UNIVERCITY OF TAMANRASSET

CYPER-ATTACK DETECTION USING UNSW-NB15

CREATED BY: HAFIDI MOUSSA & ZAKI ALAA EDDINE

SUMMARY :

the PDF likely provides insights into the application of machine learning techniques to predict cyber-attacks using the UNSW-NB15 dataset, along with discussions on challenges, evaluation metrics, and future research directions in this domain.

1-Introduction:

In today's interconnected digital landscape, cyber attacks pose significant threats to individuals, organizations, and nations alike. Detecting and mitigating these attacks is paramount to maintaining the integrity and security of digital systems. The UNSW-NB15 dataset, developed by the University of New South Wales (UNSW), serves as a valuable resource for researchers and practitioners in the field of cybersecurity. This dataset contains network traffic data captured in a controlled environment, encompassing various types of cyber attacks along with normal network activities.

The detection of cyber attacks using the UNSW-NB15 dataset involves leveraging advanced machine learning and data analysis techniques to identify patterns indicative of malicious behavior within network traffic. By analyzing features such as packet headers, payload content, and connection characteristics, it is possible to train models that can accurately classify network traffic as either benign or malicious.

The UNSW-NB15 dataset is a comprehensive collection of network traffic data specifically curated for cybersecurity research and analysis. Developed by the University of New South Wales (UNSW), this dataset offers a diverse range of network traffic scenarios, including both normal activities and various types of cyber attacks. It serves as a benchmark dataset for evaluating the effectiveness of intrusion detection systems (IDS) and machine learning algorithms in identifying and mitigating cyber threats.

2- The description UNSW-NB15 dataset :

1. Data Diversity: The dataset encompasses a wide range of network traffic scenarios, including normal traffic as well as instances of known and novel cyber attacks. This diversity enables researchers to train and test detection algorithms across different attack vectors and network conditions.

2. Attack Variety: Various types of cyber attacks are represented in the dataset, such as denial-of-service (DoS), distributed denial-of-service (DDoS), probing, infiltration, and exploitation attacks. Each attack category comprises multiple attack instances, providing a rich source of data for analysis.

3. Realistic Traffic Patterns : The network traffic data captured in the UNSW-NB15 dataset reflects realistic patterns observed in operational networks, ensuring that detection models trained on this data are applicable to real-world scenarios.

4. Annotated Labels : Each network flow in the dataset is labeled with ground truth annotations, indicating whether it corresponds to normal activity or a specific type of cyber attack. These labels facilitate supervised learning approaches for training intrusion detection models.

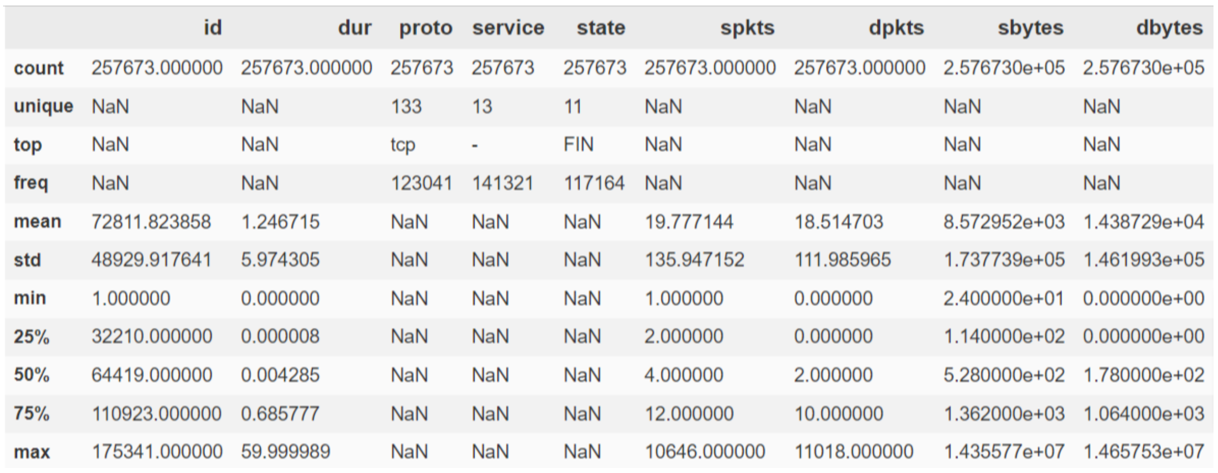
5. Rich Feature Set : The dataset includes a wide array of features extracted from network traffic data, including packet headers, flow statistics, payload content, and protocol information. These features provide valuable insights for detecting anomalous or malicious behavior within network traffic.

6. Scalability and Size : With over 2.5 million instances of network flows, the UNSW-NB15 dataset offers a large-scale dataset suitable for training and evaluating robust intrusion detection systems capable of handling high-volume network traffic.

Overall, the UNSW-NB15 dataset serves as a valuable resource for cybersecurity researchers, enabling them to develop and benchmark advanced intrusion detection techniques against a diverse range of cyber threats. Its realistic and comprehensive nature makes it an indispensable tool for enhancing the resilience of digital systems against evolving cyber attacks.

3- Statistical analysis results with discussion

1- Describe The data using df.describe():



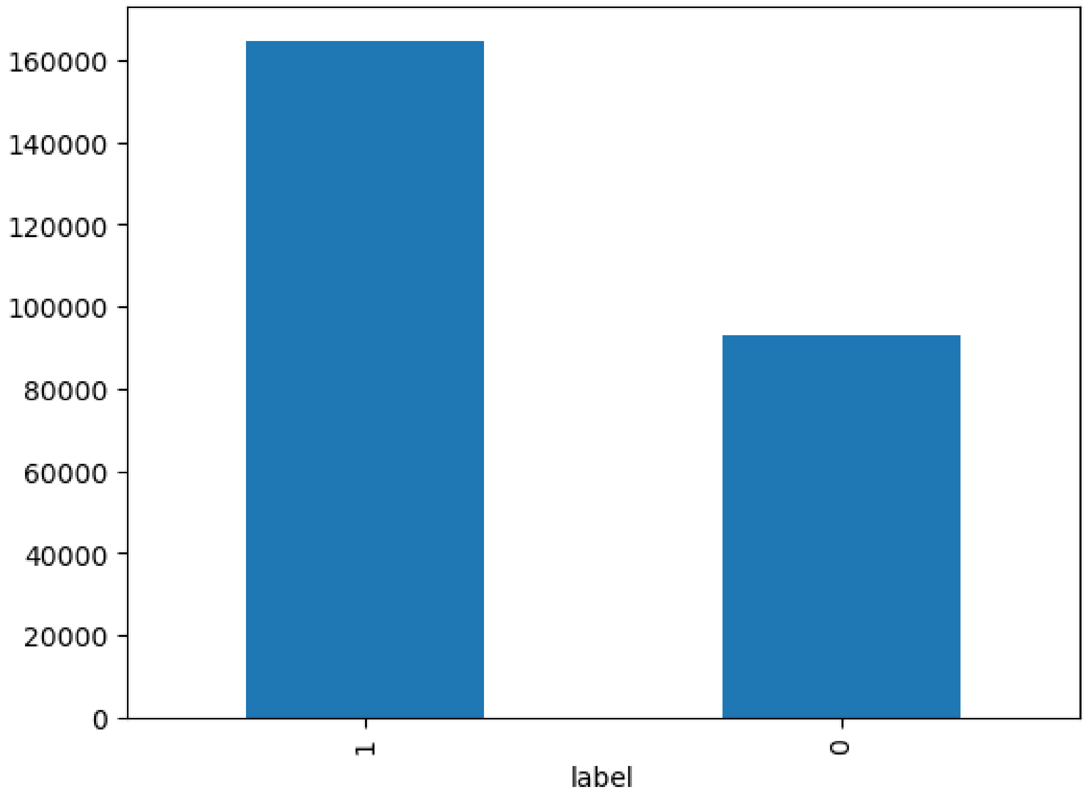
	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes
count	257673.000000	257673.000000	257673	257673	257673	257673.000000	257673.000000	2.576730e+05	2.576730e+05
unique	NaN	NaN	133	13	11	NaN	NaN	NaN	NaN
top	NaN	NaN	tcp	-	FIN	NaN	NaN	NaN	NaN
freq	NaN	NaN	123041	141321	117164	NaN	NaN	NaN	NaN
mean	72811.823858	1.246715	NaN	NaN	NaN	19.777144	18.514703	8.572952e+03	1.438729e+04
std	48929.917641	5.974305	NaN	NaN	NaN	135.947152	111.985965	1.737739e+05	1.461993e+05
min	1.000000	0.000000	NaN	NaN	NaN	1.000000	0.000000	2.400000e+01	0.000000e+00
25%	32210.000000	0.000008	NaN	NaN	NaN	2.000000	0.000000	1.140000e+02	0.000000e+00
50%	64419.000000	0.004285	NaN	NaN	NaN	4.000000	2.000000	5.280000e+02	1.780000e+02
75%	110923.000000	0.685777	NaN	NaN	NaN	12.000000	10.000000	1.362000e+03	1.064000e+03
max	175341.000000	59.999989	NaN	NaN	NaN	10646.000000	11018.000000	1.435577e+07	1.465753e+07

2- Drop The duplicates

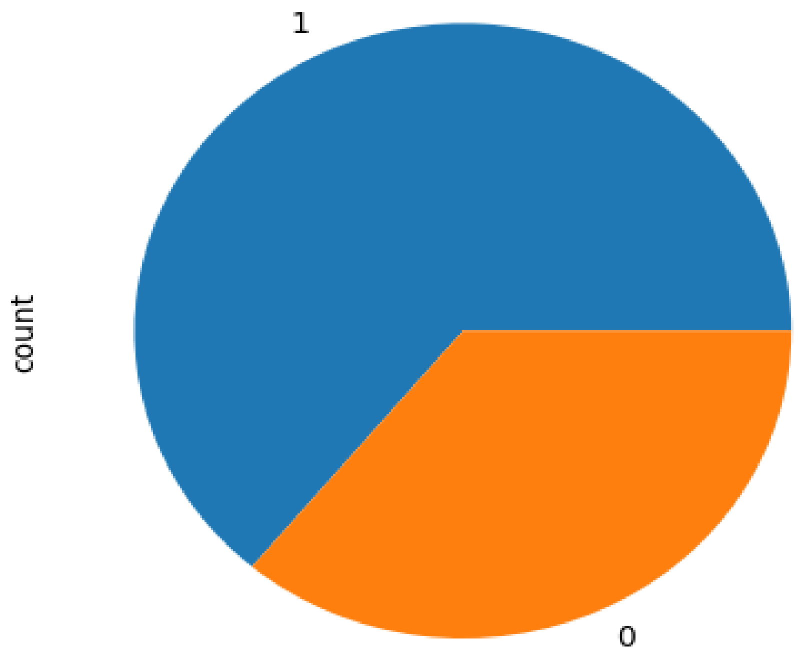
3- Fill the missing values :
   a- Filling the categorical features with the mode of the dataset
   b- Filling the numerical features with the median of the dataset

4- Check the balance of our dataset using the ‘label’ feature which is a binary feature

-Bar plot :



-Pie Plot

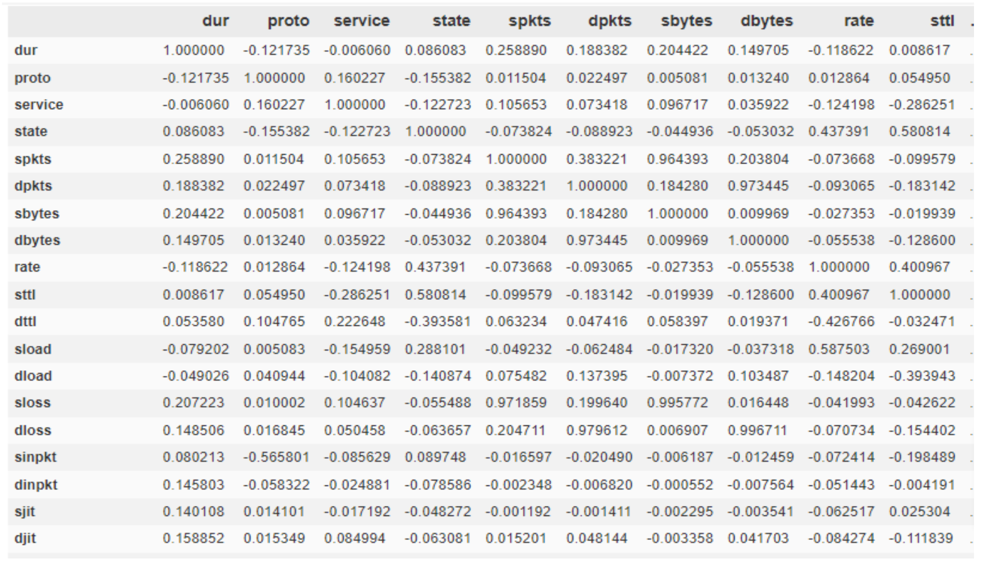


The Data set is quite not balanced but we can use it to fit our model

5- Encoding categorical features using LabelEncoder

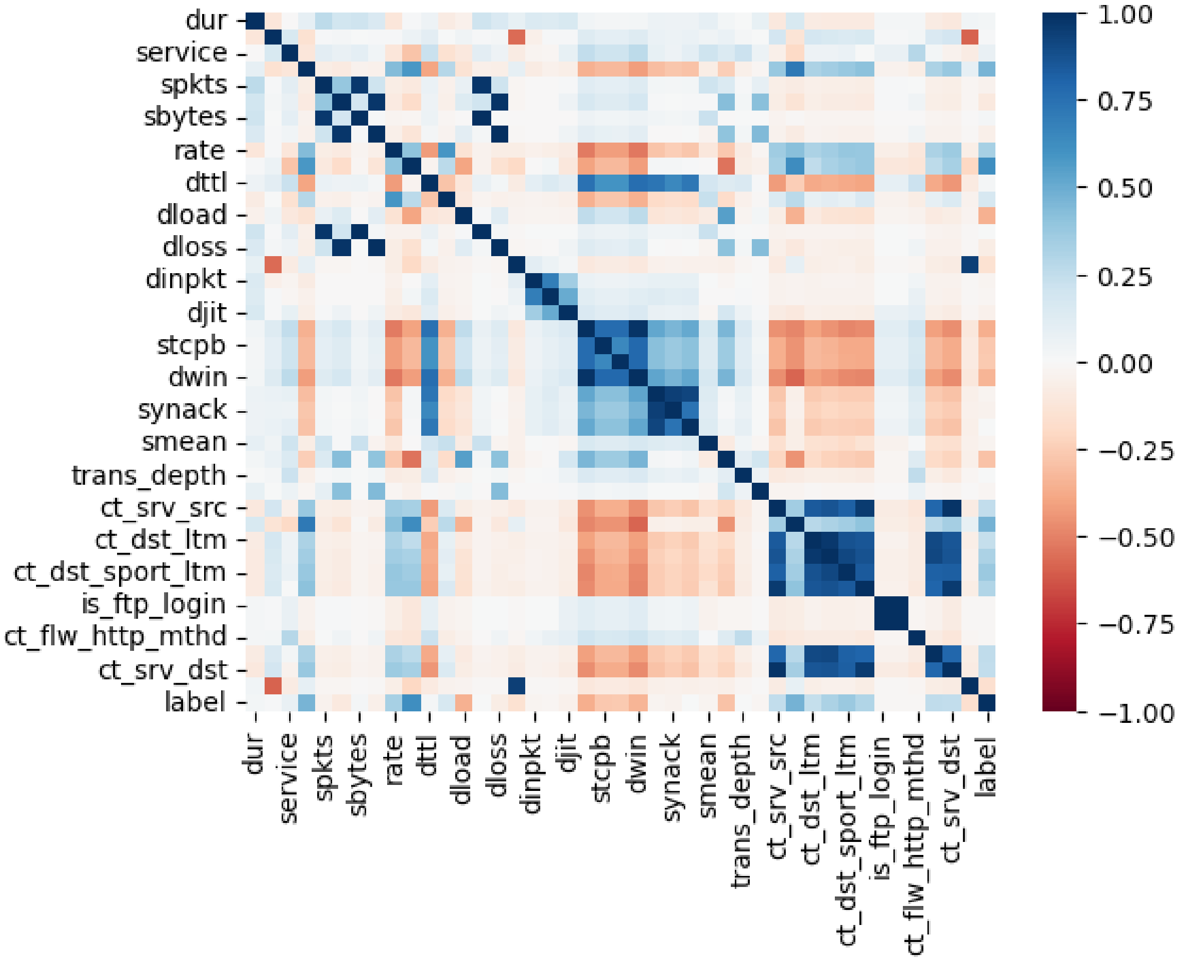
6- Dropping unnecessary features [‘id’ , ‘attack_cat’] because we are using a binary classification ‘0’ for the normal and ‘1’ for the attack

7- Dropping The features that have a high correlation :
   1- Calculating the correlation matrix of the dataset
   2- Drop the feature that have a correlation greater than’ 0.80’



	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl
dur	1.000000	-0.121735	-0.006060	0.086083	0.258890	0.188382	0.204422	0.149705	-0.118622	0.008617
proto	-0.121735	1.000000	0.160227	-0.155382	0.011504	0.022497	0.005081	0.013240	0.012864	0.054950
service	-0.006060	0.160227	1.000000	-0.122723	0.105653	0.073418	0.096717	0.035922	-0.124198	-0.286251
state	0.086083	-0.155382	-0.122723	1.000000	-0.073824	-0.088923	-0.044936	-0.053032	0.437391	0.580814
spkts	0.258890	0.011504	0.105653	-0.073824	1.000000	0.383221	0.964393	0.203804	-0.073668	-0.099579
dpkts	0.188382	0.022497	0.073418	-0.088923	0.383221	1.000000	0.184280	0.973445	-0.093065	-0.183142
sbytes	0.204422	0.005081	0.096717	-0.044936	0.964393	0.184280	1.000000	0.009969	-0.027353	-0.019939
dbytes	0.149705	0.013240	0.035922	-0.053032	0.203804	0.973445	0.009969	1.000000	-0.055538	-0.128600
rate	-0.118622	0.012864	-0.124198	0.437391	-0.073668	-0.093065	-0.027353	-0.055538	1.000000	0.400967
sttl	0.008617	0.054950	-0.286251	0.580814	-0.099579	-0.183142	-0.019939	-0.128600	0.400967	1.000000
dttl	0.053580	0.104765	0.222648	-0.393581	0.063234	0.047416	0.058397	0.019371	-0.426766	-0.032471
sload	-0.079202	0.005083	-0.154959	0.288101	-0.049232	-0.062484	-0.017320	-0.037318	0.587503	0.269001
dload	-0.049026	0.040944	-0.104082	-0.140874	0.075482	0.137395	-0.007372	0.103487	-0.148204	-0.393943
sloss	0.207223	0.010002	0.104637	-0.055488	0.971859	0.199640	0.995772	0.016448	-0.041993	-0.042622
dloss	0.148506	0.016845	0.050458	-0.063657	0.204711	0.979612	0.006907	0.996711	-0.070734	-0.154402
sinpkt	0.080213	-0.565801	-0.085629	0.089748	-0.016597	-0.020490	-0.006187	-0.012459	-0.072414	-0.198489
dinpkt	0.145803	-0.058322	-0.024881	-0.078586	-0.002348	-0.006820	-0.000552	-0.007564	-0.051443	-0.004191
sjit	0.140108	0.014101	-0.017192	-0.048272	-0.001192	-0.001411	-0.002295	-0.003541	-0.062517	0.025304
djit	0.158852	0.015349	0.084994	-0.063081	0.015201	0.048144	-0.003358	0.041703	-0.084274	-0.111839

-The correlation heatmap :



```
spkts  &  sbytes  =  0.964393217155871
spkts  &  sloss  =  0.9718594116510776
dpkts  &  dbytes  =  0.9734453279792796
dpkts  &  dloss  =  0.979612084935938
sbytes  &  sloss  =  0.9957715772406667
dbytes  &  dloss  =  0.996711133831455
sinpkt  &  is_sm_ips_ports  =  0.9421206232319264
swin  &  dwin  =  0.9804584289136496
tcprtt  &  synack  =  0.9430527539834922
tcprtt  &  ackdat  =  0.9202175911910121
ct_srv_src  &  ct_dst_ltm  =  0.8409951124111326
ct_srv_src  &  ct_src_dport_ltm  =  0.8618545947587102
ct_srv_src  &  ct_dst_sport_ltm  =  0.8148560554331661
ct_srv_src  &  ct_dst_src_ltm  =  0.9539519771685965
ct_srv_src  &  ct_srv_dst  =  0.9794668130725528
ct_dst_ltm  &  ct_src_dport_ltm  =  0.9615176986568883
ct_dst_ltm  &  ct_dst_sport_ltm  =  0.871117809397576
ct_dst_ltm  &  ct_dst_src_ltm  =  0.8574940265843254
ct_dst_ltm  &  ct_src_ltm  =  0.9015817705704372
ct_dst_ltm  &  ct_srv_dst  =  0.8530854336257614
ct_src_dport_ltm  &  ct_dst_sport_ltm  =  0.9083371617418114
ct_src_dport_ltm  &  ct_dst_src_ltm  =  0.871846565750148
ct_src_dport_ltm  &  ct_src_ltm  =  0.9094302805367733
ct_src_dport_ltm  &  ct_srv_dst  =  0.8667171818381377
ct_dst_sport_ltm  &  ct_dst_src_ltm  =  0.8364220970632356
ct_dst_sport_ltm  &  ct_src_ltm  =  0.8177468268817962
ct_dst_sport_ltm  &  ct_srv_dst  =  0.8228732004059827
ct_dst_src_ltm  &  ct_src_ltm  =  0.8025514215953938
ct_dst_src_ltm  &  ct_srv_dst  =  0.9603212131064117
is_ftp_login  &  ct_ftp_cmd  =  0.9988554882935726
```

The Dropped features :

```
{ spkts ,spkts ,dpkts, dpkts ,sbytes ,dbytes ,sinpkt ,swin ,tcprtt ,tcprtt ,ct_srv_src ,
ct_srv_src ,ct_srv_src ,ct_srv_src ,ct_srv_src ,ct_dst_ltm ,ct_dst_ltm ,ct_dst_ltm ,
ct_dst_ltm ,ct_dst_ltm ,ct_src_dport_ltm ,ct_src_dport_ltm ,ct_src_dport_ltm ,
ct_src_dport_ltm ,ct_dst_sport_ltm ,ct_dst_sport_ltm ,ct_dst_sport_ltm  ,ct_dst_src_ltm
,ct_dst_src_ltm  ,is_ftp_login }
```

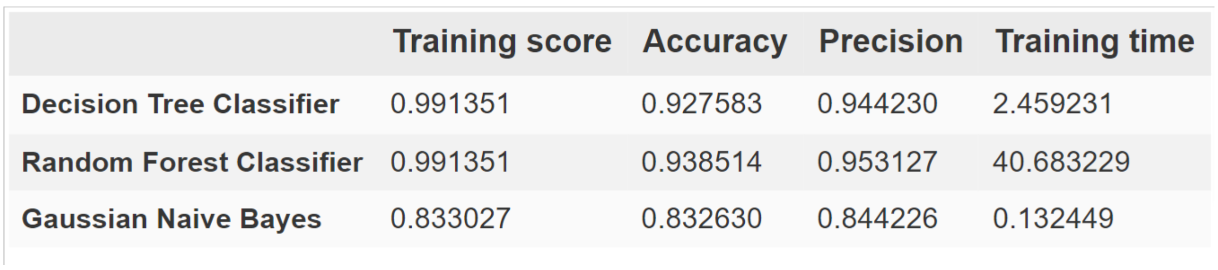
8- Splitting the training and the testing data using train_test_split() function

9- Scaling all features using StandardScaler.

4- Model used :

- Before we choose the best model for fitting our dataset we predict the result using three classification models 'Decision Tree Classifier' , 'Random Forest Classifier', 'Gaussian Naive Bayes' than we display the scores of each model

- Models comparison :



	Training score	Accuracy	Precision	Training time
Decision Tree Classifier	0.991351	0.927583	0.944230	2.459231
Random Forest Classifier	0.991351	0.938514	0.953127	40.683229
Gaussian Naive Bayes	0.833027	0.832630	0.844226	0.132449

- 1. Training Score : Both Decision Tree and Random Forest classifiers achieve high training scores, indicating good performance on the training data. Gaussian Naive Bayes has a slightly lower training score but still performs reasonably well.

- 2. Accuracy and Precision : Random Forest Classifier achieves the highest accuracy and precision among the models, indicating its effectiveness in correctly classifying instances and minimizing false positives.

- 3. Training Time : Gaussian Naive Bayes has the lowest training time, while Random Forest Classifier has the highest. Decision Tree Classifier falls in between in terms of training time.

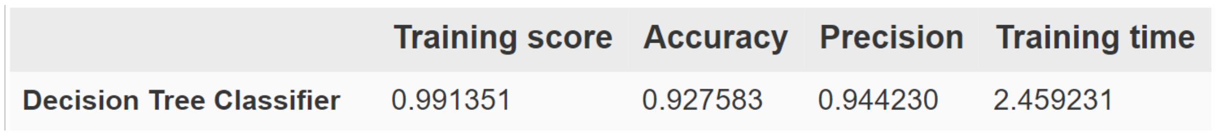
-Choosing the best model:

According to Computational Efficiency: Decision tree training is typically faster compared to ensemble methods like Random Forest, especially when dealing with large datasets. This makes decision trees more suitable for situations where computational resources are limited or training time needs to be minimized

According to Accuracy and Precision :the Random Forest Classifier appears to be the best model in this scenario. It achieves the highest accuracy and precision while maintaining a relatively high training score, indicating robust performance on both the training and potentially unseen data.

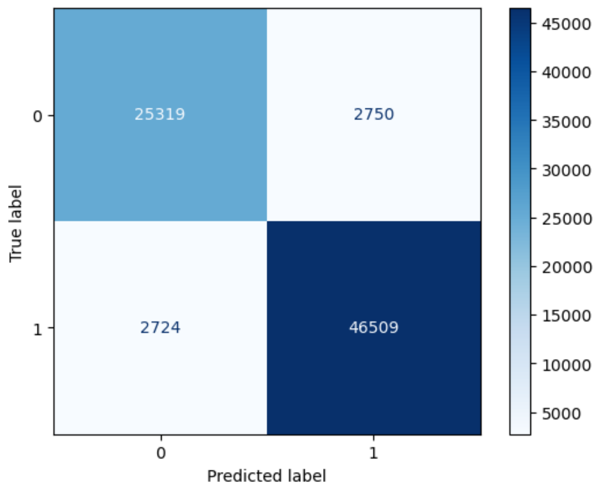
- According to the both criterion 'Accuracy and Precision' & 'Computational Efficiency' The best model is 'Decision tree classifier' because the Decision tree and Random forest are so close in the Accuracy and Precision but they are significantly different in the training time the 'Decsion tree classifier' is way better than the Rendom forest .

5-Model Result :

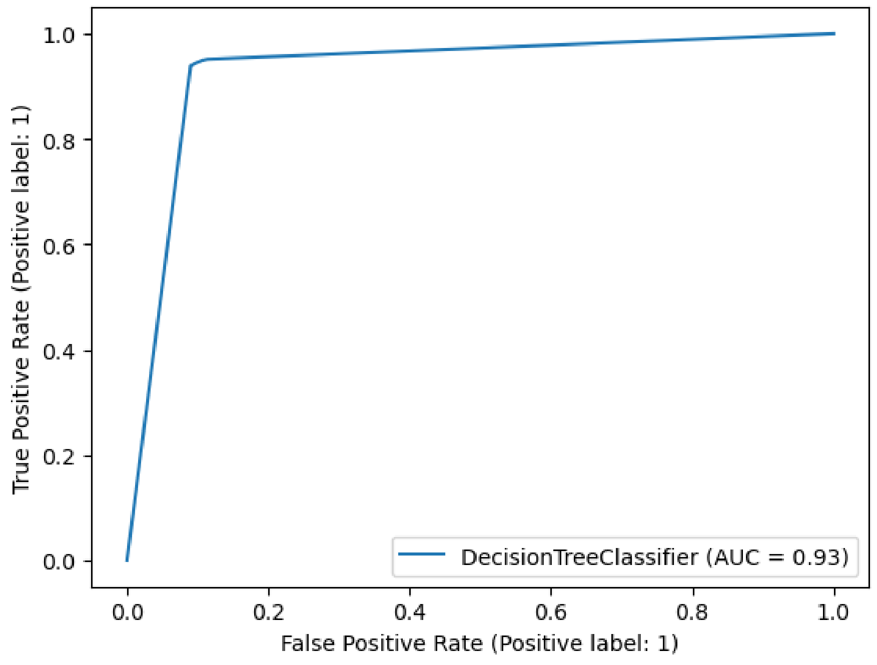


	Training score	Accuracy	Precision	Training time
Decision Tree Classifier	0.991351	0.927583	0.944230	2.459231

a- Confusion matrix :



b- Roc Curve :



6- Conclusion on the Decision Tree Model

- Overview

The Decision Tree Classifier in the UNSW_NB15 dataset achieved a high training score of 0.991351, indicating that the model fits the training data very well. The test accuracy is also relatively high at 0.927583, and the precision score is 0.944230. The training time for this model was 2.459231 seconds.

A - <u>Limitations</u>

1. Overfitting : The significant difference between the training score (0.991351) and the test accuracy (0.927583) suggests that the model might be overfitting the training data. This means that while the model performs exceptionally well on the training data, it might not generalize as well to unseen data.

2. Bias-Variance Tradeoff: Decision Trees can be prone to high variance, especially if they are deep. This high variance means the model might be very sensitive to the specific training data it was given, leading to overfitting.

3. Interpretability in Large Trees : While Decision Trees are generally easy to interpret, this interpretability diminishes as the tree becomes larger and more complex. Large trees can become very difficult to understand and visualize.

4. Computational Cost : Although the training time of 2.459231 seconds is reasonable, the computational cost can increase significantly with the size of the dataset and the complexity of the tree.

B- Future Directions

1. Pruning : Implementing pruning techniques can help reduce overfitting. By trimming the branches of the tree that have little importance, we can make the model simpler and more generalized.

2. Ensemble Methods : Consider using ensemble methods such as Random Forests or Gradient Boosting Trees. These methods build multiple trees and combine their predictions, often resulting in better performance and reduced overfitting compared to a single decision tree.

3. Hyperparameter Tuning : Optimize hyperparameters such as the maximum depth of the tree, minimum samples per leaf, and minimum samples per split using techniques like Grid Search or Random Search to improve the model's performance.

4. Cross-Validation : Use cross-validation to get a better estimate of the model's performance and to ensure that the model generalizes well to unseen data.

5. Feature Engineering : Further explore feature engineering techniques to create more informative features that could improve the model's performance.

6. Regularization Techniques : Introduce regularization techniques to penalize overly complex models, which can help mitigate overfitting.

C- Lessons Learned

1. Importance of Model Evaluation : It's crucial to evaluate the model on a separate test set to ensure that the model generalizes well. The high training score but lower test accuracy highlighted the need for careful model evaluation.

2. Model Complexity : Balancing model complexity is essential. While complex models may fit the training data better, they may not generalize well to new data. Techniques like pruning and using ensemble methods can help achieve this balance.

3. Feature Selection : The quality of the features can significantly impact model performance. Feature selection and engineering are critical steps in building a robust model.

4. Computational Efficiency : Although Decision Trees are relatively fast, computational efficiency becomes important as the dataset size and tree complexity increase.

5. Bias-Variance Tradeoff : Understanding and addressing the bias-variance tradeoff is crucial in model development. Overfitting and underfitting can significantly affect model performance and generalization.

In conclusion, while the Decision Tree Classifier showed promising results with high accuracy and precision, addressing its limitations through advanced techniques and careful tuning can lead to better performance and generalization, making the model more reliable for practical applications.