Predicting Cyber Attacks

Shannon School of Business

Cape Breton University, Sydney, NS, Canada

Group 11 | MGSC-5125: Predictive Modelling and Analytics
Professor: Enayat Rajabi

Submitted by-

Ajay Pratap Singh Rathore (0242630)

Manik Sood (0242207)

Jinal Ajay Patel (245097)

Ryan Philip Alfabeto (0251738)

Table of Contents

Business Understanding

There are at least 2 million cyberattacks each day through different various forms of cyber threats. According to security firm Kaspersky Lab, cyberattacks are increasing every year. Cyber-attacks are an important issue faced by all organizations. Securing information systems is critical. Organizations should be able to understand the ecosystem and predict attacks. Predicting attacks quantitatively should be part of risk management. The cost impact due to worms, viruses, or other malicious software is significant. This project proposes an analytical model to predict the cyber-attack on systems (Destination IP) of an organization so they can avoid and try to implement the countermeasures.

Objective

We are preparing an analytical model to predict the cyber-attack (Generic) on set of end systems (Destination IP) of an organization.

Problem Statement

After analyzing our dataset we found that there were different types of attacks reported on the end systems of an organization, so to avoid the future attacks of the same type targeted for some specific machines (website hosting servers) we are preparing a predictive model.

Dataset

Link to the dataset: https://www.kaggle.com/iamranjann/exploring-attacks-cybersec/data

Tools and Language Used in this project

Jupyter Notebook using Python

Predictive Modelling Steps

The task for predictive modelling is divided into 6 main steps which we will discuss in detail and they are as follows:

Step 1: Data Understanding

We are going to read a .csv file that contains information about different cybersecurity events that have occurred in a specific time frame. In this way, the following attributes are available in the data:

Time: Start and end date of the attack in timestamp format.

Attack category: Type of registered cybersecurity attack.

Attack subcategory: Subcategory of the type of cybersecurity attack registered

Protocol: Protocol used for the attack.

Source IP: IPv4 address where the attack came from.

Source Port: Logical port where the attack came from.

Destination IP: Destination IPv4 address.

Destination Port: Logical destination port.

Attack Name: Technical name for the cybersecurity attack.

Attack Reference: Common Vulnerabilities and Exposures (CVE) reference of the type of cybersecurity attack.

The following types of attacks are included in the dataset:

Reconnaissance: It can be defined as a probe in which the attacker collects information about a computer network to evade its security controls.

Fuzzers: The attacker attempts to uncover security loopholes in a program, operating system, or network by feeding it with massive random data entry to block it.

Analysis: A type of intrusion that penetrates web applications through ports (for example, port scans), emails (for example, spam), and web scripts (for example, HTML files).

Backdoor: A stealthy technique to avoid normal authentication to ensure unauthorized remote access to a device.

Exploit: Sequence of instructions that exploits a flaw (vulnerability) caused by involuntary or unsuspected behaviour on a host or network.

Generic: A technique that attempts against block encryption using a hash function for collision regardless of encryption settings.

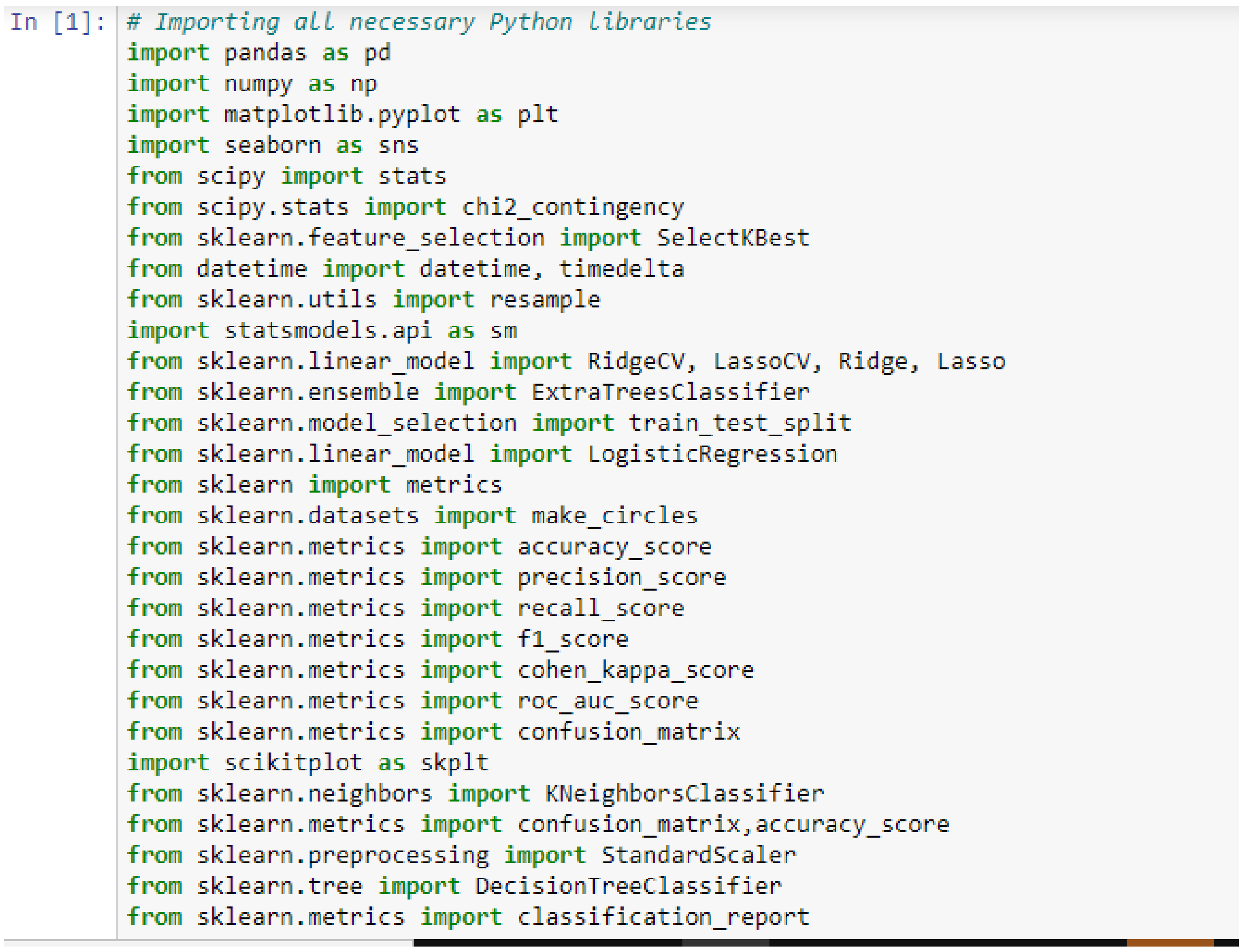
Shellcode: The attacker penetrates a small piece of code from a shell to control the compromised machine.

Worm: The attack replicates malicious script to spread it to other computers. Often, it uses a computer network to spread, depending on security flaws in the destination computer.

DoS: An intrusion that disrupts computer resources, often through memory, to be extremely busy to prevent unauthorized requests from accessing a device.

Imported two csv files from Kaggle - "**Cybersecurity_attacks.csv**" and "**TCP-ports.csv**" using pandas library.

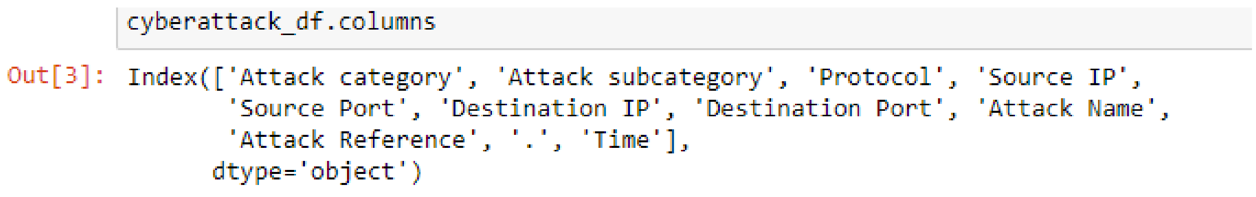
Imported following Libraries which we had used:



```
In [1]: # Importing all necessary Python libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from scipy import stats
        from scipy.stats import chi2_contingency
        from sklearn.feature_selection import SelectKBest
        from datetime import datetime, timedelta
        from sklearn.utils import resample
        import statsmodels.api as sm
        from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
        from sklearn.ensemble import ExtraTreesClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn import metrics
        from sklearn.datasets import make_circles
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import precision_score
        from sklearn.metrics import recall_score
        from sklearn.metrics import f1_score
        from sklearn.metrics import cohen_kappa_score
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import confusion_matrix
        import scikitplot as skplt
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import confusion_matrix,accuracy_score
        from sklearn.preprocessing import StandardScaler
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import classification_report
```

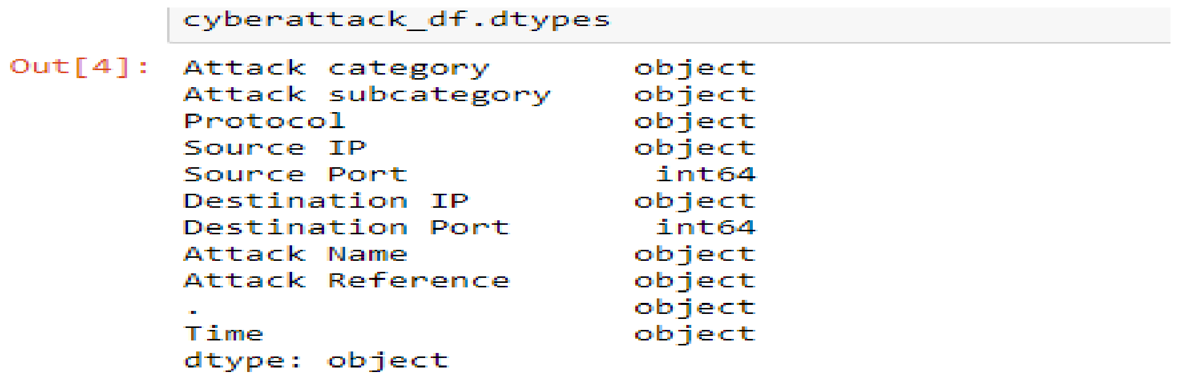
Step 2: Data Cleaning and Preparation

1. Determining the columns



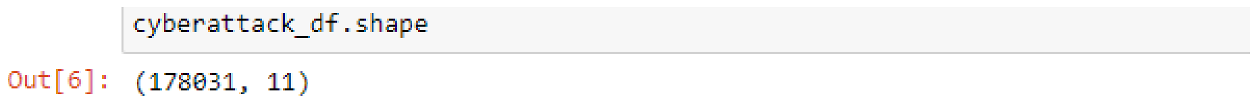
```
        cyberattack_df.columns

Out[3]: Index(['Attack category', 'Attack subcategory', 'Protocol', 'Source IP',
               'Source Port', 'Destination IP', 'Destination Port', 'Attack Name',
               'Attack Reference', '.', 'Time'],
              dtype='object')
```

2. Data types of each Column



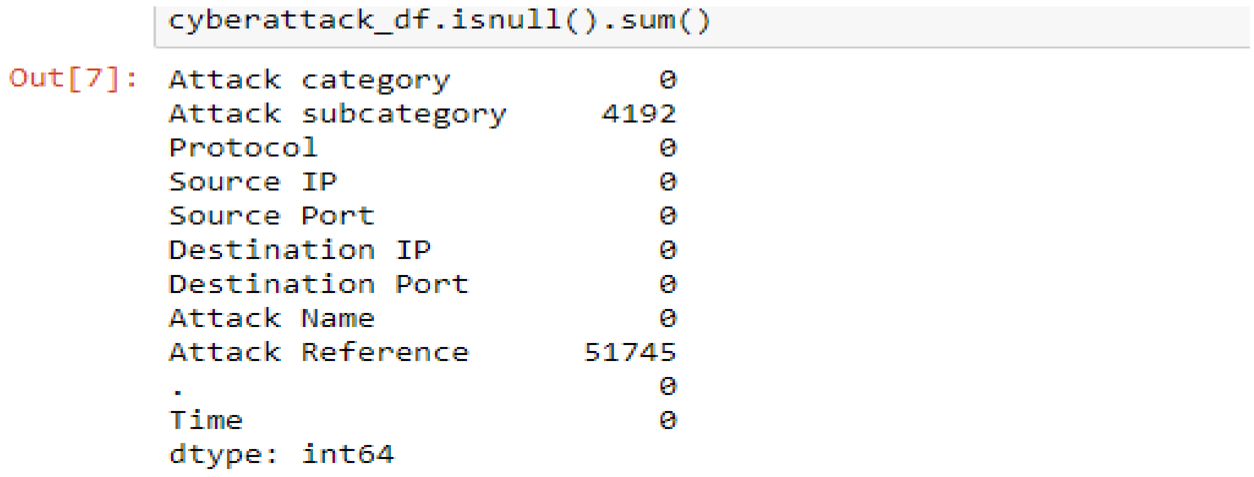
```
         cyberattack_df.dtypes

Out[4]:  Attack category        object
         Attack subcategory     object
         Protocol               object
         Source IP              object
         Source Port             int64
         Destination IP         object
         Destination Port        int64
         Attack Name            object
         Attack Reference       object
         .                      object
         Time                   object
         dtype: object
```

3. Shape of dataset



```
         cyberattack_df.shape

Out[6]:  (178031, 11)
```

4. Checked for Duplicates and Null Values in the dataset



```
         cyberattack_df.isnull().sum()

Out[7]:  Attack category           0
         Attack subcategory     4192
         Protocol                  0
         Source IP                 0
         Source Port               0
         Destination IP            0
         Destination Port          0
         Attack Name               0
         Attack Reference      51745
         .                         0
         Time                      0
         dtype: int64
```

As there are 2 columns having null data i.e. "Attack category" and "Attack Reference" we will clean in the following way:

- Converting all Null values in Attack category to Not Registered
- Dropping the column Attack Reference as it will not help further in our Prediction model

5. Cleaned column data having inconsistencies like:

- Spaces in data
- Inconsistent case
- Data separated with special characters
- Converting data

6. Spliting Time data to be readable format

cyberattack_df[['Start time','End time']] = cyberattack_df['Time'].str.split('-',expand=True)

7. As TCP_ports.csv is cleaned already, merging 2 data sets for Data Exploration

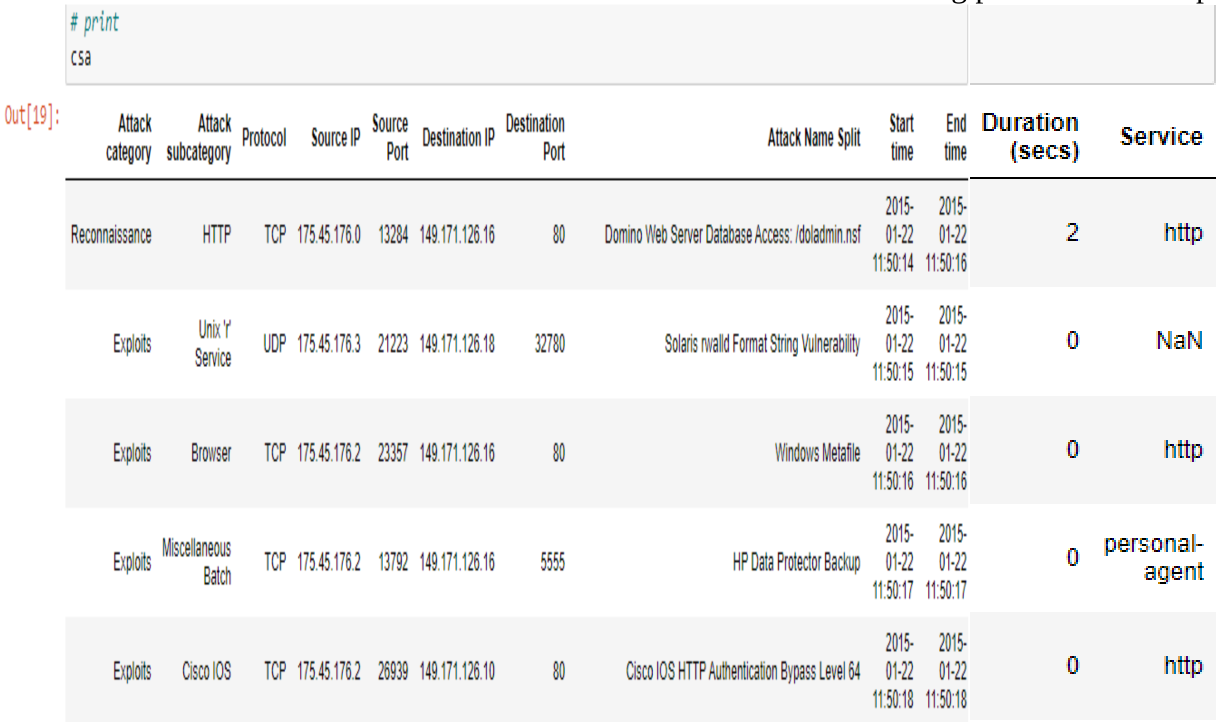
#new data frame - merging 2 data frames on Destination Port and Service columns

csa = pd.merge(cyberattack_df, tcp_ports_df[['Port','Service']], left_on='Destination Port', right_on='Port', how='left')

8. Dropping unused columns

csa.drop(columns =["Attack Reference", ".", "Time", "Attack Name Misc","Port"], inplace = True)

Final cleaned dataset has178031 rows and 12 columns after all the cleaning processes and step:



```
# print
csa
```

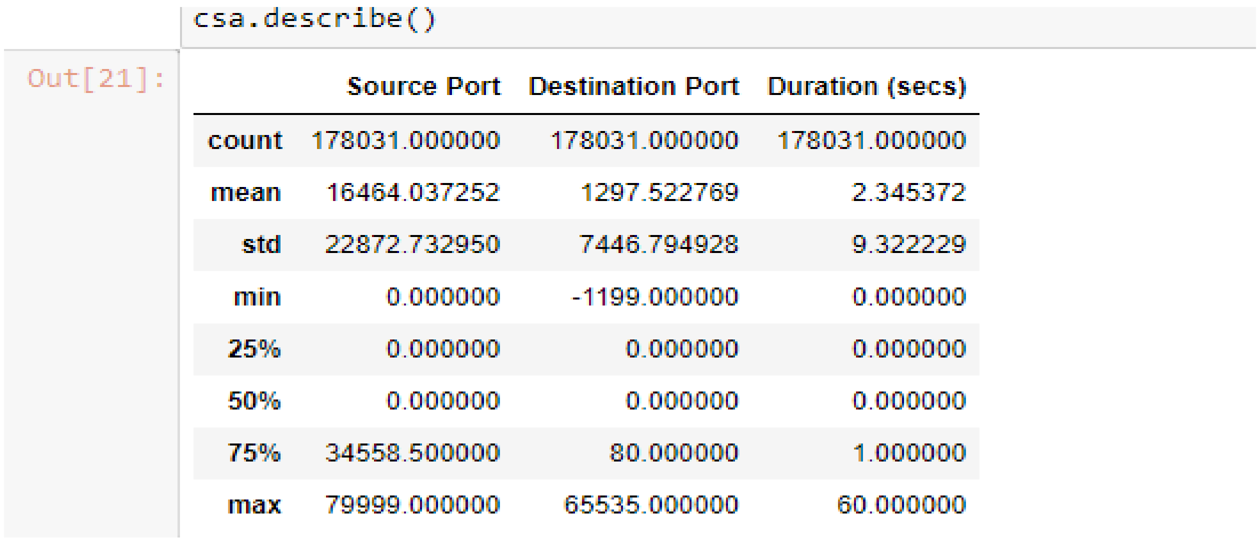
Out[19]:

	Attack category	Attack subcategory	Protocol	Source IP	Source Port	Destination IP	Destination Port	Attack Name Split	Start time	End time	Duration (secs)	Service
	Reconnaissance	HTTP	TCP	175.45.176.0	13284	149.171.126.16	80	Domino Web Server Database Access: /doladmin.nsf	2015-01-22 11:50:14	2015-01-22 11:50:16	2	http
	Exploits	Unix 'r' Service	UDP	175.45.176.3	21223	149.171.126.18	32780	Solaris rwalld Format String Vulnerability	2015-01-22 11:50:15	2015-01-22 11:50:15	0	NaN
	Exploits	Browser	TCP	175.45.176.2	23357	149.171.126.16	80	Windows Metafile	2015-01-22 11:50:16	2015-01-22 11:50:16	0	http
	Exploits	Miscellaneous Batch	TCP	175.45.176.2	13792	149.171.126.16	5555	HP Data Protector Backup	2015-01-22 11:50:17	2015-01-22 11:50:17	0	personal-agent
	Exploits	Cisco IOS	TCP	175.45.176.2	26939	149.171.126.10	80	Cisco IOS HTTP Authentication Bypass Level 64	2015-01-22 11:50:18	2015-01-22 11:50:18	0	http

Step 3: Data Exploration

And finally, predicting which source IP, destination IP had most attacks (Attack category, source IP, destination IP).

Describing the numerical data in the csa data frame but Source Port, Destination Port are considered as Categorical data only Duration is a valid numerical data.



```
csa.describe()
```

Out[21]:

	Source Port	Destination Port	Duration (secs)
count	178031.000000	178031.000000	178031.000000
mean	16464.037252	1297.522769	2.345372
std	22872.732950	7446.794928	9.322229
min	0.000000	-1199.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	34558.500000	80.000000	1.000000
max	79999.000000	65535.000000	60.000000

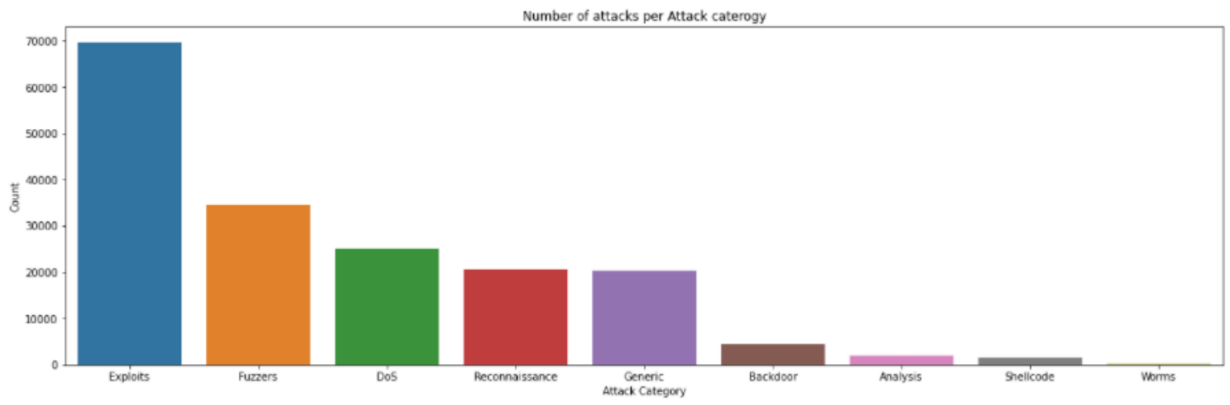
Normalization, Skewness and Kurtosis: As our target variable is "Attack category" which contains the nominal data and therefore we cannot perform normalization, skewness and kurtosis in our dataset.

Important Queries to explore the dataset and find meaningful insights and patterns:

1. Most frequent Attack name based on Attack Category

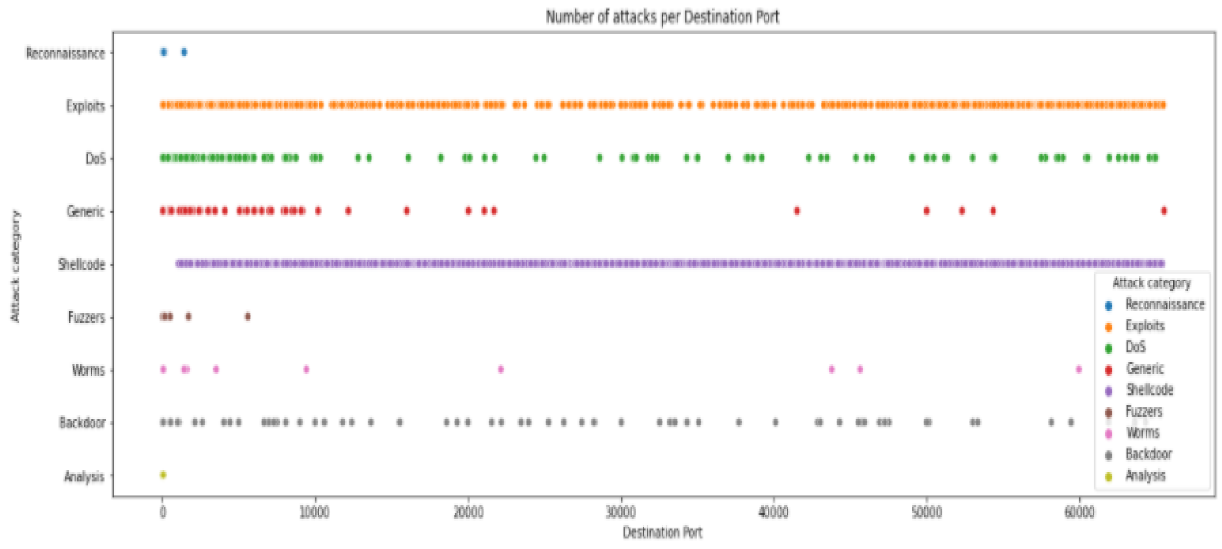
From the count & graph below, it is clear that "Exploits" had the highest number of attacks on the machines. Let us further explore the data set.

```
Text(0.5, 1.0, 'Number of attacks per Attack caterogy')
```



Trying to also exlpore Attack subcategory and is found that "Miscellaneous" were at the peak from sub-category. Therefore, we will remove it as this will not be required for our prediction model.

Exploring Destination port with Attack category to know which ports had the highest kind of attack. It is found from the visualization below that "Exploits" and "Shellcode" were high where as "Reconnaissance" and "Analysis" were the least.

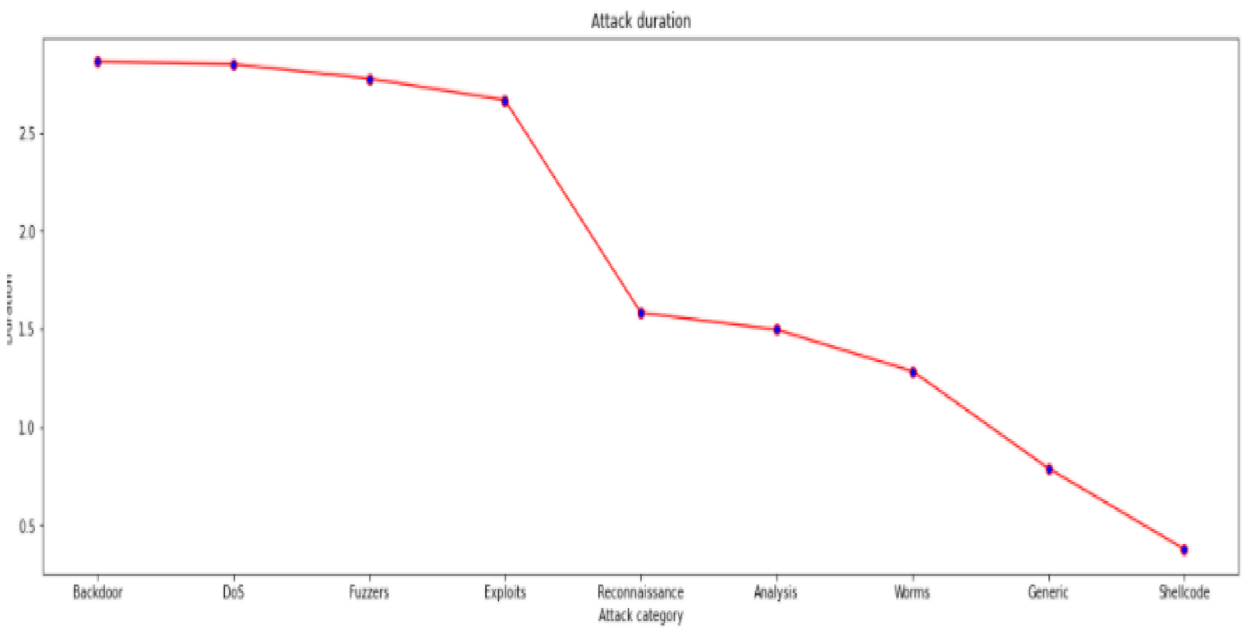


Result:

"Exploits" and "Miscellaneous" type of cyber category and sub-category were highest respectively.

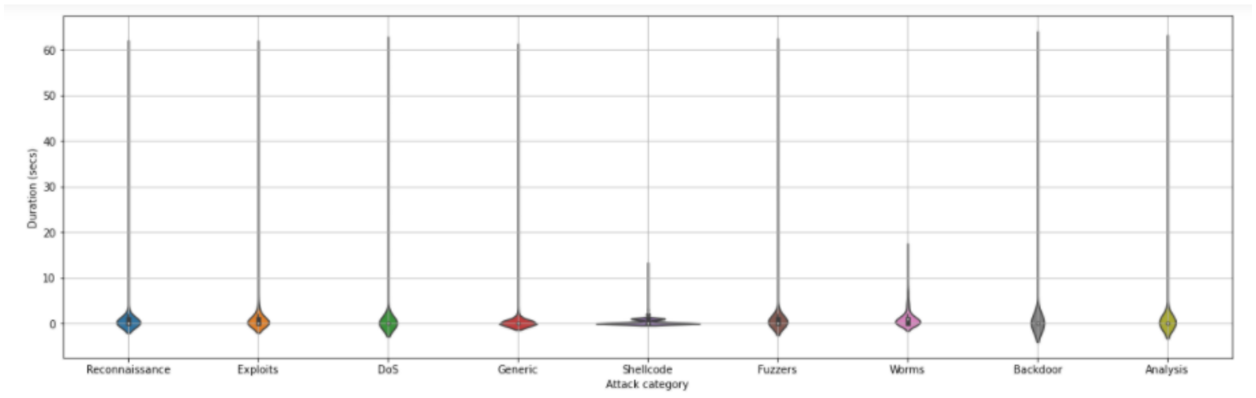
2. During what time of a day there were most attacks (Attack category and Duration)

2.1. Exploring cyber-attacks against duration (secs)



2.2. Duration Analysis

From the visualization below, two types of attacks that do not record times greater than 20 seconds are the shellcode and the worms

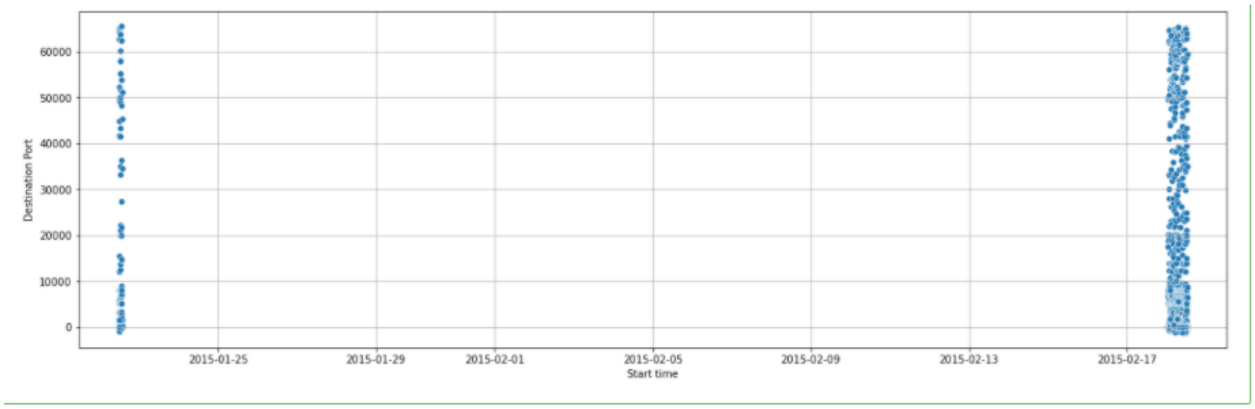


Result:

"Backdoor" was the Attack type that was impacted for the highest time in seconds, almost 2.8 seconds

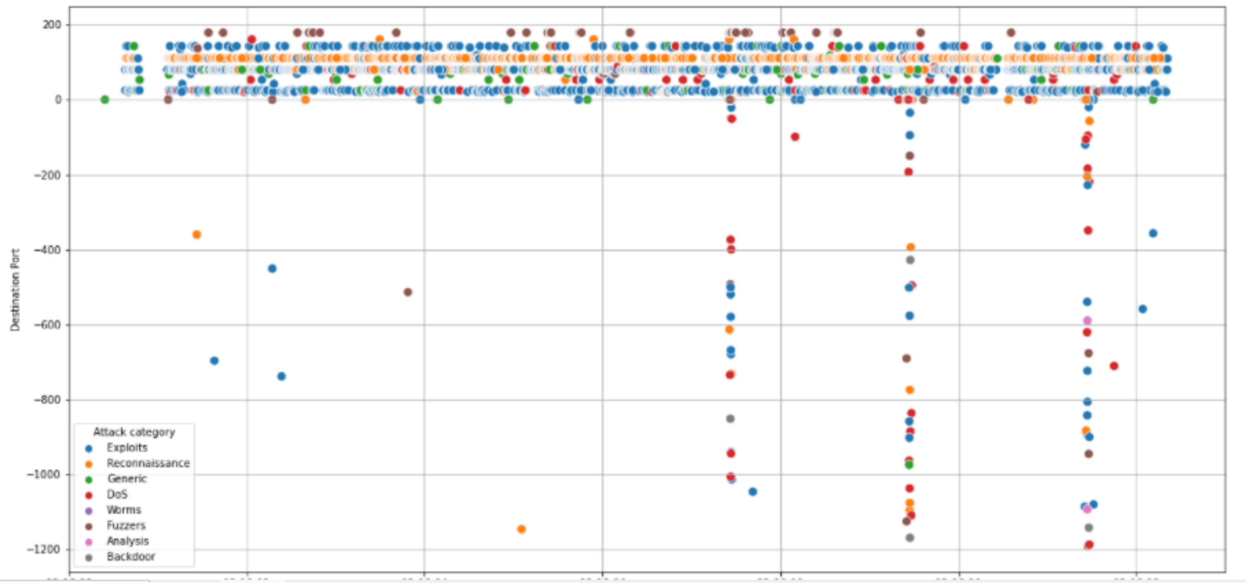
3. Most attack on destination IP (Attack category, destination IP)

Exploring most attacked destination IP address 149.171.126.17 against Start time and End time.



It can be seen from the above visualization that most of the attacks happened after Feb 17,2015 for machine "149.171.126.17" ranging with Destination Ports between 0 and 66000

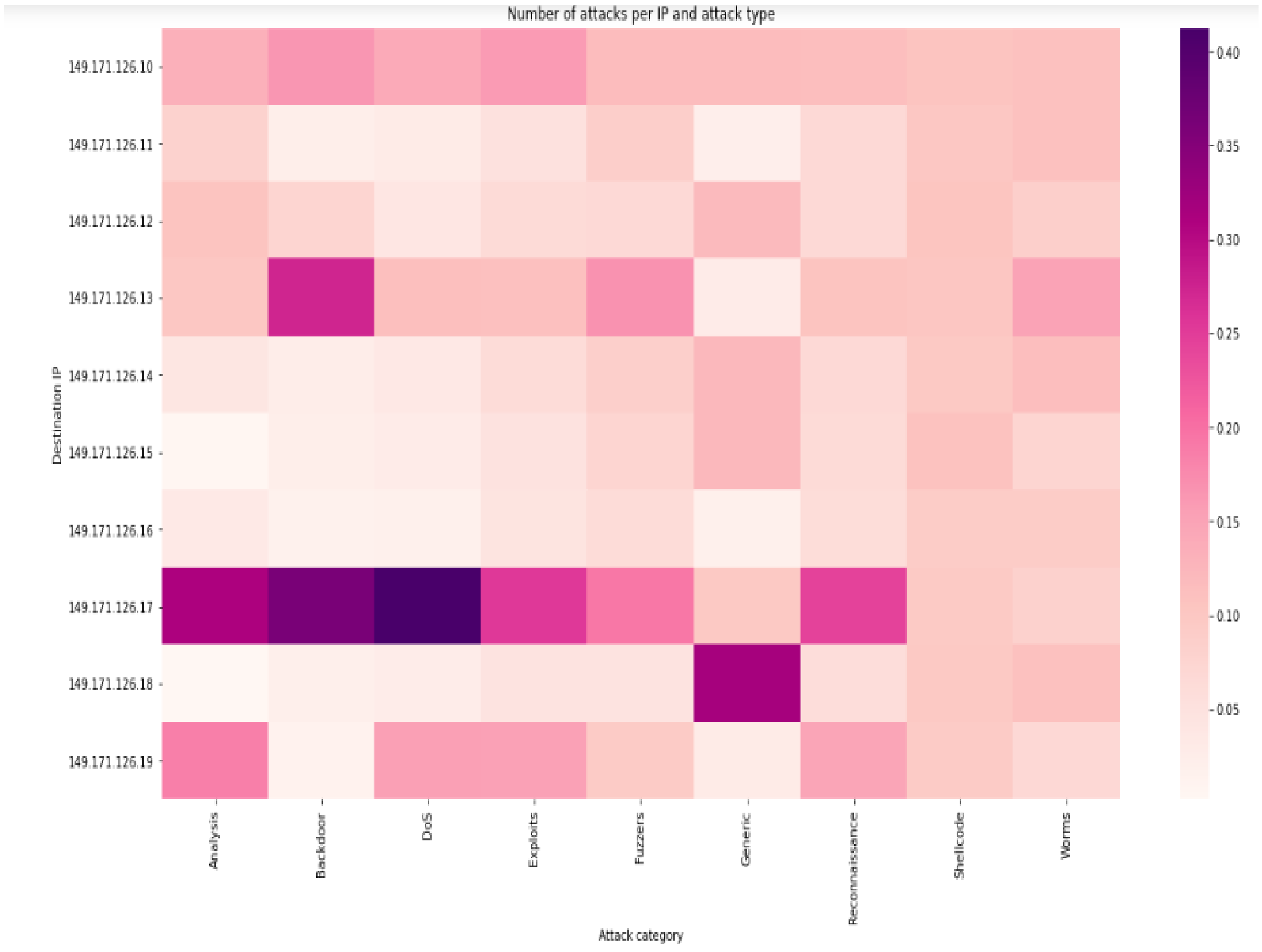
Understanding the pattern of Destination Port against start time of attack by categorizing the type of attack



Result: In fact, there is an evident pattern in the graph in which Ten clear constant trends are perceived on the values 0, 22, 25, 53, 80, 110, 111, 138, 143 and 178 of the y-axis. This pattern corresponds to the very well-known ports that provide important TCP and UDP services

4. Exploring further relationship between the time (start and end time) and attacks, Destination IP that were executed

.Heat map of the number of attacks per Destination IP and its respective Attack category



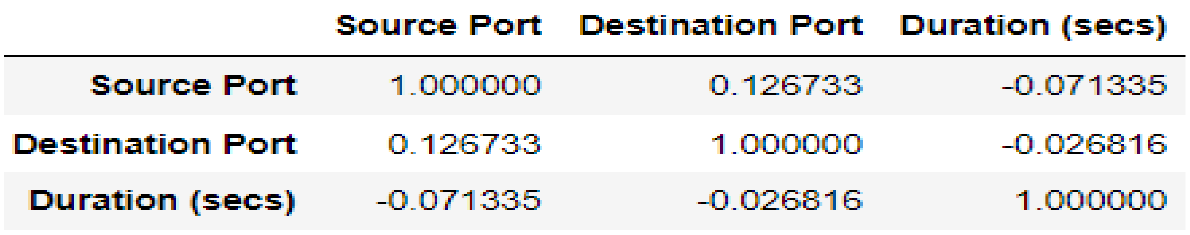
Result: The most vulnerable machine address is - "149.171.126.17". It is found that "Denial of Services (DoS)", "Exploits" and "Backdoor" attacks are clearly targeted towards specific Machines/Servers.

Step 4: Correlation and Feature Engineering

In order to understand the relationship of different columns in our data set we performed and selecting best features out of it, we used 3 methods - Pearson correlation, Embedded method and Feature importance.

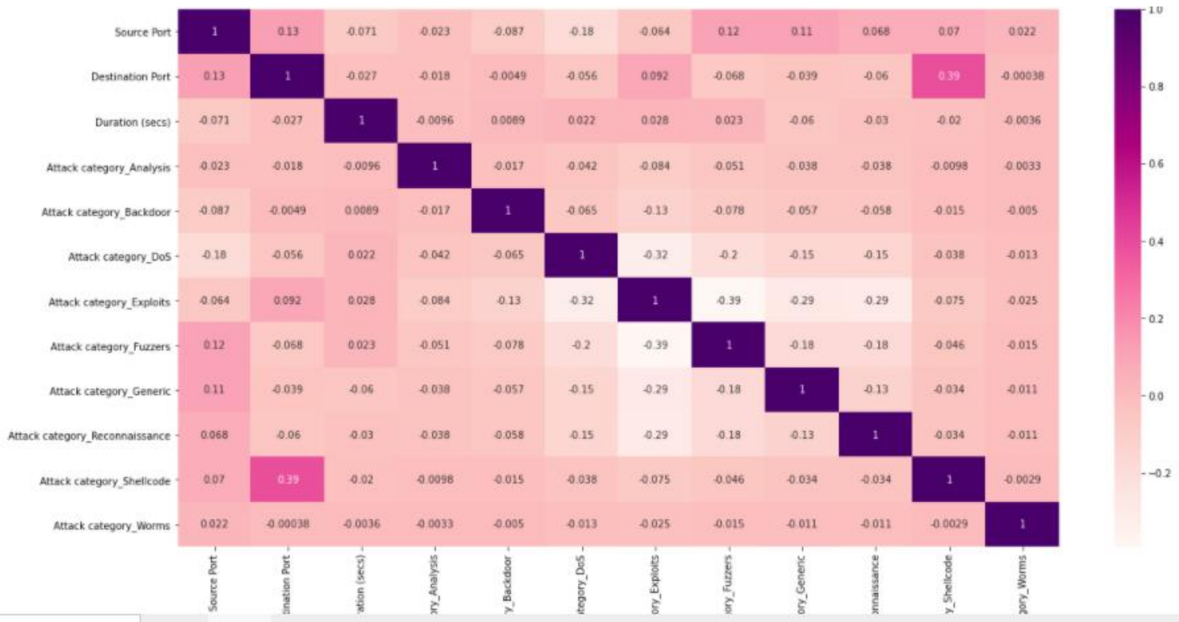
1. Pearson Correlation

Executing correlation function using Pearson method on the numerical dataset, but Source and Destination Port are Nominal data and only Duration (secs) is numerical data.



	Source Port	Destination Port	Duration (secs)
Source Port	1.000000	0.126733	-0.071335
Destination Port	0.126733	1.000000	-0.026816
Duration (secs)	-0.071335	-0.026816	1.000000

Created dummies for only "Attack category" to understand the relationship with other columns in dataframe csa. Correlation heatmap for dummies of "Attack category".

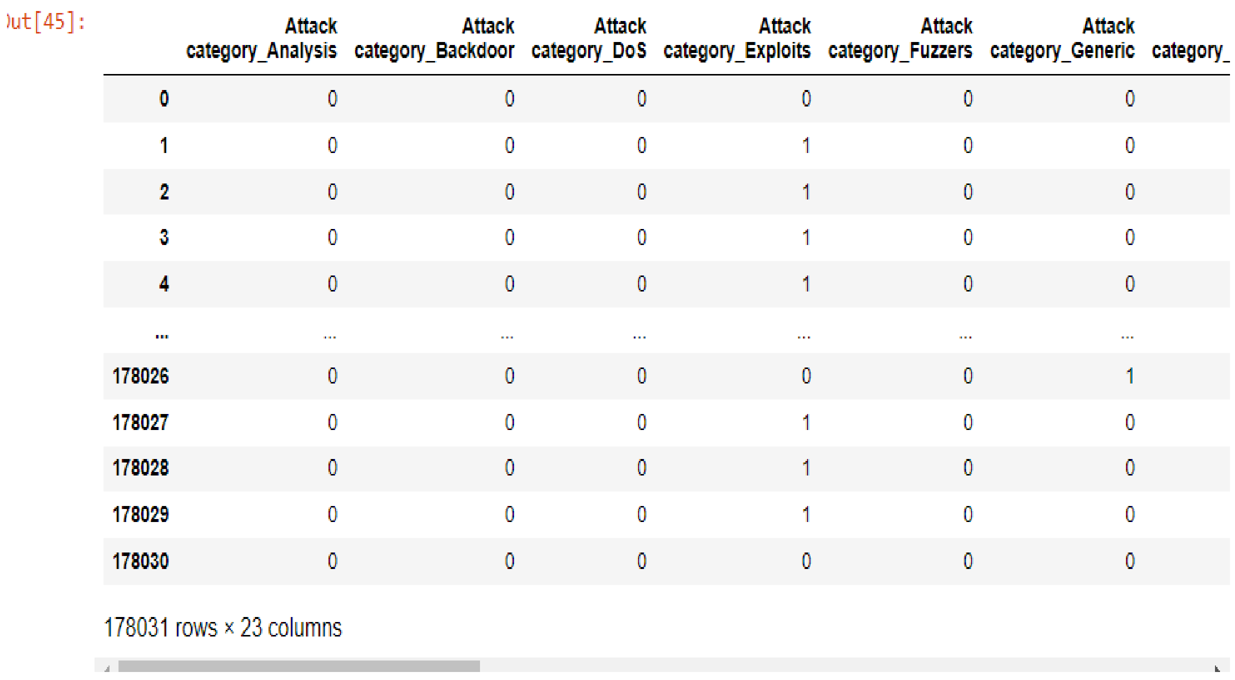


Result: By analyzing the above heat map we concluded that destination port is having the highest correlation value of 0.39 to particular attack category named shellcode. As rest all the correlation values are very small, so we have selected only destination port as the only feature from Pearson method.

Created dummy variables for Attack category, Source IP and Destination IP in order to predict the most attacks on a particular machine.

Finding correlation between Source and Destination IP per Attack type and therefore using Pearson method to understand the correlation between them.

As we created dummies, let us implement Pearson method on independent and target columns from corelation_dummy dataframe.



)ut[45]:

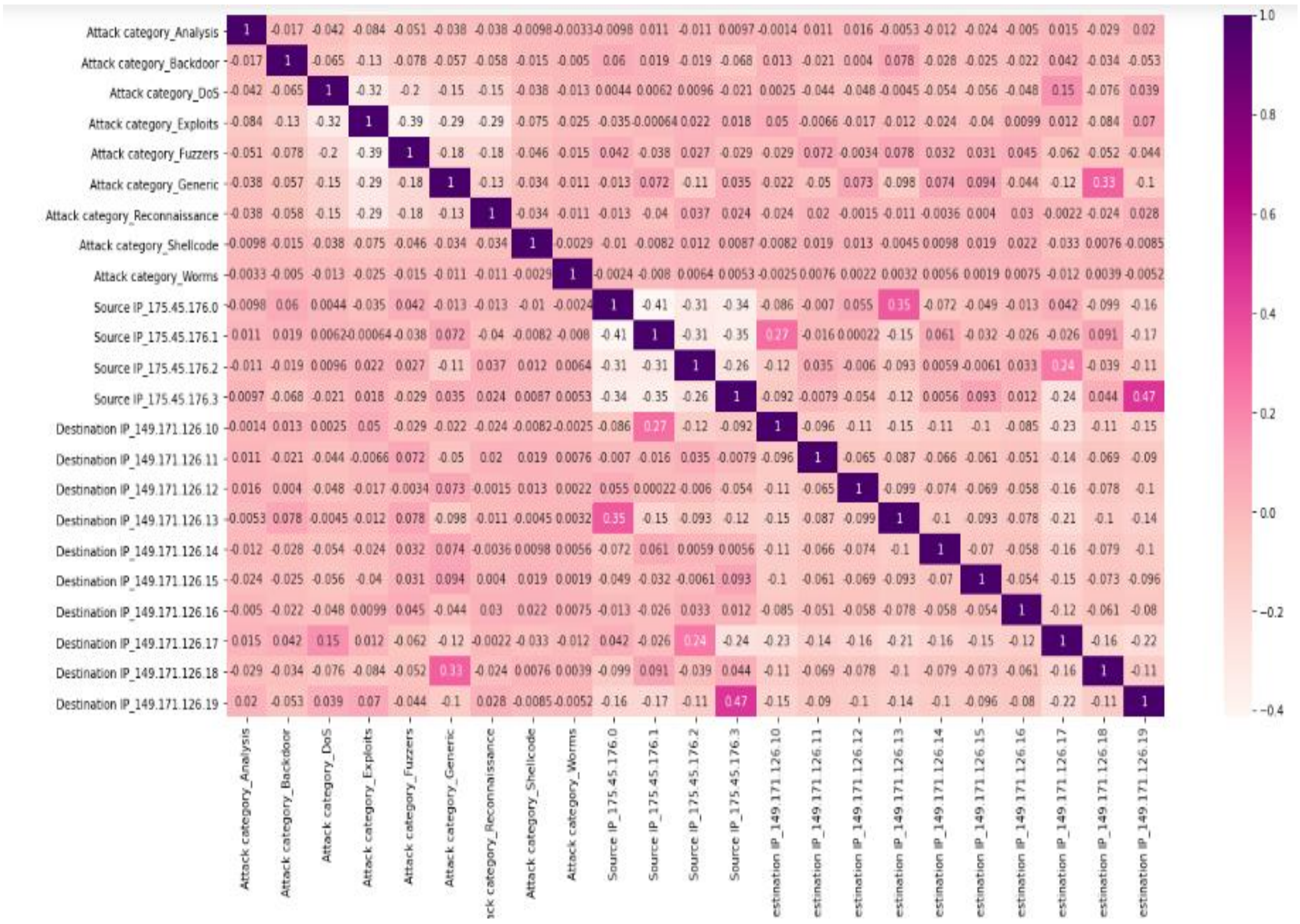
	Attack category_Analysis	Attack category_Backdoor	Attack category_DoS	Attack category_Exploits	Attack category_Fuzzers	Attack category_Generic	category_
0	0	0	0	0	0	0	
1	0	0	0	1	0	0	
2	0	0	0	1	0	0	
3	0	0	0	1	0	0	
4	0	0	0	1	0	0	
...	...	...	...	...	...	...	
178026	0	0	0	0	0	1	
178027	0	0	0	1	0	0	
178028	0	0	0	1	0	0	
178029	0	0	0	1	0	0	
178030	0	0	0	0	0	0	

178031 rows × 23 columns

X = corelation_dummy.iloc[:,9:22] #independent columns y = corelation_dummy.iloc[:,5]   #target column i.e Generic

3 features were selected are Source IP_175.45.176.2, Destination IP_149.171.126.17, Destination IP_149.171.126.18

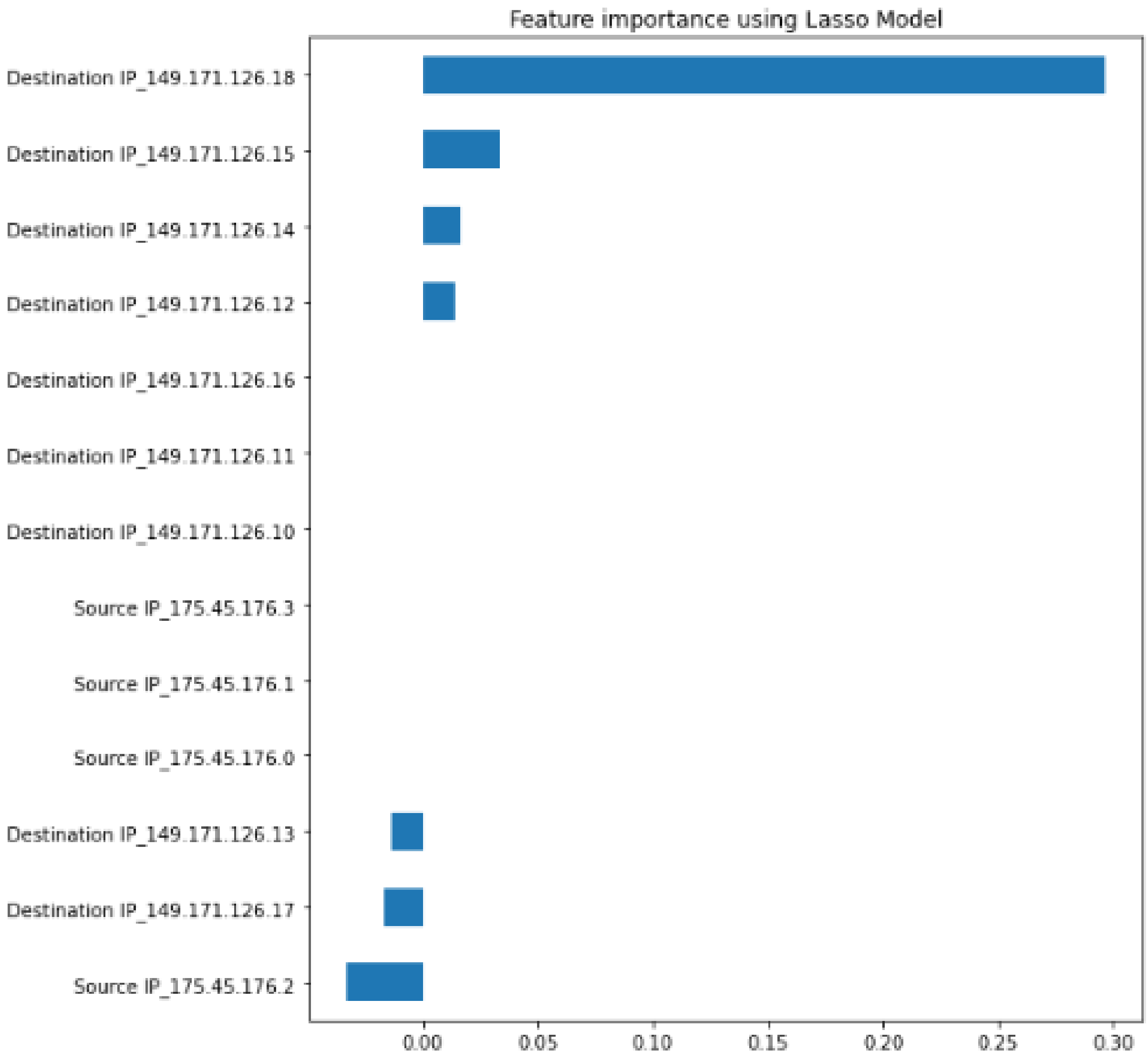
Correlation Heat map between dummy data



Result: From the heat map above, it is clear that there's a strong relation between "Generic" and "149.171.126.18" machine.

2. Embedded Method

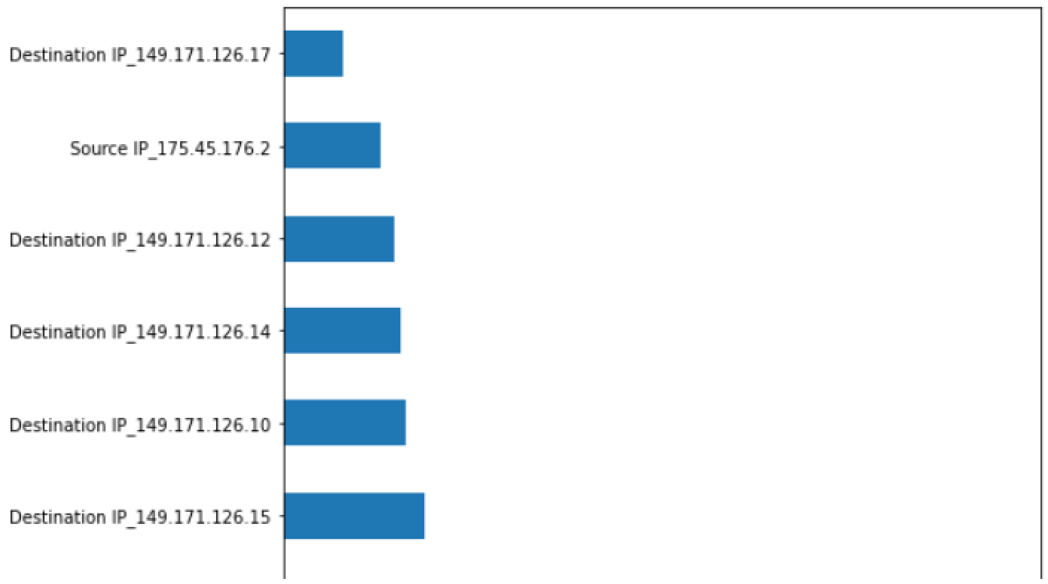
The features with coefficient = 0 are removed and the rest are taken. If the feature is irrelevant, lasso penalizes its coefficient and make it 0. Hence the features with coefficient = 0 are removed and the rest are taken.

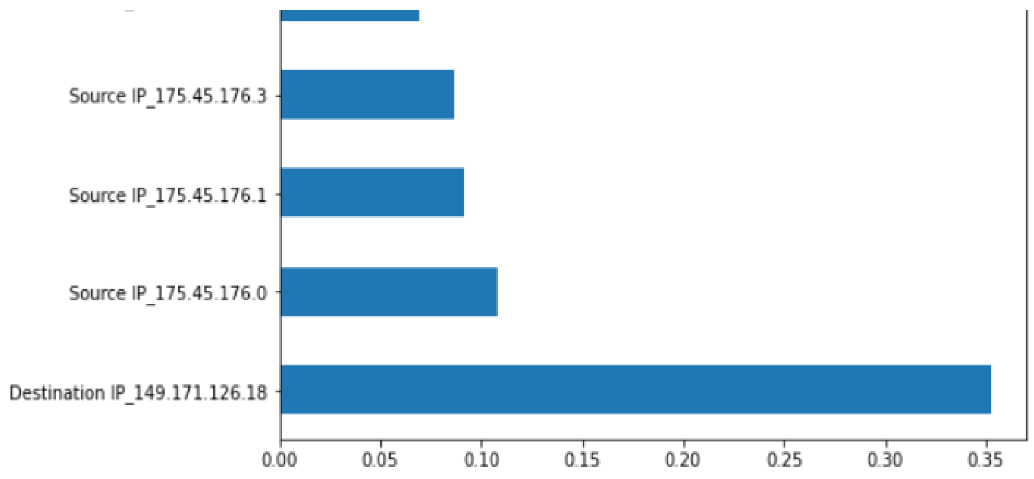


Result: From Embedded method, we select Destination IP "149.171.126.18" as one of the most important input variable that is related to output/target variable "Generic"

3. Feature importance

In order to get the best feature, we also implemented feature importance method. Feature importance gives you a score for each feature of your data, the higher the score more important or relevant is the feature towards your output variable.





Result: From Feature Importance method, we select Destination IP "149.171.126.18" as one of the most important input variable that is related to output/target variable "Generic"

Step 5: Model Creation and Selection

From Feature Engineering, we got that there's a strong relationship between Destination IP "149.171.126.18" and "Generic" type of attack which will be used and predicted further using 3 modeling methods because our data set is "Binary Classification" and assessing the same with Receiver Operating Characteristic (ROC) and Area under the curve (AUC).

We have already defined independent and dependent columns in previous step.

```
X = corelation_dummy.iloc[:,9:22] #independent columns

y = corelation_dummy.iloc[:,5] #target column i.e Generic
```

Splited the input and output variables as training and testing data

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=42)
```

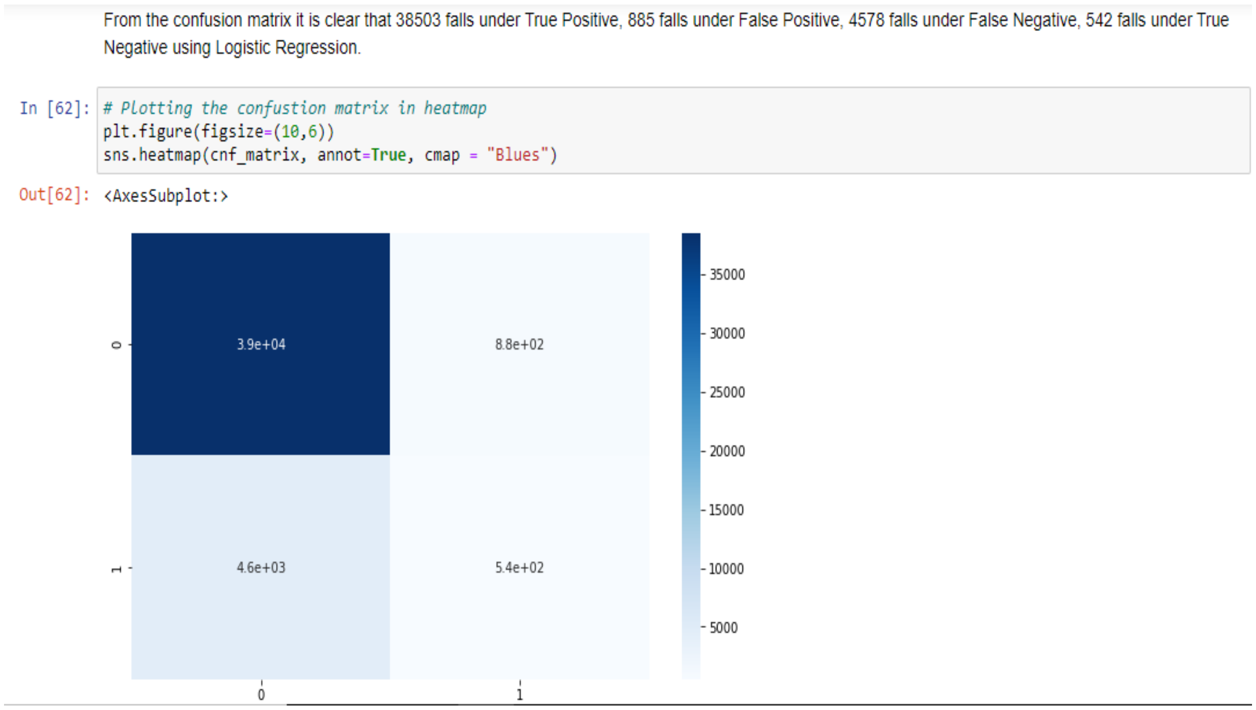
Displaying the shape of training and testing dataframes

```
display(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

(133523, 13)

(133523,)

(44508, 13)

(44508,)
```

1. Logistic Regression

Logistic regression is a statistical analysis method that uses independent variables to predict the dependent variable, just like Linear Regression, but with a difference that the dependent variable should be categorical variable. Independent variables can be numeric or categorical variables, but the dependent variable will always be categorical.

Creating confusion matrix of testing and predicting the data frames of target variable


From the confusion matrix it is clear that 38503 falls under True Positive, 885 falls under False Positive, 4578 falls under False Negative, 542 falls under True Negative using Logistic Regression.

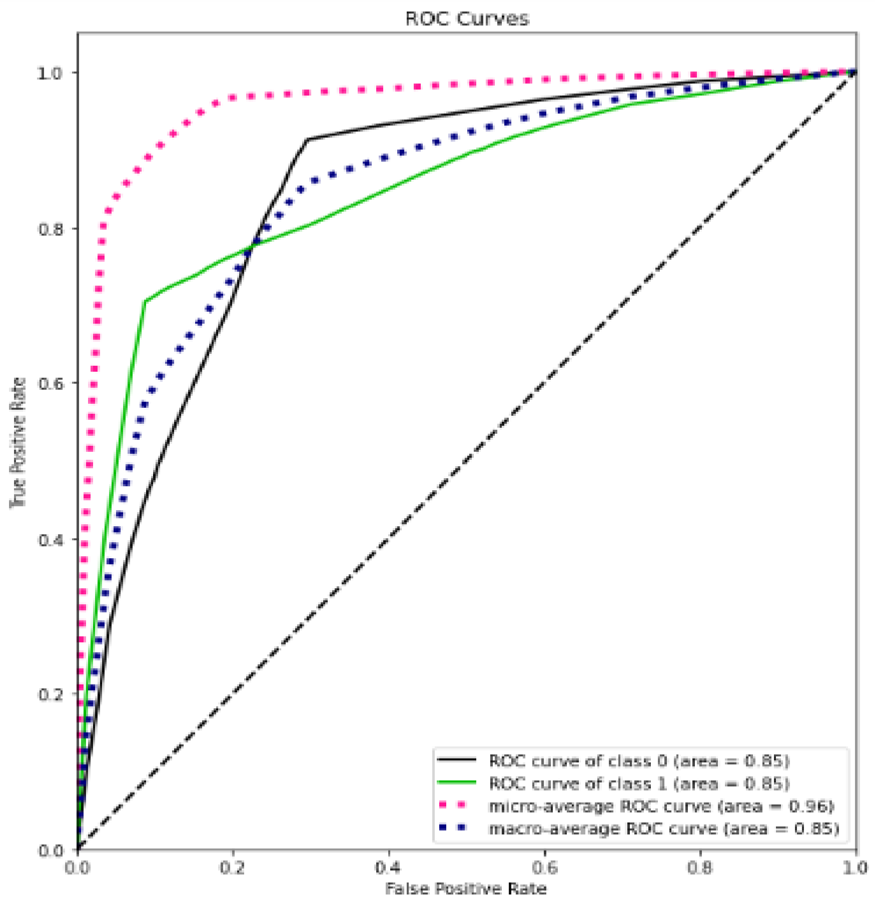
```
In [62]: # Plotting the confustion matrix in heatmap
         plt.figure(figsize=(10,6))
         sns.heatmap(cnf_matrix, annot=True, cmap = "Blues")

Out[62]: <AxesSubplot:>
```

Calculating Accuracy, Precision, Recall, F1, ROC AUC, and Confusion matrix of Logistic Regression model

```
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: %f' % accuracy)
precision = precision_score(y_test, y_pred, average='micro')
print('Precision: %f' % precision)
recall = recall_score(y_test, y_pred, average='micro')
print('Recall: %f' % recall)
f1 = f1_score(y_test, y_pred, average='micro')
print('F1 score: %f' % f1)
auc = roc_auc_score(y_test, y_pred, multi_class='ovr')
print('ROC AUC: %f' % auc)
print(cnf_matrix)
```

```
Accuracy: 0.877258
Precision: 0.877258
Recall: 0.877258
F1 score: 0.877258
ROC AUC: 0.541695
[[38503   885]
 [ 4578   542]]
```

Plotting the ROC curve i.e. a visual trade-offs between False Positive and True Positive rates



Explanation: As ROC curve is close to 1 and values on y axis are higher which states that it has higher number of True positive values therefore we can conclude that it has high accuracy (85%).

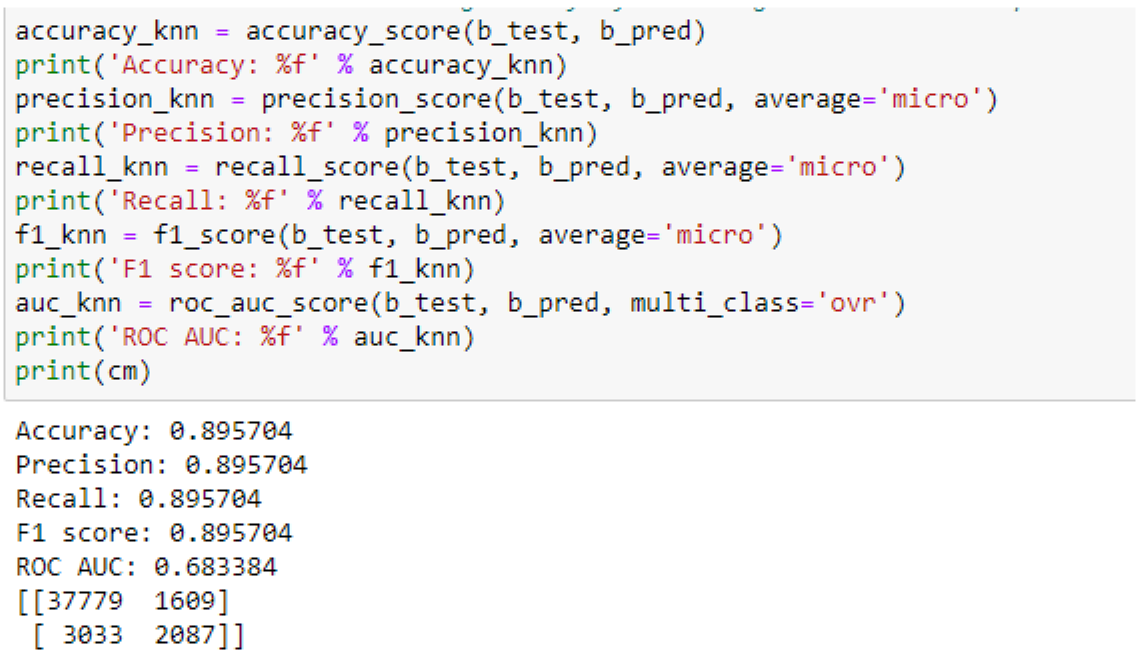
2. kNN method

K-nearest neighbors (kNN) is a supervised machine learning algorithm that can be used to solve both classification and regression tasks. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.

Making the Confusion Matrix

```
array([[37779,  1609],
       [ 3033,  2087]], dtype=int64)
```

From the confusion matrix it is clear that 37779 falls under True Positive, 1609 falls under False Positive, 3033 falls under False Negative, 2087 falls under True Negative using kNN method. Calculating Accuracy, Precision, Recall, F1, ROC AUC, and Confusion matrix of kNN method

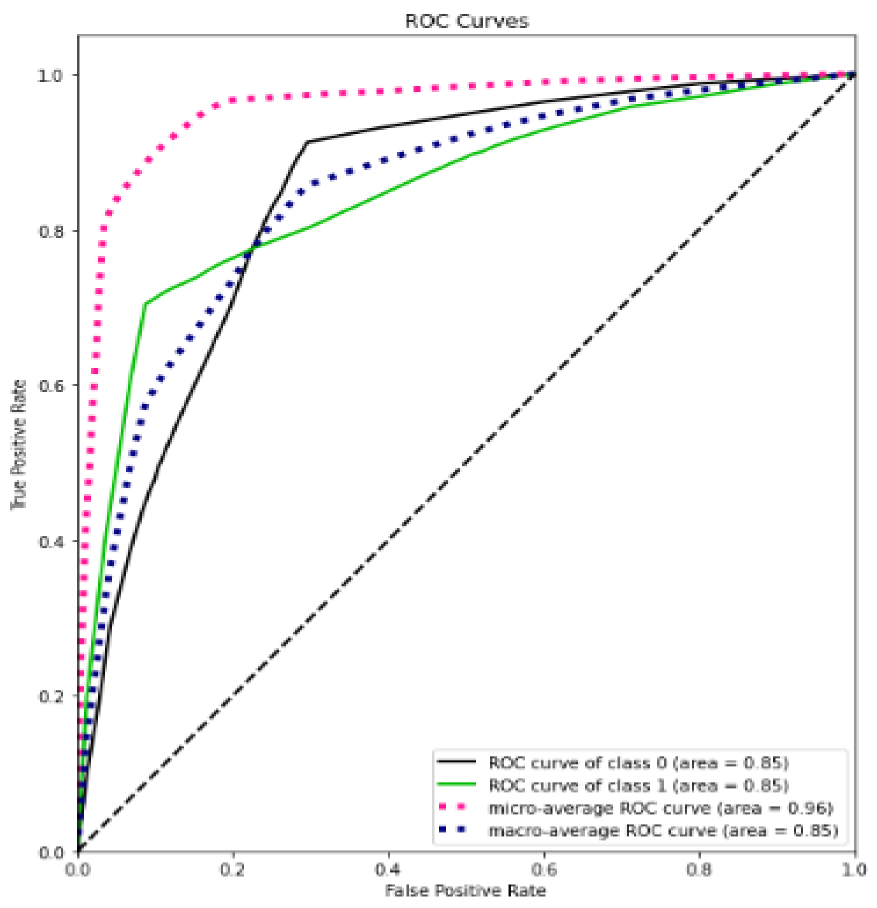


```python
accuracy_knn = accuracy_score(b_test, b_pred)
print('Accuracy: %f' % accuracy_knn)
precision_knn = precision_score(b_test, b_pred, average='micro')
print('Precision: %f' % precision_knn)
recall_knn = recall_score(b_test, b_pred, average='micro')
print('Recall: %f' % recall_knn)
f1_knn = f1_score(b_test, b_pred, average='micro')
print('F1 score: %f' % f1_knn)
auc_knn = roc_auc_score(b_test, b_pred, multi_class='ovr')
print('ROC AUC: %f' % auc_knn)
print(cm)
```

```
Accuracy: 0.895704
Precision: 0.895704
Recall: 0.895704
F1 score: 0.895704
ROC AUC: 0.683384
[[37779  1609]
 [ 3033  2087]]
```

Calculating accuracy for k from 1 to 30 and found that the best k is 18 for kNN method which is too high and computationally expensive.

Plotting the ROC curve i.e. a visual trade-offs between False Positive and True Positive rates



Explanation: As ROC curve is close to 1 and values on y axis are higher which states that it has higher number of True positive values therefore we can conclude that it has high accuracy.

3. Decision tree

In decision tree, each node specifies a test on an attribute, each branch descending from that node corresponds to one of the possible values from that attribute. We used Decision tree because it handles non-linear data sets effectively and as the k value is too high we planned to use this algorithm for our prediction.

Implementing decision tree classification and fitting the input and output variables in the tree

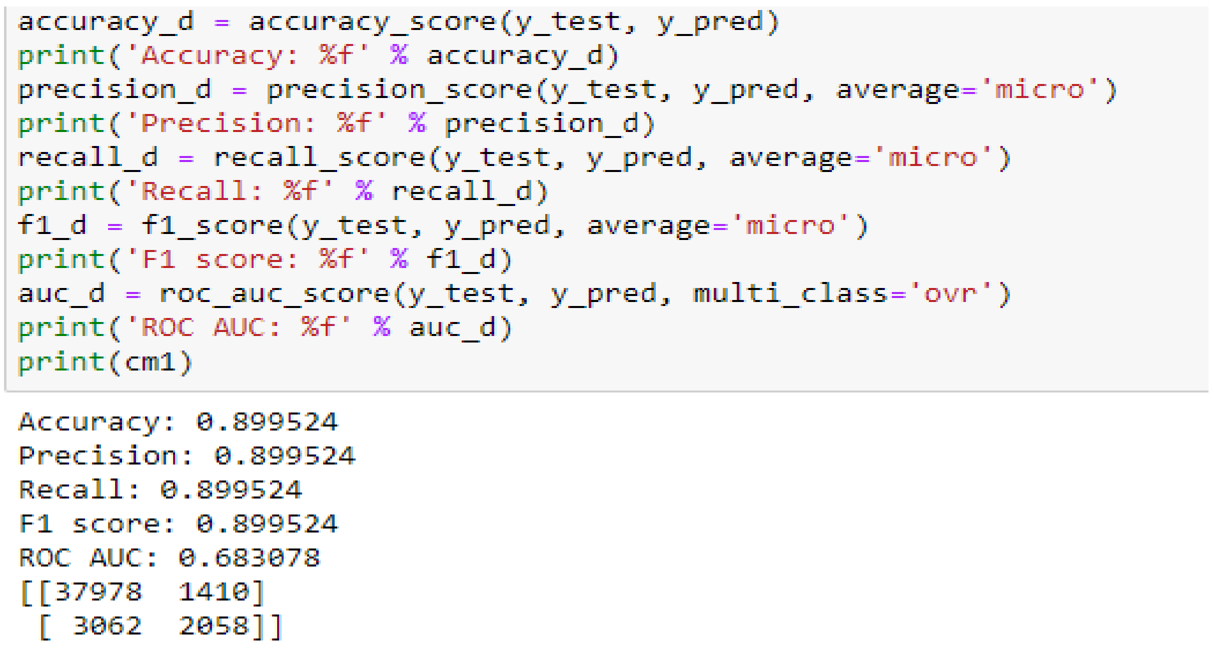
Creating confusion matrix of target variable

array([[37978, 1410],

[3062, 2058]], dtype=int64)

From the confusion matrix it is clear that 37978 falls under True Positive, 1410 falls under False Positive, 3062 falls under False Negative, 2058 falls under True Negative using Decision tree classification method.

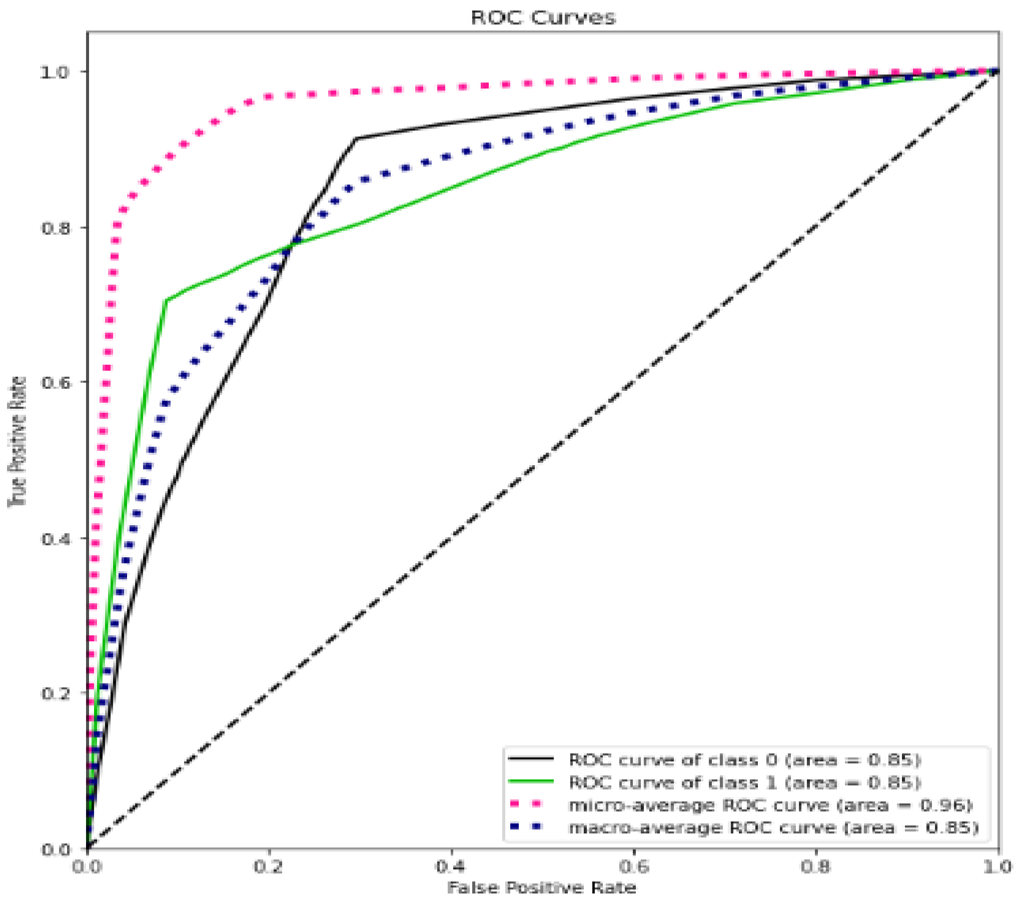
Calculating Accuracy, Precision, Recall, F1, ROC AUC, and Confusion matrix of Decision tree classification model



```
accuracy_d = accuracy_score(y_test, y_pred)
print('Accuracy: %f' % accuracy_d)
precision_d = precision_score(y_test, y_pred, average='micro')
print('Precision: %f' % precision_d)
recall_d = recall_score(y_test, y_pred, average='micro')
print('Recall: %f' % recall_d)
f1_d = f1_score(y_test, y_pred, average='micro')
print('F1 score: %f' % f1_d)
auc_d = roc_auc_score(y_test, y_pred, multi_class='ovr')
print('ROC AUC: %f' % auc_d)
print(cm1)
```

```
Accuracy: 0.899524
Precision: 0.899524
Recall: 0.899524
F1 score: 0.899524
ROC AUC: 0.683078
[[37978  1410]
 [ 3062  2058]]
```

Plotting the ROC curve i.e. a visual trade-offs between False Positive and True Positive rates



Explanation: As ROC curve is close to 1 and values on y axis are higher which states that it has higher number of True positive values therefore we can conclude that it has high accuracy.

Step 6: Model Assessment

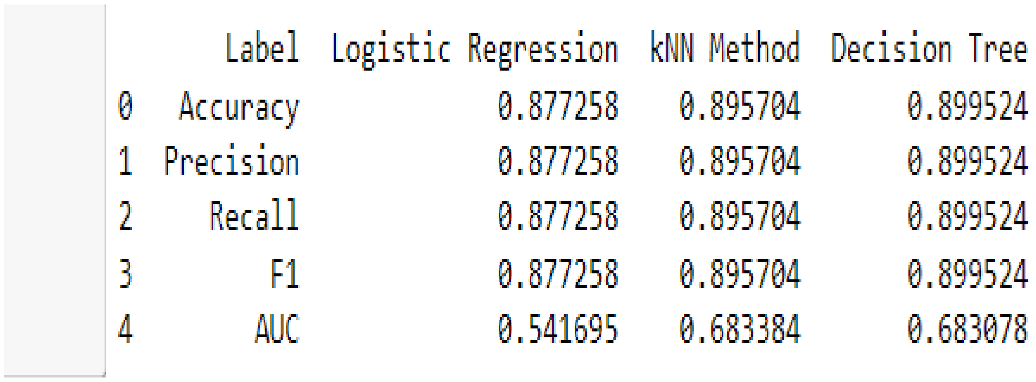
Confusion Matric Comparison from all the models created and selected

Out[76]:

	Logistic Regression	kNN Method	Decision Tree
0	[38503, 885]	[37779, 1609]	[37978, 1410]
1	[4578, 542]	[3033, 2087]	[3062, 2058]

From the confusion matrix, we can see that Decision Tree has the highest True Positive compared to others. We also take other aspects below for our assessment i.e. Accuracy, Precision, Recall, F1 and AUC for getting the best model.

Assessment



```
       Label  Logistic Regression  kNN Method  Decision Tree
0   Accuracy             0.877258    0.895704       0.899524
1  Precision             0.877258    0.895704       0.899524
2     Recall             0.877258    0.895704       0.899524
3         F1             0.877258    0.895704       0.899524
4        AUC             0.541695    0.683384       0.683078
```

Conclusion:

To sum up, we can predict "Generic" type of attack can be most vulnerable to destination machines as we received accuracy of 85%-90% using Logistic regression, kNN method and Decision tree algorithms. The most precise prediction was found to be in Decision tree algorithm with almost 90% of accuracy and precision. We can also, generate model for most vulnerable machines which has the highest correlation (0.47) between source and destination IP, but because we wanted to focus on the type of Attack that can possibly happen in future - we didn't created model based on source and destination IP.