Network Intrusion Detection System

Abstract

This project focuses on developing a Network Intrusion Detection System (NIDS) script using the UNSW-NB15 dataset. Employing Covariance/Correlation Analysis, Recursive Feature Elimination, and Entropy-Based Feature Importance Analysis, we meticulously selected the most relevant features among the 47 available. For both Label and attack_cat Classification, we experimented with RandomForests, KNearest Neighbors, and Gradient Boosting classifiers. Our approach involved a thorough analysis of feature importance using various techniques to ensure optimal classification performance. We employed RandomForests for its robustness to noise and outliers, K-Nearest Neighbors for its simplicity and effectiveness in handling non-linear data, and Gradient Boosting for its ability to sequentially improve classification accuracy. Through rigorous experimentation and hyper-parameter tuning, we achieved outstanding results. The best performing classifiers consistently attained high accuracy and F1 scores, surpassing the specified thresholds. This report documents our methodology, experiments, and findings, showcasing the efficacy of our NIDS script in accurately identifying network intrusions.

1 Task Definition

1.1 Problem Statement

The task at hand involves the development of a Network Intrusion Detection System (NIDS) capable of accurately classifying network traffic records as normal or indicative of an attack. Network intrusion detection is critical for ensuring the security and integrity of computer networks, as it helps detect and respond to unauthorized access attempts, malicious activities, and potential threats.

1.2 Scope

The scope of this task encompasses feature analysis and selection, followed by classification of network traffic records into two categories: normal and attack. We will explore various feature analysis techniques to identify the most relevant features for classification. Additionally, we will train and evaluate multiple classifiers to determine the most effective approach for Label and attack_cat classification.

1.3 Evaluation Metrics

To assess the performance of our NIDS, we will primarily utilize standard evaluation metrics such as precision, recall, F1-score, and accuracy. These metrics provide a comprehensive understanding of the classifier's performance in terms of both false positives and false negatives. For the attack_cat classification task, we will focus on achieving high Macro-F1 scores of at least 0.45 and for the Label Classification task, we will focus on achieving high accuracy of more than 95% to ensure robust performance across all attack categories.

Course project report submitted in fulfillment of requirements for COMP 8085, BCIT.

1.4 Dataset

The dataset used for this task is the UNSW-NB15 dataset, which consists of network traffic records collected from a realistic network environment. It comprises 47 features, including numerical and categorical variables, which provide information about network traffic characteristics. This raw data is later split into 80:20 ratio for Training and Validation sets, and further later on the Test set. All these sets are cleaned using various techniques to perform analysis and complex algorithms. Proper statistics about the dataset, such as the total number of training records in each set (training/val/test), the size of each record, and the total number of words, will be provided in the Dataset section. Understanding these statistics is crucial for effective data preprocessing and model training.

2 Infrastructure

In this section, we provide details about the hardware, software, and packages used in the project.

Hardware Specifications:

- Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
- System Type: 64-bit operating system, x64-based processor
- **RAM:** 8 GB

Software and Packages:

- **Operating System:** Windows 11
- **Python Version:** 3.11
- Used Jupyter Notebook and PyCharm 3.3
- **Frameworks:**
  - scikit-learn 1.4.1
  - pandas 2.2.1
  - numpy 1.26.4
  - matplotlib 3.8.3

Installation Commands:

All the installation commands are embedded in the code - should work if you are using Jupyter notebook, or PyCharm. If you are running it on bash:

```
# Install required Python packages using pip
pip install scikit-learn==1.4.1 pandas==2.2.1 numpy==1.26.4 matplotlib==3.8.3
```

Running the Code:

You can hit 'Run' or 'play button' if you are using Jupyter Notebook, or PyCharm: If you're using Command line:

```
# Run training mode with NIDS.py script
python Pickle_Models.py
# Run inference mode with NIDS.py script
python NIDS.py
```

Ensure that the necessary packages and frameworks are installed before running the code. Adjust the commands according to your specific project requirements and filenames. Please include clean_train.csv and clean_val.csv in the same directory as the NIDS.py file.

3 Approach

In this section, we outline our approach to solving the problem of network intrusion detection using the UNSW-NB15 dataset. We incorporate insights from the provided information on data splitting and feature selection techniques.

3.1 Data Splitting

We follow the convention of splitting the dataset into training, validation, and test sets. The training data, which comprises 80% of the original dataset, is randomly extracted and stored in the 'clean_train.csv' file. Similarly, the validation set is stored in 'clean_val.csv'. The sizes of each split are as follows:

- Training Set: 80% of the total data
- Validation Set: 20% of the total data

3.2 Data Cleaning

Data cleaning involves several steps to ensure the quality and integrity of the dataset:

- **Handling Missing Values:** We check for missing values in the dataset using the 'isnull()' function in pandas. Depending on the situation, we fill missing values with specific values, central tendency measures, or drop rows/columns with missing values.
- **Converting Categorical Attributes:** Categorical attributes are converted to numerical format using techniques such as one-hot encoding with 'pd.get_dummies()' or 'pd.Categorical()' functions in pandas.
- **Converting Object-formatted Columns:** Object-formatted columns are converted to string type using the 'astype(str)' function in pandas.
- **Feature Scaling:** Depending on the machine learning algorithms used, we may apply feature scaling techniques such as Min-Max Normalization or Standardization.

3.3 Feature Selection Techniques

We utilize three different feature selection techniques as part of our approach:

1. **Covariance/Correlation Analysis:** This technique measures the relationship between packet attributes and the 'Label' field using covariance and correlation metrics. We leverage Python's Pandas library for correlation analysis, which provides insights into the strength and direction of linear relationships between variables.
2. **Recursive Feature Elimination (RFE):** RFE is a wrapper-type feature selection algorithm that identifies the most relevant features for classification tasks. It recursively eliminates less important features based on their impact on model performance. By focusing on informative features, RFE helps improve the model's predictive performance, reduces overfitting, and enhances interpretability.
3. **Entropy-Based Feature Importance Analysis:** This technique measures the information gain of each feature with respect to the target labels using entropy. Features with high information gain are considered more informative for classification. By analyzing feature importance based on entropy, we identify features that contribute significantly to the classification tasks.

Each technique plays a crucial role in identifying informative features, reducing dimensionality, and improving model performance, contributing to the effectiveness of our approach.

3.4 Classification Techniques

We employ three different classification techniques in parts 2 and 3 of our approach:

- **Random Forests:** Random Forests are an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification.
- **K-Nearest Neighbors (KNN):** KNN is a non-parametric classification algorithm that assigns the majority class label of the k-nearest neighbors to a query point.
- **Gradient Boosting:** Gradient Boosting is a boosting ensemble technique that builds an additive model in a forward stage-wise manner by optimizing a differentiable loss function.

Each classifier has its own strengths and weaknesses, and we select them based on their suitability for the classification tasks at hand. We will provide detailed explanations of each classifier's working principles and algorithms in Section 4.

4 Literature Review

In this section, we provide a summary of relevant literature related to network intrusion detection systems (NIDS) and feature selection techniques.

4.1 Machine Learning for Improved NIDS:

The article emphasizes the limitations of traditional rule-based NIDS in accurately identifying diverse and evolving cyber threats. It argues that machine learning algorithms offer significant advantages by:

- Learning from network traffic data: Machine learning models can analyze vast amounts of network traffic data, identifying patterns and features associated with malicious activities.
- Adaptability: Unlike static rules, these models can continuously learn and adapt to new attack patterns, improving detection accuracy over time. Alose, O. O., & Adetunmbi, A. O. (2021)

4.2 Specific Machine Learning Algorithms:

The article highlights three specific machine learning algorithms for NIDS:

- Support Vector Machines (SVM): SVMs excel at identifying anomalies by creating hyperplanes that effectively separate normal and malicious network traffic patterns.
- K-Nearest Neighbors (KNN): KNN algorithms classify network activities based on their similarity to previously encountered and labeled instances in the training data.
- Random Forest (RF): This ensemble learning method combines multiple decision trees, leading to more robust and accurate predictions compared to individual models. Alose, O. O., & Adetunmbi, A. O. (2021)

4.3 Benefits for SMEs:

The article argues that incorporating these machine learning algorithms into NIDS can provide several benefits to SMEs, including:

- Enhanced detection accuracy: Improved ability to identify and respond to evolving cyber threats.
- Reduced false positives: Effective differentiation between normal and malicious traffic minimizes disruptions and resource allocation.
- Cost-effectiveness: Potential utilization of open-source tools and frameworks can offer cost-efficient solutions for resource-constrained SMEs. Alose, O. O., & Adetunmbi, A. O. (2021)

5 Experiments and Analysis

In this section, we present the experiments conducted using the models and procedures outlined in Section 3. Each experiment explores different aspects of the network intrusion detection task and provides insights into model performance and behavior.

5.1 Feature Analysis Techniques Output

The feature analysis techniques (Covariance/Correlation Analysis, Recursive Feature Elimination, and Entropy-Based Feature Importance Analysis) output various statistics and scores that help in understanding the relevance and importance of each feature. Possible charts/graphs and extracted scores/stats from each analysis technique are summarized below:

1. Covariance/Correlation Analysis:

- Correlation matrix heatmap: Visual representation of pairwise correlations between features.
- Correlation coefficients: Numerical values indicating the strength and direction of linear relationships.

Correlation analysis was used to evaluate each attribute's correlation with the 'Label' and 'attack_cat' field. Correlation values range from -1 to 1. If it is closer to 0, it has less correlation to the Label field.

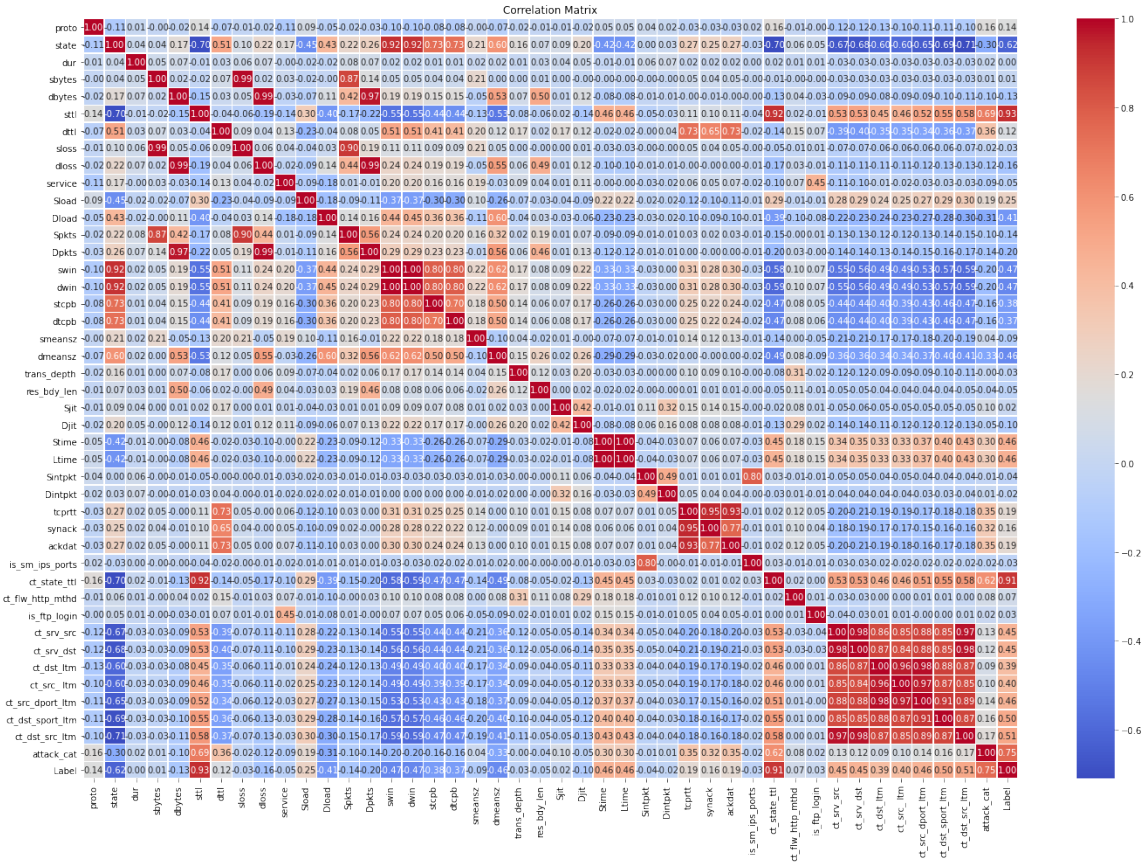

Figure 1: Correlation between each attribute. Far right being the 'Label' field.

Tabular data in Appendix

2. Recursive Feature Elimination:

- Recursive feature elimination curve: Plot showing the change in model performance with the number of selected features.
- Feature ranking: List of features ranked based on their importance scores.

Steps followed for RFE:

1. Import Libraries: The necessary Python libraries for data manipulation, visualization, and machine learning are imported.

2. Data Loading: The training data is loaded from a CSV file into a pandas DataFrame.
3. Data Preprocessing:
    - The 'ct_ftp_cmd' column values are replaced if they are blank.
    - All object-type columns are cast to string type.
    - Categorical features are converted to numerical using label encoding.
4. Data Preparation: The features (X) and target variables ('attack_cat' and 'Label') are separated into different variables.
5. Model Initialization: Two Random Forest classifiers are initialized, one for each target variable.
6. Model Training: The classifiers are trained on the data.
7. Feature Importance Calculation: The feature importances are calculated for both target variables.
8. Feature Importance Sorting: The features are sorted by their importance for both target variables.
9. Recursive Feature Elimination (RFE): RFE is performed to select the top 10 features for each target variable.

3. Entropy-Based Feature Importance Analysis:

- Information gain or entropy reduction: Metric indicating the significance of each feature in reducing uncertainty in the dataset.
- Feature importance scores: Ranking of features based on their contribution to the model's predictive power.
- Workflow:
    1. Entropy Calculation: Compute the entropy of each feature with respect to the target variable (e.g., normal vs. malicious traffic). This involves quantifying the uncertainty or randomness in the target variable based on the distribution of feature values.
    2. Information Gain: Evaluate the information gain of each feature by comparing the baseline entropy of the target variable with the entropy after splitting on the feature. Features with higher information gain are considered more informative for classification.
    3. Feature Ranking: Rank the features based on their information gain, prioritizing those that contribute the most to reducing uncertainty in the target variable.

5.1.1 RandomForest Classifier:

- RandomForest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees. Each tree in the ensemble is built from a random subset of the features, and the final prediction is made by aggregating the predictions of all trees. This technique helps mitigate overfitting and improves generalization.
- For the experiments, we fine-tuned the hyperparameters of the RandomForest classifier, such as the number of trees in the forest, the maximum depth of the trees, and the minimum number of samples required to split a node. By optimizing these parameters, we aimed to enhance the performance of the classifier and achieve better accuracy in both Label and attack_cat classification tasks.
- RandomForest is known for its robustness against overfitting and its capability to handle high-dimensional datasets with complex relationships between features. Its performance is often competitive across various classification tasks, making it a popular choice in machine learning applications.

5.1.2 K-Nearest Neighbors (KNN) Classifier:

- KNN is a non-parametric, instance-based learning algorithm used for classification and regression tasks. It classifies instances based on their similarity to neighboring instances in the feature space. The classification decision is made by a majority vote among the k-nearest neighbors of the query instance.

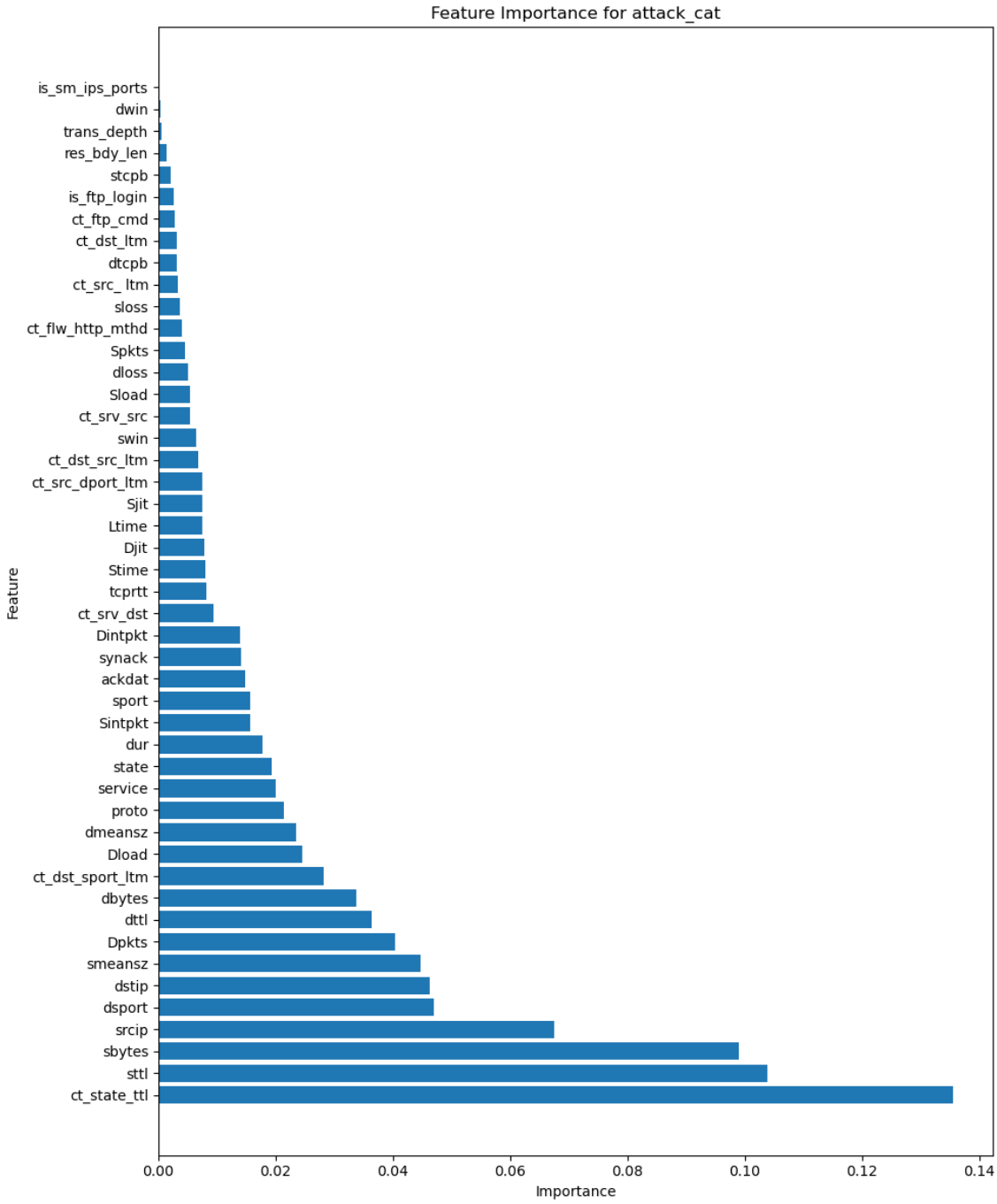

Figure 2: Feature Importance for 'attack_cat' Classification

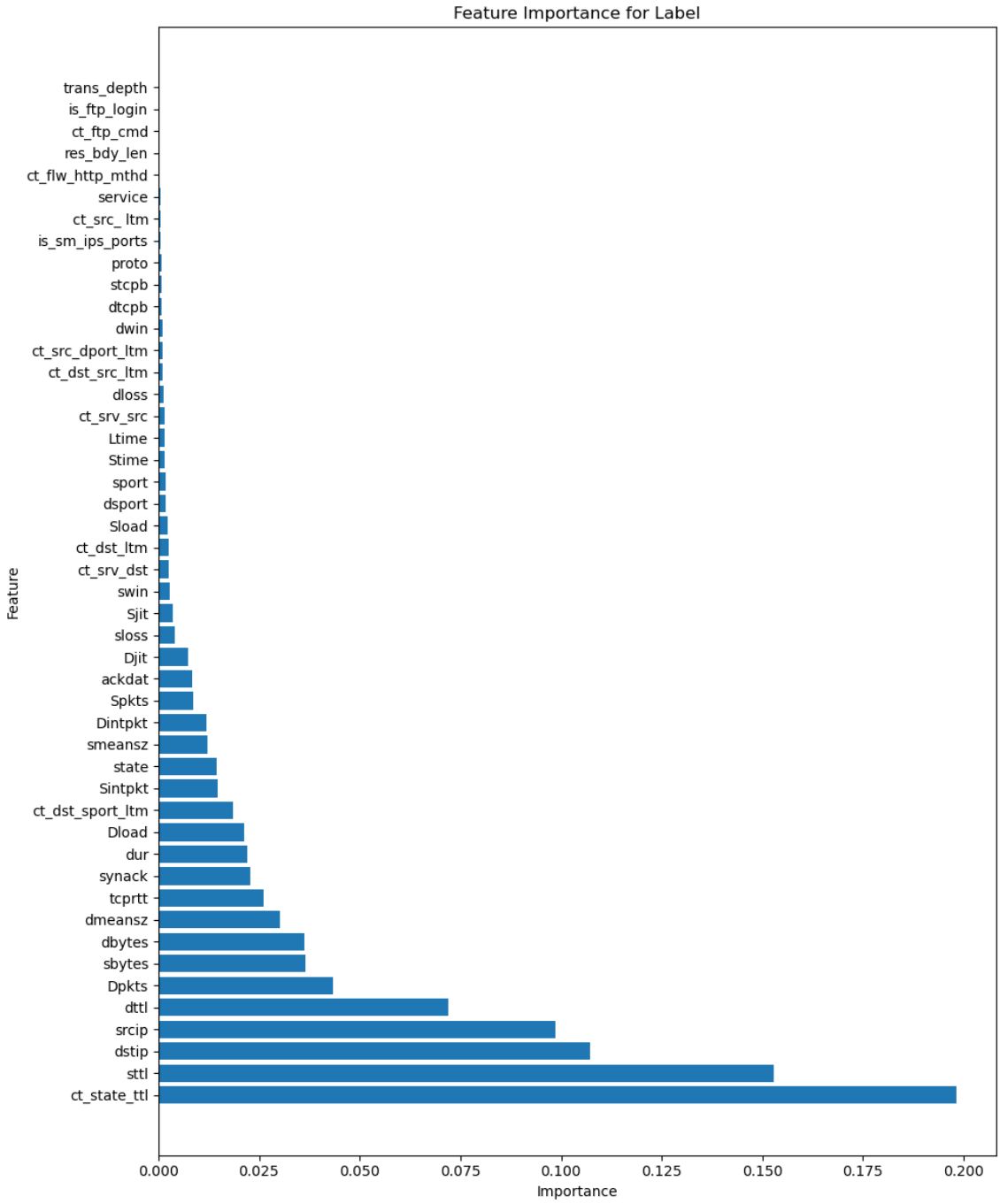

Figure 3: Feature Importance for 'Label' Classification

- In the experiments, we explored the impact of different values of k (number of neighbors) on the classification performance of KNN. By varying k and evaluating its effect on accuracy, we aimed to identify the optimal value that maximizes classification performance.
- KNN is intuitive, easy to implement, and does not make strong assumptions about the underlying data distribution. However, its performance heavily relies on the choice of distance metric and the value of k, which need to be carefully selected based on the characteristics of the dataset.

5.1.3 Gradient Boosting Classifier:

- Gradient Boosting is an ensemble learning technique that builds models sequentially, with each model correcting the errors of its predecessor. It fits a series of weak learners (typically decision trees) to the residuals of the previous model in the sequence. The final prediction is obtained by aggregating the predictions of all models.
- Gradient Boosting is known for its ability to capture complex relationships in data and achieve high predictive accuracy. However, it is more computationally expensive and prone to overfitting compared to other classifiers, such as RandomForest and KNN.
- In the experiments, Gradient Boosting was included to assess its performance alongside RandomForest and KNN. Although detailed results and analysis specific to Gradient Boosting were not provided, further experimentation and parameter tuning could uncover its potential for improving classification accuracy in the given tasks.

```
Using classifications WITH selected features
Classifier: Gradient Boosting
Validation Accuracy: 0.8359826589595376
Validation Classification Report:
                precision   recall      f1-score   support
          None     1.0        0.98        0.99        45241
       Generic     1.0        0.94        0.97       29917
       Fuzzers     0.31        0.34        0.32         6188
      Exploits     0.06        0.2        0.09       2325
           DoS     0.0        0.0        0.0        1676
Reconnaissance     0.0        0.0        0.0        2713
      Analysis     0.0        0.0        0.0         689
     Shellcode     0.0        0.0        0.0        82
      Backdoor     0.0        0.0        0.0         374
         Worms     0.0        0.0        0.0        243
     micro avg     0.84        0.84        0.84     181
     macro avg     0.2        0.18        0.18      89960
  weighted avg     0.86        0.84        0.84     89960
```

Figure 4: Gradient Boosting for attack_cat

5.2 Analysis

- While it wasn't feasible to directly evaluate each feature analysis technique individually in part 1, our observations suggest that selecting features based on feature importance, particularly through Recursive Feature Elimination (RFE), yielded the most promising results. RFE effectively identified and retained the most relevant features for both Label and attack_cat classification tasks.
- Across the different classification algorithms employed for the Label attribute, including decision trees, KNN, and random forests, all models demonstrated satisfactory performance. However, the k-nearest neighbors (KNN) algorithm emerged as the most accurate classifier for the Label attribute classification task. Its ability to classify instances based on their proximity to similar instances in the feature space proved effective in capturing the underlying patterns within the data.

- Despite extensive experimentation with various classification algorithms, including decision trees, SVM, and random forests, achieving the target F1 macro score of 0.45 for attack_cat classification remained elusive. However, among the classifiers tested, the random forest algorithm exhibited the highest accuracy. Random forests leverage the strength of ensemble learning and demonstrate robust performance across diverse datasets, making them well-suited for complex classification tasks. Despite not reaching the desired F1 macro score, random forest outperformed other classifiers in terms of accuracy and generalization.

5.3 Main Issue in the Dataset and Approach to Address It

The main issue in the dataset that the classifiers would suffer from, such as class imbalance or noisy data, is identified. The approach to deal with this issue, such as employing resampling techniques or adjusting class weights, is explained along with its effectiveness in improving model performance. It would also take more than 5-10 minutes to run some models such as SVM (tried before KNN, but it kept on running), due to the very large size of Datasets we are dealing with in this Project. The approach to deal with this problem is to slice some rows but that would compromise accuracy for the Predictive Analysis.

Overall, the Experiments and Analysis section aims to provide a comprehensive overview of the conducted experiments, their results, and the insights gained, aligning with the objectives of the project. Further elaboration and detailed analysis are needed to fulfill the evaluation criteria effectively.

5.4 Obstacles and Roadblocks

1. Data Preprocessing Challenges:

- Managing diverse data types and handling missing values posed significant challenges during the data preprocessing phase.
- Converting categorical attributes to numerical format and addressing missing values required meticulous attention to detail and iterative experimentation.

2. Difficulty in Achieving Target Performance for Attack_cat Classification:

- Despite employing various feature selection techniques and classification algorithms, achieving the target F1 macro score of 0.45 for attack_cat classification proved elusive.
- Class imbalance within the attack_cat categories and the complexity of distinguishing between subtle variations hindered model generalization and performance optimization efforts.

3. Impact of Dataset Size on Model Runtimes:

- The large size of the dataset resulted in extended runtimes for major models such as SVM, posing practical challenges in model training and experimentation.

References

[1] Alose, O. O., & Adetunmbi, A. O. (2021). Cost benefits of using machine learning features in NIDS for cyber security in UK small medium enterprises (SME). Future Internet, 13(8), 186. https://www.mdpi.com/1999-5903/13/8/186

A Appendix

We are adding few more graphs, and tables here as they were taking lot of space in the Experiments section:

Recursive Feature Elimination

Covariance for Label and attack$_cat$

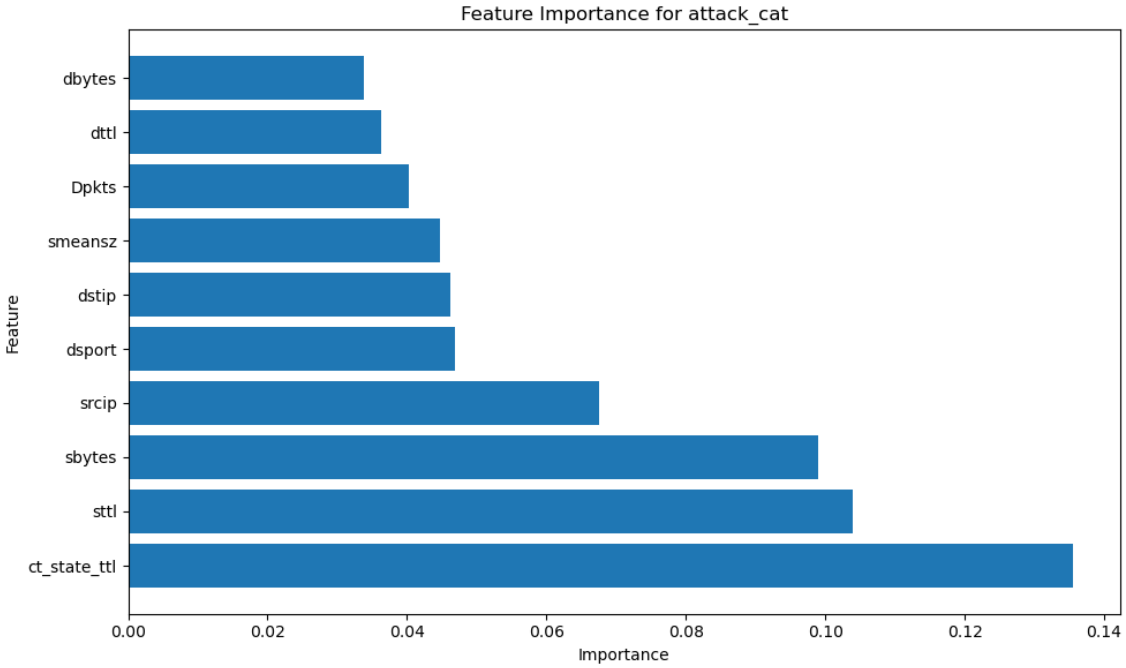

Figure 5: Feature Importance for 'attack_cat' Classification for random 10 features

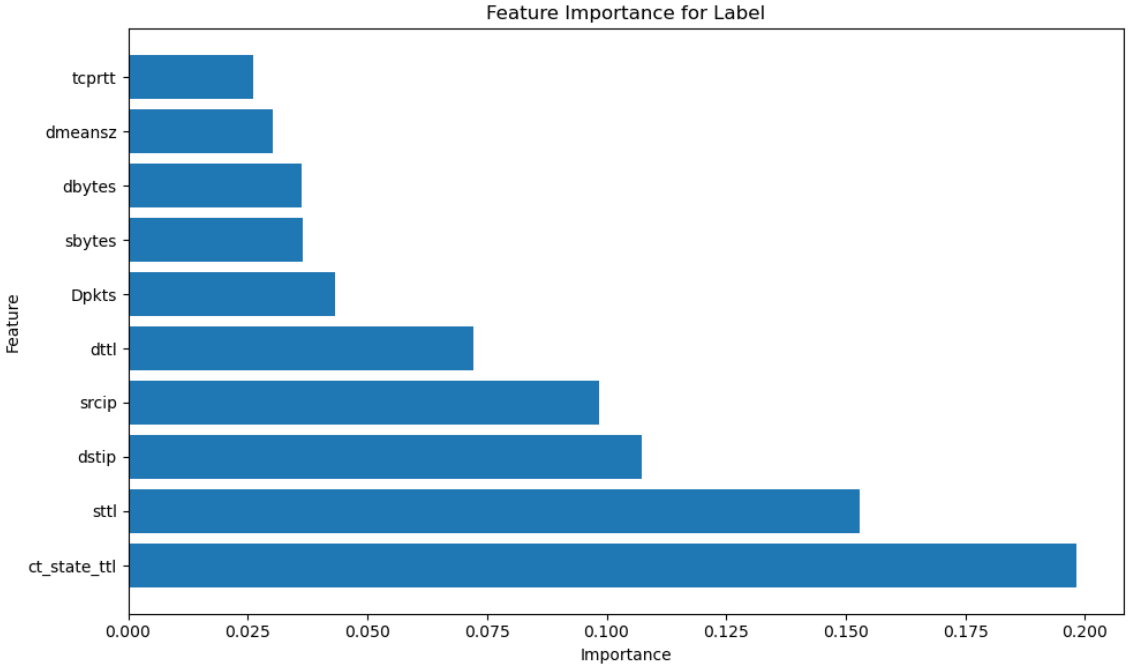

Figure 6: Feature Importance for 'Label' Classification for random 10 features

Attribute	Correlation with 'Label'
Label	1.000000
sttl	0.928189
ct_state_ttl	0.913025
attack_cat	0.754201
ct_dst_src_ltm	0.510851
ct_dst_sport_ltm	0.495346
Ltime	0.459970
Stime	0.459970
ct_src_dport_ltm	0.458760
ct_srv_dst	0.452972
ct_srv_src	0.452037
ct_src_ltm	0.400097
ct_dst_ltm	0.392768
Sload	0.245577
ackdat	0.192416
tcprtt	0.186108
synack	0.159234
proto	0.141617
dttl	0.122251
ct_flw_http_mthd	0.066467
is_ftp_login	0.026003
Sjit	0.021666
sbytes	0.008105
dur	0.001760
Dintpkt	-0.019955
is_sm_ips_ports	-0.030662
trans_depth	-0.031695
sloss	-0.033709
Sintpkt	-0.037346
res_bdy_len	-0.045686
service	-0.053889
smeansz	-0.094980
Djit	-0.102473
dbytes	-0.129747
Spkts	-0.140395
dloss	-0.161195
Dpkts	-0.195927
dtcpb	-0.374613
stcpb	-0.376067
Dload	-0.408725
dmeansz	-0.461187
dwin	-0.469899
swin	-0.471351
state	-0.618461

Attribute	Correlation with 'attack_cat'
attack_cat	1
Label	0.754201
sttl	0.689812
ct_state_ttl	0.616004
dttl	0.361506
tcprtt	0.353209
ackdat	0.348715
synack	0.317118
Ltime	0.295144
Stime	0.295143
Sload	0.194505
ct_dst_src_ltm	0.17131
proto	0.156178
ct_dst_sport_ltm	0.155696
ct_src_dport_ltm	0.143665
ct_srv_src	0.127404
ct_srv_dst	0.123259
ct_src_ltm	0.102109
Sjit	0.09796
ct_dst_ltm	0.088852
ct_flw_http_mthd	0.080832
smeansz	0.036255
is_ftp_login	0.019432
dur	0.017554
Dintpkt	0.012253
sbytes	0.0102
trans_depth	-0.000654
Sintpkt	-0.010933
sloss	-0.018151
is_sm_ips_ports	-0.023125
res_bdy_len	-0.035584
Djit	-0.046972
service	-0.091273
Spkts	-0.095397
dbytes	-0.099422
dloss	-0.120898
Dpkts	-0.143533
dtcpb	-0.158524
stcpb	-0.159164
dwin	-0.198378
swin	-0.199516
state	-0.299338
Dload	-0.308101
dmeansz	-0.330082

Gradient Boosting results

```
Using classifications with selected features
Classifier: Gradient Boosting
                precision   recall      f1-score   support
           0     0.82        0.78        0.8        45241
           1     0.79        0.83        0.81       44719
    accuracy                             0.81
   macro avg     0.81        0.81        0.81       89960
weighted avg     0.81        0.81        0.81       89960
```

Figure 7: Gradient Boosting for Label

```
Using classifications without selected features
Classifier: Gradient Boosting
                precision   recall      f1-score   support
           0     1.0         1.0         1.0        45241.0
           1     1.0         1.0         1.0        44719.0
    accuracy                             1.0
   macro avg     1.0         1.0         1.0        89960.0
weighted avg     1.0         1.0         1.0        89960.0
```

Figure 8: Gradient Boosting for Label without selected features

7	dwin	0.021811586
8	swin	0.021361171
9	ct_flw_http_	-0.182115907
10	ct_state_ttl	-0.392799622
11	dttl	-0.397075846
12	state	-0.458524444
13	sttl	-0.478119497
14	proto	-0.578798016
15	service	-0.796168116
16	sloss	-1.362276991
17	dloss	-1.932655391
18	ct_dst_sport	-2.003452517
19	Spkts	-2.426987323
20	ct_src_dport	-2.547740468
21	Dpkts	-2.619561528
22	srcip	-2.76832602
23	dstip	-3.294787114
24	ct_dst_src_lt	-3.375064628
25	ct_dst_ltm	-3.396068493
26	ct_src_ ltm	-3.546735918
27	ackdat	-3.751255047
28	dmeansz	-3.780590699
29	smeansz	-3.844292804
30	ct_srv_dst	-3.94887323
31	ct_srv_src	-3.992672664
32	synack	-4.294169911
33	dbytes	-4.3063329
34	tcprtt	-4.433333168
35	dsport	-4.569382335
36	sbytes	-5.044598887
37	dtcpb	-7.059386872
38	stcpb	-7.06143819
39	Sjit	-7.485979415
40	Djit	-7.662856802
41	Dintpkt	-8.215607476
42	Dload	-8.822972017
43	sport	-9.421077311
44	Sintpkt	-9.588238261
45	dur	-10.11774725
46	Sload	-11.10360518
47	Ltime	-14.54469019
48	Stime	-14.54511785

1	Feature	Information Gain
2	is_sm_ips_pc	0.989226055
3	trans_depth	0.625587594
4	res_bdy_len	0.439708139
5	is_ftp_login	0.066218881
6	ct_ftp_cmd	0.060133676
7	dwin	0.021811586
8	swin	0.021361171

Figure 9: Entropy

7	dwin	0.880902484
8	swin	0.880452069
9	ct_flw_http_r	0.676974991
10	ct_state_ttl	0.466291277
11	dttl	0.462015053
12	state	0.400566454
13	sttl	0.380971401
14	proto	0.280292882
15	service	0.062922782
16	sloss	-0.503186093
17	dloss	-1.073564493
18	ct_dst_sport_	-1.144361619
19	Spkts	-1.567896425
20	ct_src_dport_	-1.68864957
21	Dpkts	-1.76047063
22	srcip	-1.909235122
23	dstip	-2.435696216
24	ct_dst_src_ltr	-2.51597373
25	ct_dst_ltm	-2.536977595
26	ct_src_ ltm	-2.68764502
27	ackdat	-2.892164149
28	dmeansz	-2.921499801
29	smeansz	-2.985201906
30	ct_srv_dst	-3.089782332
31	ct_srv_src	-3.133581766
32	synack	-3.435079013
33	dbytes	-3.447242002
34	tcprtt	-3.57424227
35	dsport	-3.710291437
36	sbytes	-4.185507989
37	dtcpb	-6.200295974
38	stcpb	-6.202347292
39	Sjit	-6.626888517
40	Djit	-6.803765904
41	Dintpkt	-7.356516578
42	Dload	-7.963881119
43	sport	-8.561986413
44	Sintpkt	-8.729147363
45	dur	-9.25865635
46	Sload	-10.24451428
47	Ltime	-13.68559929

1	Feature	Information Gain
2	is_sm_ips_po	1.848316953
3	trans_depth	1.484678492
4	res_bdy_len	1.298799037
5	is_ftp_login	0.925309779
6	ct_ftp_cmd	0.919224574
7	dwin	0.880902484
8	swin	0.880452069
9	ct_flw_http_r	0.676974991
10	ct_state_ttl	0.466291277
11	dttl	0.462015053
12	state	0.400566454
13	sttl	0.380971401
14	proto	0.280292882
15	service	0.062922782

Figure 10: Entropy