Take a moment to review the details of this assignment below and gather any necessary files. Once you're ready to submit your assignment, move on to Step 2.

Assessment Description

This assignment includes a theoretical part and an application part. Demonstrate your understanding of the concepts discussed in this topic, research a technique, and implement it.

Part 1: Theory

1. What is the role of a *confusion matrix* in the evaluation of a machine trained for a pattern recognition task? In your answer, refer to a concrete example, either from literature or one you created. Anchor you answer in relevant literature.

**A confusion matrix serves as a form of evaluation, by comparing predicted to actual values. It analyzes the performance of predictive models and its error. The matrix is composed of two binary variables that are pushed into a contingency table; this table can be broken down into 4 parts. There is the true and false and the positive and negative weaving together. The true represents the predictions that are correct, and the false are errors. The positive in itself can be noted as 1 or “true” and the negative as 0 or “false”, reflecting the binary nature of the variable.**

Confusion Matrix

<https://i.stack.imgur.com/D1Lk4.png>

**From the model above, a variety of interpretations can be formed in regard to the predicted data. There is Accuracy, Precision, Recall, Specificity, F-Measure, and AUROC. The AUROC is also referred to as the ROC curve and AUC. These two are expanded on further in Theory 2.**

**, also explains for Sensitivity.**

**The confusion matrix is also utilized with the cost matrix, the cost matrix places focus on the cost of false positives, whereas false negatives don’t hold an inherent loss in many real-world applications, such as advertising. Showing an advertisement to the wrong person can be seen as an unnecessary cost, but the company incurs no cost on a targeted consumer that is missed, though the potential is lost.**

1. What is the role of the ROC curve? How would you use it to compare the performance of several classifiers? In your answer, refer to concrete examples of classifiers, either from literature or one you created. Illustrate the ROC curves and anchor your answer in relevant literature.

**Known as the “Receiver Operating Characteristics” (ROC) curve, it is an influential part of evaluating the performance of classification models. ROC serves as a visualized interpretation of the probability curve. It is also inferred with the “Area Under the Curve” (AOC). The ROC is also related to the confusion matrix, borrowing elements of “True Positive” and “False Positive.” These elements are interpreted into rates and graphed to form a ROC curve.**

Chart, pie chart

Description automatically generated

**The model above is capable of measuring separability, which is a measured estimate of the average number of observations whose nearest neighbor holds the same label or classification. A positive or negative interpretation of this model relies on the AUC; the closer the AUC is to 1 the better the separability and the closer to 0 the worse. The entire area within the model, above and below the ROC curve, equals to 1. When separability is high there is evidence of a line capable of sufficiently dividing groups. If there are no false negative and false positive then the AUC will equal 1, but with the false introduced an overlap occurs that will lower the AUC.**

Chart, scatter chart

Description automatically generated

<https://ars.els-cdn.com/content/image/3-s2.0-B9780128189467000020-f02-06-9780128189467.jpg>

Part 2: Application

You are tasked to build an image classifier for the MNIST dataset of handwritten numbers, implementing the *k-nearest neighbors (k-NN)* algorithm. You will need the following:

* The MNIST dataset, available on multiple servers on the Internet. For example:
  + <http://yann.lecun.com/exdb/mnist/>
  + <http://www.pymvpa.org/datadb/mnist.html>
* The Python package ***neighbors.KNeighborsClassifier:*** <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

The input to your classifier program is an image containing a digit, 0-9. Your program must correctly identify the digit with an accuracy of 95%. Here the outline of your task, but you will have to do a bit of research on your own (and increasingly so throughout the program) to fill in the details:

Familiarize yourself with the MNIST dataset

Familiarize yourself with the k-NN algorithm and its Python implementation in sklearn

Create a Jupyter notebook for this assignment and implement the k-NN algorithm:

* Import the package kNeighborsClassifier.
* Be mindful of the train-test split and set the parameters accordingly (justify your choice).
* Identify the variables in the dataset and define the Euclidean distance between an element in the test set and the training set.
* Calculate the distance between the test element and each of if its k nearest neighbors.
* Count the occurrence of each digit within the k nearest neighbors and identify the most popular digit.
* Identify the test element as the digit voted as most popular in the set of the k nearest neighbors.
* Classify the test element accordingly (i.e. based on the popular vote).
* Calculate the error.

**Here is my Python Jupyter Notebook Coding:**

[**https://github.com/DouglasBui/GCU/tree/main/DSC-540/Assignment2**](https://github.com/DouglasBui/GCU/tree/main/DSC-540/Assignment2)

Review the article “Handwritten Digit Recognition Using K-Nearest Neighbour Classifier.” Note the algorithms used, but focus on the way the authors:

* Present the findings
* Discuss the findings
* Calculate the accuracy of the results
* Write the article, using professional terminology and content organization

Write a technical report (i.e., not a full-fledged academic paper) to accompany the Jupyter notebook that implements the classifier, using the aforementioned article as a guide on what to address and how to present the mini-project and report the findings.

Address the potential role of a confusion matrix in your report (refer to Part 1).

Address the potential role of ROC curve in your report (refer to Part 1).

**Technical Report**

**David D. Bui**

**Grand Canyon University**

**DSC-540**

**Professor Aiman Darwiche**

**November 10rd, 2021**

**Part 2 Link for Python Code:**

**https://github.com/DouglasBui/GCU/tree/main/DSC-540/Assignment2**

In the analysis of digital recognition, the utilization of machine learning algorithms has allowed computers to derive understanding in cooperation with physical material. A prime example of application can be found in the MNIST dataset, which is an extensive database of 70,000 records, all of which are image files of hand-written numbers. There are a wide variety of classification methods that give algorithms accurate discernment. The primary method that is demonstrated within this report is the K-Nearest Neighbors (KNN) algorithm.

The data has gone through a process of transformation, starting out with individual training and testing sets that were combined and then split again to obtain an ideal split with an 8 to 2 ratio. There is a large range of split preferences within the data community, but most settle towards a 70:30 to 80:20 split, and there are some that view a 50:50 split more ideal. The first column of the data is noted as the target variable while the rest of the variables were pieces of unique images. One issue that I noticed with the data was that some variables were consistent in value, and with such a large range of dimension, I decided to perform a dimensional reduction where I removed variables that only contained the number 0. This reduced my runtime, but slightly decreased my accuracy. In the end it held a better depiction.

The KNN is a lazy algorithm that brute forces its way through each record and is the simplest of nonparametric classifier. The notebook coding goes through a step-by-step process of exactly how this algorithm is implemented. Step 1: Calculating the Euclidean distance between a random newly added test point and the training set. The K value is an important parameter that effects the performance of the model. Unfortunately, K doesn’t have any methods of finding its optimal value, except for iterating through each element and searching for the max for each instance of K, and K can equal up to the number of observations in the dataset.

Step 2: Find the K number nearest neighbors and chart out their number frequency. The method I chose to use involved two layers, the first was distance, and the second was frequency. The frequency held priority, but in instances where there are multiple equal frequencies the distance would become the deciding factor. I chose a point that best illustrated this which later resulted in a correct prediction. Step 3: Involves Classifying the test elements and forming a prediction. I utilized the KNN features to predict the classifications of X\_test.

Step4: Evaluation the results. There are several ways of evaluating the results the model. The first is the overall accuracy and error of the model, the 2nd is the K-Fold Cross Validation method, and confusion matrix. The accuracy is the simplest way of validating the model, but also is the weakest form measure. This was done though the KNN score function and subtracting its value from 1 to depict the error rate. K-Fold CV examines multiple instances of the model before taking the mean of all outcomes. Then lastly is the confusion matrix, that can calculate efficiency but requires a great deal of handling before implementation. All three methods were exploited to evaluate the KNN method, unfortunately I wasn’t able to correctly handle confusion matrix functions so it can only serve as a visual example. Both the accuracy and K-Folds Cross Validation methods yielded high results with over 95% accuracy.

There is a fair bit of difficult in utilizing a confusion matrix in with a KNN that has 10 classes. The traditional 4 box matrix is drastically expanded on in all directions to fit each class, each class will contain its own true/false and positive/negative measures, making the model’s interpretation complex. This same goes for the Receiver Operating Characteristic Curve. The model adapts into multiclass featuring, with each class introduced there is an additional curve added to the model that individually represents the level of separation. This also equates to a great deal of overlap and noise being brought into play. Neither the less, its capabilities of evaluation and interpretation remain.

The overall conclusion is that the K-Nearest Neighbor algorithm was excellent in predicting the classification of digital handwritten numbers. The model was quite intensive in run-time, which is one of the cons it holds, but its results made up for it. Another con is figuring out an optimal K value, the model can suffer significantly if the K value overreaches its similarly labeled neighboring centroid. There is a great deal of sensitivity behind the parameters of this model that makes it susceptible to outliers and needs a great deal of scaling. KNN also finds difficulty when working with datasets with high levels of dimensions, but it works well with images since the variables are pixel-based slices that form easily discerned patterns. The advantages of simplicity and lacking in need or learning, this model is perfect for digital recognition.

**References:**

Narkhede, Sarang, (2018), Understanding AUC-ROC Curve, Towards Data Science, <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

Biome Res Int, (2017), K Important Neighbors: A Novel Approach to Binary Classification in High Dimensional Data, NCBI, DOI: [10.1155/2017/7560807](https://dx.doi.org/10.1155%2F2017%2F7560807)