DARKNET TRAFFIC CLASSIFICATION USING MACHINE LEARNING

Aslam Baig
Siva Sai
Sumana Vemuri
Karthik Ramanarayana
Suresh Devi

Aim & Purpose of this project

- **Aim of the Project :**
  - The aim of this project is to build ML model to predict the type of the application used based on the its traffic attributes in a Darknet.

- **Problem statement :**
  - Darknet is the unused address space of the internet. Any communication from the dark space is considered skeptical owing to its passive listening nature.
  - Due to the absence of legitimate hosts in the darknet, any traffic is contemplated to be unsought and is characteristically treated as probe, backscatter, or misconfiguration.
  - So ,Classification is of network traffic is of paramount importance to categorize the real-time applications from illicit malicious activity software.

- **Why it is so important ?**
  - Analyzing darknet traffic helps in early monitoring of malware before onslaught and detection of malicious activities after outbreak.

Goal and approaches to problem

- **Goal of the Project:**
  - In this project, we are trying to analyze the traffic flow of various type of applications in a darknet and predict the type of the application been using without checking the encrypted information at packet level using supervised ML models for Classification.

- **Approaches to the achieve goal:**
  - From the problem statement , we concur that it is purely a classification problem with multiple labels where we are going to classify the type of the network flow into multiple categories of real-time application classes.

  - This classification must be done based on the network's attributes , flow constants etc.., So ,the dataset must contain the key attributes that separates the different categories of the applications in a darknet.

  - After that, an Intuitive supervised machine learning model must be built on top of this dataset that handles the desired classification accurately.

Data sets sources

- The darknet datasets are gathered from the opensource research-based platforms as there require special configurations on normal open network accessible to everyone. Wireshark and tcpdump tools are used to capture the Pcap file from both non-VPN and VPN based networks connected to a darknet.

- UNB ISCX Network Traffic (VPN) ISCXVPN2016 dataset and UNB CIC Network Traffic (Tor-nonTor) ISCXTor2016 datasets offered by Gerard Drapper provide the Network traffic data flown from encrypted VPN and non-VPN networks , Tor and nonTor browser datasets .This dataset consists of labeled network traffic, including full packet in csv formats.

- A hybrid dataset of darknet traffic is created by amalgamating out ISCXTor2016 and ISCXVPN2016 datasets to create a complete darknet dataset covering Tor and VPN traffic respectively called CICDarknet2020.

- The dataset contains the details of darknet traffic which constitutes the categories of applications used the darknet which forms our classification labels. They labels are Audio-Stream, Browsing, Chat, Email, P2P, Transfer, Video-Stream & VOIP .

Data set Description

Table 1: Darknet Network Traffic Details

Traffic Category	Applications used
Audio-Stream	Vimeo and Youtube
Browsing	Firefox and Chrome
Chat	ICQ, AIM, Skype, Facebook and Hangouts
Email	SMTPS, POP3S and IMAPS
P2P	uTorrent and Transmission (BitTorrent)
Transfer	Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla and an external service
Video-Stream	Vimeo and Youtube
VOIP	Facebook, Skype and Hangouts voice calls

Existing vs Proposed

- The existing solutions and related research is based on the classification of the encrypted traffic flow in the open network or VPN based network ,whereas we are trying to classify the type of the application in a darknet in the both cases of the VPN and non-VPN , tor and Non-Tor access scenarios.

- The hybrid dataset CICDarknet2020 of darknet traffic is created by amalgamating out ISCXTor2016 and ISCXVPN2016 datasets to create a complete darknet dataset covering Tor and VPN traffic which forms the complex scenarios.

- We will be building a supervised classifier on this complex dataset that classifies the appropriate category of the application more precisely.

Why switched to New Dataset

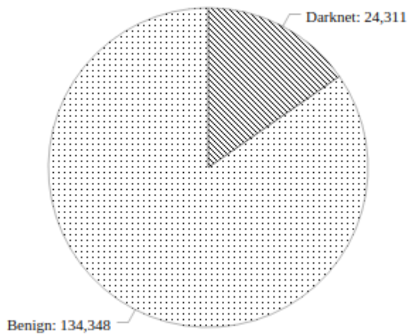
- The aim of building the network traffic classifier remain intact but the capturing real-time dataset from home-based networks doesn't suffice the requirements of dataset size and variants of classes used for classification. Moreover , some additional configurations and software are required to access the darknet from personal computers.

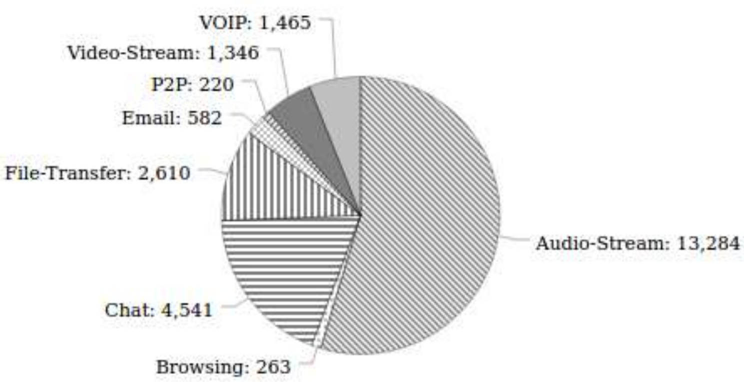
- So,UNB’s research-based dataset CICDarknet2020 (ISCXTor2016 & SCXVPN2016) dataset has been used as it fits best for our requirements in terms of the dataset size , number of features and labelled classes that we are going to predict.

- Reference : https://www.unb.ca/cic/datasets/darknet2020.html

Dataset – Description

- In CICDarknet2020 dataset, a two-layered approach is used to generate benign and darknet traffic at the first layer. The darknet traffic constitutes Audio-Stream, Browsing, Chat, Email, P2P, Transfer, Video-Stream and VOIP which is generated at the second layer. To generate the representative dataset, UNB amalgamated datasets ISCXTor2016 & ISCXVPN2016, and combined the respective VPN and Tor traffic in corresponding Darknet categories.





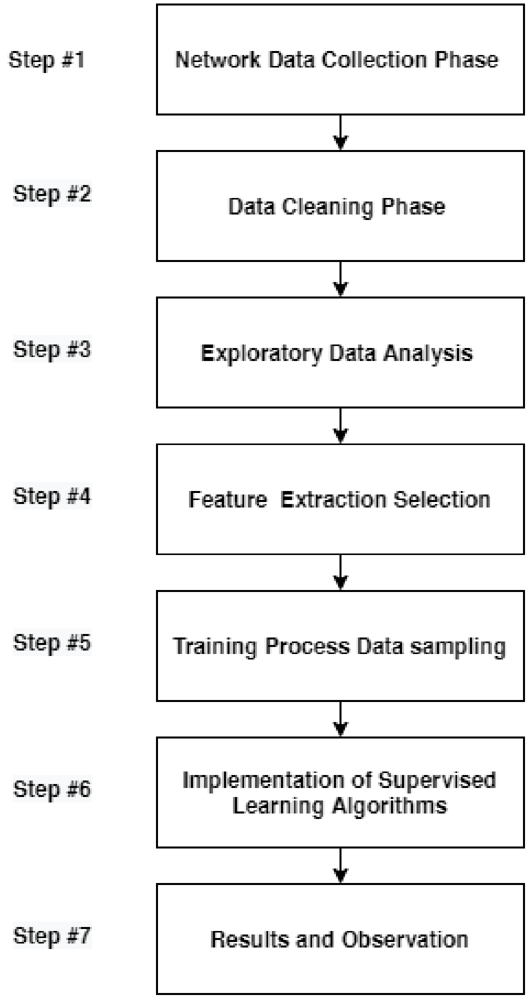
- UNB's research-based dataset CICDarknet2020 (ISCXTor2016 & SCXVPN2016) contains the details of darknet traffic which constitutes the categories of applications used the darknet which forms our classification labels. They labels are Audio-Stream, Browsing, Chat, Email, P2P, Transfer, Video-Stream & VOIP .

Project Pipeline

- The entire process of the building a Darknet traffic classification model starts with data collection phase and end with evaluating the model performance against darknet dataset.

- It is a seven step approach where each stage have the sub steps .

- At the final step, out of the built classifiers best classification model is select that has high performance rate.



Project Pipeline

- This project encompasses the below steps
  1. Dataset Collection
  2. Dataset cleaning and Pre-processing
  3. EDA
  4. Feature Selection and Engineering
  5. Model building
  6. Evaluating the model Performance
  7. Hyperparameter tuning

1. Data Collection Phase

- The Darknet traffic data is collected from the Network monitoring tools which are subjected to flow from both Tor and Non-Tor browsers. The UNB's CICDarknet2020 data set is one of such datasets which is amalgamated out ISCXTor2016 and ISCXVPN2016 datasets

- It contains 84 features and 141531 samples , a perfect match for our objectives and satisfy all the three main components of a good dataset, which are real-world, substantial and diverse.

- Once, this data is collected , preliminary analysis on the distribution of the data must be done using manual approaches and was sent to next phase of the pipeline.

```
#Analysing various aspects of dataset
df.describe()
```

	Src Port	Dst Port	Protocol	Flow Duration	Total Fwd Packet	Total Bwd packets	Total Length of Fwd Packet	Total Length of Bwd Packet	Fwd Packet Length Max	Fwd Packet Length Min
count	141530.000000	141530.000000	141530.000000	1.415300e+05	141530.000000	141530.000000	1.415300e+05	1.415300e+05	141530.000000	141530.000000
mean	38450.268819	18124.647333	10.350427	2.061280e+07	152.800749	154.642062	1.126211e+05	1.304530e+05	208.920420	15.817078
std	19124.801990	22202.197159	5.431807	3.809155e+07	2378.323352	3418.715287	3.251367e+06	4.588180e+06	649.432333	31.312298
min	0.000000	0.000000	0.000000	0.000000e+00	1.000000	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000
25%	32425.500000	80.000000	6.000000	1.778100e+04	1.000000	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000
50%	43528.000000	5366.000000	6.000000	4.162820e+05	2.000000	1.000000	4.400000e+01	0.000000e+00	34.000000	0.000000
75%	53338.000000	40020.000000	17.000000	1.181470e+07	4.000000	3.000000	2.160000e+02	2.160000e+02	103.000000	31.000000
max	65534.000000	65535.000000	17.000000	1.200000e+08	238161.000000	470862.000000	7.693074e+08	6.704287e+08	64240.000000	1350.000000

```
df.dtypes

Flow ID          object
Src IP           object
Src Port          int64
Dst IP           object
Dst Port          int64
                  ...
Idle Std        float64
Idle Max        float64
Idle Min        float64
Label            object
Label.1          object
Length: 85, dtype: object
```

1.Data Collection Phase-Dataset

- The CICDarknet2020 data set contains 84 features and 141531 samples .
- Features pre-processing : The 84 features better describes the type of the traffic in the darknet. However, most of the features like Flow ID , Source IP and Destination IP address etc., are not useful for our purpose prediction
- These features were engineered, and best effective features were selected in further phases.

```
df.columns

Index(['Flow ID', 'Src IP', 'Src Port', 'Dst IP', 'Dst Port', 'Protocol',
       'Timestamp', 'Flow Duration', 'Total Fwd Packet', 'Total Bwd packets',
       'Total Length of Fwd Packet', 'Total Length of Bwd Packet',
       'Fwd Packet Length Max', 'Fwd Packet Length Min',
       'Fwd Packet Length Mean', 'Fwd Packet Length Std',
       'Bwd Packet Length Max', 'Bwd Packet Length Min',
       'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
       'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
       'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std',
       'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean',
       'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags',
       'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length',
       'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
       'Packet Length Min', 'Packet Length Max', 'Packet Length Mean',
       'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count',
       'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',
       'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio',
       'Average Packet Size', 'Fwd Segment Size Avg', 'Bwd Segment Size Avg',
       'Fwd Bytes/Bulk Avg', 'Fwd Packet/Bulk Avg', 'Fwd Bulk Rate Avg',
       'Bwd Bytes/Bulk Avg', 'Bwd Packet/Bulk Avg', 'Bwd Bulk Rate Avg',
       'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets',
       'Subflow Bwd Bytes', 'FWD Init Win Bytes', 'Bwd Init Win Bytes',
       'Fwd Act Data Pkts', 'Fwd Seg Size Min', 'Active Mean', 'Active Std',
       'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max',
       'Idle Min', 'Label', 'Label.1'],
      dtype='object')
```

2. Data Cleaning & Pre-processing phase

- In this phase, the dataset is set to free from inconsistencies such as missing data and supplicates .These kinds of impurities were removed by cleansing the data by dropping the duplicates and filling missing values with value values close to the mean of that feature.

- Next , the transformation of the categorical data and output labels is performed as most of the models only handles the numeric data .

```
In [9]: #As the volume of null values is small, thus dropping those
        df.dropna(axis = 0, inplace = True)

In [10]: df.isnull().values.any()

Out[10]: False

In [11]: #Dropping duplicated values
         df = df.loc[:,~df.columns.duplicated()]
```

```
In [13]: df['Label'] = df['Label'].astype('str')
         df['Label'].dtype
         df['Label'] = df['Label'].apply(lambda name : name.upper())

In [14]: df['Label.1'] = df['Label.1'].astype('str')
         df['Label.1'].dtype
         df['Label.1'] = df['Label.1'].apply(lambda name : name.upper())

In [15]: #Converting categorical values into numerical values using label encode
         labelencoder= LabelEncoder()
         df['Label'] = labelencoder.fit_transform(df['Label'])
```

3.Exploratory data analysis phase

- In this phase, the filtered dataset is then analyzed to identify the underlying relationships among each feature in the dataset. The highly correlated values and outliers are detected and removed from the dataset

- First , highly correlated data is identified by defining the correlation matrix and the values having the high values of correlation coefficient r > 9 is removed from the dataset.

```python
#Finding correlation among columns
cor = df.corr()
cor.head()
```

	Src Port	Dst Port	Protocol	Flow Duration	Total Fwd Packet	Total Bwd packets	Total Length of Fwd Packet	Total Length of Bwd Packet	Fwd Packet Length Max	Fwd Packet Length Min	...
Src Port	1.000000	-0.245809	-0.097933	0.065040	-0.036318	-0.014283	-0.019744	-0.008293	0.076338	-0.090378	...
Dst Port	-0.245809	1.000000	-0.320981	0.039493	0.022131	0.014801	0.004468	0.010882	0.004597	-0.178557	...
Protocol	-0.097933	-0.320981	1.000000	-0.267141	-0.034756	-0.026179	-0.023051	-0.020883	-0.195232	0.563995	...
Flow Duration	0.065040	0.039493	-0.267141	1.000000	0.142106	0.100285	0.072527	0.057006	0.340708	-0.069027	...
Total Fwd Packet	-0.036318	0.022131	-0.034756	0.142106	1.000000	0.744834	0.457390	0.635688	0.125570	-0.020993	...

5 rows × 80 columns

```python
#deleting columns which have r-value>9
col_cor = set()
for i in range(len(cor.columns)):
    for j in range(i):
        if (abs(cor.iloc[i,j])>=0.8)) and (cor.columns[j] not in col_cor):
            col_name = cor.columns[i]
            col_cor.add(col_name)


print("cols are: ",col_cor)
print("no of features: ", len(col_cor))
```

```
cols are:  {'Fwd IAT Total', 'Bwd Packets/s', 'Packet Length Max', 'Bwd IAT Total', 'Idle Max', 'Fwd Packet Length Std', 'Fwd I
AT Mean', 'Packet Length Std', 'Fwd IAT Max', 'Bwd Header Length', 'Bwd Packet Length Std', 'Fwd Segment Size Avg', 'Flow IAT M
ax', 'Subflow Bwd Bytes', 'Idle Min', 'Fwd Packets/s', 'Bwd Segment Size Avg', 'ACK Flag Count', 'Bwd IAT Mean', 'Average Packe
t Size', 'Subflow Fwd Bytes', 'Total Length of Bwd Packet', 'Fwd Header Length', 'Bwd Packet/Bulk Avg'}
no of features:  24
```

3.Exploratory data analysis phase-Cont

- Secondly , The dataset is sent to Outlier Treatment. The outliers in the dataset are detected based on the distribution of the values outside the Inter Quartile Range (IQR). $IQR = Q_3 - Q_1$
- Once, the Outliers are detected the observations are dropped from the dataset which are beyond the IQR to make the data distribution much standard distribution.
-

```
#Finding inter quartile range
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3-Q1
print("The IQR of all data: ",IQR)
```

```
#Dropping the outliers
df_new = df[~((df < (Q1 - 1.5*IQR)) | (df > (Q3 + 1.5*IQR))).any(axis=1)]
df_new.head()
```

- This pruned dataset is then sent for feature selection and best features are engineered from the dataset that better classifies the traffic.

4. FEATURE EXTRACTION & SELECTION

- After preforming the EDA, dropping the less useful features, outlier treatment using Inter-quartile method, The dataset has been pre-processed and has some useful features.

- Contextual Anomalies might cause inconsistencies in the Prediction and must be handled by detecting using the tree based unsupervised Isolation Forest algorithm. Minority classes and the attribute-values which differs from standard normal distribution are eliminated using this method.

```
from sklearn.ensemble import IsolationForest

random_state = np.random.RandomState(42)

model=IsolationForest(n_estimators=100,max_samples='auto',contamination=float(0.2),random_state=random_state)

for col in X.columns:
        model.fit(X[[col]])
        print("Result of " + col + ":")
        print(model.get_params())
```

4. Feature Extraction & Selection-Cont

- Then the data has been sent for SMOTE (Synthetic Minority Oversampling Technique) to overcome the imbalanced classification by to oversample the minority class. This is mainly used to generated synthetic instances for the minority class which has less observations.

```
# Splitting the dataset into test train split
X_train, X_test, y_train, y_test = train_test_split(Xn, Yn, test_size=0.20, shuffle=True)

#applying SMOTE (Synthetic Minority Oversampling Technique) to overcome the imbalanced classification by to oversample the minor
smote= SMOTE('auto')
X_sm,y_sm = smote.fit_resample(X_train, y_train)
print(np.shape(X_sm), np.shape(y_sm))

# saving the output labels in a dictionary format
dic = {}
for i in y_sm:
    if i in dic.keys():
        dic[i]+=1
    else:
        dic[i]=1
print(dic)
```

```
(9014, 56) (9014,)
(45689, 56) (45689,)
{5: 6527, 4: 6527, 0: 6527, 2: 6527, 1: 6527, 6: 6527, 3: 6527}
```

4. Feature Extraction & Selection-Cont

- Top Useful features for classification are selected based on the Techniques of **SelectKBest Class and Extra Tree Classifier** for extracting the top features for the datasets.
- In the SelectKBestClass method, Chi-squared (chi²) statistical techniques are used for non-negative features better defines the best useful features to be selected.
- The most useful feature have the best scores.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

X_ordered_rank_feat = SelectKBest(score_func=chi2, k='all')
X_ordered_feat= X_ordered_rank_feat.fit(X_sm, y_sm)
```

```
dfscores=pd.DataFrame(X_ordered_feat.scores_,columns=["Score"])
dfcolumns=pd.DataFrame(X_col_names)
features_rank=pd.concat([dfcolumns,dfscores],axis=1)

# Feature vs score i.e its effect on target variable
features_rank.columns=['Features','Score']
features_rank.sort_values(by=['Score'],ascending= False)
```

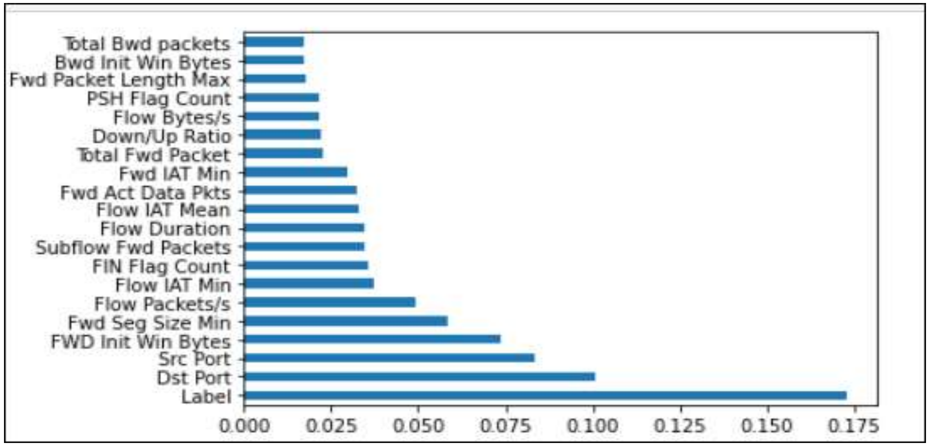
	Features	Score
53	Idle Mean	2.388827e+18
3	Flow Duration	6.925753e+09
18	Fwd IAT Std	5.982073e+09
16	Flow IAT Std	5.702904e+09
15	Flow IAT Mean	2.840596e+09
1	Dst Port	8.858334e+08
19	Fwd IAT Min	6.493460e+08
17	Flow IAT Min	2.984355e+08
46	FWD Init Win Bytes	2.282206e+08
0	Src Port	4.677719e+07

4. Feature Extraction & Selection-Cont

- Alternatively using inbuilt class feature_importances of tree-based classifiers such as **ExtraTreeClassifier** can also be used for extracting the top n. features from the dataset used for the classification.
- From the available ordered features, Top 20 features are considered for the classification than that of the others.

```
#Feature importance is an inbuilt class that comes
#we will be using Extra Tree Classifier for extract
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X_sm,y_sm)
```

```
ExtraTreesClassifier()
```



4. FEATURE EXTRACTION & SELECTION-CONT

- From feature selection techniques ,the most useful features list is extracted, and data is set to be prepared from this list to model building .

- ***Feats*** *= ['Label','Dst Port','Fwd Seg Size Min','Src Port','FWD Init Win Bytes','Total Fwd Packet','Bwd Init Win Bytes','Flow Duration', 'Flow IAT Mean','Total Bwd packets', 'FIN Flag Count','Flow Bytes/s','Packet Length Mean','Total Length of Fwd Packet','Fwd Packet Length Mean','Down/Up Ratio','Fwd IAT Std','Packet Length Variance']*

5. Model Building -Classification

- For building a multi-Label classifier, the below set of the models provides high classification rate on the Darknet dataset.

I. Decision Tree
II. Random Forest
III. KNN Model

- Hence, the best model which has the high classification rate is selected and evaluated for the desired results.

5. Model – Decision Tree

- The decision tree classifier use numeric and categorical data for the classification problems. It also supports nonlinear relationships between features.
- When the Decision Tree classifier is trained over the training split. It provided the score of the 98% accuracy

Decision Tree

```
In [49]: print("Starting to train")
         dt = DecisionTreeClassifier()
         dt.fit(X_train , y_train)

         Starting to train

Out[49]: DecisionTreeClassifier()

In [50]: dt.tree_.node_count, dt.tree_.max_depth

Out[50]: (961, 26)

In [51]: dt.score(X_test, y_test)

Out[51]: 0.9861342207432058
```

5. Model – Decision Tree

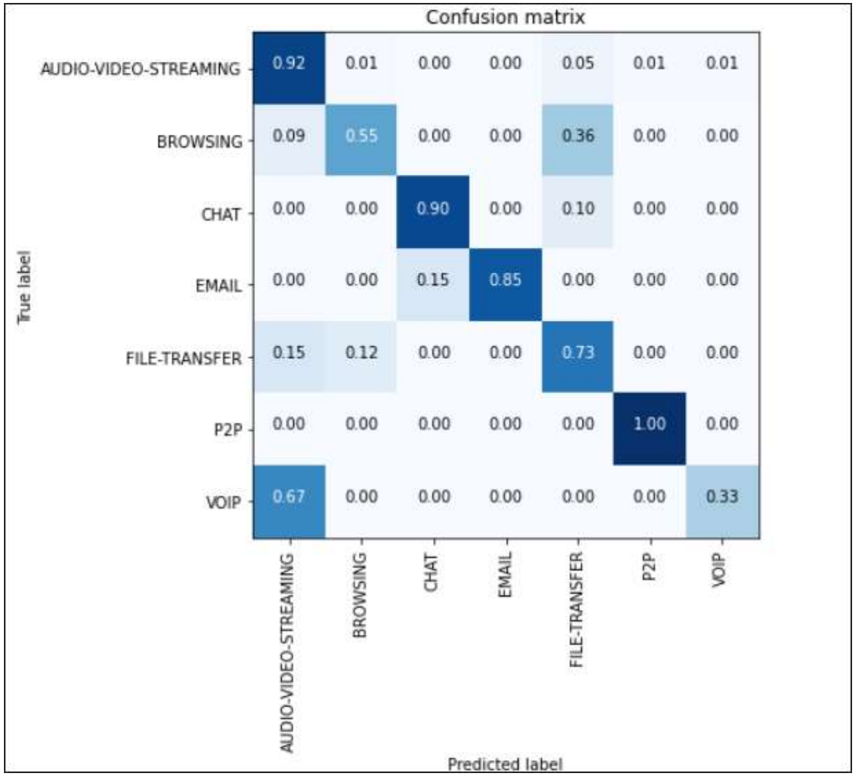
- Classification report of the Decision Tree Classifier shows the precision recall f1 score and support metrics for each of the output labels classified by the Decision Tree. Below fig represents the Classification report.

```
y_pred = dt.predict(X_test)
print('accuracy %s' % accuracy_score(y_test, y_pred))
print("Classes: ", le.inverse_transform([0,1,2,3,4,5,6]))
print(classification_report(y_test, y_pred, target_names= le.inverse_transform([0,1
```

```
accuracy 0.9861342207432058
Classes:  ['AUDIO-VIDEO-STREAMING' 'BROWSING' 'CHAT' 'EMAIL' 'FILE-TRANSFER' 'P2P'
 'VOIP']
                       precision    recall  f1-score   support

AUDIO-VIDEO-STREAMING       0.93      0.92      0.93       106
             BROWSING       0.60      0.55      0.57        11
                 CHAT       0.82      0.90      0.86        10
                EMAIL       1.00      0.85      0.92        13
        FILE-TRANSFER       0.66      0.73      0.69        26
                  P2P       1.00      1.00      1.00      1634
                 VOIP       0.50      0.33      0.40         3

            micro avg       0.99      0.99      0.99      1803
            macro avg       0.69      0.66      0.67      1803
         weighted avg       0.99      0.99      0.99      1803
```

5. Model – Decision Tree

- The Confusion matrix of this classifier against the given output classes better depicts the Decision Tree classifer's performance on the Darknet dataset.
-



5. Model – K nearest Neighbor

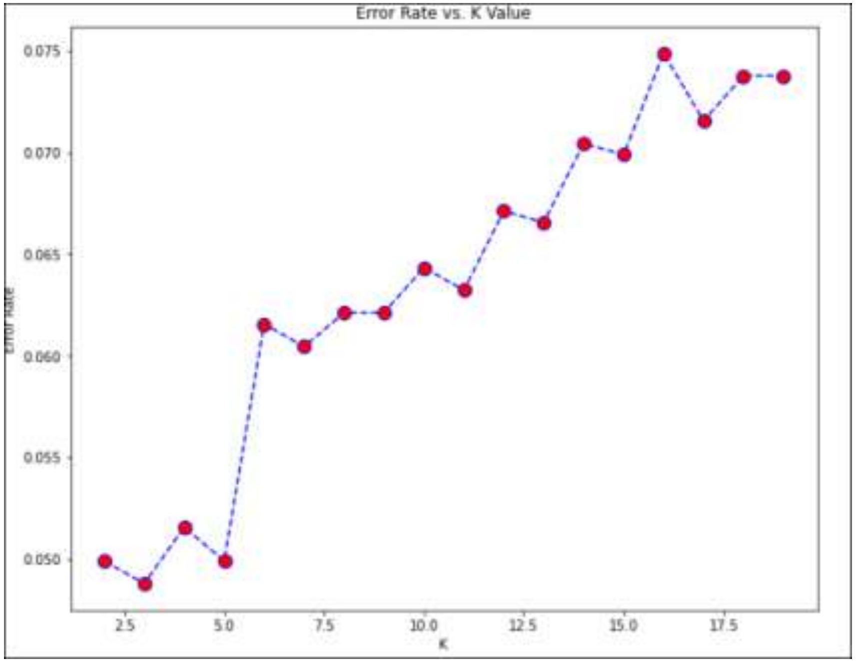
- Classification report of the Knn Classifier shows the precision recall f1 score and support metrics for each of the output labels classified by the Knn. Below fig represents the Classification report.

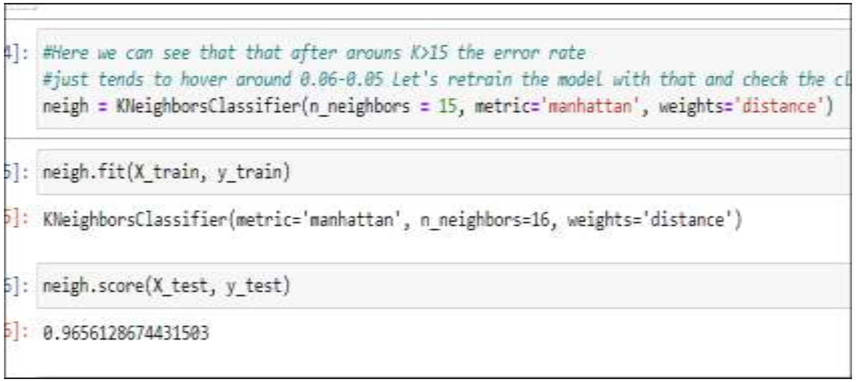
```
: print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.88      0.77      0.82       108
           1       0.40      0.43      0.41        14
           2       0.79      0.73      0.76        15
           3       1.00      0.85      0.92        13
           4       0.24      0.38      0.29        21
           5       1.00      0.99      1.00      1631
           6       0.08      1.00      0.14         1

    accuracy                           0.97      1803
   macro avg       0.63      0.74      0.62      1803
weighted avg       0.98      0.97      0.97      1803
```

5. Model – K-Nearest neighbor

- Knn Model is a robust model that can work with noisy data and perform better if the training data set is large.
- On comparing the accuracies of the KNN model under each K, the better suited value for K is 15. Thus, this model better performs with 15 neighbors

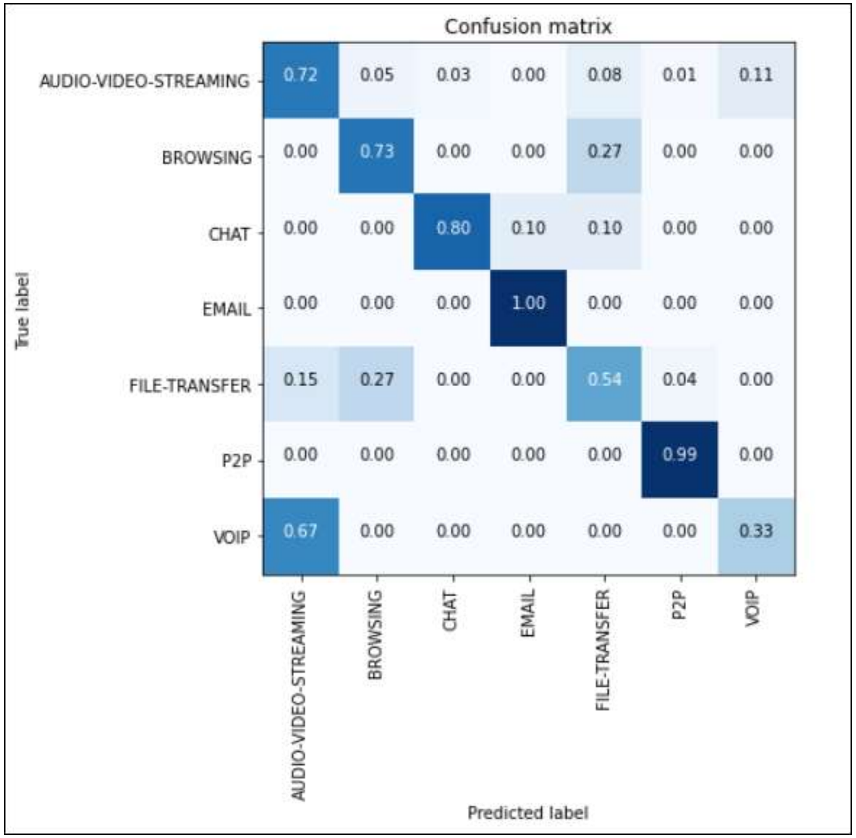
- When the KNN classifier is trained over the training split. It provided the score of the 96% accuracy





5. Model – K-Nearest neighbor

- The Confusion matrix of this classifier against the given output classes better depicts the Knn classifer's performance on the Darknet dataset.
-



5. Model – Random Forest

- The Random Forest classifier use numeric and categorical data for the classification problems and is able to handle large datasets
- When the Random Forest classifier is trained over the training split. It provided the score of the 99% accuracy

Random Forest

```
#rf = RandomForestClassifier(max_depth=60
rf = RandomForestClassifier()
rf.fit(X_train , y_train)
```

```
RandomForestClassifier()
```

```
rf.score(X_test, y_test)
```

```
0.9900166389351082
```

5. Model – Random Forest

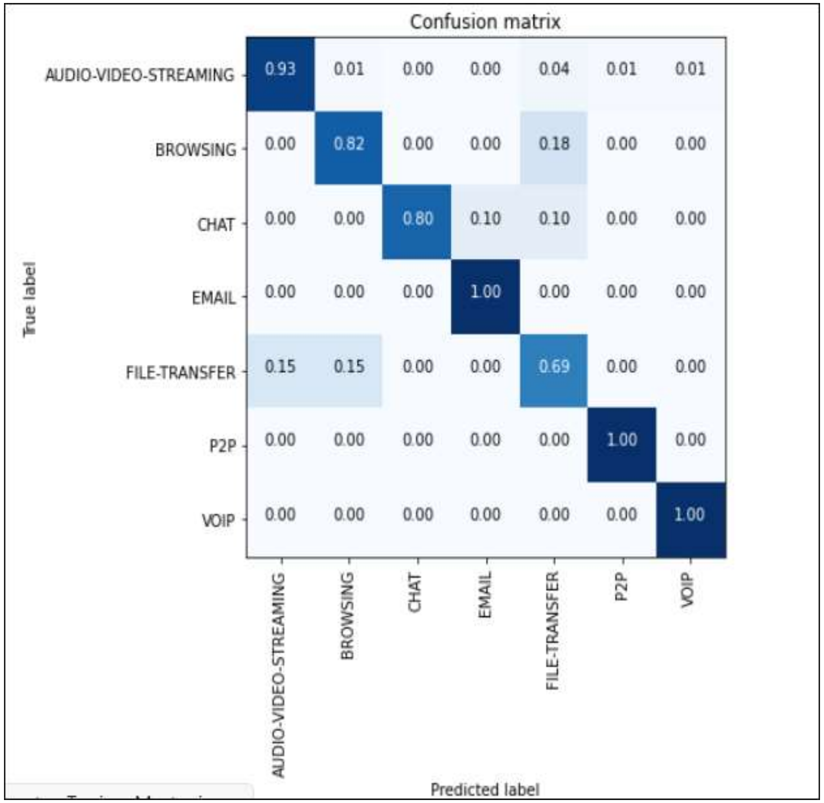
- Classification report of the Random Forest Classifier shows the precision recall f1 score and support metrics for each of the output labels classified by the Random Forest. Below fig represents the Classification report.

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       106
           1       0.64      0.82      0.72        11
           2       1.00      0.80      0.89        10
           3       0.93      1.00      0.96        13
           4       0.75      0.69      0.72        26
           5       1.00      1.00      1.00      1634
           6       0.75      1.00      0.86         3

    accuracy                           0.99      1803
   macro avg       0.86      0.89      0.87      1803
weighted avg       0.99      0.99      0.99      1803
```

5. Model – Random Forest

- The Confusion matrix of this classifier against the given output classes better depicts the Random Forest classifier's performance on the Darknet dataset.



6. Comparing Model Performances

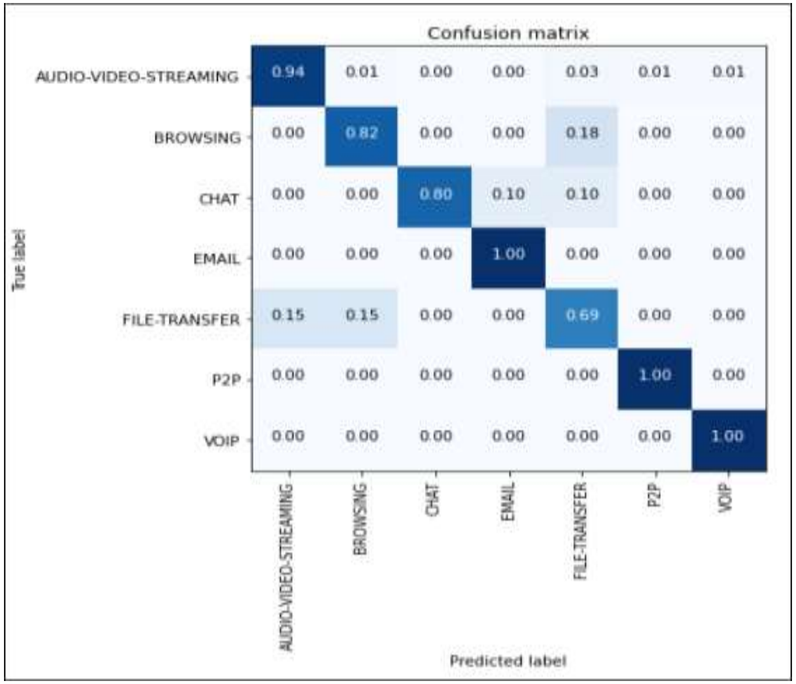
- On comparing the Confusion matrix and Classification reports of all the 3 classifers Random Forest has the best classification rate and can better define the Output Classes of the Darknet .

Classification model accuracies	
Model Name	Accuracy
Decision Tree	98.44%
K-NN Model	96.56%
Random Forest	99.0 %

7. Tuning the RF model

- The classification rate of Random Forest Classifier can be further improved by Hyperparameter tuning the model and selecting the best parameters to improve the model prediction rate.
- On tuning the parameters of the RF model with the below params list the classification rate of RF model further improved shown from the Confusion Matrix .

param_grid *= {'criterion': 'gini', 'max_depth': 50, 'max_features': 'log2', 'n_estimators': 40}*



8.Conclusion

- The comparative analysis of three machine learning classifiers shows the significant way of handling the classification task.

- It can be seen that traffic classification using Supervised machine learning algorithms provides good results in classifying the Darknet traffic from both VPN and Non-VPN based networks. In the near future, these models are scaled further to readily incorporated into the network devices to classify the darknet traffic.

- Also, we infer that for classification task , the features: Flow duration, Src Port, Dst Port, Total Fwd packets,flow byte count and average packet size better defines the type of Traffic in the Darknet.

- Our attempt to train this Darknet dataset across Neural Net models RNN showed significantly less performance in classification which is less than that of above three classifiers

References

1.https://www.unb.ca/cic/datasets/darknet2020.html

2. https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

3. https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb

4. https://www.scikit-yb.org/en/latest/api/cluster/elbow.html#:~:text=K%2Dmeans%20is%20a%20simple,number%20(k)%20of%20clusters.&text=The%20elbow%20method%20runs%20k,average%20score%20for%20all%20clusters.

THANK YOU