**CSE-3024 Web Mining**

**Lab Assignment 8**

**Alokam Nikhitha**

**19BCE2555**

**Experiment 8**

Aim

Using a Decision Tree Classifier, divide the given network intrusion dataset into normal and abnormal categories. Along with the classification, the following items must be printed:

* Confusion Matrix
* Accuracy of model on Test data
* Decision Tree visualization.

**Dataset Used:** The network intrusion dataset from Kaggle.

Link to which is: <https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?select=Train_data.csv>

**Procedure:**

- First, we import the necessary numpy, pandas, matplotlib, and tree libraries.

- The dataset is then imported into our workspace. The set of independent and dependent attributes is also defined.

- Next, we used a 7.5:2.5 ratio to divide the dataset into training and test sets.

- Then, using DecisionTreeClassifier from sklearn.tree, we train our decision tree model.

- Next, we look for the test set results that our model anticipated.

- Then, using the expected and test set findings, we print our confusion matrix.

- Similarly, we print the model's accuracy based on the test set and anticipated result.

- Finally, we visualise our model using the sklearn tree.

**Code:**

#19BCE2555

#Importing libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn import tree

#Importing dataset

dataset = pd.read\_csv("Train\_data.csv")

X = dataset.iloc[:, 4:41].values

y = dataset.iloc[:, -1].values

#Splitting the dataset

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=0)

#Fitting our model

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy' ,random\_state = 0)

classifier.fit (X\_train, y\_train)

#Predicting the Test set Results

y\_pred = classifier.predict(X\_test)

#Printing the confusion matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

#Printing the accuracy of our model

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y\_test, y\_pred)

print(accuracy)

#Defining the labels of our dataset

classes = ["Anamoly", "Normal"]

#Printing the visualized decision tree

fig = plt.figure(figsize=(25,20))

\_ = tree.plot\_tree(classifier,

feature\_names=dataset.columns,

class\_names=classes,

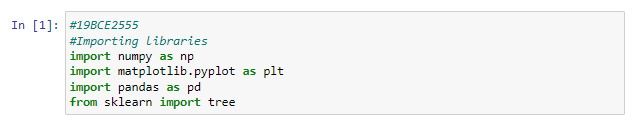
filled=True)

#Printing the feature wise break points of our decision tree

test\_representation = tree.export\_text(classifier)

print(test\_representation)

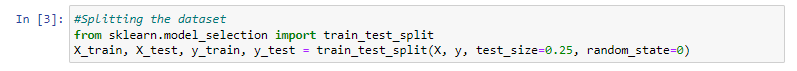
**Code Snippets and Outputs:**



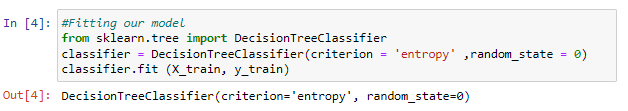
We're importing our libraries right now. Nupmy is imported as np, pandas is imported as pd, matplotlib's pyplot extension is imported as plt, and finally tree is imported from sklearn.



We're using pandas to import our Network Intrusion Dataset into our workspace. Then a set of dependent and independent qualities is defined. The set of independent qualities is labelled X, while the set of dependent attributes is labelled y.



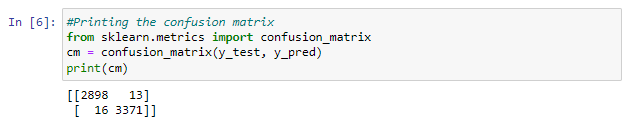
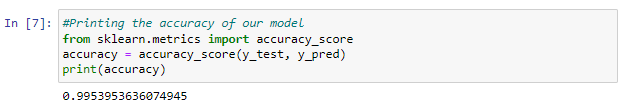
We're going to divide our dataset into two parts: a training set and a test set. We're going to maintain 25% of the dataset in the test set and 75% in the training set.



We're taking data from the training set to train our model. For our decision tree classifier, we employed "entropy" as the deciding factor.



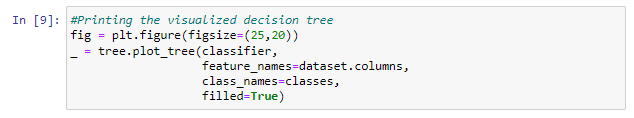
We're collecting our anticipated test set results from the classifier and saving them in the y pred variable.

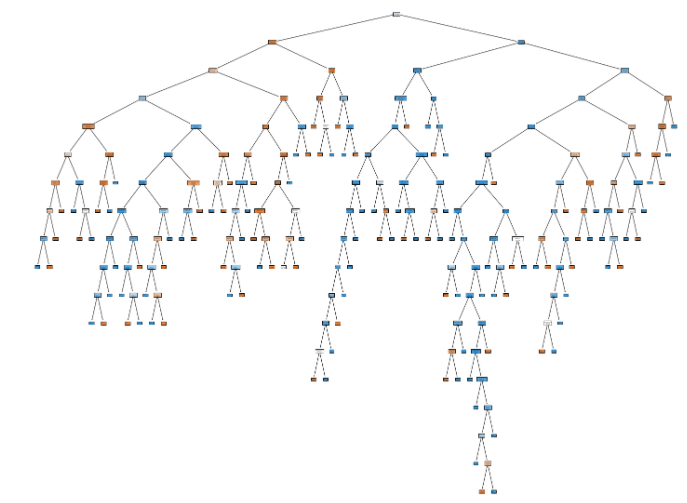
 

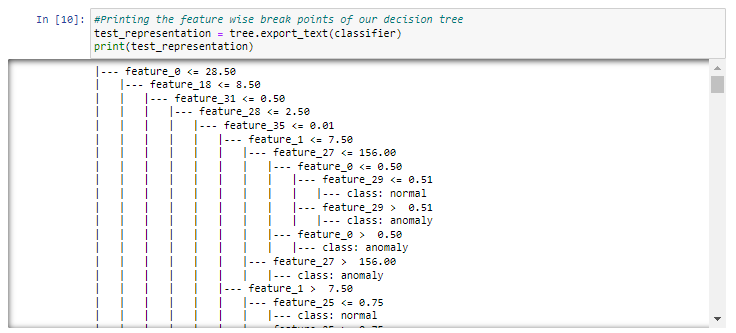
The confusion matrix and accuracy of our decision tree classifier are printed here. Our model's accuracy with the test dataset is 99.53953636 percent.



We're using sklearn's tree library to visualise our decision tree.







The categorization criterion of our decision tree is presented here. We can see that feature 0 is our classifier's root node, followed by multiple middle nodes.

**Results and Output**

Confusion Matrix:



This is our confusion matrix.

True Negatives: 3488

True Positives: 4032

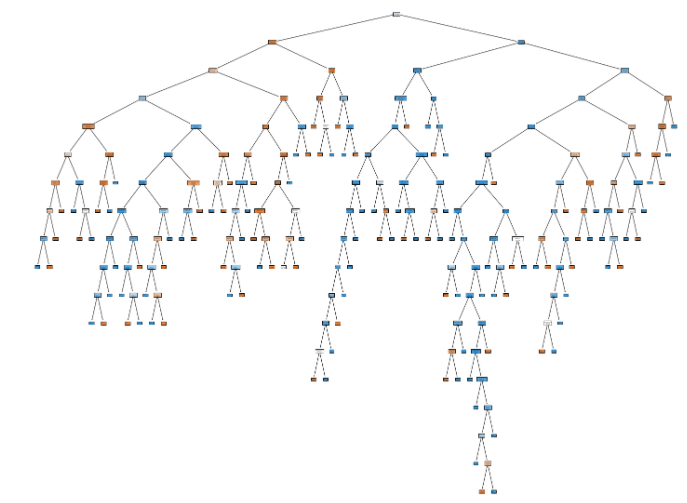
False Positives: 17

False Negatives: 21

Accuracy:

The accuracy of our model stands at 99.49%

Decision Tree Visualization:



Classification Points:

