We are using KNN to classify a region of a country by a few of the country's attributes, like wealth, happiness, health, etc. The dataset was obtained through Kaggle and is a list of 158 countries over 10 regions of the world with detailed attributes of each country. There are other datasets from more recent years but currently we are only focusing on the 2015 dataset.

Stats of each class (max, min, mean, median, mode, and standard deviation) for the happiness score:

- Western Europe:7.59, 4.86, 6.69, 6.94, 4.86, 6.69, 0.80
- Middle East and Northern Africa: 7.28, 3.00, 5.41, 5.26, 3.00, 5.41, 1.07
- Latin America and Caribbean: 7.23, 4.51, 6.14, 6.15, 4.5, 6.14, 0.71
- Southeastern Asia: 6.80, 3.82, 5.32, 5.36, 3.82, 5.32, 0.90
- Central and Eastern Europe: 6.51, 4.22, 5.33, 5.29, 4.22, 5.33, 0.56
- Eastern Asia: 6.30, 4.87, 5.63, 5.73, 4.88, 5.63,0.51
- Sub-Saharan Africa: 5.48, 2.84, 4.20, 4.27, 2.84, 4.20, 0.60
- Southern Asia: 5.25, 3.58, 4.58, 4.57, 3.58, 4.58, 0.53

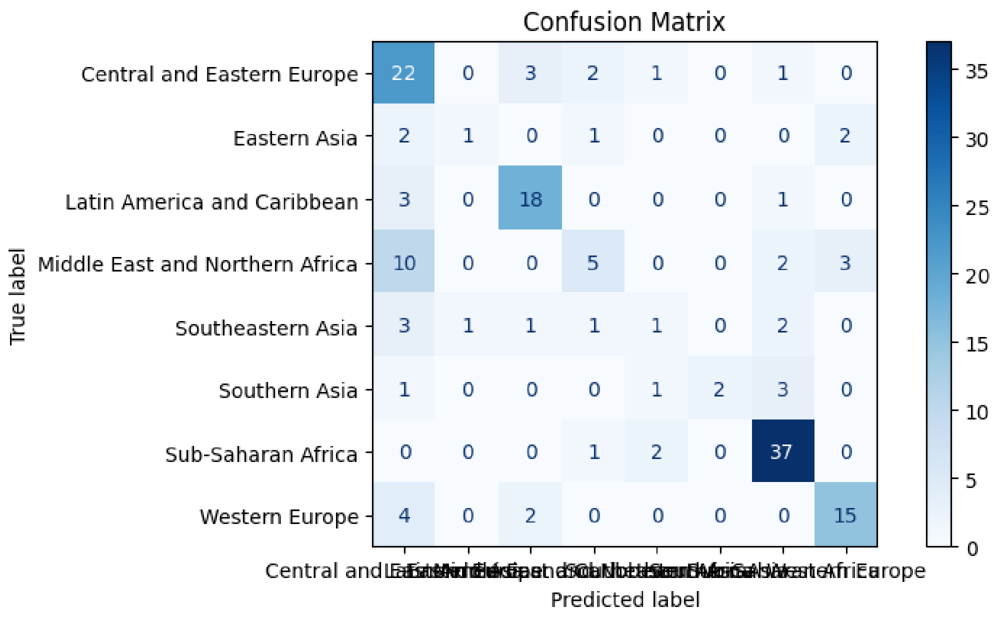
There are also two other classes, but we excluded them from the project because they only had two instances, and would require a small K value and skew the results.

KNN is a very simple supervised learning algorithm that can take a set of data and attempt to classify an unknown point by examining the K nearest neighbors. It is an algorithm that is resource heavy during testing (since you have to find the neighbors) but requires little to train. The algorithm's success is dependent on the application, training data, and the picked K value.

Classification Results: (Matrix bottom row is same order as left from top to bottom)

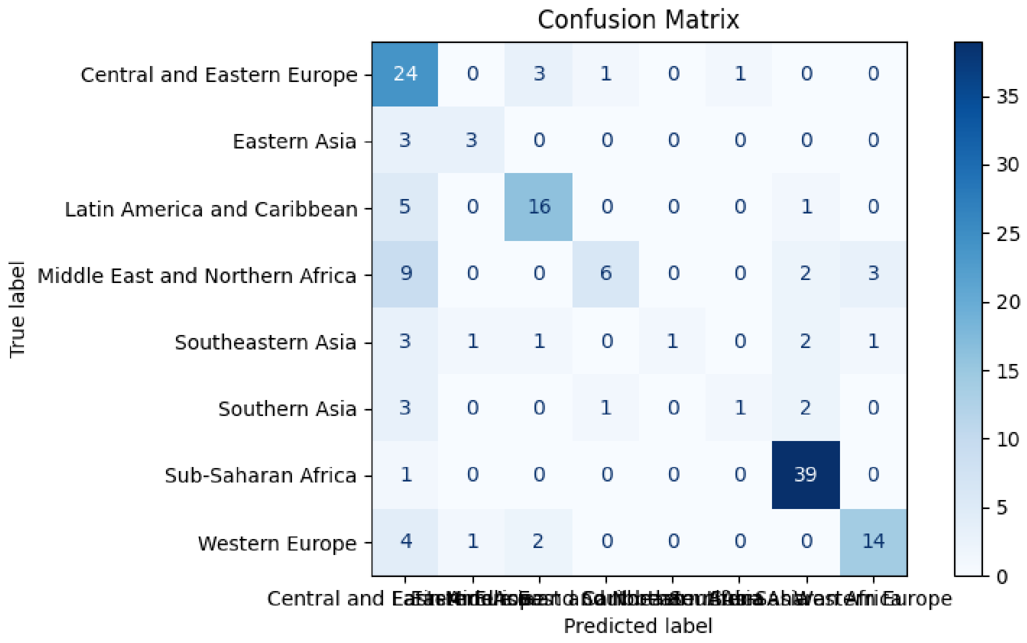
K = 5, Split = 70% train, metric = Euclidean

- ACC: 0.57



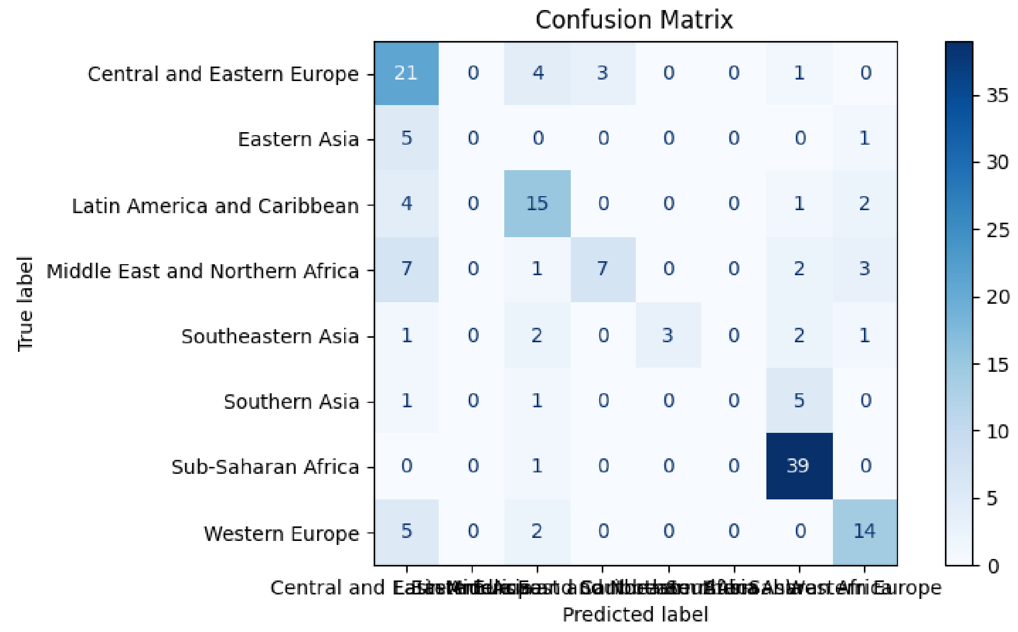
K = 7, Split = 70% train, metric = Euclidean

- ACC: 0.66



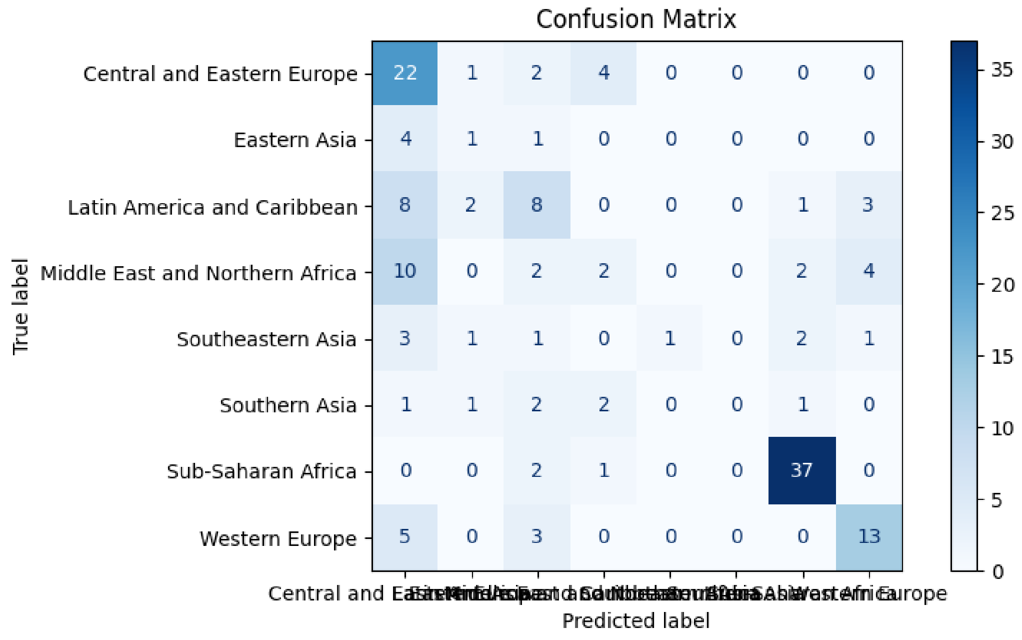
K = 9, Split = 70% train, metric = Euclidean

- ACC: 0.64



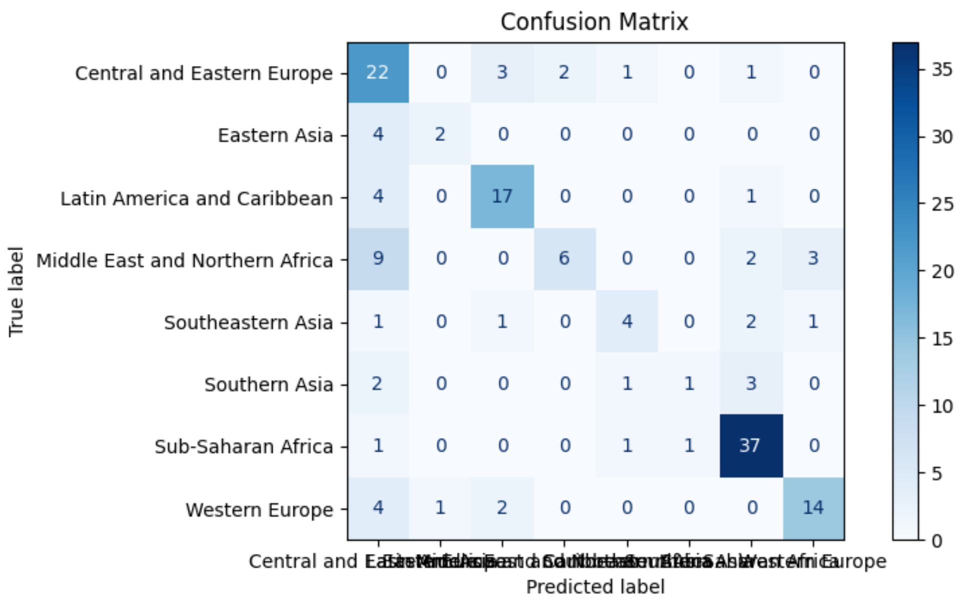
K = 7, Split = 40% train, metric = Euclidean

- ACC: 0.56



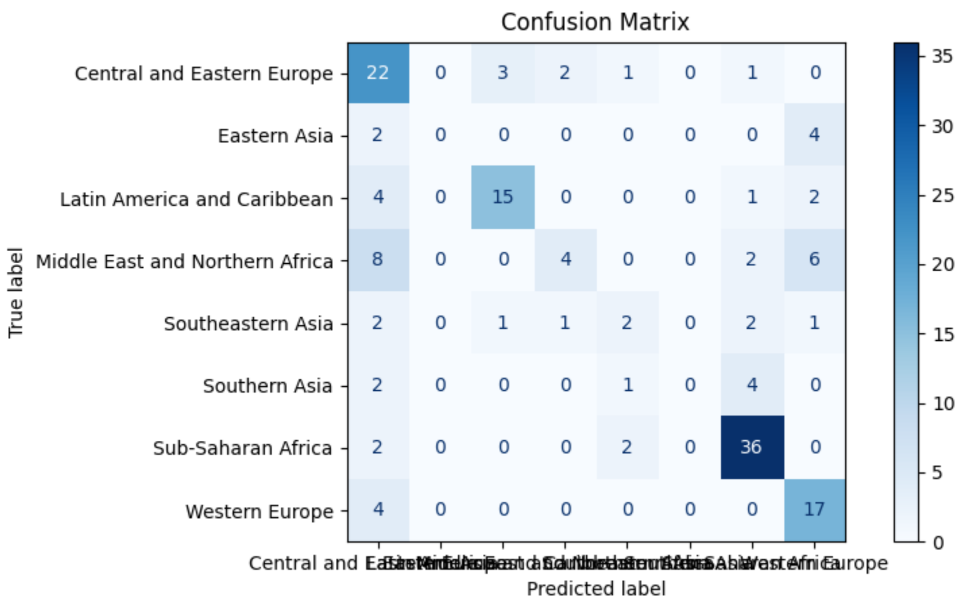
K = 7, Split = 90% train, metric = Euclidean

- ACC: 0.56



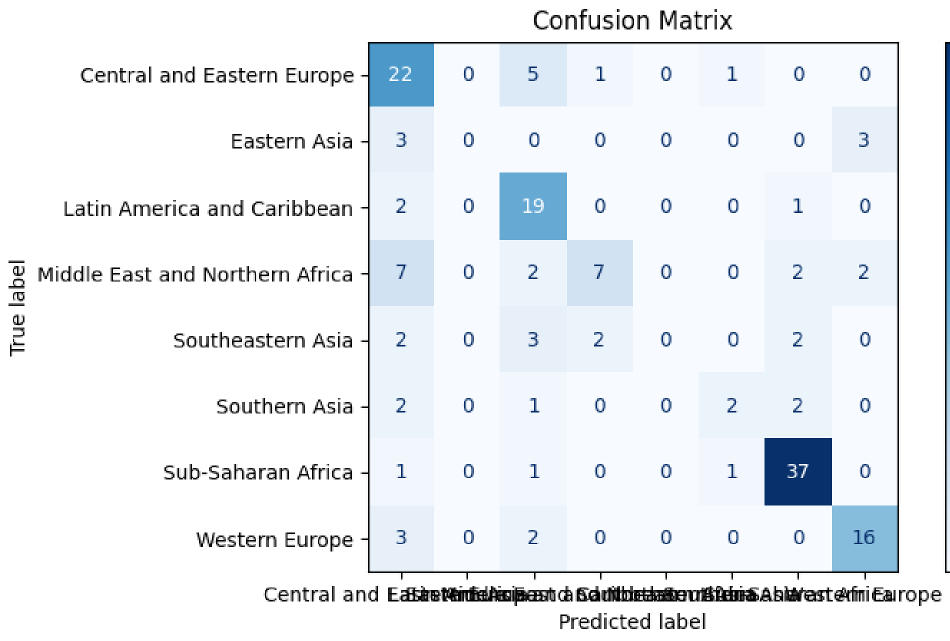
K = 7, Split = 70% train, metric = Minkowski

- ACC: 0.49



K = 7, Split = 70% train, metric = Manhattan

- ACC: 0.57



For our project we decided to use the euclidean distance since most of the data is relatively small, ranging from zero up to eight. So far the best performance for K has been seven. Ideally the K value would be a little higher for 154 instances but two classes have 7 or less instances so the value was reduced so that it could more accurately predict. Currently we partition the dataset into 70% for training and 30% for testing. The accuracy is dependent on the partition because too little training partition and the classifier will not have as many neighbors to compare and will likely be off. The bigger the training subset the more accurate your model will be, however you'll most likely want to increase the K value and which will require the model to make more comparisons during testing. The data leakage problem is using some of the data you have trained on in testing and double dipping, causing a skewed accuracy because the points used twice will have a very high true prediction rate. Data leakage is not a problem for our model because we use a function before we input the training data to split the entire dataset into two partitions.

This project has shown me all the different hyper variables a KNN model has and how each can influence the others, both positively and negatively. KNN is a very easy to implement model, but can be fairly inaccurate sometimes and takes a bit of resources everytime you want to classify a point, which would infinitely scale with the size of the dataset.