Harshit Mishra (19BCE0799)

CSE-3024 WEB MINING

LAB ASSIGNMENT 9

Aim: To verify the performance of decision tree with change in hyper-parameters

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:

- Firstly, we import the necessary libraries of numpy, pandas, matplotlib and tree.
- Next, we import the dataset into our workspace. We also define the set of independent and dependent attribute.
- We use only the first 500 rows of our dataset in to get a better visualisation of small tree.
- Next, we split the dataset into training set and test set using a ratio of 7:3.
- Then we train our decision tree model using DecisionTreeClassifier from sklearn.tree
- Here we specify the criteria to entropy.
- Next, we find the test set results as predicted by our model.
- Then, we print the accuracy of our model using test set result and predicted result.
- We also train a decision tree from sklearn.tree and this time, we don’t use any hyper-parameters because, by default the criteria is set to gini.
- Again, we print the accuracy score of this classifier after predicting the y_pred variable from X_test result set.
- Finally, using the tree of sklearn, we visualize both the classifiers and check the difference in their spatial structure.

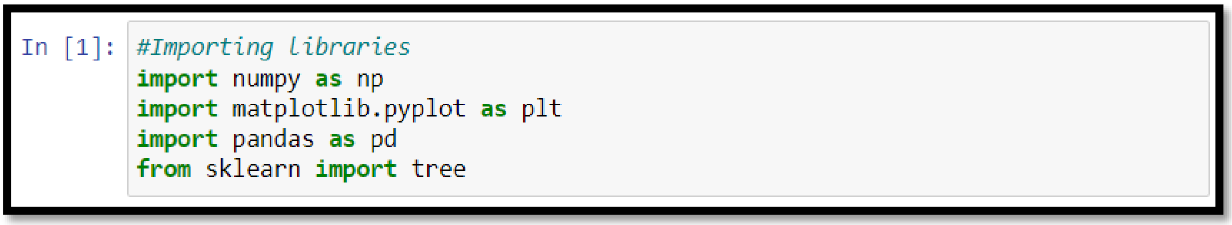
Harshit Mishra (19BCE0799)

Code:

```
#Importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import tree


#Importing datasets
dataset = pd.read_csv('Train_data.csv')
X = dataset.iloc[1:500, 4:41].values
y = dataset.iloc[1:500, -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=0.3, random_state=0)


#Fitting the classifier with entropy criteria
from sklearn.tree import DecisionTreeClassifier
classifier_1 = DecisionTreeClassifier(criterion="entropy", random_state=0)
classifier_1.fit(X_train, y_train)


#Test set predictions
y_pred_1 = classifier_1.predict(X_test)


#Accuracy score
from sklearn.metrics import accuracy_score
accuracy_1 = accuracy_score(y_test, y_pred_1)
print("Accuracy with entropy criteria is: ", accuracy_1)


#Fitting the classifier with gini criteria
classifier_2 = DecisionTreeClassifier(random_state=0)
classifier_2.fit(X_train, y_train)


#Test set prediction
y_pred_2 = classifier_2.predict(X_test)


#Accuracy Score
accuracy_2 = accuracy_score(y_test, y_pred_2)
print("Accuracy with gini criteria is: ", accuracy_2)
```

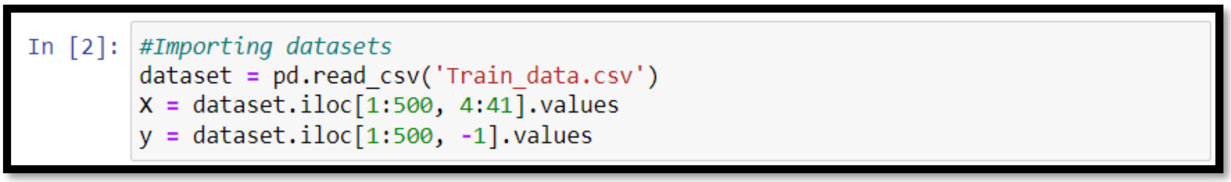
```
#Printing the decision tree with entropy
fig = plt.figure(figsize = (25,20))
_ = tree.plot_tree(classifier_1,
            feature_names= dataset.columns,
            class_names = ["anamoly", "normal"],
            filled =True)


#Printing the decision tree with gini
fig = plt.figure(figsize = (25,20))
_ = tree.plot_tree(classifier_2,
            feature_names= dataset.columns,
            class_names = ["anamoly", "normal"],
            filled =True)
```

Code Snippet and Outputs:



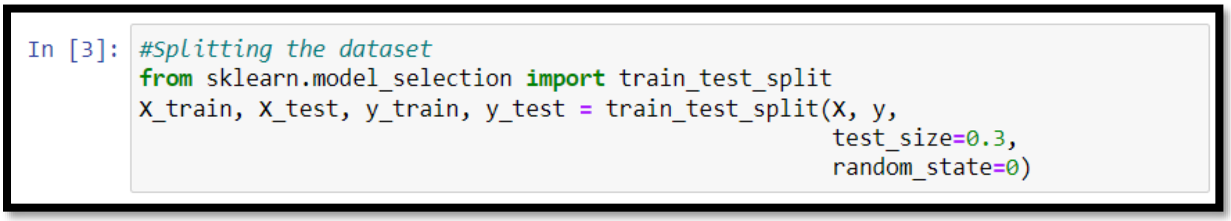
```
In [1]: #Importing libraries
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn import tree
```

Here we are importing our libraries. We import nupmy as np, pandas as pd, matplotlib's pyplot extension as plt and finally we import tree from sklearn.



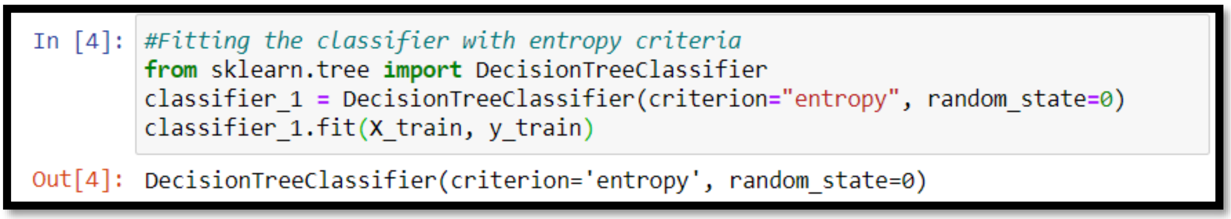
```
In [2]: #Importing datasets
        dataset = pd.read_csv('Train_data.csv')
        X = dataset.iloc[1:500, 4:41].values
        y = dataset.iloc[1:500, -1].values
```

Here we are importing our Network Intrusion Dataset into our workspace using pandas. Then we are defining set of dependent and independent attributes. Set of independent attributes are labelled X and dependent ones are labelled y. We use only the first 500 rows of our dataset to visualize the tree better.



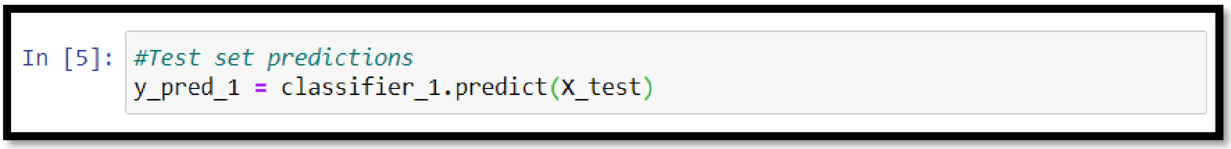
```
In [3]: #Splitting the dataset
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                            test_size=0.3,
                                                            random_state=0)
```

Here we are splitting our dataset into training set and test set. We are keeping 30% of the dataset into test set and 70% of it in training set.



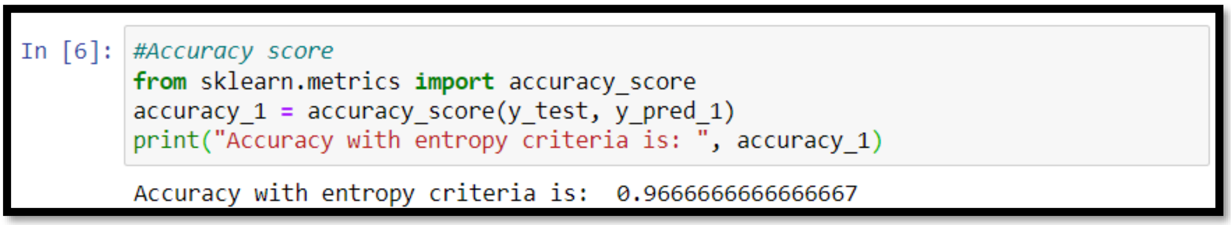
```
In [4]: #Fitting the classifier with entropy criteria
        from sklearn.tree import DecisionTreeClassifier
        classifier_1 = DecisionTreeClassifier(criterion="entropy", random_state=0)
        classifier_1.fit(X_train, y_train)

Out[4]: DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Here we are training our model using training set data. We have used "entropy" as the decisive criteria for our decision tree classifier.



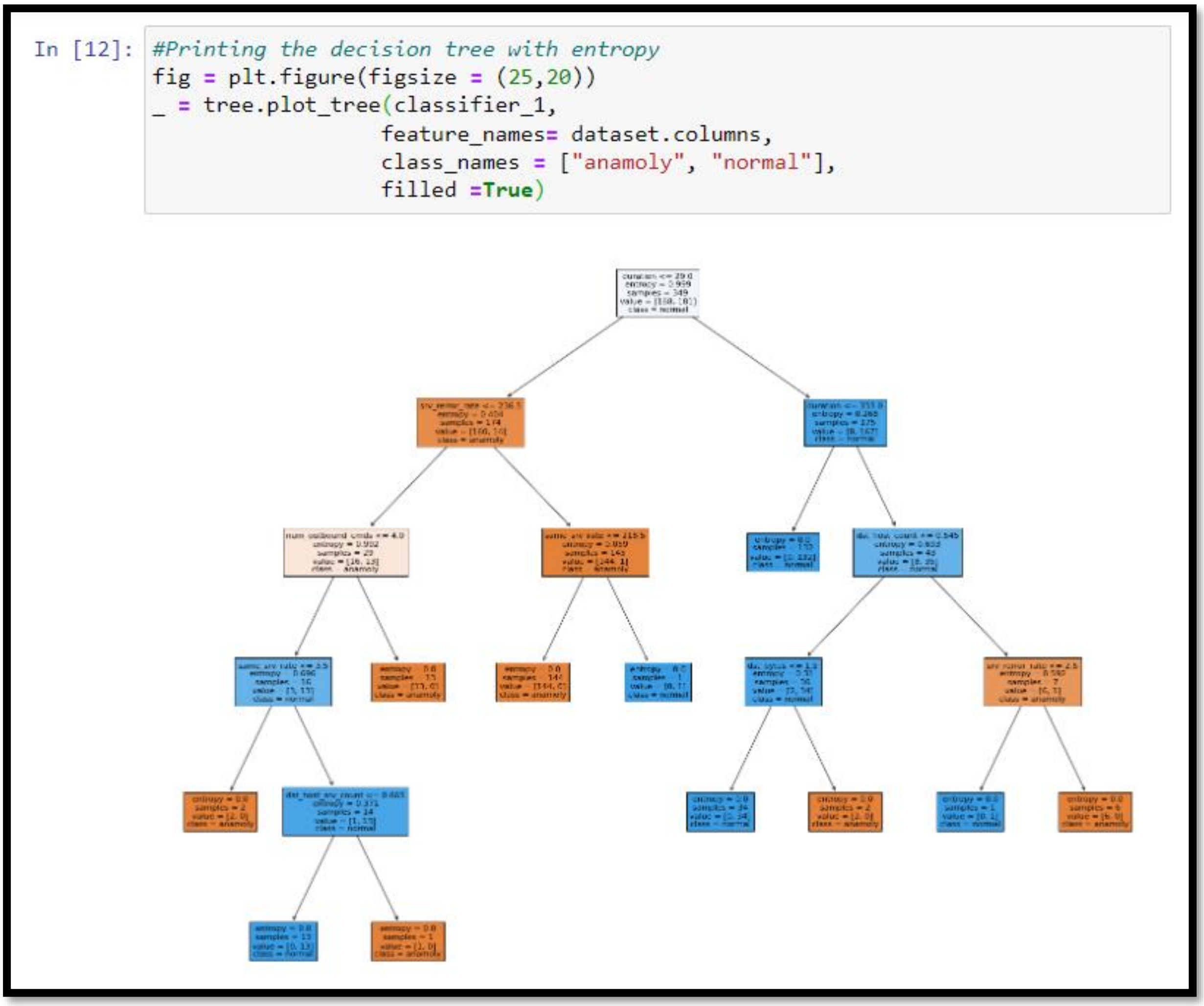
```
In [5]: #Test set predictions
        y_pred_1 = classifier_1.predict(X_test)
```

Here we are getting our predicted results of test set from the classifier and then are storing it in y_pred variable.



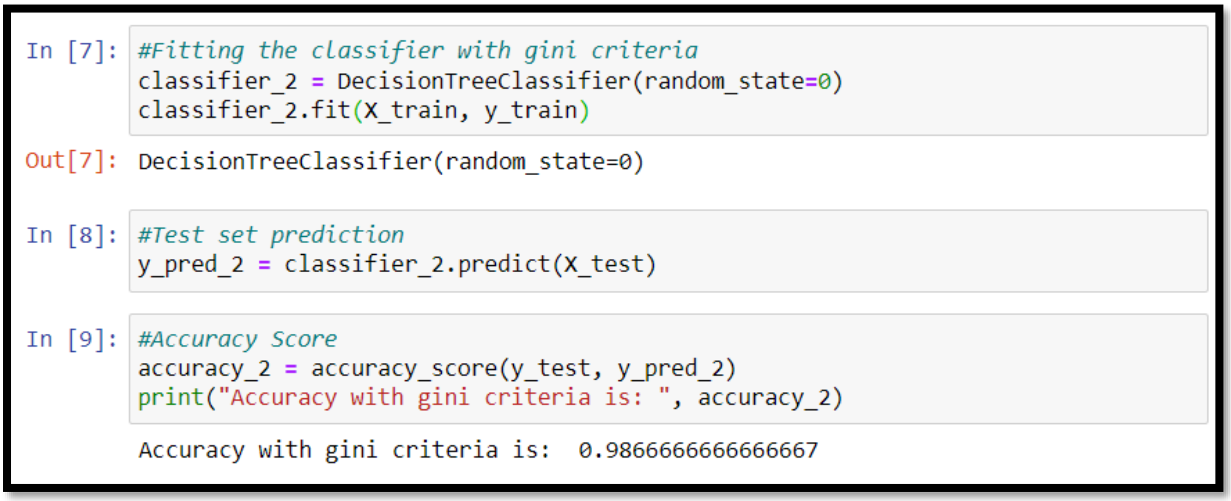
```
In [6]: #Accuracy score
        from sklearn.metrics import accuracy_score
        accuracy_1 = accuracy_score(y_test, y_pred_1)
        print("Accuracy with entropy criteria is: ", accuracy_1)

        Accuracy with entropy criteria is:  0.9666666666666667
```

Here we are printing the accuracy score of our model with entropy as splitting criteria. We can see that the accuracy is 96.67%.



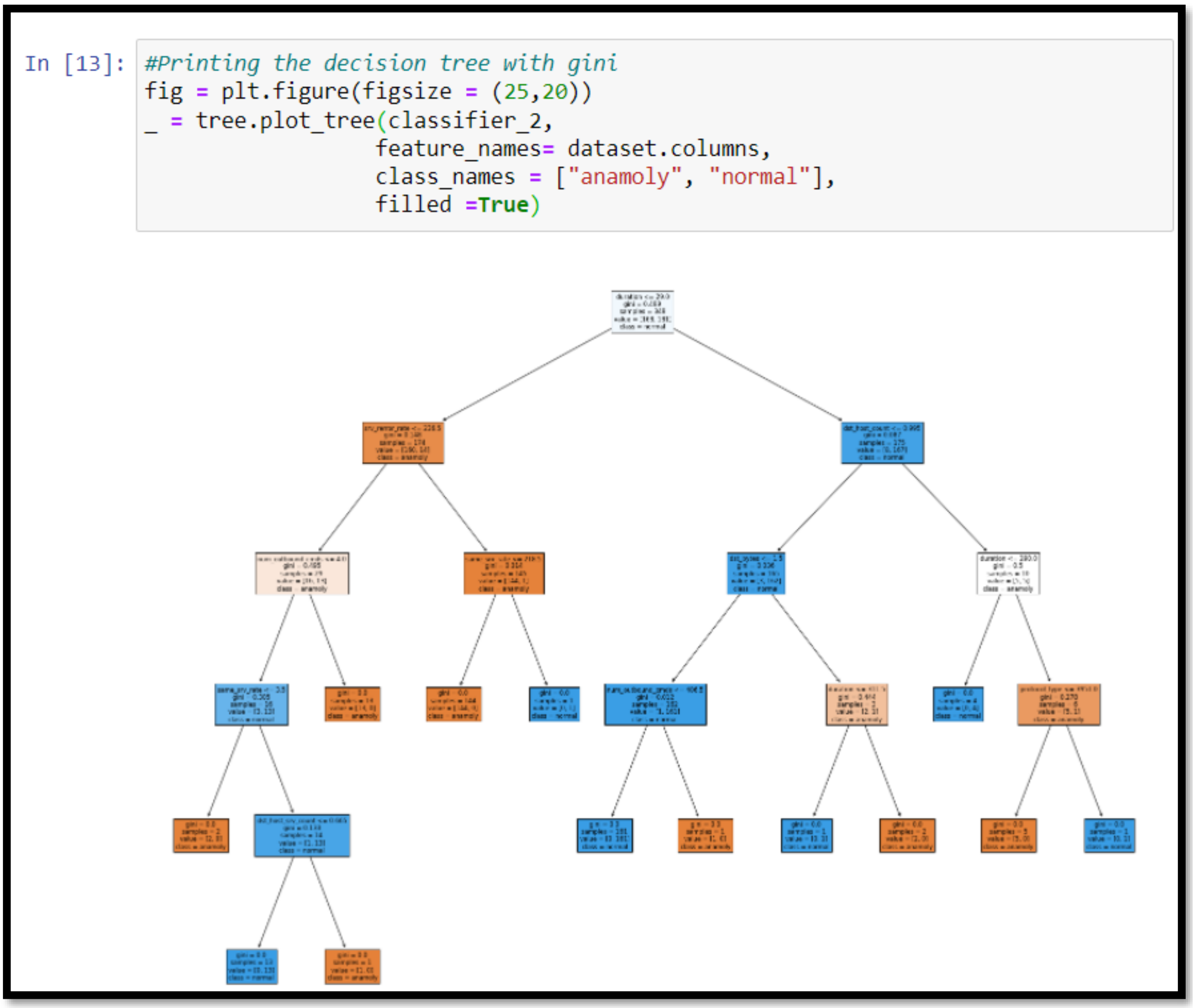
```
In [12]: #Printing the decision tree with entropy
         fig = plt.figure(figsize = (25,20))
         _ = tree.plot_tree(classifier_1,
                            feature_names= dataset.columns,
                            class_names = ["anamoly", "normal"],
                            filled =True)
```

Here we are visualizing our decision tree using sklearn’s tree library



```
In [7]: #Fitting the classifier with gini criteria
        classifier_2 = DecisionTreeClassifier(random_state=0)
        classifier_2.fit(X_train, y_train)

Out[7]: DecisionTreeClassifier(random_state=0)

In [8]: #Test set prediction
        y_pred_2 = classifier_2.predict(X_test)

In [9]: #Accuracy Score
        accuracy_2 = accuracy_score(y_test, y_pred_2)
        print("Accuracy with gini criteria is: ", accuracy_2)

        Accuracy with gini criteria is:  0.9866666666666667
```

Similar to entropy criteria, here we are printing the accuracy score with gini as our criteria for splitting the attributes. We can see that the accuracy here is 98.66%



```
In [13]: #Printing the decision tree with gini
         fig = plt.figure(figsize = (25,20))
         _ = tree.plot_tree(classifier_2,
                            feature_names= dataset.columns,
                            class_names = ["anamoly", "normal"],
                            filled =True)
```

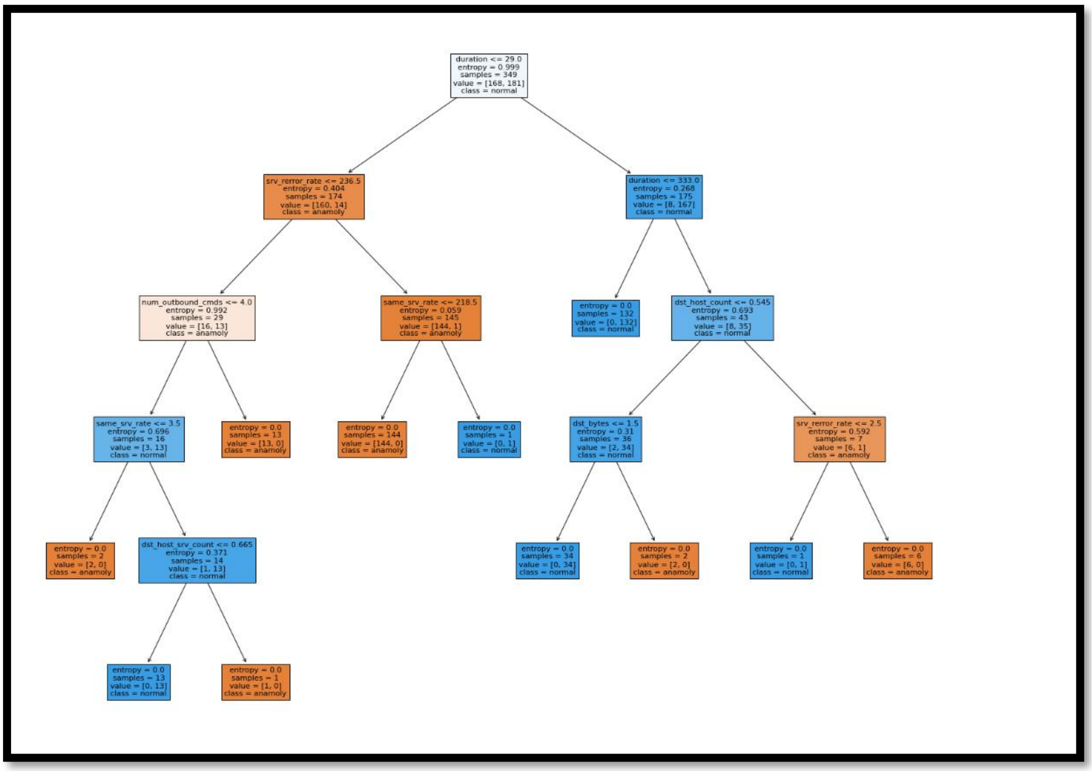
Here we are again visualising our decision tree, but with Gini criteria to see the spatial structure and its difference with entropy-based tree.

Results:

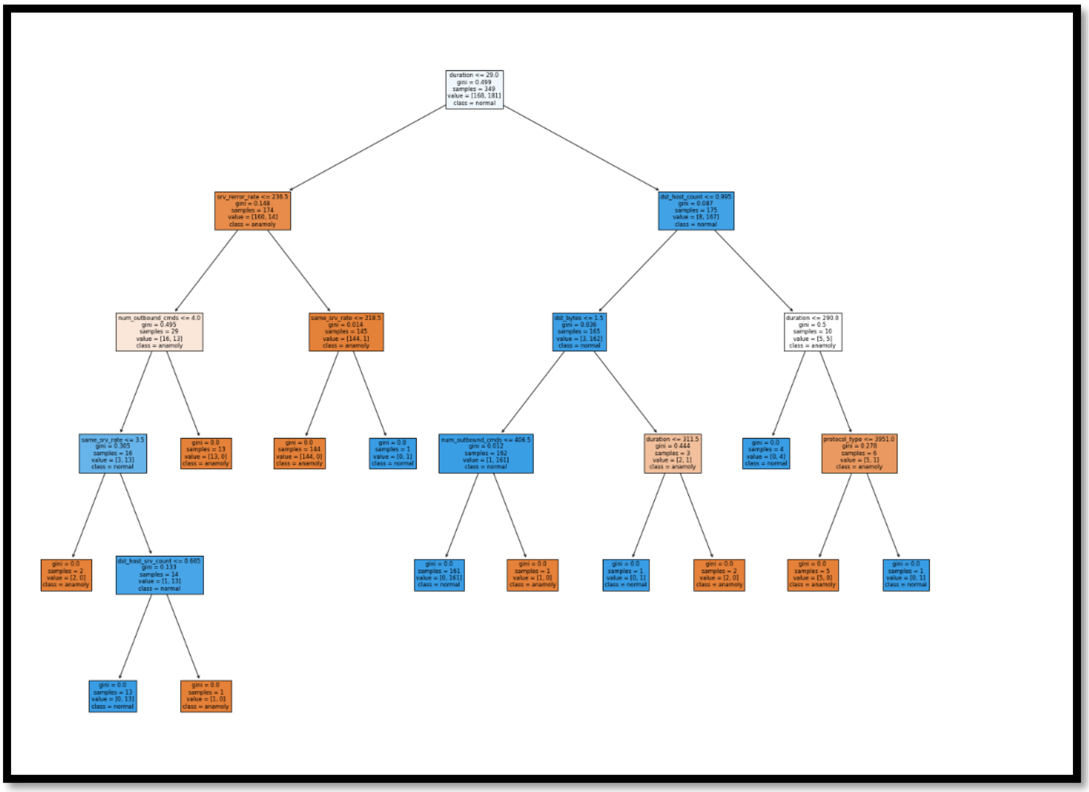
Accuracy with Entropy = 96.66%

Accuracy with Gini = 98.66%

Decision Tree with Entropy:


duration <= 29.0 entropy = 0.999 samples = 349 value = [168, 181] class = normal
srv_rerror_rate <= 236.5 entropy = 0.404 samples = 174 value = [160, 14] class = anamoly
duration <= 333.0 entropy = 0.268 samples = 175 value = [8, 167] class = normal
num_outbound_cmds <= 4.0 entropy = 0.992 samples = 29 value = [16, 13] class = anamoly
same_srv_rate <= 218.5 entropy = 0.059 samples = 145 value = [144, 1] class = anamoly
entropy = 0.0 samples = 132 value = [0, 132] class = normal
dst_host_count <= 0.545 entropy = 0.693 samples = 43 value = [8, 35] class = normal
same_srv_rate <= 3.5 entropy = 0.696 samples = 16 value = [3, 13] class = normal
entropy = 0.0 samples = 13 value = [13, 0] class = anamoly
entropy = 0.0 samples = 144 value = [144, 0] class = anamoly
entropy = 0.0 samples = 1 value = [0, 1] class = normal
dst_bytes <= 1.5 entropy = 0.31 samples = 36 value = [2, 34] class = normal
srv_rerror_rate <= 2.5 entropy = 0.592 samples = 7 value = [6, 1] class = anamoly
entropy = 0.0 samples = 2 value = [2, 0] class = anamoly
dst_host_srv_count <= 0.665 entropy = 0.371 samples = 14 value = [1, 13] class = normal
entropy = 0.0 samples = 34 value = [0, 34] class = normal
entropy = 0.0 samples = 2 value = [2, 0] class = anamoly
entropy = 0.0 samples = 1 value = [0, 1] class = normal
entropy = 0.0 samples = 6 value = [6, 0] class = anamoly
entropy = 0.0 samples = 13 value = [0, 13] class = normal
entropy = 0.0 samples = 1 value = [1, 0] class = anamoly

Decision Tree with Gini:


duration <= 29.0
gini = 0.499
samples = 349
value = [168, 181]
class = normal
srv_rerror_rate <= 236.5
gini = 0.148
samples = 174
value = [160, 14]
class = anamoly
dst_host_count <= 0.995
gini = 0.087
samples = 175
value = [8, 167]
class = normal
num_outbound_cmds <= 4.0
gini = 0.495
samples = 29
value = [16, 13]
class = anamoly
same_srv_rate <= 218.5
gini = 0.014
samples = 145
value = [144, 1]
class = anamoly
dst_bytes <= 1.5
gini = 0.036
samples = 165
value = [3, 162]
class = normal
duration <= 290.0
gini = 0.5
samples = 10
value = [5, 5]
class = anamoly
same_srv_rate <= 3.5
gini = 0.305
samples = 16
value = [3, 13]
class = normal
gini = 0.0
samples = 13
value = [13, 0]
class = anamoly
gini = 0.0
samples = 144
value = [144, 0]
class = anamoly
gini = 0.0
samples = 1
value = [0, 1]
class = normal
num_outbound_cmds <= 406.5
gini = 0.012
samples = 162
value = [1, 161]
class = normal
duration <= 311.5
gini = 0.444
samples = 3
value = [2, 1]
class = anamoly
gini = 0.0
samples = 4
value = [0, 4]
class = normal
protocol_type <= 3951.0
gini = 0.278
samples = 6
value = [5, 1]
class = anamoly
gini = 0.0
samples = 2
value = [2, 0]
class = anamoly
dst_host_srv_count <= 0.665
gini = 0.133
samples = 14
value = [1, 13]
class = normal
gini = 0.0
samples = 161
value = [0, 161]
class = normal
gini = 0.0
samples = 1
value = [1, 0]
class = anamoly
gini = 0.0
samples = 1
value = [0, 1]
class = normal
gini = 0.0
samples = 2
value = [2, 0]
class = anamoly
gini = 0.0
samples = 5
value = [5, 0]
class = anamoly
gini = 0.0
samples = 1
value = [0, 1]
class = normal
gini = 0.0
samples = 13
value = [0, 13]
class = normal
gini = 0.0
samples = 1
value = [1, 0]
class = anamoly