Topic 2: Supervised Learning and Classification

Jonathan Ibifubara Pollyn

GitHub Repository: https://github.com/JonathanPollyn/Machine-Learning-for-Data-Science

College of Science, Engineering and Technology, Grand Canyon University

DSC-540: Machine Learning for Data Science

Dr. Aiman Darwiche

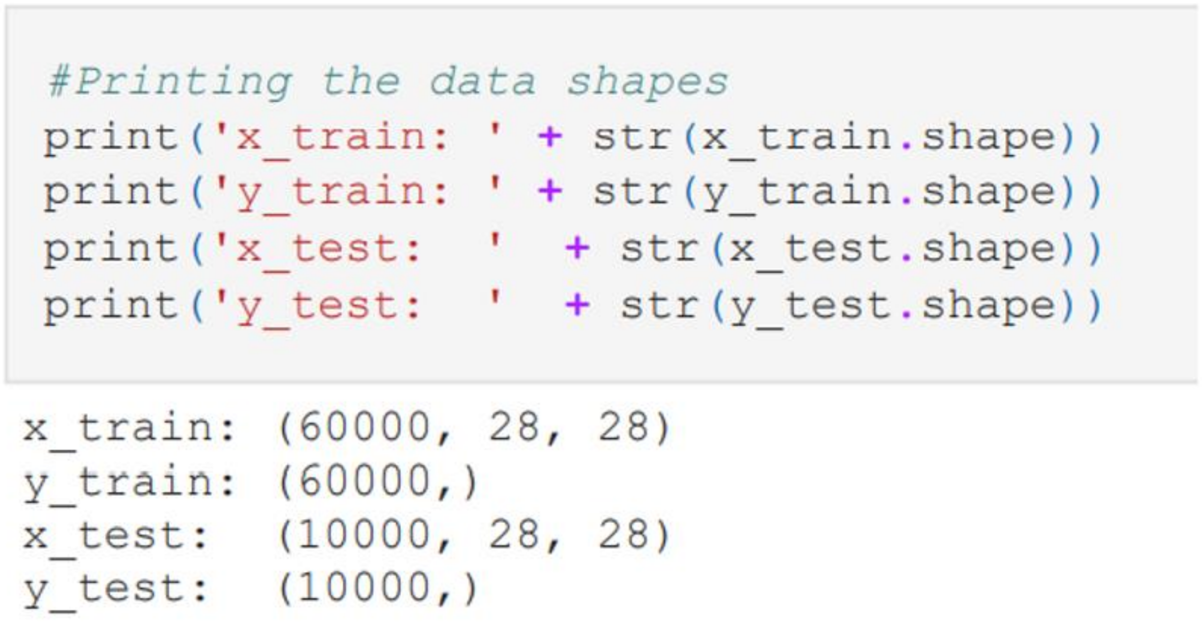
November 10, 2021

Confusion Matrix and ROC Curve

After cleaning, pre-processing, and wrangling the data, we first feed it to an excellent model and, of course, obtain output in probabilities. But these models must be evaluated for effectiveness, where higher effectiveness equals better performance, which is the goal of building a model. The confusion matrix is a performance metric for machine learning classification problems with two or more classes as output. It is excellent for assessing things like recall, precision, specificity, accuracy, and, most crucially, AUC-ROC curves and, for instance, using the TP, FP, FN, and TN through the similarity of pregnancy. We can interpret the True Positive as expecting a favorable outcome and indeed getting our expectations right. For example, correctly guessing that a woman is pregnant, and she is pregnant. The true negative can be interpreted as expecting a terrible outcome, and it came true, for instance, correctly suggesting that a man is not pregnant, and indeed he is not. On the other hand, the False Negative can be predicted as expecting something positive, but it turned out to be false, for instance, expecting that a man is pregnant, but he is not. Finally, the False Negative can be interpreted as expecting a negative outcome, but it was incorrect. For example, correctly guessed that a woman is not pregnant, yet she is. The AUC (Area Under the Curve) and ROC (Receiver Operating Characteristics) curve is used to verify or depict the performance of a multi-class classification issue. ROC curve is a performance statistic for classification issues at various threshold levels. AUC indicates the degree or measure of separability, whereas ROC is a probability curve. It reveals how well the model can discriminate between classes. The AUC indicates how well the model predicts 0 classes like 0 and 1 courses as 1. The higher the AUC, the better the model predicts 0 types as 0 and 1 as 1. By analogy, the higher the AUC, the better the model distinguishes between people who have the condition and those who do not.

Image classification application using MNIST dataset

In machine learning literature, the MNIST dataset is one of the most well-studied datasets. It's a benchmark and a standard against which machine learning algorithms are compared in many circumstances. The purpose of this dataset is to categorize the handwritten numbers from 0 to 9 correctly. The dataset contains 60,000 training and 10,000 testing images. Python programming is used to model the data in a Jupiter notebook. The shape of the data shows that the training and test data are in 2D format with the training data at (60000, 28,28) and the test data at (10000, 28,28), which indicates that the pictures are 28x28 pixels in size. With the picture having a resolution of 28 pixels by 28 pixels, it must be restructured to enable access to every pixel in the image. We can only use deep learning principles and assign color codes to every pixel if we access every pixel. See figure 1: Shape of training and test data.

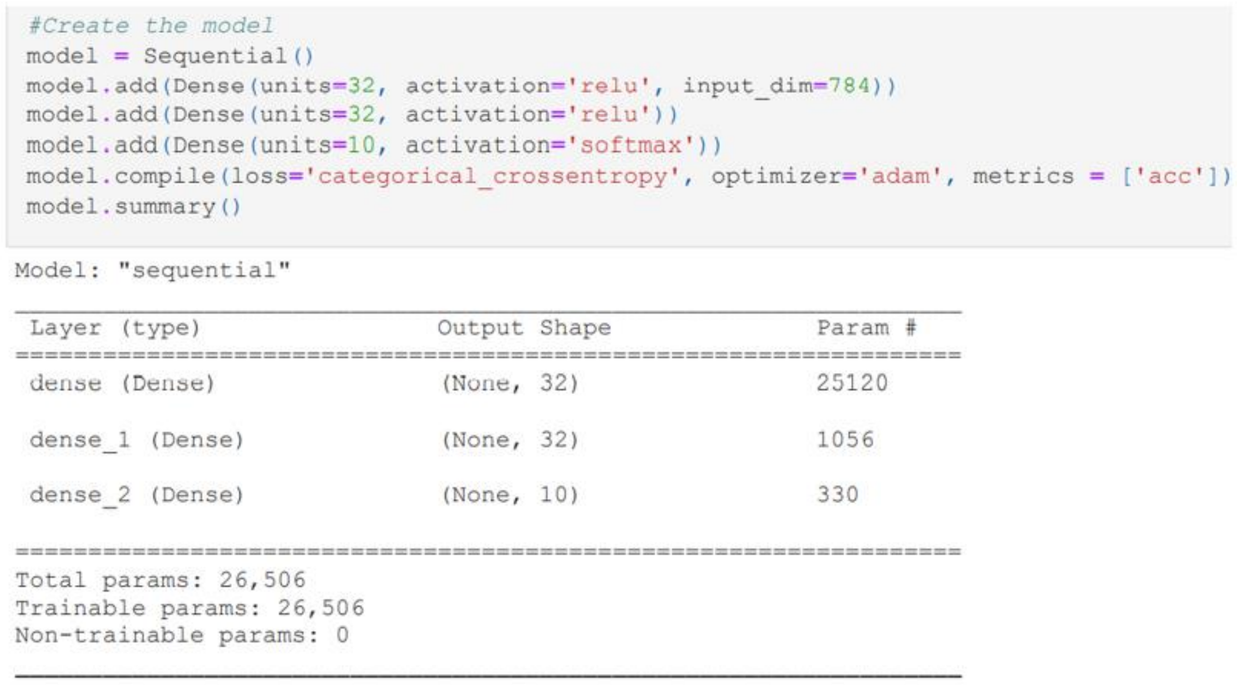


```
#Printing the data shapes
print('x_train: ' + str(x_train.shape))
print('y_train: ' + str(y_train.shape))
print('x_test:  '  + str(x_test.shape))
print('y_test:  '  + str(y_test.shape))
```

```
x_train: (60000, 28, 28)
y_train: (60000,)
x_test:  (10000, 28, 28)
y_test:  (10000,)
```

Figure 1: Shape of training and test data

The function is then stored in the variable model, making it easy to retrieve it whenever needed. Instead of typing the function each time, the variable is utilized by calling it the variable. Then, using the 'relu' as the activation function, turn the picture into a dense pool of layers and stack each layer one over the other. Then 'softmax' is the activation function used to build a couple of additional layers. The complete model is built using cross-entropy as our loss function, Adam as our optimizer, and accuracy as our measure for evaluation. To gain a quick overview of our model, 'model.summary(),' was used to get the overview of the model see figure 2: Trained Model.

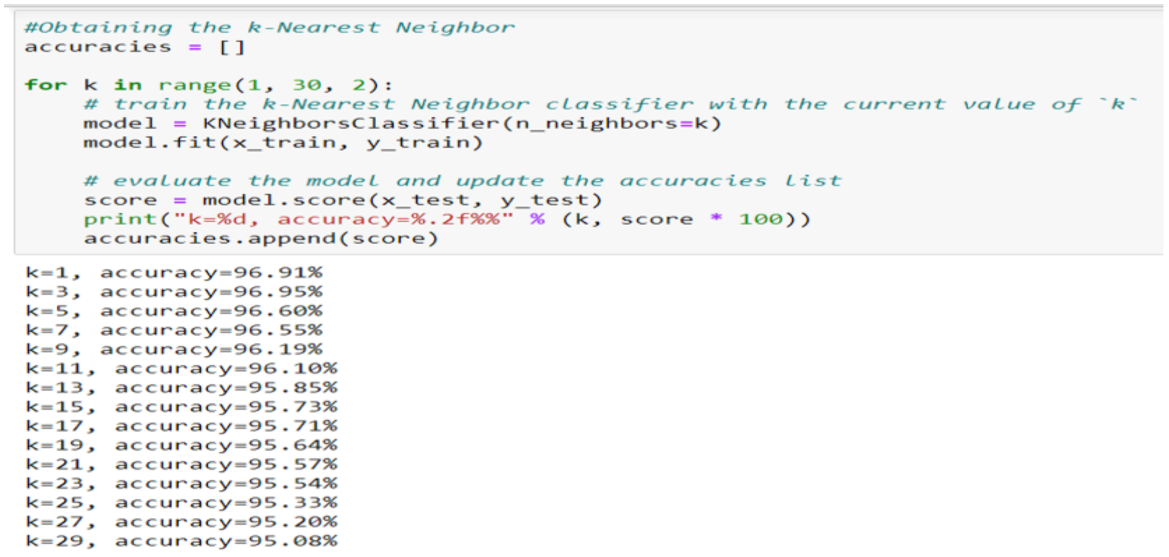


```
#Create the model
model = Sequential()
model.add(Dense(units=32, activation='relu', input_dim=784))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics = ['acc'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                25120

 dense_1 (Dense)             (None, 32)                1056

 dense_2 (Dense)             (None, 10)                330

=================================================================
Total params: 26,506
Trainable params: 26,506
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Trained Model

In the final stage, the model is trained using only one line of code. The .fit() method is used, which accepts the train set of the dependent and independent and dependent variables as input and sets epochs = 40 and batch size to 32. x_train and y_train are the names of the train sets. The term "epoch" refers to a single cycle of training the neural network with all training data. Each epoch consists of one or more batches in which the neural network is trained using a portion of the dataset. To achieve a high level of accuracy, the model is submitted multiple times. Depending on how the model performs, the number of epochs might be modified. The batch size of training samples used in a single iteration is called batch size in machine learning. So, for each iteration, we submit a batch of 32 images to train. As a result, the model achieved a 95.32 percent accuracy for the training data set after training the model. It's now time to test the model in the test set and check if we've reached the necessary accuracy. The model is validated using the test dataset; the scores variable is created to record the value and see how well the model works. The test set of the dependent and independent variables is passed to the evaluate() method. This calculates the model's loss and accuracy in the test set. We print just the correctness since we are focused on accuracy. The score approach examines how many of the k-NN classifier's predictions were correct (the higher the score, the better, indicating that the classifier correctly labeled the digit a higher percentage of the time). Then the score is used to update the list of accuracies, making it possible to discover the value of k that resulted in the highest validation accuracy see figure 3.



```
#Obtaining the k-Nearest Neighbor
accuracies = []

for k in range(1, 30, 2):
    # train the k-Nearest Neighbor classifier with the current value of `k`
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(x_train, y_train)

    # evaluate the model and update the accuracies list
    score = model.score(x_test, y_test)
    print("k=%d, accuracy=%.2f%%" % (k, score * 100))
    accuracies.append(score)
```

```
k=1, accuracy=96.91%
k=3, accuracy=96.95%
k=5, accuracy=96.60%
k=7, accuracy=96.55%
k=9, accuracy=96.19%
k=11, accuracy=96.10%
k=13, accuracy=95.85%
k=15, accuracy=95.73%
k=17, accuracy=95.71%
k=19, accuracy=95.64%
k=21, accuracy=95.57%
k=23, accuracy=95.54%
k=25, accuracy=95.33%
k=27, accuracy=95.20%
k=29, accuracy=95.08%
```

Figure 3: K Parameter tuning

Handwritten Digit Recognition Using K-Nearest Neighbor Classifier

Ravi Babu, U., Kumar Chintha, A., & Venkateswarlu, Y. (2014). Handwritten digit recognition using structural, statistical features and k-nearest neighbor classifier. *International Journal of Information Engineering and Electronic Business*, *6*(1), 62–68. https://doi.org/10.5815/ijieeb.2014.01.07

The authors affirmed that one of the most practical difficulties in pattern recognition applications is offline handwritten character recognition. They stated that digit recognition is used in postal mail sorting, bank check processing, form data entry, and other applications. Digit recognition is critical to the whole system's performance (accuracy and speed). Many algorithms and classification techniques have been proposed in recent years. The authors stipulated that in the performance of an offline character recognition system, the feature extraction and classification approach are critical. For character recognition systems, many feature extraction algorithms have been presented. Handwritten number recognition difficulties have been investigated using dynamic programming, HMM, neural networks, knowledge systems, and combinations of these techniques. The authors stated that due to the available forms, writing style varies from person to person, and many types of noise present, recognition of handwritten English numerals is a complex process. Noise breaks the number character's continuity and changes its topology. In a numerical recognition system, the authors say, feature extraction is crucial. A variety of feature extraction approaches, such as template matching, projection histogram, zoning, and other moment procedures, are described in the literature to enable specialized applications. The author's research introduced a new feature extraction approach based on maximum profile distance. The collection of features comprises the number of holes in an image, features based on the Water Reservoir Principle, maximum profile distances, and filling hole density. The authors organized in order of Database and Preprocessing, a method for extracting features, suggested algorithm, Classification's approach. Specifics of the experiment as well as the outcomes gathered and finally, conclusion.

For the database, the authors suggested the MNIST database, a 60,000-image training set, and a 5000-image test set. The training and test sets are NIST digit base subsets. The MNIST digit database featured fixed-size pictures, with the digit image aligned in the middle of the background pixels. Because the MNIST digit database requires less time for noise reduction in preprocessing, it is helpful for learning techniques and pattern recognition algorithms. NIST's Special Databases, which include grayscale photographs of handwritten numbers, were initially used to create the MNIST database.

Based on the preprocessing step, the authors stated that handwritten numeral recognition might also reach good performance. Pictures in the MNIST handwritten digit database are grayscale images; transform them into binary images. It uses the threshold value to convert grayscale photos to binary. After converting photos to binary, the author stated that excess components might appear in unwanted locations in the backdrop image, referred to as noise. It is essential to eliminate the image's noise; an algorithm must eliminate these undesired noises from the background.

The authors stated that feature extraction is crucial in any recognition system, especially in numerical recognition. For the identification of numbers in this work, structural elements are utilized. The numerical recognition is based on the number of loops in the picture (1 feature), the water reservoir principle (4 characteristics), maximum profile distances (4 features), and fill hole density (1 feature). Each image yields a total of ten unique attributes.

According to the authors, the suggested technique uses the k-nearest neighbor classification algorithm to classify MNIST digit pictures in the test database using the training database's feature vector. A k-nearest neighbor algorithm is a classification approach that uses training information to categorize objects. The authors affirmed that the k-nearest neighbor algorithm's role is to specify the calculations until classification is completed, regardless of the learning techniques used. The authors stated that, in general, there are two learning approaches used in K-NN. They are slowly learning approaches that are based on instances. Next is the K-nearest neighbor approach because the computations are straightforward and the simplest classification technique. The classification of things is based on the votes of their neighbors, denoted by the letter k. In K-NN, objects are assigned to a class based on the most votes.

K-NN computes the distance between the test sample's feature values and every training picture's feature vector value in the k-Nearest neighbor classification. The Euclidian distance measurements are used to determine the majority class among the k-nearest training samples.

Conclusion

In conclusion, the authors affirmed that the recognition of handwritten numbers, ten (10) structural elements were utilized in this study. The most significant difficulty in any recognition process is handling feature extraction and classification methodologies. In terms of accuracy and time complexity, the suggested approach aims to address both criteria. The recognition procedure achieves an overall accuracy of 96.94 percent. This approach is unique in that it is thinning-free, size-normalization-free, accurate, and independent of digit size and writing style/ink. It is also fast and precise. This study is being done as a first attempt, and the goal of the article is to make Telugu OCR more robust. Our long-term goal is to improve this algorithm and create a more robust handwritten Telugu OCR with a high recognition rate and the ability to recognize offline handwritten digits with fewer parameters and without needing a typical classification technique.

References List

Narkhede, S. (2018, May 9). *Understanding confusion matrix | by sarang narkhede | towards data science*. Medium. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Narkhede, S. (2018, June 26). *Understanding auc - roc curve | by sarang narkhede | towards data science*. Medium. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Ravi Babu, U., Kumar Chintha, A., & Venkateswarlu, Y. (2014). Handwritten digit recognition using structural, statistical features and k-nearest neighbor classifier. *International Journal of Information Engineering and Electronic Business*, *6*(1), 62–68. https://doi.org/10.5815/ijieeb.2014.01.07

K-nearest neighbor classification – pyimagesearch. (n.d.).
https://customers.pyimagesearch.com/lesson-sample-k-nearest-neighbor-classification/