Configuration Manual

MSc Research Project
MSc in Cybersecurity

Ajay Ashok Kumbhar
Student ID: 21138222

School of Computing
National College of Ireland

Supervisor: Dr Arghir-Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ajay Ashok Kumbhar		
Student ID:	21138222		
Programme:	MSc in Cybersecurity	**Year:**	2022
Module:	.........MSc Research Project...………		
Supervisor:	...... Dr Arghir-Nicolae Moldovan ....………		
Submission Due Date:	.................15-Dec-2022..………		
Project Title:	End-to-end attack detection based on ML and spark...........………		
Word Count:	...782.................**Page Count**.................12...............................……...		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ....Ajay Kumbhar...

Date: ....15 Dec 2022...

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ajay Ashok Kumbhar
X21138222

1 Introduction

The required procedure and configuration process has been mentioned in the configuration Manual. Using below methodology we can implement and perform the testing of the model.

2 System Specification

The required configuration in system for the proposed model.

- Required operation system: Windows 11
- Processor: Intel i7
- Hard Drive: 1.5 TB SSD
- RAM: 24 GB DD4 Ram
- Language used: spark and python

3 Tools and Used software:

Below required languages has been installed on the local machine.

- Python is installed on the local machine and the python version is 3.10.2.

```
C:\Users\Ajay>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

Fig .1 Version of python

- Java is installed and added the entries in the environment variable to run spark.

Java has version 1.8.0 in our machine.

```
C:\Users\Ajay>java -version
java version "1.8.0_351"
Java(TM) SE Runtime Environment (build 1.8.0_351-b10)
Java HotSpot(TM) Client VM (build 25.351-b10, mixed mode)
```

Fig 2. Java Version

- Pyspark

The most important factor in this proposal is Apache spark. I installed the apache spark in the local system which will help to process high data.
To install the Spark, we have to setup the JAVA and Python in the local machine. Once both languages are installed in the machine next step is to set the environment variable path for JAVA, python and SPARK.

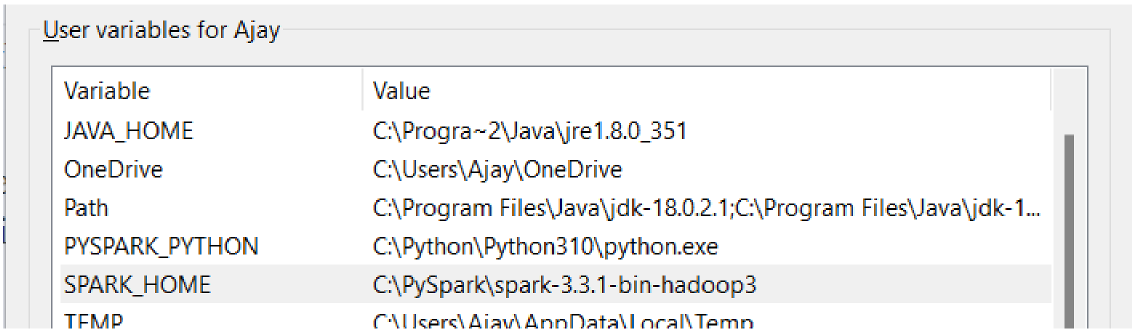

Fig 3. SPARK and PYSPARK path

Above image display the required path for the pyspark and the python to run on the local machine.

To confirm Pyspark is installed in the system need to run pyspark in CMD which will automatically run using the python.

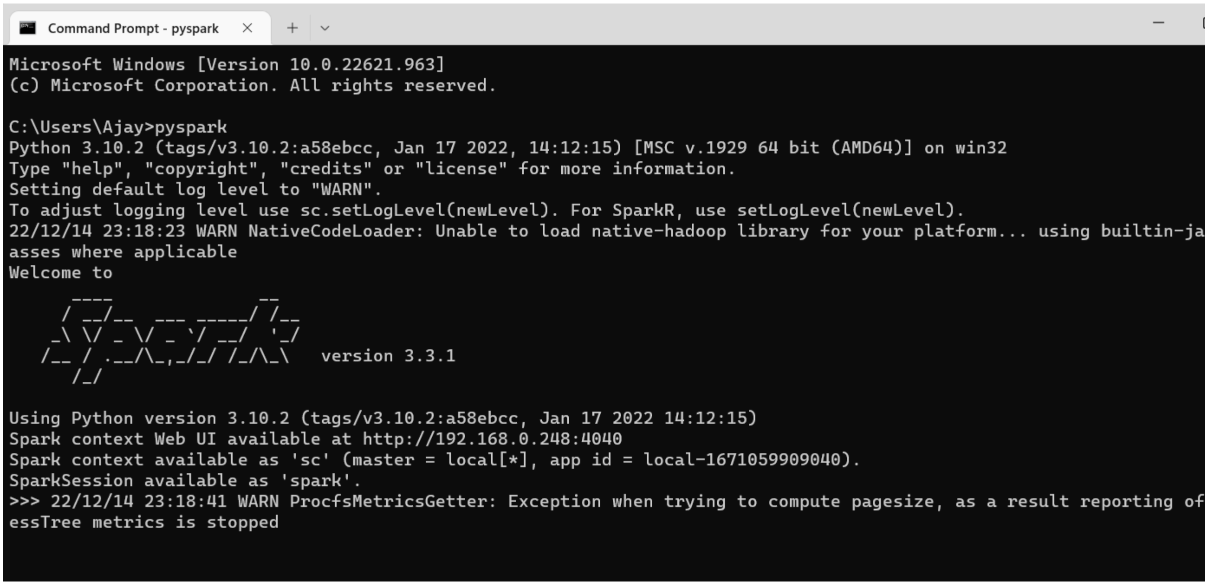

Fig 3. Pyspark Installed

After successful installation of the spaark we can verify is the server is running or not. Below images shows the localhost URL which displays the SPARK computing process.

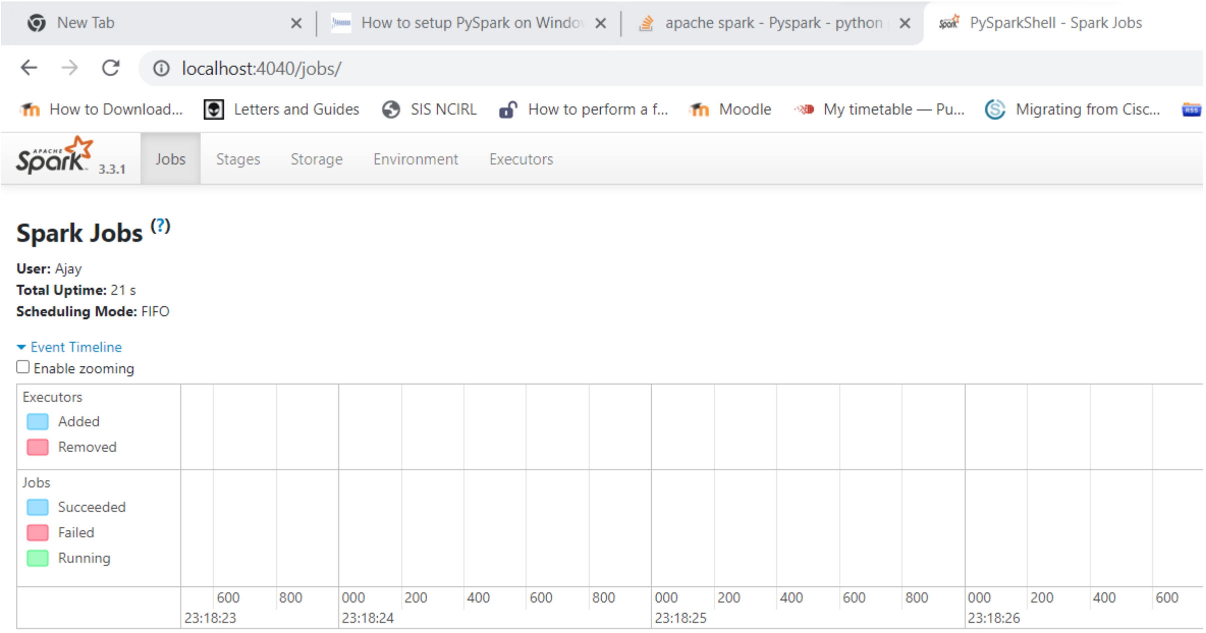

Fig 4. Spark localhost GUI

- To write the I used the open-source platform which is Jupyter noteboon and visual studio code.

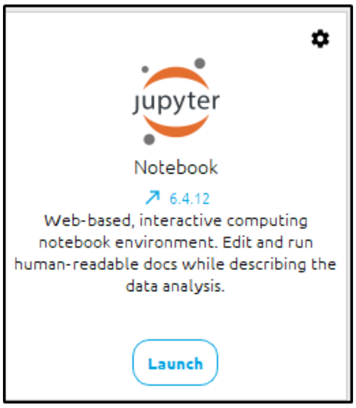

Fig 5. Jupyter version

4 Implementation model

Researched the latest dataset in the online platform which could provide better accuracy of the system.
Step 1: Downloaded the latest dataset from the open-source platform.

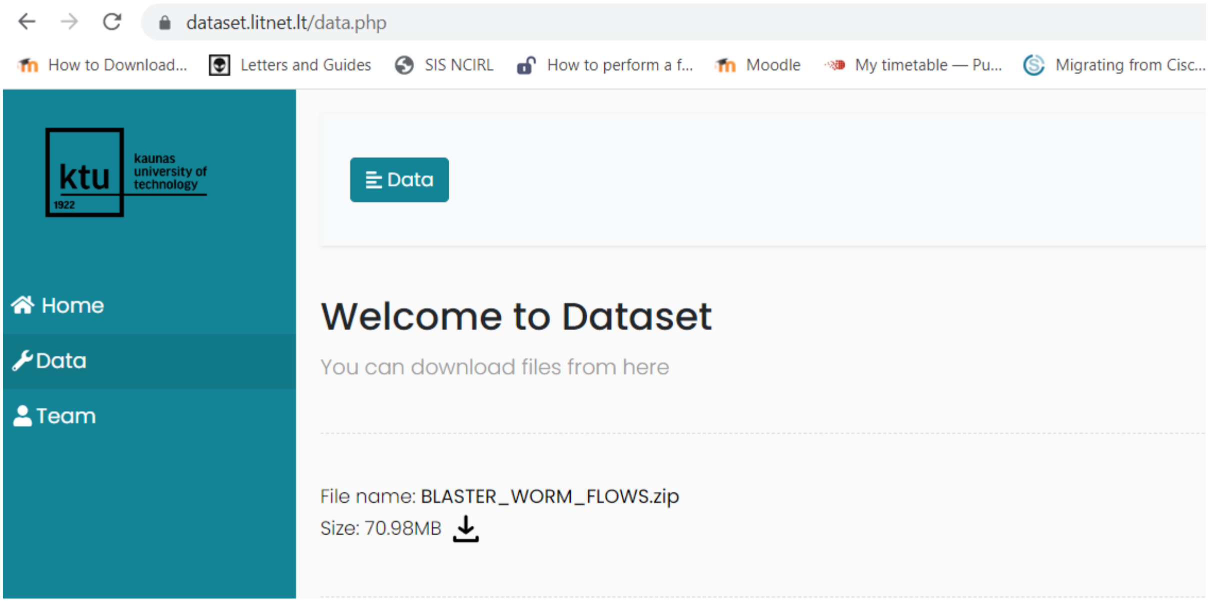

Fig 6. Open-source dataset

Step 2: After downloading the dataset the next step is to install anaconda and Jupyter notebook on the local machine to run the code. After successfully installation of anaconda and Jupyter notebook we ran anaconda prompt and website is opened to write the code.

Step 2: The next step is to import the libraries required for the model. Required libraries are the Numpy, pandas and findspark which checks the spark functionality is working or not.

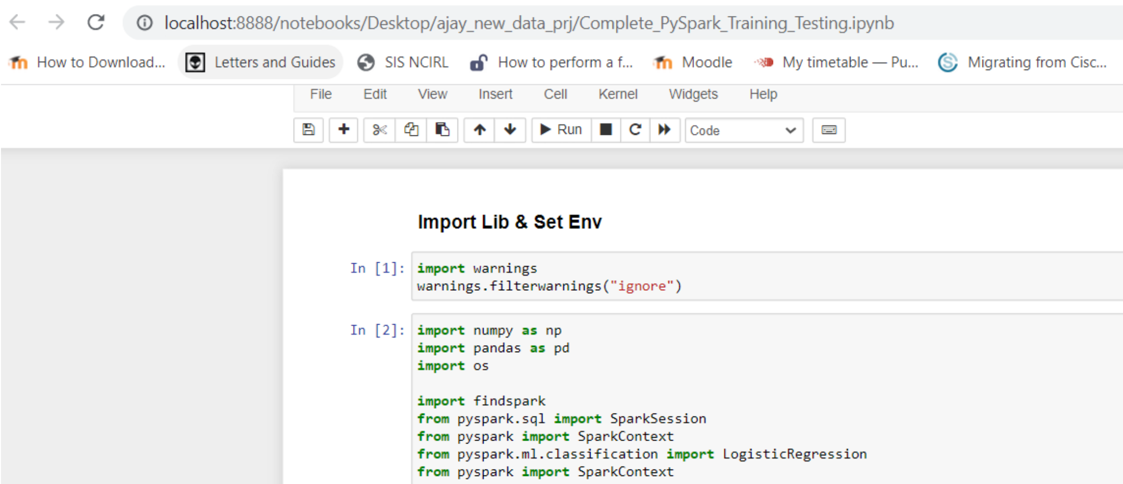

Fig 7. Imported libraries

Step 3: The next stpes was to work with worker node of the spark functionality. SparkContext was helped to interact with the cluster and help to create the required values such as accumulator, broadcast and RDD variable. The below syntax help to do this job.

```
In [4]: sc = SparkContext.getOrCreate()
```

```
In [8]: df = spark.read.csv('LITNET_2020.csv', inferSchema = True, header = True)
        df.show(5)
```

In [9]:
```
df.printSchema()
```

```
root
 |-- ID: integer (nullable = true)
 |-- ts_year: integer (nullable = true)
 |-- ts_month: integer (nullable = true)
 |-- ts_day: integer (nullable = true)
 |-- ts_hour: integer (nullable = true)
 |-- ts_min: integer (nullable = true)
 |-- ts_second: integer (nullable = true)
 |-- te_year: integer (nullable = true)
 |-- te_month: integer (nullable = true)
 |-- te_day: integer (nullable = true)
 |-- te_hour: integer (nullable = true)
 |-- te_min: integer (nullable = true)
 |-- te_second: integer (nullable = true)
 |-- td: double (nullable = true)
 |-- sa: string (nullable = true)
 |-- da: string (nullable = true)
 |-- sp: integer (nullable = true)
```

Fig 8. Imported dataset

Above images shows the uploaded dataset using the spark and the variables which are present in the dataset. Printschema display the available fields in the dataset.

Step 4: The next step is to preprocess the data. In this, I verfied the available features and removed the unwanted features from the dataset with the help of the df.drop.

Data Preprocessing

In [10]:
```
df = pd.read_csv('LITNET_2020.csv')
```

In [11]:
```
df.isna().sum()
```

Out[11]:
```
ID               0
ts_year          0
ts_month         0
ts_day           0
ts_hour          0
                ..
udp_p_r_range    0
p_range_dst      0
udp_src_p_0      0
attack_t         0
attack_a         0
Length: 85, dtype: int64
```

In [12]:
```
df.head()
```

Out[12]:

	ID	ts_year	ts_month	ts_day	ts_hour	ts_min	ts_second	te_year	te_month	te_day	...	tcp_src_tftp	tcp_src_kerb	tcp
0	78	2020	1	31	15	34	58	2020	1	31	...	blank	blank	
1	141	2020	1	31	15	34	58	2020	1	31	...	blank	blank	
2	147	2020	1	31	15	34	58	2020	1	31	...	blank	blank	
3	170	2020	1	31	15	34	58	2020	1	31	...	blank	blank	
4	225	2020	1	31	15	34	58	2020	1	31	...	blank	blank	

5 rows × 85 columns

```
In [14]: lh=df.drop(['ID', 'ts_year', 'ts_month', 'ts_day', 'ts_hour', 'ts_min', 'ts_second',
                'te_year', 'te_month', 'te_day', 'te_hour', 'te_min', 'te_second', 'td',
                'sa', 'da','fwd', 'stos','smk', 'dmk', 'dtos', '_dir', 'nh', 'nhb',
                'svln', 'dvln', 'ismc', 'odmc', 'idmc', 'osmc', 'mpls1', 'mpls2',
                'mpls3', 'mpls4', 'mpls5', 'mpls6', 'mpls7', 'mpls8', 'mpls9', 'mpls10',
                'cl', 'sl', 'al', 'ra', 'eng', 'exid', 'tr','icmp_dst_ip_b', 'icmp_src_ip', 'udp_dst_p', 'tcp_f_s', 'tcp_f_n_a',
                'tcp_f_n_f', 'tcp_f_n_r', 'tcp_f_n_p', 'tcp_f_n_u', 'tcp_dst_p',
                'tcp_src_dst_f_s', 'tcp_src_tftp', 'tcp_src_kerb', 'tcp_src_rpc',
                'tcp_dst_p_src', 'smtp_dst', 'udp_p_r_range', 'p_range_dst',
                'udp_src_p_0'], axis=1)
```

Fig 9. Perfomred data preprocessing

Step 5: After successful verifying the dataset, I recongised that only below features are essentilas and other features are not having better values.

While checking the data in excel these fetures only have usefull info

```
In [16]: lh.columns

Out[16]: Index(['sp', 'dp', 'pr', '_flag1', '_flag2', '_flag3', '_flag4', '_flag5',
                '_flag6', 'ipkt', 'ibyt', 'opkt', 'obyt', '_in', 'out', 'sas', 'das',
                'attack_t', 'attack_a'],
               dtype='object')

In [17]: a = lh[['sp', 'dp', 'pr', '_flag1', '_flag2', '_flag3', '_flag4', '_flag5',
                '_flag6', 'ipkt', 'ibyt', 'opkt', 'obyt', '_in', 'out', 'sas', 'das',
                'attack_t', 'attack_a']]
```

Fig 10. Verified and mentioned required features.

Step 6: Requested model to display dataset information and observed the some features was having categorical values which needs to be convert in the numerical form.

```
In [24]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21713 entries, 0 to 21712
Data columns (total 19 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sp        21713 non-null  int64
 1   dp        21713 non-null  int64
 2   pr        21713 non-null  object
 3   _flag1    21713 non-null  object
 4   _flag2    21713 non-null  object
 5   _flag3    21713 non-null  object
 6   _flag4    21713 non-null  object
 7   _flag5    21713 non-null  object
 8   _flag6    21713 non-null  object
 9   ipkt      21713 non-null  int64
 10  ibyt      21713 non-null  int64
 11  opkt      21713 non-null  int64
 12  obyt      21713 non-null  int64
 13  _in       21713 non-null  int64
 14  out       21713 non-null  int64
 15  sas       21713 non-null  int64
 16  das       21713 non-null  int64
 17  attack_t  21713 non-null  object
 18  attack_a  21713 non-null  int64
dtypes: int64(11), object(8)
memory usage: 3.1+ MB
```

Fig 11. Displyed categorical values

Step 7: Converted values in numerical form to understand the machine.

Convert Categry into Numerical

```
In [26]: LE = LabelEncoder()

In [27]: for i in c_name:
             df[i] = LE.fit_transform(df[i])

In [28]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21713 entries, 0 to 21712
Data columns (total 19 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sp        21713 non-null  int64
 1   dp        21713 non-null  int64
 2   pr        21713 non-null  int32
 3   _flag1    21713 non-null  int32
 4   _flag2    21713 non-null  int32
 5   _flag3    21713 non-null  int32
 6   _flag4    21713 non-null  int32
 7   _flag5    21713 non-null  int32
 8   _flag6    21713 non-null  int32
 9   ipkt      21713 non-null  int64
 10  ibyt      21713 non-null  int64
 11  opkt      21713 non-null  int64
 12  obyt      21713 non-null  int64
 13  _in       21713 non-null  int64
 14  out       21713 non-null  int64
 15  sas       21713 non-null  int64
 16  das       21713 non-null  int64
 17  attack_t  21713 non-null  int32
 18  attack_a  21713 non-null  int64
dtypes: int32(8), int64(11)
memory usage: 2.5 MB
```

Fig 12. Converted values to numerical

Step 8: After visualisation of dataset, I recongnzed the dataset is unbalanced and need to balance. To balance the dataset SMOTE library has been used and KNN which balance the dataset basis in closed neighbor values.

Balancing the Data

In [30]:
```
from imblearn import under_sampling, over_sampling
```

Installed the module imblearn in anaconda command prompt and then installed imblearn package.
https://stackoverflow.com/questions/50376990/modulenotfounderror-no-module-named-imblearn

In [31]:
```
# Need to install imblearn lib
import sklearn.utils._cython_blas
from imblearn.over_sampling import SMOTE
```

In [32]:
```
over_sampler = SMOTE(k_neighbors=2)
```

In [33]:
```
X = df.iloc[:, 0:-1]
y = df.iloc[:, -1]
```

In [34]:
```
X_res, y_res = over_sampler.fit_resample(X, y)
```

In [35]:
```
print(f"Training target statistics: {y_res.shape}")
print(f"Testing target statistics: {y.shape}")
```

```
Training target statistics: (36126,)
Testing target statistics: (21713,)
```

In [36]:
```
X_res['Class2'] = y_res.values
```

In [37]:
```
X_res.to_csv("Cleaned_Data.csv",index=False)
```

#End of Dataframe we cleaned and balance the data

Fig 13. Balanced data

Step 9: After transferring code to spark session performed the splitting of the data and performed machine algorithm on the cleaned data. Total three algorithm has been implemented which are decision tree, random forest and binary classification to detect the attack.

In [46]:
```
#split the data
train, test = final_data.randomSplit([0.6,0.4])
```

In [47]:
```
train
```

Out[47]:
```
DataFrame[features: vector, Class2: int]
```

In [48]:
```
#Build Logistic model
lr = LogisticRegression(labelCol="Class2",featuresCol="features")
```

In [49]:
```
binary_lr_model=lr.fit(train)
predict_train=binary_lr_model.transform(train)
predict_test=binary_lr_model.transform(test)
predict_test.select("Class2","prediction").show(10)
```

Fig 14. Binary classification

Decison Tree

In [55]:
```
from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import StringIndexer, VectorIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.util import MLUtils


# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = final_data.randomSplit([0.7, 0.3])

# Train a DecisionTree model.
dt = DecisionTreeClassifier(labelCol="Class2", featuresCol="features")

# Train model.
model = dt.fit(trainingData)

# Make predictions.
predictions = model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "Class2", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(
    labelCol="Class2", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
```

Fig 15. Decision Tree

Random Forest

In [60]:
```
from pyspark.ml.feature import StringIndexer, VectorIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.util import MLUtils
# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = final_data.randomSplit([0.7, 0.3])

# Train a DecisionTree model.
# dt = DecisionTreeClassifier(labelCol="Class2", featuresCol="features")

from pyspark.ml.classification import RandomForestClassifier

rf_model = RandomForestClassifier(featuresCol = 'features', labelCol = 'Class2')

# Train model.
rf_model = rf_model.fit(trainingData)

# Make predictions.
predictions = rf_model.transform(testData)

# Select example rows to display.
predictions.select("prediction", "Class2", "features").show(5)

# Select (prediction, true label) and compute test error
evaluator = MulticlassClassificationEvaluator(
    labelCol="Class2", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)

y_true = predictions.select(['Class2']).collect()
y_pred = predictions.select(['prediction']).collect()

print(classification_report(y_true, y_pred))
sns.heatmap(confusion_matrix(y_true, y_pred),annot=True)

print("Accuracy ",int(accuracy_score(y_true, y_pred)*100),"%")
```

Fig 16. Random Forest

Step 10: Socket session has been created in the server and scripted python file at the client will transfer the file to server using this socket. Once session receive the file it get predicted as per the machine learning algorithm and provides the output.

```
import socket
import ftplib
import time
from datetime import datetime

print("-----------------------------------")

ip_address = socket.gethostbyname(socket.gethostname())
port = 5002

print("[STARTED]  > Server running at : ", ip_address, " ", port)

print()

s = socket.socket()
s.bind((ip_address, port))
s.listen(5)

print("[LISTENING] > Waiting for connection ..")

while True:
    c, addr = s.accept()
    client_ip = addr[0]
    print()
    print('[CONNECTED]  > Connection got from ' + str(client_ip))
    print()

    msg = c.recv(1024)
    msg = msg.decode("utf-8")
    print("[MESSAGE RECEIVED] > ", msg)

    msg = "Hello... send the packet"
    c.send(msg.encode("utf-8"))
    print("[MESSAGE SENT] > ", msg)
    print('')
```

```
    msg = c.recv(20480)
    print("[MESSAGE RECEIVED] >  file writing")
    print(" ")
    file_name = 'in_folder/test.xlsx'
    f = open(file_name, 'wb')
    print(file_name)
    f.write(msg)
    f.close()
    time.sleep(40)

    print("[MODEL PREDICTION] > Predicting...")

    result = model_testing(file_name)
    date = datetime.today()

    if result == 'Normal':
        insert_into_excel(date, ip_address, 'NO', '-')
    else:
        insert_into_excel(date, ip_address, 'Yes', result)

    print("[MODEL PREDICTED RESULT] > ", result)

    c.close()
    print('-----------------------------------------')

    print("[LISTENING] Waiting for new connection ..")
```

```
-------------------------------------
[STARTED]  > Server running at :  192.168.0.248   5002

[LISTENING] > Waiting for connection ..
```

Fig 17. Socket opened at server side

Step 11: Multiple client will use the python script to share the traffic to the central server to predict is there any attack occurred on the device or not. Below is the python script which help to get connected with server.

```
# ip = "192.168.1.118"

ip = input("Enter ip number :")
port = 5002


class Client:
    def __init__(self):
        self.content = None
        self.ip = ip
        self.port = port

    def send_file(self, p_name):
        file = "test_files/" + p_name
        with open(file, 'rb') as f:
            self.content = f.read()
        return self.content

    def connect(self):
        try:
            s = socket.socket()
            s.connect((ip, port))
            msg = input("Type here >")
            s.send(msg.encode("utf-8"))

            msg = s.recv(1024)
            msg = msg.decode("utf-8")
            print("[MESSAGE RECEIVED] ", msg)

            packet_name = input("Enter packet name : ")
            content = self.send_file(packet_name)
            s.send(content)
            s.close()

        except Exception as e:
            print("[ERROR] Opps something went wrong, check below error message")
            print("[ERROR MESSAGE] ", e)
```

Fig 18. Client python script