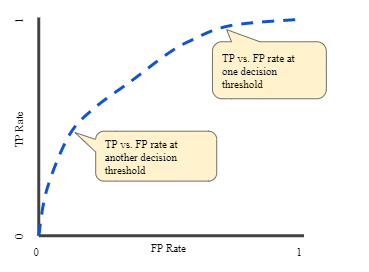
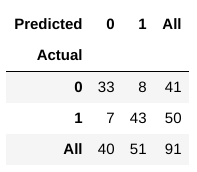
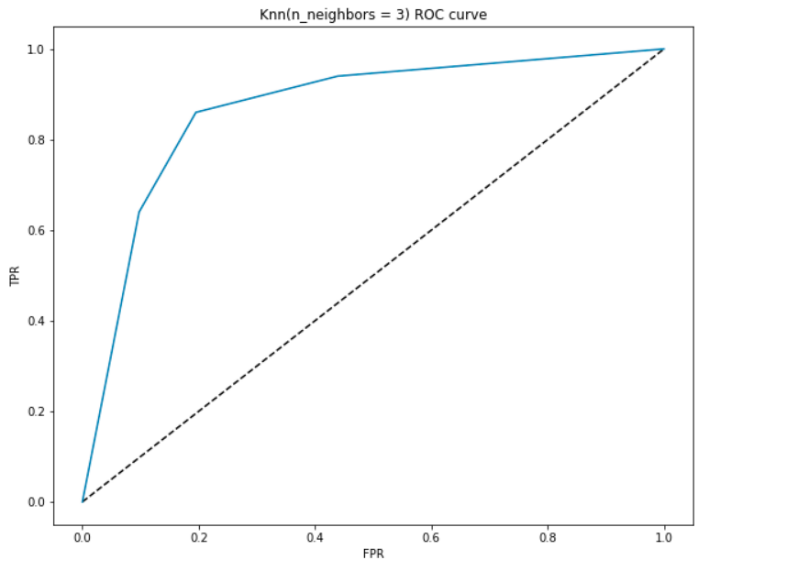
One important tool for evaluation of machine learning models is that of a confusion matrix. The need for this table arises due to the nature of models classifying some records of a dataset correctly and some incorrectly. If we have data points in a testing set, each of which is either relevant or irrelevant to a query , then a model evaluating what class they belong to can produce four possible classifications: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP corresponds to a record that was correctly classified as relevant, whereas TN implies it was correctly identified as irrelevant. FP indicates an irrelevant point was classified as relevant, and FN means a relevant point was classified as irrelevant. The table of these four values can be created as such below:

|  |  |  |
| --- | --- | --- |
|  | Predicted Relevant | Predicted Irrelevant |
| Actual Relevant |  |  |
| Actual Irrelevant |  |  |

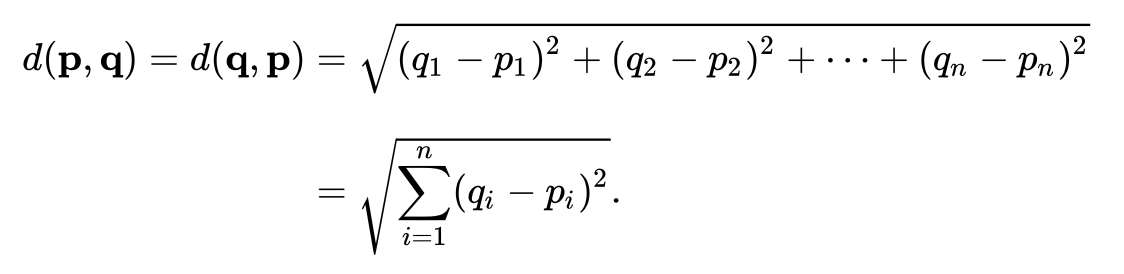
Obviously, we would like to maximize the ratio of true classifications to false ones, as these correspond to a model that is very accurate at predicting whether an input belongs to query . Although this accuracy, defined as the percentage of all results that were correctly classified, is a good measure of the power of the model, more comprehensive understanding of its performance necessitates the introduction of terms and . Precision defines the percentage of retrieved items that are relevant , and recall provides the percentage of relevant items that were retrieved. These figures should also be maximized, but it is very possible that they can differ significantly. Referring to this matrix as retrieval from a query allows us to expand the modeling algorithm beyond binary “True/False” outcomes.

Another metric that is commonly used in conjunction with the confusion matrix is the **receiver operating characteristic** (ROC) curve. Of interest here is the dual rates of TP’s and FP’s. The true positive rate is measured mathematically identically to recall, whereas the false positive rate is simialry calculated as . An ROC curve is one that plots both of these values at different classification thresholds, producing a graph such as the one depicted here. The area under this curve, or AUC, is computed numerically like an integral from 0 to 1. A theoretically perfect ROC curve will run along the left and top lines of this 1x1 cube, giving the AUC a value of 1.

To show the implementation and benefits of these metrics, consider the simple example in the article by Huilgol (2021). Using the popular Heart Disease Dataset from the UCI repository, she performs the kNN classification algorithm on a set of patients to see if machine learning can predict whether or not they have heart disease. The confusion matrix is reproduced below:

As we can see, the precision is calculated to be and the recall comes out as We can also calculate the familiar accuracy measure, which is The theoretical implications of using this matrix allows us to diagnose the strengths and weaknesses of our model. If the precision metric was much higher than the recall, this mathematically indicates a large amount of false positives, meaning we are retrieving records too often. In the opposite scenario where precision is lower than recall, this would naturally imply we are leaving out correct classifications. In the practical sense, that being of this example, precision defines the percentage of patients who actually have heart disease that we correctly identified. A low precision means a high number of sick patients are not treated for the disease. Since recall then explains the amount of diseased patients that we correctly identified as sick, low recall would imply too many healthy people are being told they have heart disease. It can therefore be inferred that low precision carries the risk of medical casualties due to undiagnosed and untreated sick people, whereas low recall brings the possibility of financial consequences for doctors, hospitals, and patients as a result of healthy people having to undergo treatment for a disease they do not have. In this example, the factors at play may induce the preference to save lives over money, and focus the method primarily on maximizing precision over recall.

The ROC curve also gives us insight into the benefits of our model. As shown to the left, we can interpret the theory behind our model by looking at its shape. At low values of , the being high indicates the model is classifying records very well. The dotted line on the graph represents a perfectly linear relationship where , akin to a random guess. Since the ROC curve bends significantly above that line, this indicates that it is much better than simply guessing whether a patient has heart disease. Practically, the ROC tells us that when is small the amount of irrelevant records being left unselected is also small, and the being high means a high number of actually relevant items were retrieved. The overall AUC value of 0.878 also tells us that relying on our model alone, we should expect to correctly distinguish whether a person has heart disease about 87.8% of the time. The main benefit of the ROC curve is that it allows us to visualize both the predictive power and tradeoff between the sensitivity (TPR) and specificity (inverse of FPR) of our model in one graph.

 In our example, in the accompanying Jupyter Notebook, we implement the k-Nearest Neighbors (KNN) algorithm for classification of a set of handwritten Arabic numerals. This method, which is completely data-driven, takes a number of neighbors and for each data point, calculates the Euclidean distance (formula below) that point is away from each other point . After the of these “neighbors” have been determined to be closest, this is then used to “vote” on the classification for that data point.

The probability density function that the records are being drawn from is estimated similarly to a Reimann sum, and although in our case this task is made more accurate due to a sufficiently large , it is also made computationally longer due to a large number of features (784), necessitating computation on a large scale. For each point in the sample, the Euclidean distance away from each other point in feature matrix is found and ranked, which is then used to calculate the bin size (for the Reimann sum) in the smoothing estimating of . Since our method is being used for classification, this needs to broken down by class estimation, i.e. for all classes. The final categorization of every is given by the class label found in most of its -nearest neighbors.

In the implementation of our algorithm, our feature matrix was composed of the pixels of a 28x28 image of a handwritten number. The hue of this pixel was used as input to determine the shape of the digit. We found that our model scored an accuracy of 96.88%, meaning all but 3.12% of digits in the test dataset were correctly identified. This is shown our confusion matrix, computed with the crosstab() function output in the notebook. This allows us to look at how the model performed for each digit individually, thus making way for the computation of our very high values for precision, recall and scores. The ROC curve can be implemented, one digit at a time (it is not valid for multidimensional matrices) to demonstrate the corresponding curve. The practical implication of these findings is that we can identify handwritten numbers with a great degree of accuracy, and future endeavors might be to see if similar results can be found on letters.

In the paper by Babu et al. (2014), the same MNIST handwritten number digit recognition test dataset is used to demonstrate the use of the KNN classification method. The authors employ a number of different methods of feature extraction to detect various characteristics of the numbers to help the computer recognize the handwritten digits. These include the number of loops in the shape, top/bottom/left/right “reservoirs” or holes, the size of the shape and the density of each hole. The authors apply KNN to these extracted features to get the results of applying the training model to the test data. They present their findings by showing a sample (size ) of the data and the computed features, as well as the results of applying the KNN using a few different values of . They found, and displayed in a small chart, that lower values resulted in higher accuracy, and presented the highest accuracy of 96.94. The authors leave out the details of specifically how (and more crucially, from a theoretical standpoint, why) accuracy was calculated, but rather assume the reader already has knowledge of this. Instead, a table is provided that shows the number of images correctly and incorrectly classified. The authors also do not calculate the specificity, precision, recall or sensitivity. It is perhaps presumed that the high accuracy figure is enough to show that the model is valid, as the summary of their findings focuses more on the speed of calculation and size normalization. The authors write the entire article professionally and do well to motivate the reason for their specific method of feature generation. They also list a possible future endeavor, that being of digit recognition using fewer features in their calculations.

References:

Babu, U. R., Venkateswarlu, Y., & Chintha, A. K. (2014). Handwritten Digit Recognition Using K-Nearest Neighbour Classifier. *2014 World Congress on Computing and Communication Technologies*, 60–65. https://doi.org/10.1109/wccct.2014.7

Gopal, M. (2020). *Applied Machine Learning* (1st ed.) [E-book]. McGraw-Hill Education. https://www.gcumedia.com/digital-resources/mcgraw-hill/2019/applied-machine-learning\_1e.php

Huilgol, P. (2021, March 9). *Precision vs. Recall – An Intuitive Guide for Every Machine Learning Person*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/