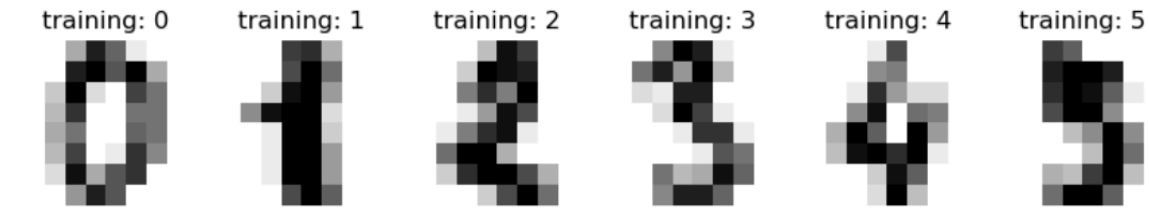
Benjino Chery

MTH 4330

Logistic classifier

Logistic regression is used to classify all the digits in the MNIST data set that are not ones from the one that are one.

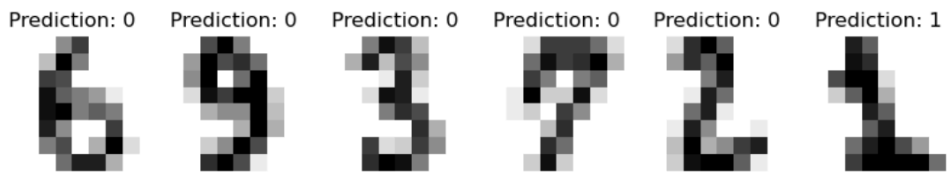


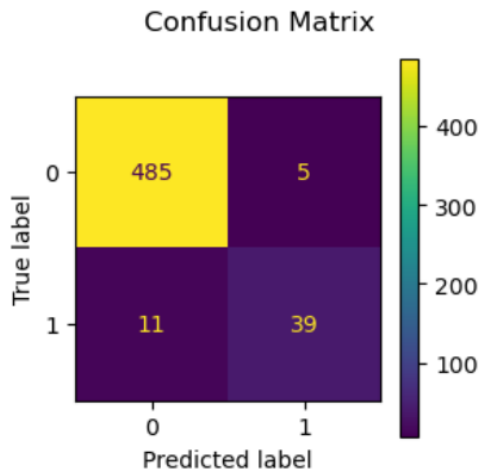
To apply a classifier on this data, we need to flatten the images, turning each 2-D array of grayscale values from shape (8, 8) into shape (64,). Subsequently, the entire dataset will be of shape (n_samples, n_features), where n_samples is the number of images and n_features is the total number of pixels in each image.

```
# flatten the images
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
#print(digits.images[1])
data.shape
#print(len(digits.target))
```

The MNIST data set is divided into 2 sets. The training set, which contains 70% of the data, will be used to build the classifier model and the test set, which contains 30%, be used to verify the accuracy of the model.

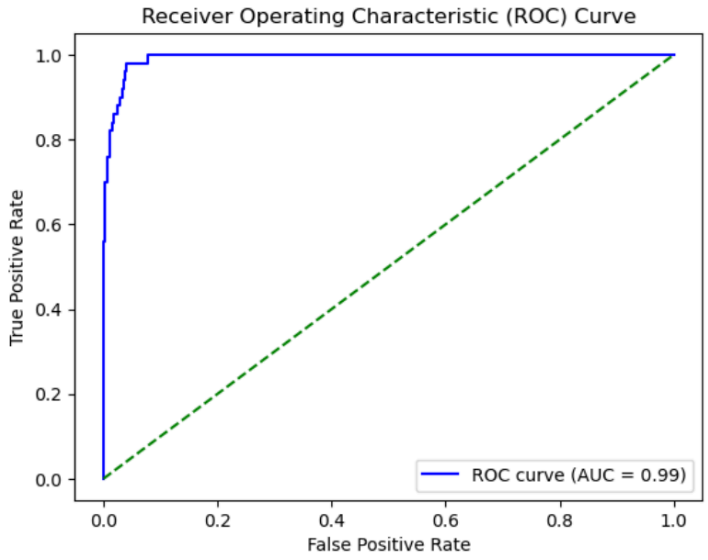
```
# Split data into 70% train and 30% test subset
X_train, X_test, y_train, y_test = train_test_split(data, my_target, random_state=42, test_size=0.3, shuffle=True)

# # Create a classifier: LogisticRegression Learn the digits on the train subset
clf =LogisticRegression().fit(X_train,y_train)

# Predict the value of the digit on the test subset
predicted = clf.predict(X_test)
```





ROC Curve

In the ROC curve for an ideal model, there would exist a threshold at which the True Positive Rate is high and the False Positive Rate is low. The more that the ROC curve hugs the top left corner of the plot, the better the model does at classifying the data.



Youden's Index

Youden's Index (or Youden's J statistic) is used to select the optimal predicted probability cut off. It is the maximum vertical distance between ROC curve and diagonal line, where the idea is to maximise the difference between true positive rate and false positive rate.

$$J = sensitivity + specificity - 1$$

Associated with any threshold value (t) is the probability of a true positive (sensitivity = q(t)) and the probability of a true negative (specificity = 1 − p(t)).

$$sensitivity = \frac{TP}{TP + FN}$$

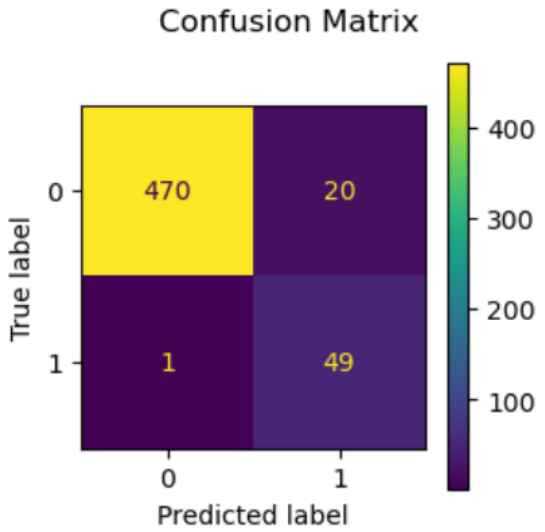
$$specificity = \frac{TN}{TN + FP} = \frac{TN}{N} = 1 - \frac{FP}{N} = 1 - FNRate$$

$$J = sensitivity + specificity - 1 = \frac{TP}{TP + FN} + 1 - FNRate - 1$$

$$J = \underset{t}{\operatorname{argmax}}(TPRate(t) - FNRate(t))$$

```
# Youden's J statistic
youden_j = tpr - fpr
max_j = np.argmax(youden_j)
best_threshold = thresholds[max_j]
best_threshold
```

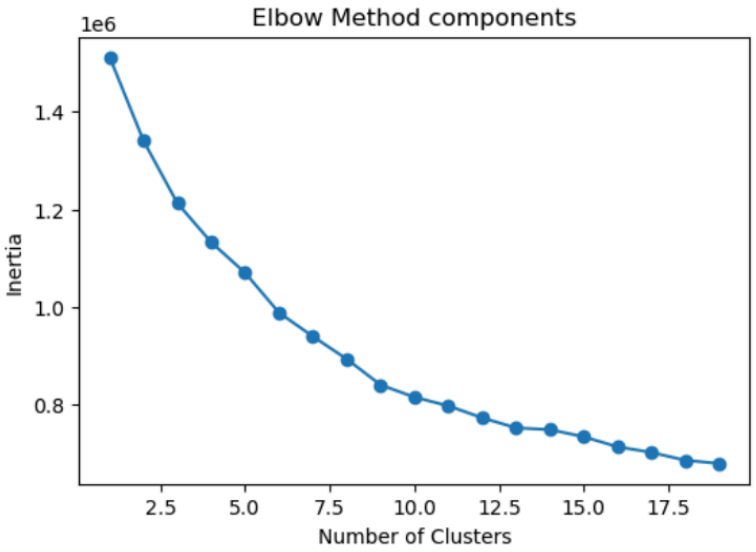
Using Youden's J statistics, the best threshold for this model is approximately 0.01352. With this threshold the classifier predicted almost all of the one correctly of the test set.



The sensitivity is 0.98 and the specificity is 0.95.

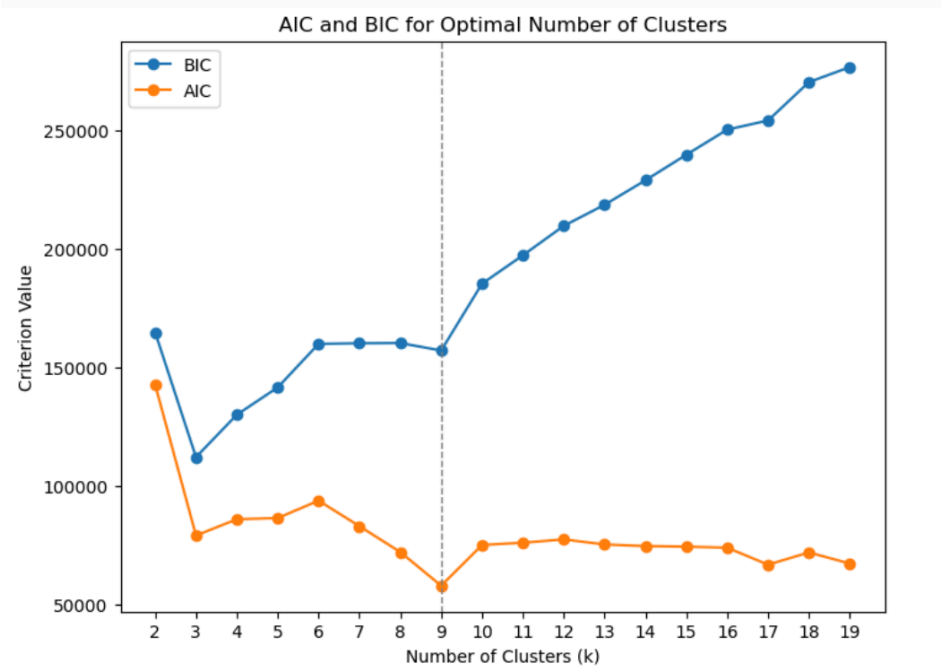
k-means to label the entire dataset

```
# inertia, or within-cluster sum-of-squares criterion:
distortion = []

for k in range (1,20):
    model = KMeans(n_clusters= k, random_state=42)
    model.fit(X_train)
    distortion.append(model.inertia_)
```

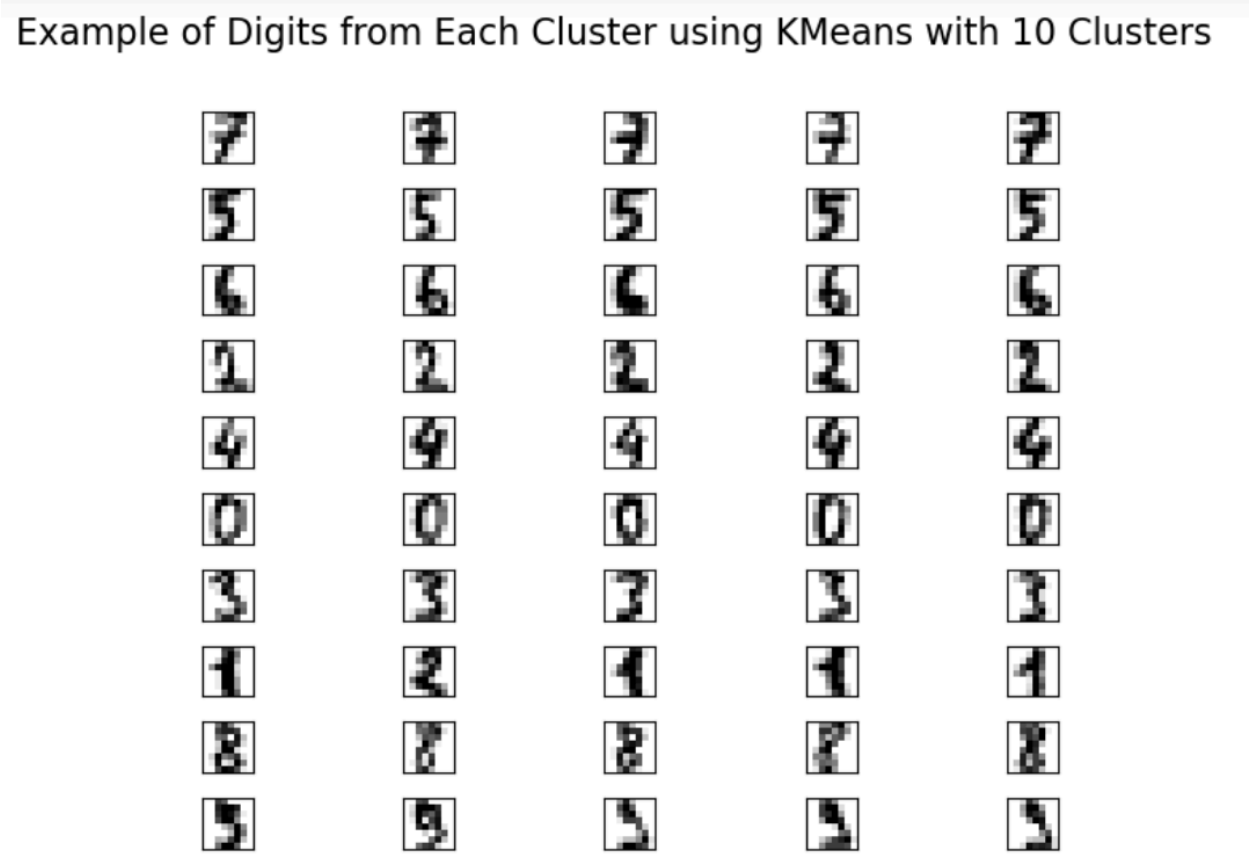


It is not evident form the elbow graph what the best value of k is. Consequently Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) might be useful. BIC and AIC are statistical measures used to assess the quality of a model while penalizing model complexity. In the context of clustering they help determine the optimal number of clusters by balancing fit and complexity; a lower value indicates a better fit.

```
BIC = []
AIC = []

for k in range (2,20):
    model = GaussianMixture(n_components= k, max_iter=400, random_state=42)
    model.fit(data)
    BIC.append(model.bic(X_train))
    AIC.append(model.aic(X_train))
```

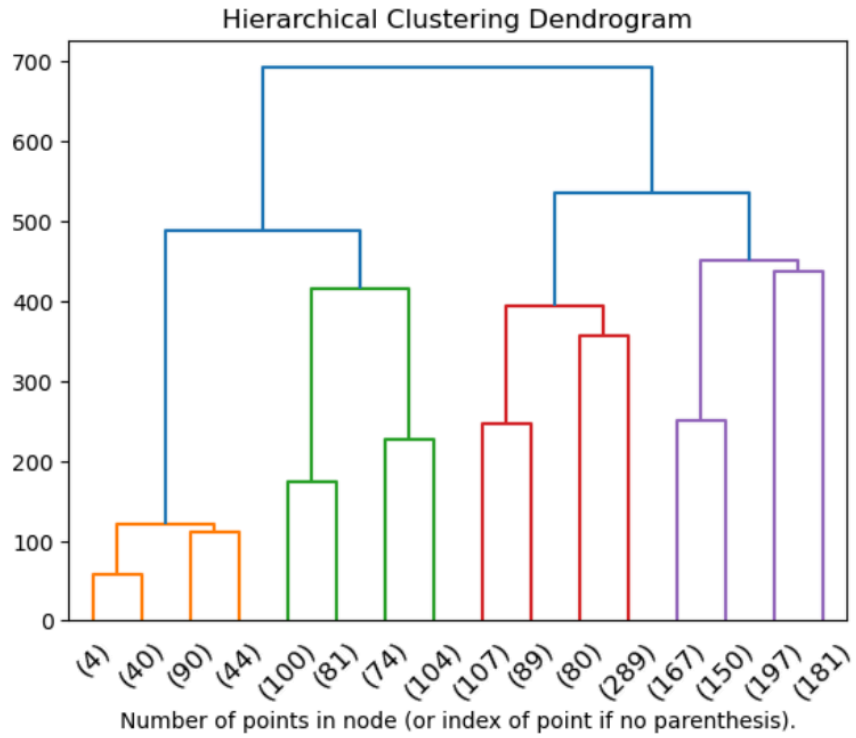


Based on the AIC and BIC graph we can assume that 10 clusters is a good approximation for the optimal number of clusters, as it balances the model fit and complexity.



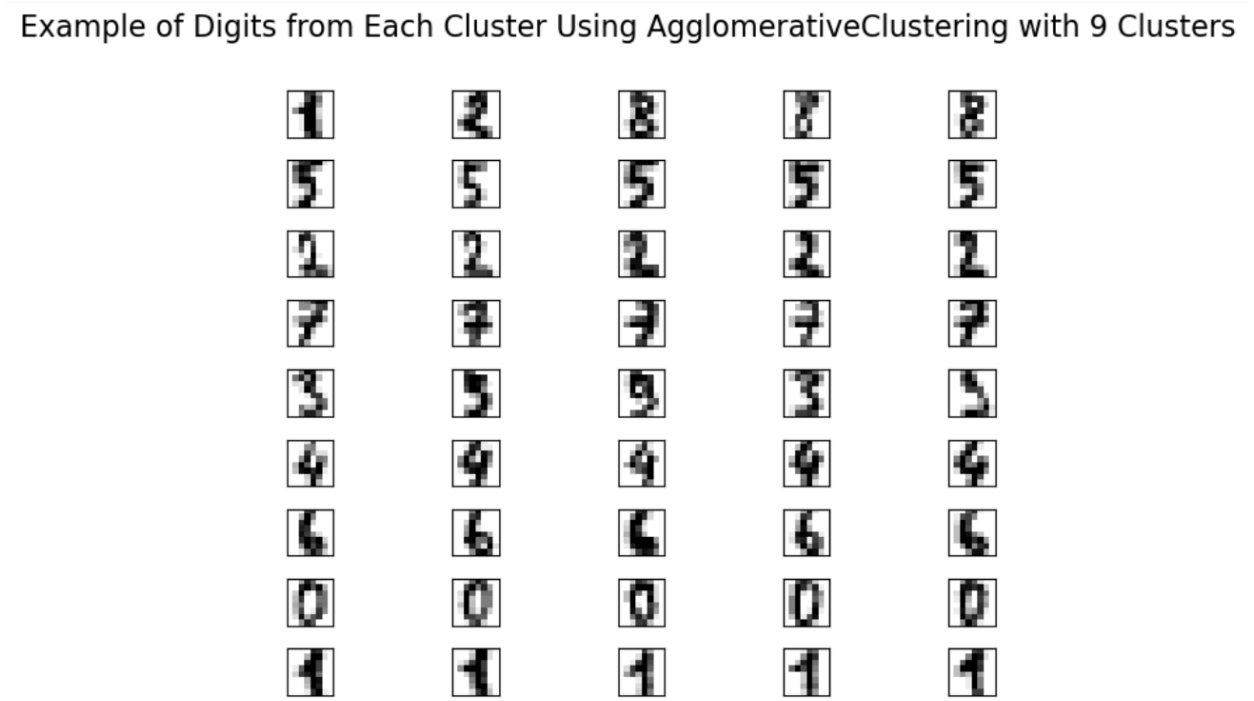
Hierarchical clustering

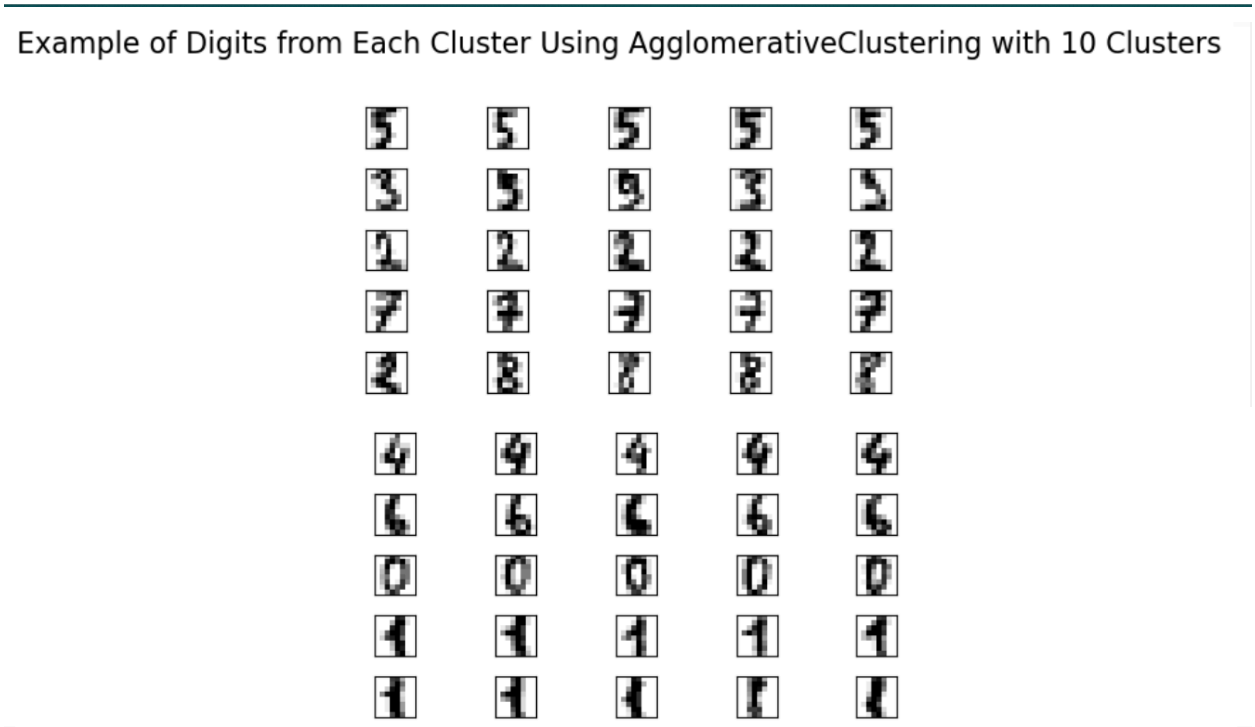
Hierarchical clustering is a method of grouping data points that builds a tree-like structure to show how clusters relate to each other. To create the clusters it uses either the bottom-up approach where each data point starts as its own cluster and merges progressively or the top-down approach where all points start in one cluster and split into smaller clusters. Both approaches use a distance matrix and their resulting Hierarchical clustering Technique can be visualized using a Dendrogram , which records the sequences of merges or splits.



Cutting the dendrogram at a height of around 300, which represents the distance or dissimilarity between the clusters being merged at that point, would result in 9 clusters.

```
model = AgglomerativeClustering(n_clusters=9, metric='euclidean',linkage='ward')
model.fit_predict(data)
labels = model.labels_
```





The K-means clustering apparently works better since it provides better-defined clusters for the MNIST dataset than the HCA, as each digit is more consistently grouped into separate clusters.