X train\_balanced, Y train balanced= smote.fit resample(X train scaled,

Y train) <Used to balance the imabalance data>

print(pd.Series(Y\_train\_balanced) .value\_counts()) <Used to print

the counts of unique values in the Pandas Series Y\_train\_balanced>

knn = KNeighborsClassifier(n\_neighbors=S) <Used to create an instance

of the KNN classifier configured to consider the 5 nearest neighbors

when making predictions>

knn.fit(X\_train\_balanced, Y\_train\_balanced) <Used to train a

KNN classifier model using the training data>

Y\_pred • knn.predict(X\_test\_scaled) < Used to make predictions on the tes

data (X\_test\_scaledJ using a trained KNN classifier>

pr int ( "Confusion Matrix: \n", con fusion\_ma tr ix (Y\_ test, Y \_pred) )

print("\nClassification Report:\n", classification\_report(Y\_test,

Y\_pred)l print("\nAccuracy Score:", accuracy\_score(Y\_test, Y\_pred)) <

Used to evaluate the performance of a ML classifier by printing out the

confusion matrix, classification report, & accuracy score>

cm• confusion\_matrix(Y\_test, Y\_pred)

sns.heatmap(cm, annot•True, fmt•'d',

cmap•'Blues') plt.title('Confusion Matrix')

plt.xlabel{'Predicted Label') plt.ylabel{'True

Label')

plt.show{) <creates a heatmap visualization of the confusion matrix using

Seaborn & Matplotlib libraries>

class O metrics= {'Precision': 0.60, 'Recall': 0.86, 'Fl-score':

0.71)-class 1 metrics= {'Precision': 0.97, 'Recall': 0.88, 'Fl-

score': 0.92)-overall\_accuracy • 0.87 macro\_avg\_fl • 0.81

weighted\_avg\_fl • 0.88

categories•

list(class\_O\_metrics.keys(ll

class\_O\_values • [class\_O\_metrics[cat] for cat in categories)

class\_l\_values • {class\_l\_rnetrics[cat] for cat in categories)

bar\_width =

0.35

index= range(len(categoriesl)

Challenges faced

fig, ax = plt.subplots() bar1 = ax.bar(index, class\_0\_values, bar\_width, label='Class 0', color='blue') bar2 = ax.bar([i + bar\_width for i in index], class\_1\_values, bar\_width, label='Class 1', color='red')

ax.axhline(y=overall\_accuracy, color='green', linestyle='-', label='Overall Accuracy') ax.axhline(y=macro\_avg\_f1, color='orange', linestyle='--', label='Macroaverage F1-score') ax.axhline(y=weighted\_avg\_f1, color='purple', linestyle=':', label='Weighted average F1-score')

ax.set\_xlabel('Metrics') ax.set\_ylabel('Percentage') ax.set\_title('Performance Metrics Comparison') ax.set\_xticks([i + bar\_width/2 for i in index]) ax.set\_xticklabels(categories) ax.legend()

plt.show() < visualizes the performance metrics comparison between two classes in a bar chart, along with horizontal lines representing overall accuracy, macro-average F1-score, & weighted average F1-score>