from sklearn.metrics import confusion\_matrix

def measurements(real,pred,Name):

Confusion\_Matrix = confusion\_matrix(real,pred,labels = [0,1])

TN= Confusion\_Matrix[0][0]

FN= Confusion\_Matrix[1][0]

FP= Confusion\_Matrix[0][1]

TP= Confusion\_Matrix[1][1]

# True Negative Rate

Acc\_Negative= TN/(TN+FP)

# True Positive Rate

Acc\_Positive = TP/(TP+FN)

Recall = Acc\_Positive

# G-mean

G\_mean = np.power(Acc\_Negative \* Acc\_Positive, 0.5)

# Precision

Precision = TP/(TP+FP)

# Weighted Accuracy

Beta= 0.5 # Here we use equal weights for both true positive rate and true negative rate; i.e., β equals 0.5

Weighted\_Accuracy= (Beta \* Acc\_Positive) + ((1-Beta)\*Acc\_Negative)

# F-measure

F\_measure = (2 \* Precision \* Recall) /(Precision + Recall)

performance\_measures = [{"Method":Name,"Acc\_Positive(Recall)":Acc\_Positive, "Acc\_Negative":Acc\_Negative, "Precision":Precision,"F\_measure":F\_measure,"G\_mean":G\_mean,"Weighted\_Accuracy":Weighted\_Accuracy}]

df = pd.DataFrame(performance\_measures)

return(df)

Call to function with Name = ‘kNN’:

measurements(yt,pred,'kNN')