**Anomaly Detection**

**Unsupervised Isolation Forest**

**x\_train,x\_test,y\_train, y\_test = train\_test\_split(df2,labels,test\_size=0.2,random\_state=42)**

**x\_val, y\_val = df\_validate, df\_validate["label"]**

***#################################### model #################################***

**isolation\_forest = IsolationForest(n\_estimators=100, max\_samples=256,contamination=0.1,random\_state =42)**

**isolation\_forest.fit(x\_train)**

**Validation set**

**anomaly\_scores = isolation\_forest.decision\_function(x\_val)**

**plt.figure(figsize=(15,10))**

**plt.hist(anomaly\_scores, bins =100)**

**plt.xlabel("Average path length",fontsize=14)**

**plt.ylabel('Number of Data Points',fontsize=14)**

**plt.show()**

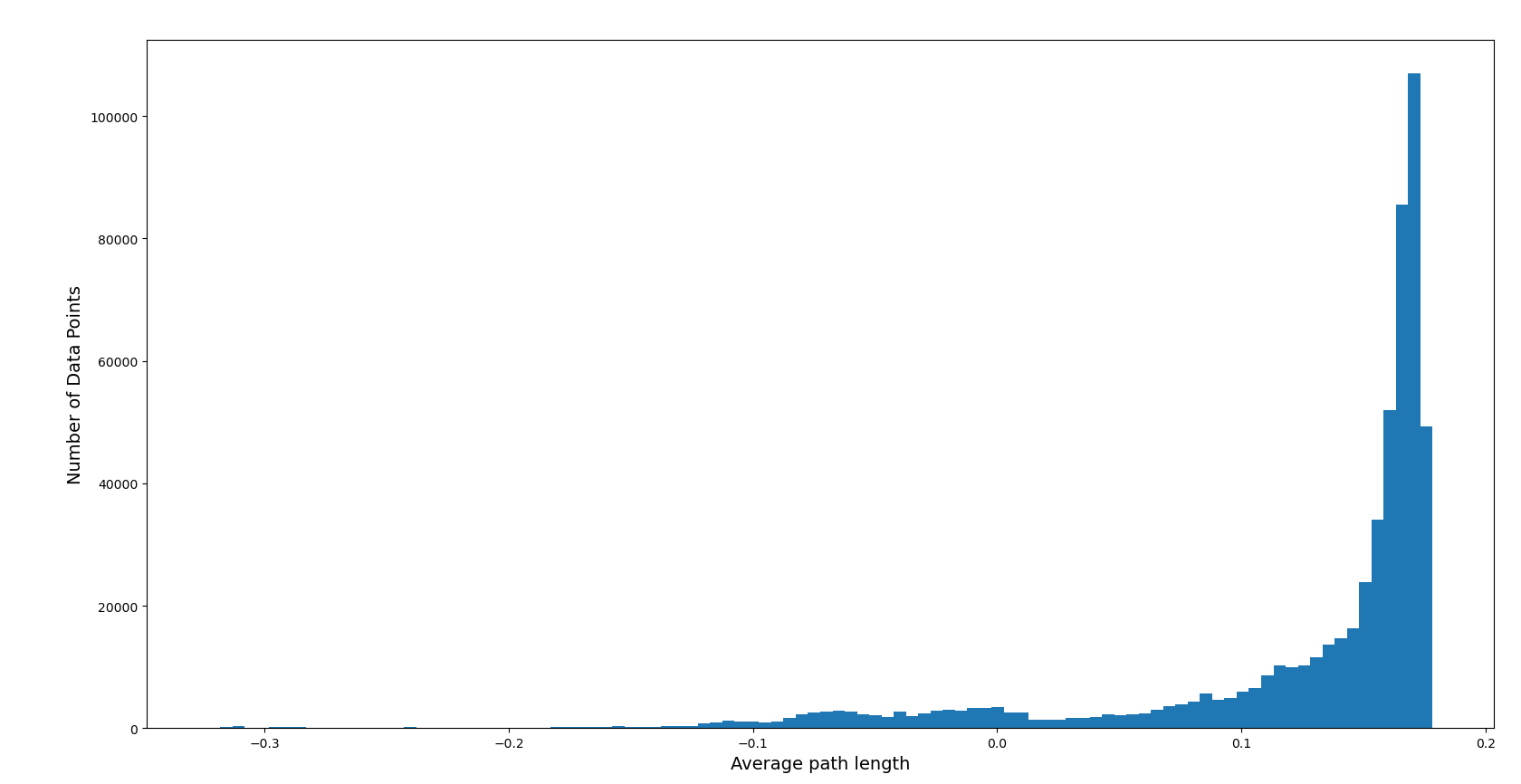
**anomalies = anomaly\_scores > -0.19**

**anomalies = np.array(anomalies).reshape(anomalies.shape[0],-1)**

**matches = y\_val == list(encoded.classes\_).index("normal.")**

**auc = roc\_auc\_score(anomalies,matches)**

**print("Validation Set AUC= ",auc)**

****

**Testing Set**

**anomaly\_scores\_test = isolation\_forest.decision\_function(x\_test)**

**plt.figure(figsize=(15,10))**

**plt.hist(anomaly\_scores\_test,bins=100)**

**plt.xlabel("Average path lengths",fontsize=14)**

**plt.ylabel('Number of Data Points',fontsize=14)**

**plt.show()**

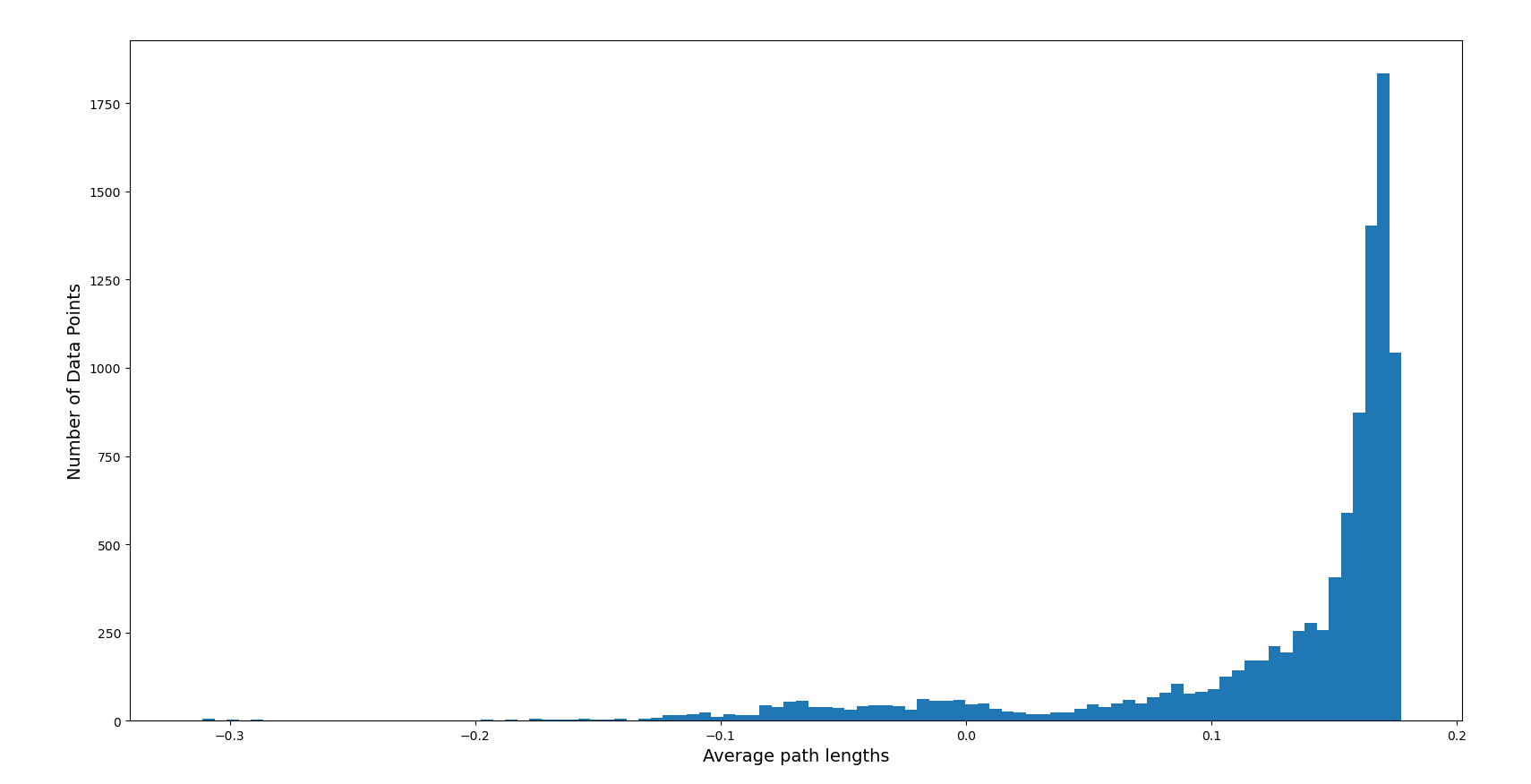
**anomalies\_test = anomaly\_scores\_test > -0.19**

**anomalies\_test = np.array(anomalies\_test).reshape(anomalies\_test.shape[0],-1)**

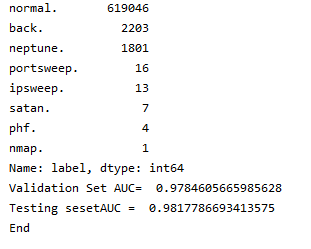
**matches = y\_test == list(encoded.classes\_).index("normal.")**

**auc = roc\_auc\_score(anomalies\_test ,matches)**

**print("Testing sesetAUC = ",auc)**

****

**AUC in Validation set vs Testing Set**

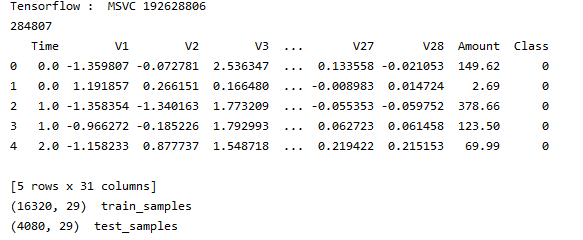
****

**Semi supervised Deep Autoencoders**

**x\_train, x\_test,y\_train, y\_test = train\_test\_split(df.drop(labels=['Time','Class'],axis=1),df['Class'],test\_size=0.2,random\_state=42)**

**print(x\_train.shape," train\_samples")**

**print(x\_test.shape, " test\_samples")**

****

**logfile\_name ="deepautoencoder"**

**encoding\_dim = 16**

**input\_dim =x\_train.shape[1]**

**inputArray = Input(shape=(input\_dim,))**

**encoded = Dense(encoding\_dim,activation='relu')(inputArray)**

**encoded = Dense(8,activation='relu')(encoded)**

**encoded = Dense(4,activation='relu')(encoded)**

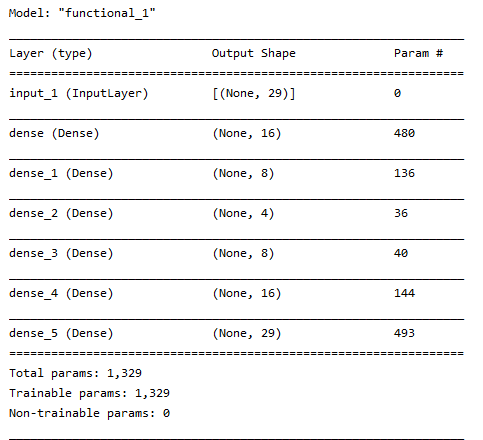
**decoded = Dense(8,activation = 'relu')(encoded)**

**decoded = Dense(encoding\_dim,activation = 'relu')(decoded)**

**decoded = Dense(input\_dim,activation = 'softmax')(decoded)**

**autoencoder = Model(inputArray,decoded)**

**autoencoder.summary()**

****

**autoencoder.compile(optimizer=RMSprop(),loss = 'mean\_squared\_error',metrics=['mae','accuracy'])**

**batch\_size = 32**

**epochs =20**

**history = autoencoder.fit(x\_train,y\_train,batch\_size=batch\_size,epochs=epochs,verbose=1,validation\_data=(x\_test,y\_test),callbacks=[TensorBoard(log\_dir='../logs/autoencoder1')])**

**threshold= [10, 1,5,15]**

**for th in threshold:**

**y\_pred = autoencoder.predict(x\_test)**

**y\_dist = np.linalg.norm(x\_test - y\_pred,axis = -1)**

**z = zip(y\_dist >= th,y\_dist)**

**y\_label=[]**

**error = []**

**for idx,(is\_anomaly,y\_dist) in enumerate(z):**

**if is\_anomaly:**

**y\_label.append(1)**

**else:**

**y\_label.append(0)**

**error.append(y\_dist)**

**y\_label = pd.Series(y\_label)**

**print(classification\_report(y\_test,y\_label))**

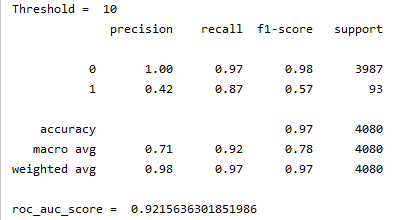
**print("roc\_auc\_score = ",roc\_auc\_score(y\_test,y\_label))**

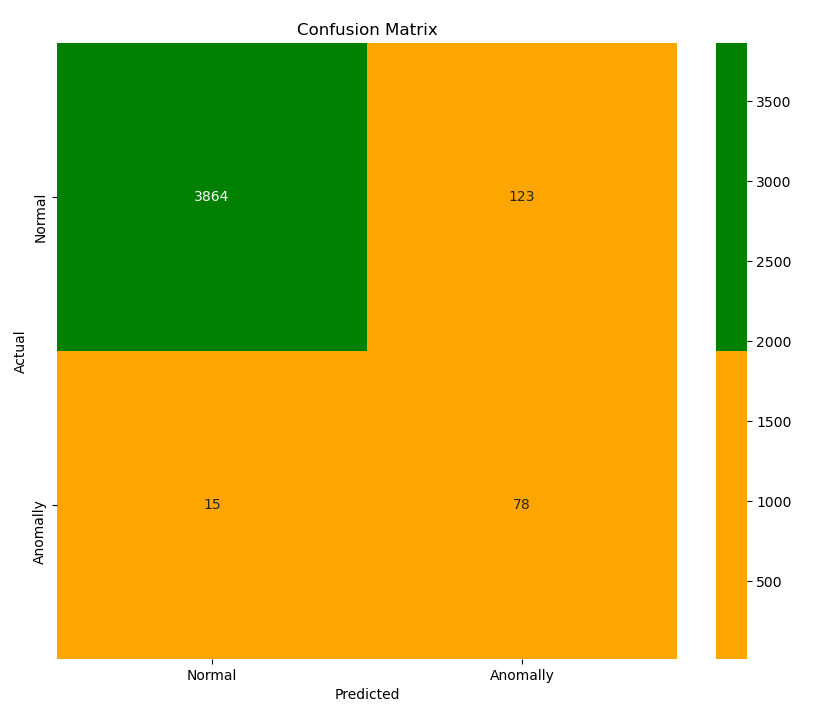
**viz = visualization()**

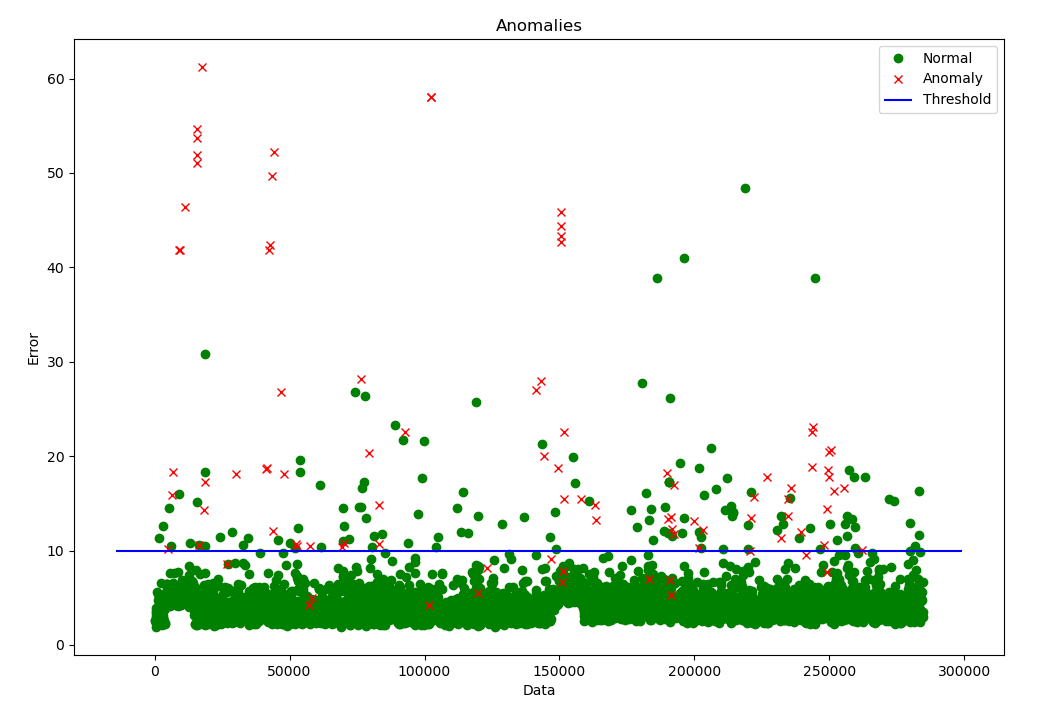
**viz.draw\_confusion\_matrix(y\_test,y\_label)**

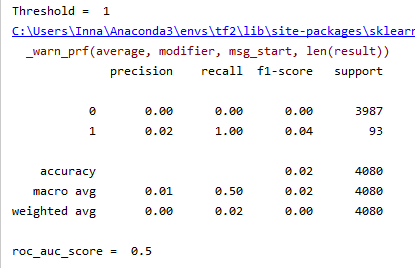
**viz.draw\_anomaly(y\_test,error,threshold)**

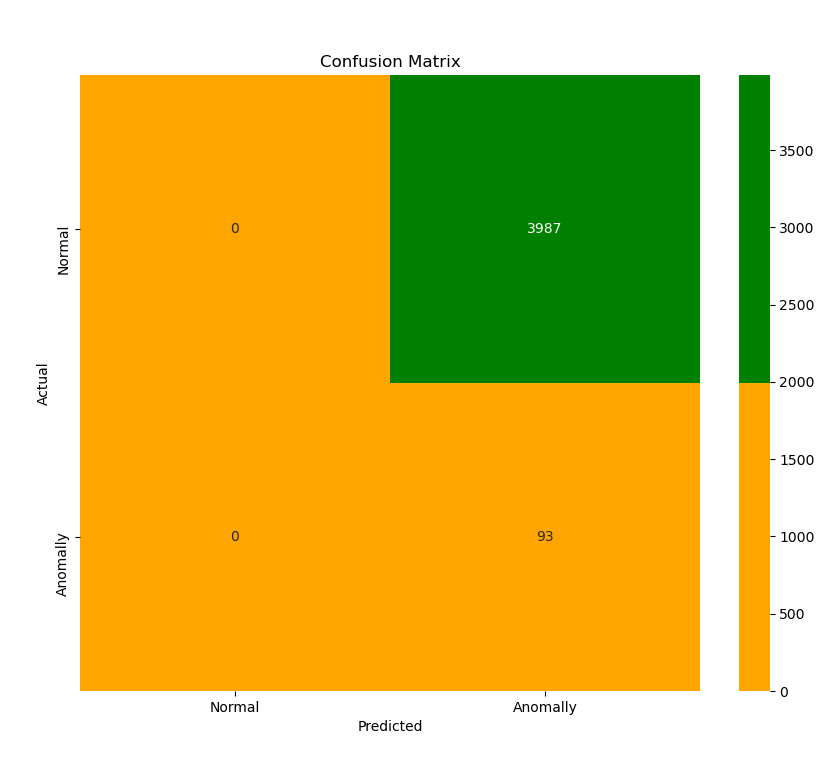
**print("End Threshold = " ,th)**

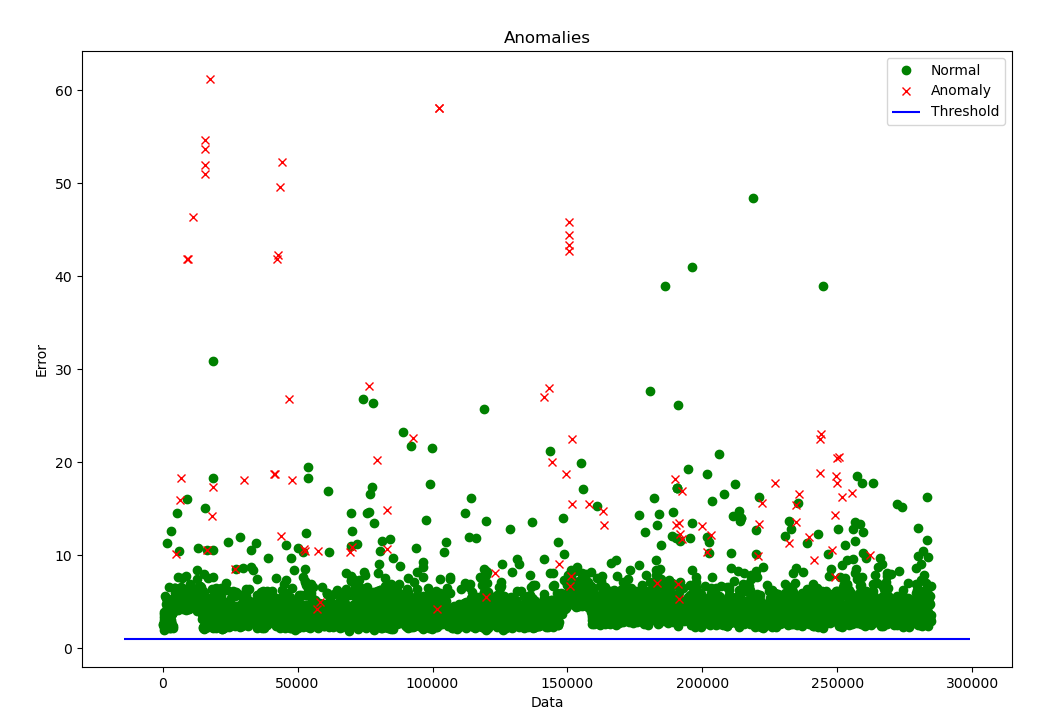
****

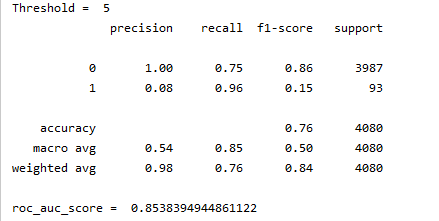
****

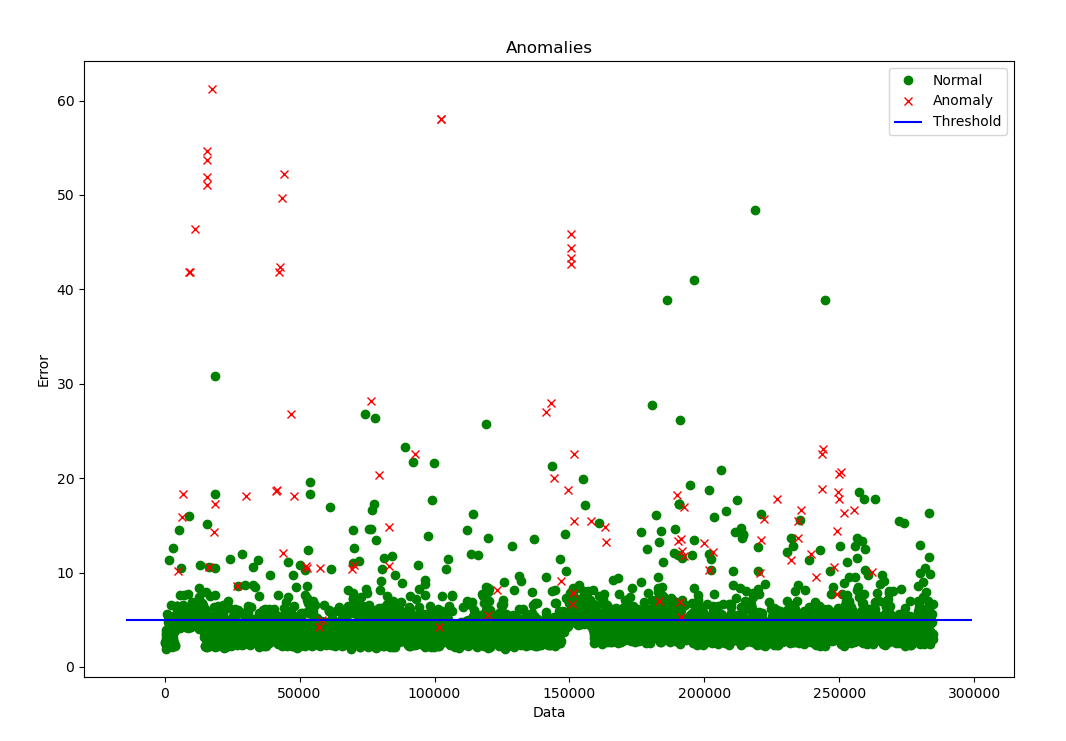
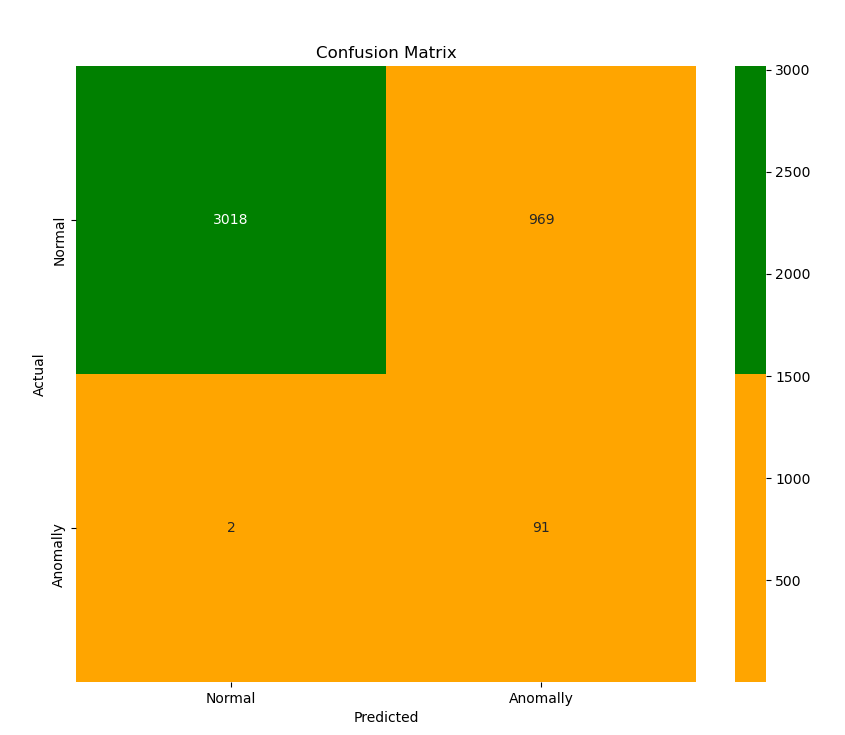
****

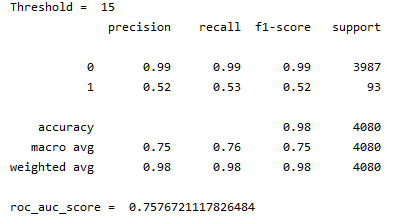
****

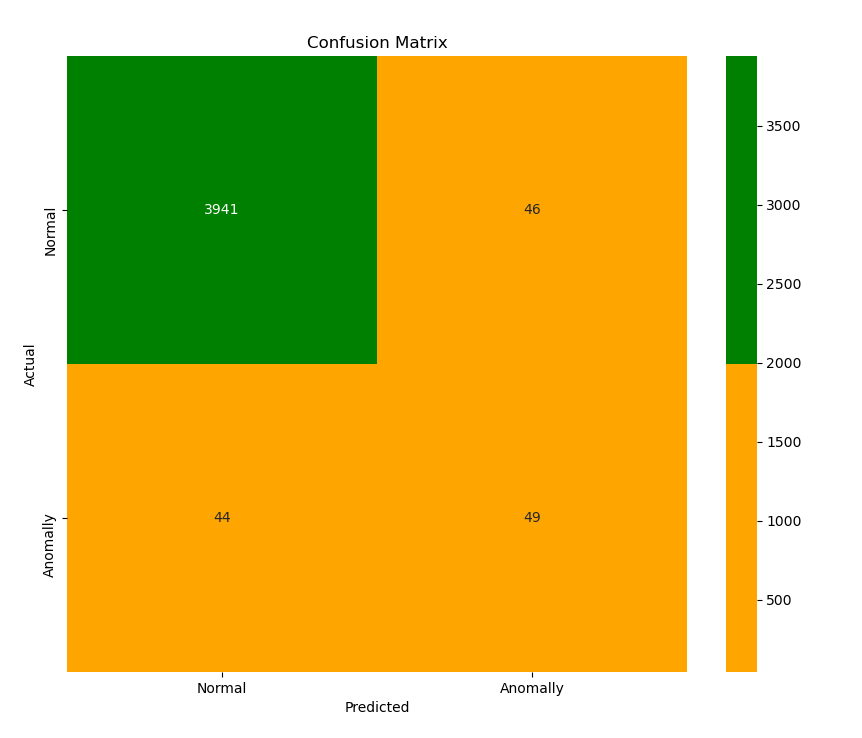
****

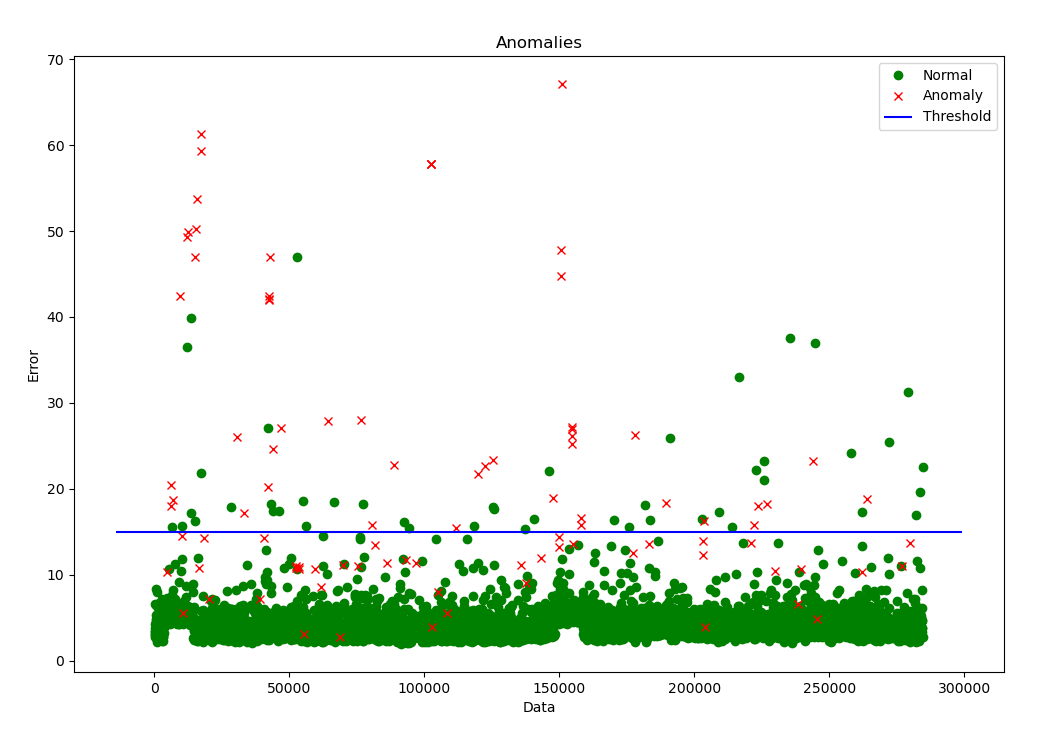
****

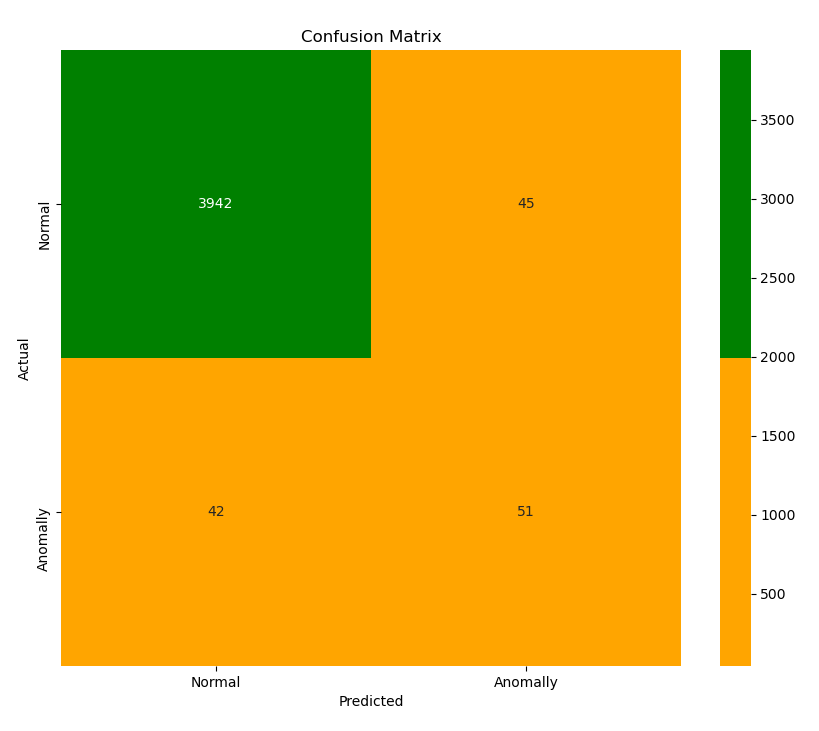
****

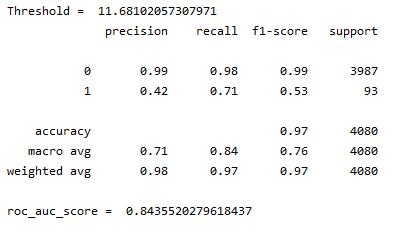
****

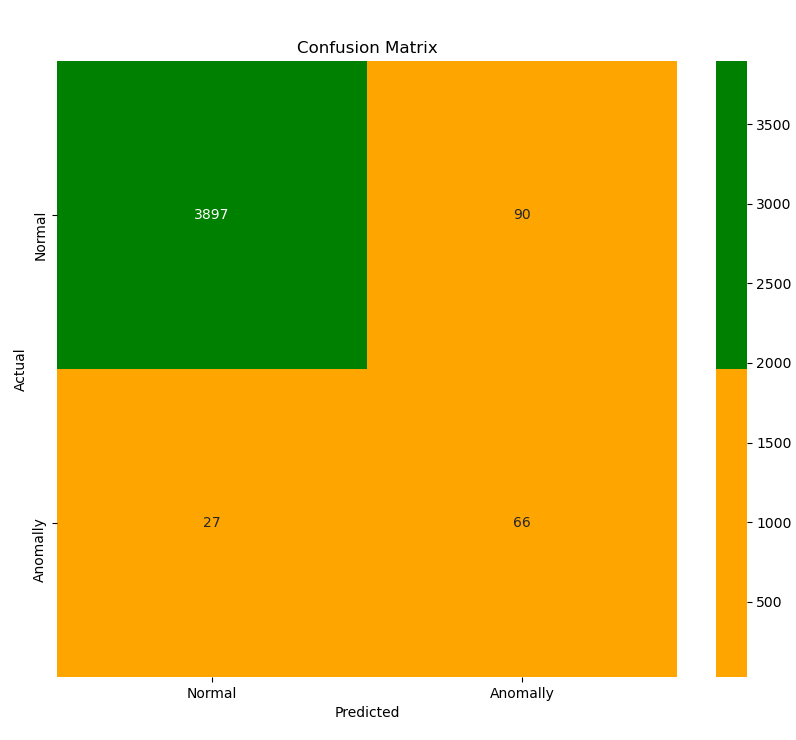
****

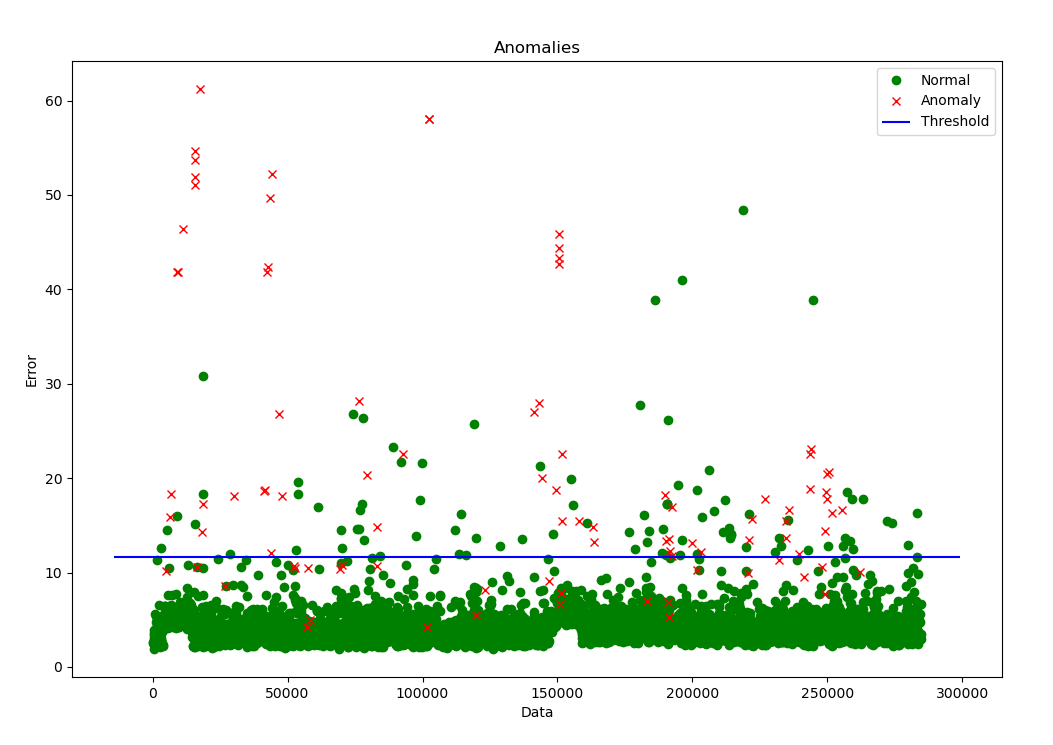




****

****

****

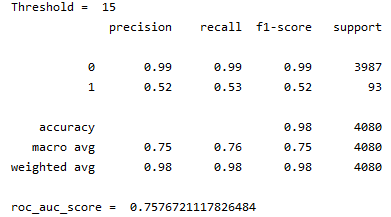
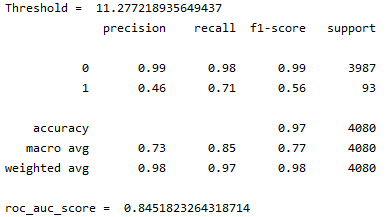
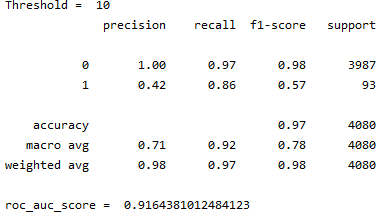
****

**We can see that the for Thresholds = [1 , 5]**

**precision ,recall is low, f1 and accuracy is low is low**

**But for Thresholds = [10, 11.277, 15] are much better**

**Compare the thresholds 10, 11.277, 15**

****

**We can see that the recall value for 1 is the best for Threshold = 11.277**

**This means that the less anomalies will be missed using this threshold and therefore**

**The best model is with threshold = 11.277**