crucial hyper parameter. (Jason, 2019) The square root of N, where N is the total number of samples, is typically determined to be the optimal K value. To find the most favorable K value, an error plot or accuracy plot needs to be plotted (Amey, 2020).

Here, for 125,973 observations in the training data set, the K would be square root of 125,973, which is 354.92. Rounding it to 355.

9 CONCLUSIONS

Organisations around the world spend millions each year to ensure their systems are secure from both internal and external threats. Data, be it corporate or individual, needs to be secured from unintended access and usages. Hence, mitigating these network intrusions are crucial to ensure network security.

In the NSL-KDD set, we used the data mining methods such as Random Forest, Naive Bayes, and K-Nearest Neighbor to analyse network intrusions. Out of the 41 attributes present in the data set, feature selection was employed to retrieve only those attributes that are beneficial to the model. To extract the key features, both the Information Gain Method separately and the Information Gain Method combined with the Correlation Method were used. It was observed that when features from the correlation technique were used over

the information gain approach, the Naive Bayes offers an accuracy of 91.2% as opposed to 78.54%. Whereas, both the Random Forest and KNN algorithms have same accuracy while using both methods. Hyper-Parameter tuning and Cross-validation were also conducted on the models to increase efficiency.

10 References

1. G.V Nadiammai and M. Hemalatha (2013) *Effective approach toward Intrusion Detection System using data mining techniques.* Available at: https://www.researchgate.net/publication/259521163_Effective_approach_toward_Intrusion_Detection_System_using_data_mining_techniques [Accessed 7 December 2022]

2. Apriorit (2022) *Using Data Mining Techniques in Cybersecurity Solutions.* Available at: https://www.apriorit.com/dev-blog/527-data-mining-cyber-security [Accessed 6 December 2022]

3. AJ Dellinger (2019) *Understanding The First American Financial Data Leak: How Did It Happen And What Does It Mean?.* Available at: https://www.forbes.com/sites/ajdellinger/2019/05/26/understanding-the-first-american-financial-data-leak-how-did-it-happen-and-what-does-it-mean/?sh=4936db0c567f [Accessed 8 December 2022]

4. Steven (2022) *OneTouchPoint's Data Breach.* Available at: https://www.idstrong.com/sentinel/onetouchpoints-data-breach/ [Accessed 8 December 2022]

5. Sean Michael Kerner (2022) *34-Cyber Security Staistics to Lose Sleep Over.* Available at:https://www.techtarget.com/whatis/34-Cybersecurity-Statistics-to-Lose-Sleep-Over-in-2020. [Accessed 08 December 2022]

6. Sausalito (2020) *Cybercrime To Cost The World $10.5 Trillion Annually By 2025.* Available at: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/ [Accessed 06 December 2022]

7. Javelin (2021) *Total Identity Fraud Losses Soar to $56 Billion in 2020.* Available at: https://javelinstrategy.com/press-release/total-identity-fraud-losses-soar-56-billion-2020 [Accessed 06 December 2022]

8. Sabrina Pagnotta (2022) *What's the cost of a data breach in 2022?.* Available at: https://www.bitsight.com/blog/cost-of-data-breach [Accessed 06 December 2022]

9. IBM (2022) *How much does a data breach cost in 2022?.* Available at: https://www.ibm.com/uk-en/security/data-breach [Accessed 7 December 2022]

10. IBM (2020) *Random Forest.* Available at: https://www.ibm.com/cloud/learn/random-forest [Accessed 7 December 2022]

11. IBM (n.d) *What is the k-nearest neighbors algorithm?*. Available at: https://www.ibm.com/uk-en/topics/knn [Accessed 10 December 2022]

12. Hee-su Chae, Byung-oh Jo, Sang-Hyun Choi Twae-kyung Park (2019) *Feature Selection for Intrusion Detection using NSL-KDD.* Available at: https://e-tarjome.com/storage/btn_uploaded/2019-07-13/1563003303_9701-etarjome-English.pdf. [Accessed 11 December 2022]

13. Rajesh Thomas and Deepa Pavithran (2018) *A Survey of Intrusion Detection Models based on NSL-KDD Data Set, 2018 Fifth HCT Information Technology Trends (ITT), pp. 286-291, doi: 10.1109/CTIT.2018.8649498.*. Available at: https://ieeexplore.ieee.org/abstract/document/8649498 [Accessed 7 December 2022]

14. Dikshant Gupta, Suhani Singhal, Shamita Malik and Archana Singh (2016) *Network intrusion detection system using various data mining techniques, 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), 2016 pp. 1-6, doi: 10.1109/RAINS.2016.7764418.* . Available at: https://ieeexplore.ieee.org/abstract/document/7764418?casa_token=36VOB8fxe_gAAAAA:Mtxa73F7sdp0dICOmlmQnoxq51I594NWkOlC8jp49eG4hN7QrLqgYyibrdZeuzgbvWAIEmyeKno [Accessed 7 December 2022]

15. Bhupendra Ingre and Anamika Yadav (2015) *Performance analysis of NSL-KDD dataset using ANN, 2015 International Conference on Signal Processing and Communication Engineering Systems, 2015, pp. 92-96, doi: 10.1109/SPACES.2015.7058223.* . Available at: https://ieeexplore.ieee.org/abstract/document/7058223?casa_token=jzgTtETZHOMAAAAA:QtwOvtcAt_ql4f6F_8x01xtVrI88CZoBPU45NnUYTLEZULaBQ_qHW6w7ucVXe14rxfoYOvCtA3E [Accessed 7 December 2022]

16. Simon Bernard, Laurent Heutte and Sébastien Adam (2009) *Influence of Hyperparameters on Random Forest Accuracy.* Available at: https://link.springer.com/chapter/10.1007/978-3-642-02326-2_18 [Accessed 7 December 2022]

17. Ramón Díaz-Uriarte and Sara Alvarez de Andrés (2006) *Gene selection and classification of microarray data using random forest.* Available at: https://link.springer.com/article/10.1186/1471-2105-7-3 [Accessed 7 December 2022]

18. Amey Band (2020) *How to find the optimal value of K in KNN?*. Available at: https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb [Accessed 8 December 2022]

19. Jason Brownlee (2019) *Tune Hyperparameters for Classification Machine Learning Algorithms.* Available at: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/ [Accessed 8 December 2022]

20. Solane Duque and Dr.Mohd. Nizam bin Omar (2015) *Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS).* Available at: https://reader.elsevier.com/reader/sd/pii/S18770

50915029750?token=390AAD20682C84EEA07C9906BFD7BC0A61A13D0E6FB647D79B2DB63F332B549A4945C4E3B6C225BB5BA78047E75A85EB&originRegion=eu-west-1&originCreation=20221213045742 [Accessed 8 December 2022]

21. Gaurav Meena and Ravi Raj Choudhary (2017) *A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA*. Available at: https://ieeexplore.ieee.org/abstract/document/8004032/authors#authors [Accessed 8 December 2022]

22. L.Dhanabal and Dr. S.P. Shantharajah (2015) *A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms*. Available at: http://faratarjome.ir/u/media/shopping_files/store-EN-1484204753-3159.pdf [Accessed 8 December 2022]

23. Arif Jamal Malik, Waseem Shahzad and Farrukh Aslam Khan (2015) *Network Intrusion Detection using hybrid binary PSO and random Forests Algorithm*. Available at: https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.508 [Accessed 8 December 2022]

24. Ajayi Adebowale, Idowu S.A and Anyaehie Amarachi A. (2013) *Comparative Study of Selected Data Mining Algorithms Used For Intrusion Detection*. Available at: https://www.researchgate.net/profile/Adebowale-Ajayi/publication/331304899_Comparative_study_of_selected_data_mining_algorithms_used_for_intrusion_detection/links/5c7181db92851c69503adeaf/Comparative-study-of-selected-data-mining-algorithms-used-for-intrusion-detection.pdf [Accessed 8 December 2022]

25. P.Natesan and P.Balasubramanie (2012) *Multi Stage Filter Using Enhanced Adaboost for Network Intrusion Detection*. Available at: https://www.researchgate.net/profile/Prof-P-Balasubramanie/publication/259561510_Multi_Stage_Filter_Using_Enhanced_Adaboost_for_Network_Intrusion_Detection/links/54dc8e1f0cf282895a3a9096/Multi-Stage-Filter-Using-Enhanced-Adaboost-for-Network-Intrusion-Detection.pdf [Accessed 8 December 2022]

11 Appendix

```
#*****************************************************************#
#*****NETWORK INTRUSION DETECTION SYSTEM CODE*******************#
#*****************************************************************#


library(FSelector)
library(caret)
library(e1071)
library(randomForest)
library(class)
library("ElemStatLearn")
library("klaR")
library(naivebayes)
library(dplyr)
```

```r
library(ggplot2)
library(psych)
library("class")

#--------------Read train and test data : start
    ---------------------------
print("Reading Train and Test data \n")
#kdd_test <- read.csv(file.choose(), header = TRUE)
#kdd_transform <- read.csv(file.choose(), header = TRUE)
kdd_train <- read.csv("/Users/meeramohanm/BCU/Assignments/
    DataMining/kdd_train.csv")
kdd_test <- read.csv("/Users/meeramohanm/BCU/Assignments/DataMining
    /kdd_test.csv")
str(kdd_transform)
str(kdd_test)
#--------------Read train and test data : end
    ----------------------


#****************************************************************
#---------------DATA PRE-PROCESSING & EDA ----------------------
#****************************************************************


#-------------- Data Cleaning : start
    -------------------------------
#1. Check for Null Values
is.null(kdd_train)

#2. Checking for missing values
table(is.na.data.frame(kdd_train))

#3. Check for duplicate values (no duplicate values if the
    statement returns the value 0)
sum(duplicated(kdd_train))
#-------------- Data Cleaning : end--------------------------------

#-------*DATA IMBALANCE CHECK START*--------


#Plot the Pie Chart
x <- table(kdd_train$labels)

pie_labels <- paste0(round(100 * x/sum(x), 2), "%") #Computing %

pie(x, labels = pie_labels, main= "Pie Chart for Labels")

legend("topright", legend <- c("Normal", "Malicious"),fill = c("
    white", "lightblue"))

#-------*DATA IMBALANCE CHECK END*--------

#-------------- Feature Selection : start-------------------------
#Information Gain Feature Selection Method
ig <- information.gain(formula(kdd_train),kdd_train)
avg_ig <- sum(information.gain(formula(kdd_train),kdd_train))/41
cat("Average Information Gain is ", avg_ig)

ig <- information.gain(formula(kdd_train),kdd_train)
avg_ig <- sum(information.gain(formula(kdd_train),kdd_train))/41
cat("Average Information Gain is ", avg_ig)

```

```
#Select features based on IG value
kdd_train <- kdd_train[,c(3,4,5,6,10,22,23,24,30,33,34,35,36,42)]
str(kdd_train)

kdd_test <- kdd_test[,c(3,4,5,6,10,22,23,24,30,33,34,35,36,42)]
str(kdd_test)

#Plot pair plot to find correlation
pairs.panels(kdd_train)

#-------------- Feature Selection : end-------------------------

#--------------Data Transformation : start--------------------
#Assign values to attributes service, flag, and labels
kdd_train$labels[which(kdd_train$labels == "normal")] <- 0
kdd_train$labels[which(kdd_train$labels == "neptune")] <- 1
kdd_train$labels[which(kdd_train$labels == "warezclient")] <- 1
kdd_train$labels[which(kdd_train$labels == "ipsweep")] <- 1
kdd_train$labels[which(kdd_train$labels == "portsweep")] <- 1
kdd_train$labels[which(kdd_train$labels == "teardrop")] <- 1
kdd_train$labels[which(kdd_train$labels == "nmap")] <- 1
kdd_train$labels[which(kdd_train$labels == "satan")]<- 1
kdd_train$labels[which(kdd_train$labels == "smurf")]<-1
kdd_train$labels[which(kdd_train$labels == "pod")]<-1
kdd_train$labels[which(kdd_train$labels == "back")]<-1
kdd_train$labels[which(kdd_train$labels == "guess_passwd")]<-1
kdd_train$labels[which(kdd_train$labels == "multihop")]<-1
kdd_train$labels[which(kdd_train$labels == "rootkit")]<-1
kdd_train$labels[which(kdd_train$labels == "buffer_overflow")]<-1
kdd_train$labels[which(kdd_train$labels == "imap")]<-1
kdd_train$labels[which(kdd_train$labels == "warezmaster")]<-1
kdd_train$labels[which(kdd_train$labels == "phf")]<-1
kdd_train$labels[which(kdd_train$labels == "land")]<-1
kdd_train$labels[which(kdd_train$labels == "loadmodule")]<-1
kdd_train$labels[which(kdd_train$labels == "spy")]<-1
kdd_train$labels[which(kdd_train$labels == "perl")]<-1
kdd_train$labels[which(kdd_train$labels == "ftp_write")]<-1

kdd_train$flag[which(kdd_train$flag == "OTH")] <- 2
kdd_train$flag[which(kdd_train$flag == "SF")] <- 3
kdd_train$flag[which(kdd_train$flag == "S0")] <- 4
kdd_train$flag[which(kdd_train$flag == "S1")] <- 5
kdd_train$flag[which(kdd_train$flag == "S2")] <- 6
kdd_train$flag[which(kdd_train$flag == "S3")] <- 7
kdd_train$flag[which(kdd_train$flag == "REJ")] <- 8
kdd_train$flag[which(kdd_train$flag == "RSTO")] <- 9
kdd_train$flag[which(kdd_train$flag == "RSTOS0")] <- 10
kdd_train$flag[which(kdd_train$flag == "RSTR")] <- 11
kdd_train$flag[which(kdd_train$flag == "SH")] <- 12

kdd_train$service[which(kdd_train$service == "aol")] <- 13
kdd_train$service[which(kdd_train$service == "auth")] <- 14
kdd_train$service[which(kdd_train$service == "bgp")] <- 15
kdd_train$service[which(kdd_train$service == "courier")] <- 16
kdd_train$service[which(kdd_train$service == "csnet_ns")] <- 17
kdd_train$service[which(kdd_train$service == "ctf")] <- 18
kdd_train$service[which(kdd_train$service == "daytime")] <- 19
kdd_train$service[which(kdd_train$service == "discard")] <- 20
kdd_train$service[which(kdd_train$service == "domain")] <- 21
kdd_train$service[which(kdd_train$service == "domain_u")] <- 22
kdd_train$service[which(kdd_train$service == "echo")] <- 23
kdd_train$service[which(kdd_train$service == "eco_i")] <- 24
```

```
kdd_train$service[which(kdd_train$service == "ecr_i")] <- 25
kdd_train$service[which(kdd_train$service == "efs")] <- 26
kdd_train$service[which(kdd_train$service == "exec")] <- 27
kdd_train$service[which(kdd_train$service == "finger")] <- 28
kdd_train$service[which(kdd_train$service == "ftp")] <- 29
kdd_train$service[which(kdd_train$service == "ftp_data")] <- 30
kdd_train$service[which(kdd_train$service == "gopher")] <- 31
kdd_train$service[which(kdd_train$service == "harvest")] <- 32
kdd_train$service[which(kdd_train$service == "hostnames")] <- 33
kdd_train$service[which(kdd_train$service == "http")] <- 34
kdd_train$service[which(kdd_train$service == "http_2784")] <- 35
kdd_train$service[which(kdd_train$service == "http_443")] <- 36
kdd_train$service[which(kdd_train$service == "http_8001")] <- 37
kdd_train$service[which(kdd_train$service == "imap4")] <- 38
kdd_train$service[which(kdd_train$service == "IRC")] <- 39
kdd_train$service[which(kdd_train$service == "iso_tsap")] <- 40
kdd_train$service[which(kdd_train$service == "klogin")] <- 41
kdd_train$service[which(kdd_train$service == "kshell")] <- 42
kdd_train$service[which(kdd_train$service == "ldap")] <- 43
kdd_train$service[which(kdd_train$service == "link")] <- 44
kdd_train$service[which(kdd_train$service == "login")] <- 45
kdd_train$service[which(kdd_train$service == "mtp")] <- 46
kdd_train$service[which(kdd_train$service == "name")] <- 47
kdd_train$service[which(kdd_train$service == "netbios_dgm")] <- 48
kdd_train$service[which(kdd_train$service == "netbios_ns")] <- 49
kdd_train$service[which(kdd_train$service == "netbios_ssn")] <- 50
kdd_train$service[which(kdd_train$service == "netstat")] <- 51
kdd_train$service[which(kdd_train$service == "nnsp")] <- 52
kdd_train$service[which(kdd_train$service == "ntp_u")] <- 53
kdd_train$service[which(kdd_train$service == "other")] <- 54
kdd_train$service[which(kdd_train$service == "pm_dump")] <- 55
kdd_train$service[which(kdd_train$service == "pop_2")] <- 56
kdd_train$service[which(kdd_train$service == "pop_3")] <- 57
kdd_train$service[which(kdd_train$service == "printer")] <- 58
kdd_train$service[which(kdd_train$service == "private")] <- 59
kdd_train$service[which(kdd_train$service == "red_i")] <- 60
kdd_train$service[which(kdd_train$service == "remote_job")] <- 61
kdd_train$service[which(kdd_train$service == "rje")] <- 62
kdd_train$service[which(kdd_train$service == "shell")] <- 63
kdd_train$service[which(kdd_train$service == "smtp")] <- 64
kdd_train$service[which(kdd_train$service == "sql_net")] <- 65
kdd_train$service[which(kdd_train$service == "ssh")] <- 66
kdd_train$service[which(kdd_train$service == "sunrpc")] <- 67
kdd_train$service[which(kdd_train$service == "supdup")] <- 68
kdd_train$service[which(kdd_train$service == "systat")] <- 69
kdd_train$service[which(kdd_train$service == "telnet")] <- 70
kdd_train$service[which(kdd_train$service == "tftp_u")] <- 71
kdd_train$service[which(kdd_train$service == "tim_i")] <- 72
kdd_train$service[which(kdd_train$service == "time")] <- 73
kdd_train$service[which(kdd_train$service == "urh_i")] <- 74
kdd_train$service[which(kdd_train$service == "urp_i")] <- 75
kdd_train$service[which(kdd_train$service == "uucp")] <- 76
kdd_train$service[which(kdd_train$service == "uucp_path")] <- 77
kdd_train$service[which(kdd_train$service == "vmnet")] <- 78
kdd_train$service[which(kdd_train$service == "whois")] <- 79
kdd_train$service[which(kdd_train$service == "X11")] <- 80
kdd_train$service[which(kdd_train$service == "Z39_50")] <- 81

kdd_train$labels[which(kdd_train$labels == "normal")] <- 0
kdd_train$labels[which(kdd_train$labels == "neptune")] <- 1
kdd_train$labels[which(kdd_train$labels == "warezclient")] <- 1
kdd_train$labels[which(kdd_train$labels == "ipsweep")] <- 1
```

```
kdd_train$labels[which(kdd_train$labels == "portsweep")] <- 1
kdd_train$labels[which(kdd_train$labels == "teardrop")] <- 1
kdd_train$labels[which(kdd_train$labels == "nmap")] <- 1
kdd_train$labels[which(kdd_train$labels == "satan")]<- 1
kdd_train$labels[which(kdd_train$labels == "smurf")]<-1
kdd_train$labels[which(kdd_train$labels == "pod")]<-1
kdd_train$labels[which(kdd_train$labels == "back")]<-1
kdd_train$labels[which(kdd_train$labels == "guess_passwd")]<-1
kdd_train$labels[which(kdd_train$labels == "multihop")]<-1
kdd_train$labels[which(kdd_train$labels == "rootkit")]<-1
kdd_train$labels[which(kdd_train$labels == "buffer_overflow")]<-1
kdd_train$labels[which(kdd_train$labels == "imap")]<-1
kdd_train$labels[which(kdd_train$labels == "warezmaster")]<-1
kdd_train$labels[which(kdd_train$labels == "phf")]<-1
kdd_train$labels[which(kdd_train$labels == "land")]<-1
kdd_train$labels[which(kdd_train$labels == "loadmodule")]<-1
kdd_train$labels[which(kdd_train$labels == "spy")]<-1
kdd_train$labels[which(kdd_train$labels == "perl")]<-1
kdd_train$labels[which(kdd_train$labels == "ftp_write")]<-1

kdd_test$flag[which(kdd_test$flag == "OTH")] <- 2
kdd_test$flag[which(kdd_test$flag == "SF")] <- 3
kdd_test$flag[which(kdd_test$flag == "S0")] <- 4
kdd_test$flag[which(kdd_test$flag == "S1")] <- 5
kdd_test$flag[which(kdd_test$flag == "S2")] <- 6
kdd_test$flag[which(kdd_test$flag == "S3")] <- 7
kdd_test$flag[which(kdd_test$flag == "REJ")] <- 8
kdd_test$flag[which(kdd_test$flag == "RSTO")] <- 9
kdd_test$flag[which(kdd_test$flag == "RSTOS0")] <- 10
kdd_test$flag[which(kdd_test$flag == "RSTR")] <- 11
kdd_test$flag[which(kdd_test$flag == "SH")] <- 12

kdd_test$service[which(kdd_test$service == "aol")] <- 13
kdd_test$service[which(kdd_test$service == "auth")] <- 14
kdd_test$service[which(kdd_test$service == "bgp")] <- 15
kdd_test$service[which(kdd_test$service == "courier")] <- 16
kdd_test$service[which(kdd_test$service == "csnet_ns")] <- 17
kdd_test$service[which(kdd_test$service == "ctf")] <- 18
kdd_test$service[which(kdd_test$service == "daytime")] <- 19
kdd_test$service[which(kdd_test$service == "discard")] <- 20
kdd_test$service[which(kdd_test$service == "domain")] <- 21
kdd_test$service[which(kdd_test$service == "domain_u")] <- 22
kdd_test$service[which(kdd_test$service == "echo")] <- 23
kdd_test$service[which(kdd_test$service == "eco_i")] <- 24
kdd_test$service[which(kdd_test$service == "ecr_i")] <- 25
kdd_test$service[which(kdd_test$service == "efs")] <- 26
kdd_test$service[which(kdd_test$service == "exec")] <- 27
kdd_test$service[which(kdd_test$service == "finger")] <- 28
kdd_test$service[which(kdd_test$service == "ftp")] <- 29
kdd_test$service[which(kdd_test$service == "ftp_data")] <- 30
kdd_test$service[which(kdd_test$service == "gopher")] <- 31
kdd_test$service[which(kdd_test$service == "harvest")] <- 32
kdd_test$service[which(kdd_test$service == "hostnames")] <- 33
kdd_test$service[which(kdd_test$service == "http")] <- 34
kdd_test$service[which(kdd_test$service == "http_2784")] <- 35
kdd_test$service[which(kdd_test$service == "http_443")] <- 36
kdd_test$service[which(kdd_test$service == "http_8001")] <- 37
kdd_test$service[which(kdd_test$service == "imap4")] <- 38
kdd_test$service[which(kdd_test$service == "IRC")] <- 39
kdd_test$service[which(kdd_test$service == "iso_tsap")] <- 40
kdd_test$service[which(kdd_test$service == "klogin")] <- 41
kdd_test$service[which(kdd_test$service == "kshell")] <- 42
```

```
kdd_test$service[which(kdd_test$service == "ldap")] <- 43
kdd_test$service[which(kdd_test$service == "link")] <- 44
kdd_test$service[which(kdd_test$service == "login")] <- 45
kdd_test$service[which(kdd_test$service == "mtp")] <- 46
kdd_test$service[which(kdd_test$service == "name")] <- 47
kdd_test$service[which(kdd_test$service == "netbios_dgm")] <- 48
kdd_test$service[which(kdd_test$service == "netbios_ns")] <- 49
kdd_test$service[which(kdd_test$service == "netbios_ssn")] <- 50
kdd_test$service[which(kdd_test$service == "netstat")] <- 51
kdd_test$service[which(kdd_test$service == "nnsp")] <- 52
kdd_test$service[which(kdd_test$service == "ntp_u")] <- 53
kdd_test$service[which(kdd_test$service == "other")] <- 54
kdd_test$service[which(kdd_test$service == "pm_dump")] <- 55
kdd_test$service[which(kdd_test$service == "pop_2")] <- 56
kdd_test$service[which(kdd_test$service == "pop_3")] <- 57
kdd_test$service[which(kdd_test$service == "printer")] <- 58
kdd_test$service[which(kdd_test$service == "private")] <- 59
kdd_test$service[which(kdd_test$service == "red_i")] <- 60
kdd_test$service[which(kdd_test$service == "remote_job")] <- 61
kdd_test$service[which(kdd_test$service == "rje")] <- 62
kdd_test$service[which(kdd_test$service == "shell")] <- 63
kdd_test$service[which(kdd_test$service == "smtp")] <- 64
kdd_test$service[which(kdd_test$service == "sql_net")] <- 65
kdd_test$service[which(kdd_test$service == "ssh")] <- 66
kdd_test$service[which(kdd_test$service == "sunrpc")] <- 67
kdd_test$service[which(kdd_test$service == "supdup")] <- 68
kdd_test$service[which(kdd_test$service == "systat")] <- 69
kdd_test$service[which(kdd_test$service == "telnet")] <- 70
kdd_test$service[which(kdd_test$service == "tftp_u")] <- 71
kdd_test$service[which(kdd_test$service == "tim_i")] <- 72
kdd_test$service[which(kdd_test$service == "time")] <- 73
kdd_test$service[which(kdd_test$service == "urh_i")] <- 74
kdd_test$service[which(kdd_test$service == "urp_i")] <- 75
kdd_test$service[which(kdd_test$service == "uucp")] <- 76
kdd_test$service[which(kdd_test$service == "uucp_path")] <- 77
kdd_test$service[which(kdd_test$service == "vmnet")] <- 78
kdd_test$service[which(kdd_test$service == "whois")] <- 79
kdd_test$service[which(kdd_test$service == "X11")] <- 80
kdd_test$service[which(kdd_test$service == "Z39_50")] <- 81

kdd_train$flag <- as.numeric(as.factor(kdd_train$flag))
kdd_train$service <- as.numeric(as.factor(kdd_train$flag))
kdd_train$labels <- as.numeric(as.factor(kdd_train$labels))

kdd_train$flag <- as.numeric(as.factor(kdd_train$flag))
kdd_test$service <- as.numeric(as.factor(kdd_test$service))
kdd_test$labels <- as.numeric(as.factor(kdd_test$labels))

#Select features based on correlation is done here to avoid
#multiple data transformation operations
kdd_train_new <- kdd_train[c(-3,-4)]
kdd_test_new <- kdd_test[c(-3,-4)]

#--------------Data Transformation : end------------------------

#************************************************************
#---------------DATA MINIG TECHNIQUES ----------------------
#************************************************************

#--------------Naive Bayes : Start------------------------
#NB model with features selected using IG method
set.seed(10)
```

```
split <- 0.60
kdd_train[ ] <- lapply(kdd_train,as.factor)
x_train = kdd_train[-14]
y_train = as.factor(kdd_train$labels)
x_test = kdd_test[-14]
y_test = as.factor(kdd_test$labels)
# Train the model with Naive Bayes and cross validate.


nb_model = train(x_train, y_train, method = 'nb',
                 trControl = trainControl(method = 'cv', number =
    3))

# Compute
preds = predict(nb_model$finalModel, x_test)$class
tbl = table(y_test, at = preds)
sum(diag(tbl)) / sum(tbl)


preds = predict(nb_model, newdata = x_test)
confusionMatrix(preds, y_test)

#NB model with features selected using IG method and Correlation
set.seed(10)
split <- 0.60
kdd_train_new[ ] <- lapply(kdd_train_new,as.factor)
x_train = kdd_train_new[-14]
y_train = as.factor(kdd_train_new$labels)
x_test = kdd_test_new[-14]
y_test = as.factor(kdd_test_new$labels)
# Train the model with Naive Bayes and cross validate.


nb_model = train(x_train, y_train, method = 'nb',
                 trControl = trainControl(method = 'cv', number =
    3))

# Compute
preds = predict(nb_model$finalModel, x_test)$class
tbl = table(y_test, at = preds)
sum(diag(tbl)) / sum(tbl)


preds = predict(nb_model, newdata = x_test)
confusionMatrix(preds, y_test)

#--------------Naive Bayes : End-------------------------

#--------------KNN : Start-------------------------------
#---NB model with features selected using IG method----

#normalise train data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
kdd_train_n <- as.data.frame(lapply(kdd_train[1:11], normalize))

#normalize test data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
kdd_test_n <- as.data.frame(lapply(kdd_test[1:13], normalize))

kdd_train_labels <- kdd_train_n$labels
```

```
kdd_test_labels <- kdd_test_n$labels

y_train = as.factor(kdd_train$labels)
y_test = as.factor(kdd_test$labels)

#Calculate the square root of observation. Here, for 125,973
    observations in the training data set, the K is 354.92.Rounding
     it to 355.
knn_test_pred_355 <- knn(train = kdd_train_n, test = kdd_test_n, cl
    = y_train, k = 355)

table(y_test,knn_test_pred_355)
plot(knn_test_pred_355)

#Computing Accuracy
confusionMatrix(knn_test_pred_355 ,y_test)

#*********************************************
#---NB model with features selected using IG method and Correlation
    ----
#normalise train data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
kdd_train_n <- as.data.frame(lapply(kdd_train_new[1:11], normalize)
    )

#normalize test data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
kdd_test_n <- as.data.frame(lapply(kdd_test_new[1:13], normalize))

kdd_train_new_labels <- kdd_train_n$labels
kdd_test_new_labels <- kdd_test_n$labels

y_train = as.factor(kdd_train_new$labels)
y_test = as.factor(kdd_test_new$labels)

#Calculate the square root of observation. Here, for 125,973
    observations in the training data set, the K is 354.92.Rounding
     it to 355.
knn_test_pred_355 <- knn(train = kdd_train_new_n, test = kdd_test_n
    , cl= y_train, k = 355)

table(y_test,knn_test_pred_355)
plot(knn_test_pred_355)

#Computing Accuracy
confusionMatrix(knn_test_pred_355 ,y_test)
#--------------KNN : End---------------------------------

#--------------Random Forest : Start-------------------------
#---NB model with features selected using IG method----
model1 <- randomForest(labels ~.,data=kdd_train,importance = TRUE)
model1
kdd_test$pred <- predict(model1, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#---NB model with features selected using IG method and Correlation
    ----
model2 <- randomForest(labels ~.,data=kdd_train_new,importance =
```

```
    TRUE)
model2
kdd_test_new$pred <- predict(model1, newdata = kdd_test,type="class
    ")
kdd_test_new$pred <- as.factor(kdd_test_new$pred)
kdd_test_new$labels <- as.factor(kdd_test_new$labels)
confusionMatrix(kdd_test_new$pred, kdd_test_new$labels)

#---Find best ntree value-----------------

#RF with ntree = 300
model <- randomForest(labels ~.,data=kdd_train, ntree = 300,
    importance = TRUE)
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#RF with ntree = 400
model <- randomForest(labels ~.,data=kdd_train, ntree = 400,
    importance = TRUE)
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#RF with ntree = 500
model <- randomForest(labels ~.,data=kdd_train, ntree = 500,
    importance = TRUE)
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#RF with ntree = 600
model <- randomForest(labels ~.,data=kdd_train, ntree = 600,
    importance = TRUE)
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#RF with ntree = 700
model <- randomForest(labels ~.,data=kdd_train, ntree = 700,
    importance = TRUE)
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#RF with ntree = 800
model <- randomForest(labels ~.,data=kdd_train, ntree = 800,
    importance = TRUE)
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
```

```
485 confusionMatrix(kdd_test$pred, kdd_test$labels)
486
487 #---Find best value of mtry-----------------
488
489 #RF with ntree = 400 and mtry = 1
490 model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
        1, importance = TRUE)
491 model
492 kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
493 kdd_test$pred <- as.factor(kdd_test$pred)
494 kdd_test$labels <- as.factor(kdd_test$labels)
495 confusionMatrix(kdd_test$pred, kdd_test$labels)
496
497 #RF with ntree = 400 and mtry = 3
498 model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
        3, importance = TRUE)
499 model
500 kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
501 kdd_test$pred <- as.factor(kdd_test$pred)
502 kdd_test$labels <- as.factor(kdd_test$labels)
503 confusionMatrix(kdd_test$pred, kdd_test$labels)
504
505 #RF with ntree = 400 and mtry = 4
506 model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
        4, importance = TRUE)
507 model
508 kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
509 kdd_test$pred <- as.factor(kdd_test$pred)
510 kdd_test$labels <- as.factor(kdd_test$labels)
511 confusionMatrix(kdd_test$pred, kdd_test$labels)
512
513 #RF with ntree = 400 and mtry = 5
514 model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
        5, importance = TRUE)
515 model
516 kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
517 kdd_test$pred <- as.factor(kdd_test$pred)
518 kdd_test$labels <- as.factor(kdd_test$labels)
519 confusionMatrix(kdd_test$pred, kdd_test$labels)
520
521 #---Cross Validation-----------------
522
523 #k=10
524 model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
        3, importance = TRUE,trControl=trainControl(method="cv",number
       =10))
525 model
526 kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
527 kdd_test$pred <- as.factor(kdd_test$pred)
528 kdd_test$labels <- as.factor(kdd_test$labels)
529 confusionMatrix(kdd_test$pred, kdd_test$labels)
530
531 #k=7
532 model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
        3, importance = TRUE,trControl=trainControl(method="cv",number
       =7))
533 model
534 kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
535 kdd_test$pred <- as.factor(kdd_test$pred)
536 kdd_test$labels <- as.factor(kdd_test$labels)
537 confusionMatrix(kdd_test$pred, kdd_test$labels)
538
```

```
#k=5
model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
     3, importance = TRUE,trControl=trainControl(method="cv",number
    =5))
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#k=3
model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
     3, importance = TRUE,trControl=trainControl(method="cv",number
    =3))
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#k=1
model <- randomForest(labels ~.,data=kdd_train, ntree = 400, mtry =
     3, importance = TRUE,trControl=trainControl(method="cv",number
    =1))
model
kdd_test$pred <- predict(model, newdata = kdd_test,type="class")
kdd_test$pred <- as.factor(kdd_test$pred)
kdd_test$labels <- as.factor(kdd_test$labels)
confusionMatrix(kdd_test$pred, kdd_test$labels)

#--------------Random Forest : End-------------------------
```