**Homework 1**

Time series classification with K-nearest neighbors (KNN)

Shruti Avinash Pawar

May 30, 2020

**Class:** ALY 6020 – Predictive Analytics

**Professor:** Dr. Marco Montes de Oca

Businesses in various domains have to make yes/no decisions such as determining a consumer class or classifying diagnosis of diseases. Classification algorithms assist in the binary decision-making process. One of the classification algorithms is K-nearest neighbors (KNN). KNN can be used to build regression prediction problems as well however, this model is widely used for classification. K nearest neighbor classification is based on distances between training data values and test data values. K is the number of neighbors that we will look for to determine the class of the test data value. When k = 1, the classifier will consider the nearest neighbor however when k>1 the classifier will choose the class with highest number of nearest neighbors (Peterson, 2020). Three majorly used distances in KNN are Euclidean distance, Manhattan distance and Minkowski Distance. These distances are respectively defined as below:

Euclidean Distance - 

Manhattan Distance –



Minkowski Distance –



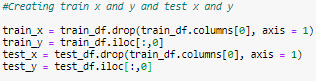
We can see that Euclidean is a summation of squared distance under a root. If in the function of Minkowski, we replace the value of p with 2, we get the Euclidean function and if we replace the value of p with 1, output will be Manhattan distance function(Han & Pei, 2012). In this report, Minkowski distance will be used to build a KNN classification model on a time series data extracted from a thesis formatted by R.Olszewski. The data is a time series of electric activity recorded during one heartbeat. There are two classes -1 and 1 which represent normal heartbeat and Myocardinal Infraction. Python is the tool used to build and predict the KNN classification model.

**KNN Classification Model:**

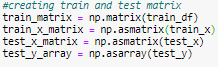
Step 1: The data is available in two txt files named as train and test. Pandas dataframe is used to load the data with space as a delimiter.



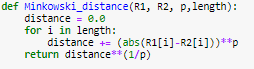
After loading the data, it is segregated into train x and test x which has all the columns apart from class along with train y and test y which only has class of the data stored.

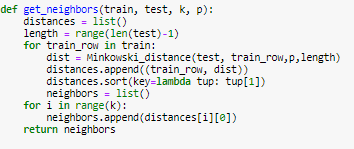


Minkowski distance is a matrix which stores values of distance between test and train samples. Thus, the pandas dataframe is coverted into a matrix using numpy,asmatrix.

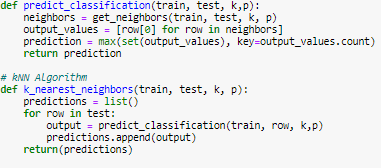


In the above code, test\_y data is stored as an array to compare with the array of predicted values to calculate confusion matrix. The Minkowski distance function is used in determining the class of neighbors. When get\_neighbors function executes, it calculates distance between the train tuples and text matrix for the respective tuples and appends the values of distances and neighbor class in the list. This list is then used in the classification algorithm to predict the class of samples in the test data.

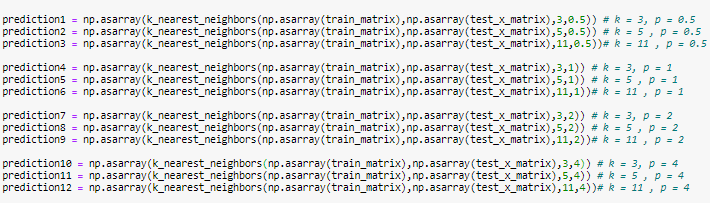




Predict\_classification function predicts the class of the data values in the test dataset and



K\_nearest\_neighbors function loops through every row in the matrix and appends the predicted class in the predictions list. As we can see in the prediction code, the data value is classified based on the maximum number of neighbors within k classes. All the functions mentioned above are interconnected and can be used to call different values of k and p. Using the call function, 12 different models are processed, and accuracy is calculated using the confusion matrix and classification report from library sklearn.metrics.



**Table1:** Summary table of KNN model output for different k and p values

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| p | k | accuracy | True Positive Class | False Positive class | False Negative class | True Negative class |
| 0.5 | 3 | 91% | 29 | 7 | 2 | 62 |
| 0.5 | 5 | 86% | 25 | 11 | 3 | 61 |
| 0.5 | 11 | 81% | 25 | 8 | 11 | 53 |
| 1 | 3 | 91% | 28 | 8 | 1 | 63 |
| 1 | 5 | 87% | 25 | 11 | 2 | 62 |
| 1 | 11 | 84% | 27 | 9 | 7 | 57 |
| 2 | 3 | 88% | 25 | 11 | 1 | 63 |
| 2 | 5 | 85% | 23 | 13 | 2 | 62 |
| 2 | 11 | 82% | 23 | 13 | 5 | 59 |
| 4 | 3 | 74% | 10 | 26 | 0 | 64 |
| 4 | 5 | 76% | 13 | 23 | 1 | 63 |
| 4 | 11 | 74% | 13 | 23 | 3 | 61 |

As we can see in table 1, the model output with k = 3 has the highest accuracy irrespective of the value of p. This is because, the lower the k, the class will be based on the nearest neighbor. Higher values of k will also consider neighbors that are comparatively far to accommodate for the k value. The precision of the model is also highest at k = 3 and since our data is related to healthcare, the motive should be to have more precision i.e. least false negative values in the model output. The model has highest precision at k =3 and p=4 with 0 false negative values. Looking at both precision and accuracy, k =3 and p=1 is the best case that can be chosen from table 1 to finalize the classification model. Alternatively, we can determine the value of k using cross validation. A cross validation function loops through different values of k and calculates the accuracy for all those values. We can then choose the k with highest accuracy to build the classification model. Another thing to notice from table 1 is that the accuracy is dropping with increasing value of p. Thus, generally value of p ranges between 1 to 2 in KNN classification algorithms which would mean Manhattan and Euclidean distances are widely used.

**Conclusion:**

Couple of learnings from this homework are as follows:

1. The structure of the data imported in python is of utmost importance as incorrect data structure results into errors like ‘invalid index to scalar matrix’, ‘int object is not iterable’, etc and incase of KNN its is essential to load the data in a matrix to calculate distances efficiently and to avoid the afore stated errors.
2. KNN algorithm does not learn from the training dataset like other classification models but it calculates distance between each training row and the entire sample and then makes prediction about the class.

**Citation:** Peterson, L. (2020). K-nearest neighbor. Retrieved 31 May 2020, from <http://www.scholarpedia.org/w/index.php?title=K-nearest_neighbor&action=cite&rev=137311>

Srivastava, T. (2018). K Nearest Neighbor | KNN Algorithm | KNN in Python & R. Retrieved 31 May 2020, from <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>

Han, J., & Pei, J. (2012). Minkowski Distance - an overview | ScienceDirect Topics. Retrieved 31 May 2020, from <https://www.sciencedirect.com/topics/computer-science/minkowski-distance>