# Documentation for Determining the Optimal Threshold for Attack Detection Using ROC Curve

## Overview

This document describes the process of determining the optimal threshold for detecting attacks using Receiver Operating Characteristic (ROC) curve analysis. Two datasets, attack.csv and no\_attack.csv, are used. The optimal threshold is identified using the ROC curve and the Area Under the Curve (AUC) is calculated to assess the model’s performance.

## Why ROC Analysis?

Performance Assessment: ROC analysis provides a comprehensive way to assess the performance of a binary classification model. It considers the trade-offs between the true positive rate (TPR) and the false positive rate (FPR) across various threshold settings.  
  
Threshold Selection: In many classification tasks, the output of the model is a probability score or a continuous value. Setting a threshold on this score determines the classification outcome (e.g., positive or negative). ROC analysis helps in selecting an optimal threshold that balances between correctly identifying positive instances (attacks, in this case) while minimizing false alarms (misclassifying non-attacks).

## Datasets

attack.csv: Contains data with known attacks.  
no\_attack.csv: Contains data with no attacks.

Both datasets include a column named ratio which is used for the analysis.

## How to run

Open the “Threshold using ROC method.ipynb” in a google colab, or jupyter notebook and use the attack.csv and no\_attack.csv files to display the ROC curve.

# Code Description

## Importing Libraries

python

import pandas as pd  
import numpy as np  
from sklearn.metrics import roc\_curve, roc\_auc\_score  
import matplotlib.pyplot as plt

## Loading the Data

The CSV files are loaded into Pandas DataFrames.

python

no\_attack\_df = pd.read\_csv('no\_attack.csv', delimiter=',')  
attack\_df = pd.read\_csv('attack.csv', delimiter=',')

## Checking Columns

Ensure that both DataFrames contain the ratio column.

python

print("No Attack DataFrame columns:", no\_attack\_df.columns)  
print("Attack DataFrame columns:", attack\_df.columns)  
  
if 'ratio' not in no\_attack\_df.columns:  
 raise KeyError("The 'ratio' column is not found in no\_attack\_df")  
if 'ratio' not in attack\_df.columns:  
 raise KeyError("The 'ratio' column is not found in attack\_df")

## Data Preprocessing

Convert the ratio column to numeric and drop rows with NaN values.

python

no\_attack\_df['ratio'] = pd.to\_numeric(no\_attack\_df['ratio'], errors='coerce')  
attack\_df['ratio'] = pd.to\_numeric(attack\_df['ratio'], errors='coerce')  
  
no\_attack\_df.dropna(subset=['ratio'], inplace=True)  
attack\_df.dropna(subset=['ratio'], inplace=True)

## Labeling Data

Label the data for ROC curve calculation. no\_attack data is labeled as 0 and attack data as 1.

python

no\_attack\_df['label'] = 0  
attack\_df['label'] = 1

## Combining the Data

Combine the data into a single DataFrame.

python

combined\_df = pd.concat([no\_attack\_df[[‘ratio’, ‘label’]], attack\_df[[‘ratio’, ‘label’]]], ignore\_index=True)

## Extracting Ratios and Labels

Extract the ratio values and corresponding labels.

python

data = combined\_df['ratio'].values  
labels = combined\_df['label'].values

## Calculating ROC Curve and AUC

Calculate the ROC curve and the AUC score.

python

fpr, tpr, thresholds = roc\_curve(labels, data)  
roc\_auc = roc\_auc\_score(labels, data)  
  
print(f"AUC: {roc\_auc}")

## Finding the Optimal Threshold

Calculate the distance to the top-left corner (0,1) of the ROC curve and find the threshold that minimizes this distance.

python

distances = np.sqrt((fpr - 0)\*\*2 + (tpr - 1)\*\*2)  
optimal\_index = np.argmin(distances)  
optimal\_threshold = thresholds[optimal\_index]  
  
print(f"Optimal Threshold: {optimal\_threshold}")

## Applying the Threshold

Define a function to classify data points based on the optimal threshold and calculate false positive and true positive rates.

python

def is\_attack(ratio):  
 return ratio > optimal\_threshold  
  
false\_positives = [is\_attack(ratio) for ratio in no\_attack\_df['ratio']]  
true\_positives = [is\_attack(ratio) for ratio in attack\_df['ratio']]  
  
false\_positive\_rate = sum(false\_positives) / len(false\_positives)  
true\_positive\_rate = sum(true\_positives) / len(true\_positives)  
  
print(f"False Positive Rate: {false\_positive\_rate \* 100}%")  
print(f"True Positive Rate: {true\_positive\_rate \* 100}%")

## Plotting the ROC Curve

Plot the ROC curve and highlight the optimal threshold.

python

plt.figure()  
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:0.2f})')  
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  
plt.scatter(fpr[optimal\_index], tpr[optimal\_index], color='red', label='Optimal Threshold', zorder=10)  
plt.xlim([0.0, 1.0])  
plt.ylim([0.0, 1.05])  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('Receiver Operating Characteristic')  
plt.legend(loc="lower right")  
plt.show()

## Conclusion

The optimal threshold for detecting attacks based on the ROC curve analysis is 87.6. This threshold provides a balance between the true positive rate and the false positive rate, optimizing the model’s performance in identifying attacks.