CSE 5243 - Introduction to Data Mining

Instructor: Jason Van Hulse

Homework 2

Submitted by:

Siva Meenakshi Renganathan
meenakshirenganathan.1@osu.edu

1. OBJECTIVE:

The objective of this homework is to implement a k-Nearest Neighbor (kNN) classification algorithm for both the Iris and Income datasets and analyze the results

i.e. For a given dataset with n tuples the output will be 4n dataframe as follows

Transaction ID	Actual class	Predicted class	Posterior probability
1	Setosa	Setosa	0.55

There are two different datasets:

1) Iris dataset
2) Income dataset.

2. IMPLEMENTATION AND DESIGN:

The program can be split into five major parts:

1) Read input from csv and convert them into dataframes
2) Preprocess the dataframes and convert it into matrices
3) Predict the class for a given k using a distance calculation function
4) Compute confusion matrices and error rates
5) Plot the ROC curve

2.1 READ:

Read.csv() function in R reads the dataset and loads it into a dataframe. The difference between a dataframe and matrix in R is that dataframe can hold different attribute types while matrices cannot.

2.2 PREPROCESS:

Iris and Income datasets have different types of attributes, so each needs some specific pre-processing which is discussed later.

2.2.1 IRIS DATASET:

Iris dataset has five attributes: Petal_length, Petal_width, Sepal_length, Sepal_width and a class attribute. Since Iris is a complete dataset without any missing values or outliers it does not need any pre-processing.

2.2.2 INCOME DATASET:

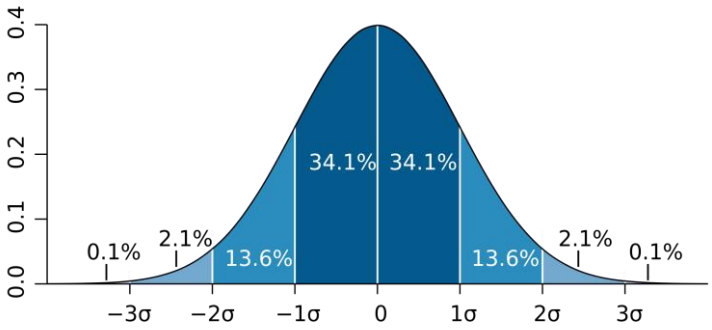
Income dataset is a relatively larger dataset with different attribute types. It also has a lot of missing values and outliers. These have to be handled before calculating the distances.

Handling Missing values aka NA:

- If the type of the missing value is numeric it is replaced with the mean of that column.
  Mean = Sum of all values/No of values.
- If the type of the missing value is non-numeric it is replaced with the mode of that column.
  Mode = Most frequently occurring variable

Handling Outliers:

Outliers can have a significant impact on the range of the data thereby affecting normalization and distance computation. These outliers can be omitted or altered in such a way that it does not cause any anomaly. This program uses Standard deviation to determine the outliers. SD is a measure that is used to quantify the amount of variation of a set of data values. **95.45 percent of the values in any dataset lies between (mean – 2SD) to (mean + 2SD).**



So the code loops over each numeric column to find out values that are either greater than (mean + 2SD) or lesser than (mean - 2SD) and curbs them so that the maximum value of that column will be (mean +2SD) and the minimum value in that column is (mean – 2SD). In some cases like age Mean – 2SD was negative, So the code uses min(column) or (mean – 2SD) as the minimum value.

minval = meanval - 2*sdval
maxval = meanval + 2*sdval
rangeval = maxval – minval

if(minval< min(column)) then minval = min(column)
if(maxval> max(column)) then maxval = max(column)

Age, Capital_gain, Capital_loss, Hours_per_week are the numeric columns here and they are processed for outliers using the formula above.

Normalization:

Normalization or Feature Scaling is the process of converting the range of an attribute to 0 to 1

Normalized value = Value – Min value/(Max value –Min value)

The dataset contains different attributes like age, capital_gain, capital_loss, working hours per week. The range of these values are not the same and hence each of these columns will have a different weight on the distance calculation without normalization. In some cases the attribute fnlwgt is too high that without normalization it eclipses every other attributes and takes the lion share in distance computation. To avoid it the data is normalized.

Age, Education_category, Capital_gain, Capital_loss, Hours_per_week are the numeric columns here and they are normalized using the formula above.

Categorical Attributes:

Income dataset has categorical attributes like Workclass, Occupation, Relationship, Marital_status, Race, Gender, Native_country. Distance calculation for these attribues is simple. If two tuples have the same value for any column, the distance contribution if that column to the total distance is zero and if they have different values, the distance is 1.

Education column is omitted because it is represented by an Ordinal attribute called Education_cat. Fnlwgt is also omitted by the knn distance calculation algorithm as weights are not considered.

2.2.3 CONVERSION INTO MATRICES

Since R processes dataframes very slowly all the data has to be converted into matrices. The categorical attributes in the dataset are given a numeric value, this is different from ordinal attributes as there is no ordering here. Columns like work class, marital status, relationship, race, gender, native country are converted into factors for easier computation.

2.3 DISTANCE CALCULATION AND PREDICTION:

2.3.1 EUCLIDEAN DISTANCE:

Each row in the test set is reinforced with all rows in the training set and the Euclidean distance of that row with other rows are calculated and it is stored in a temporary matrix. This temporary matrix is then sorted based on the distance in ascending order.

$$dist = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2}$$

Where p and q are the two rows and it has n columns.

2.3.2 KNN CLASSIFIER:

A Knn classifier calculates the distance of a tuple in test set with all tuples in the training set and based on the class of its k-neighbors, the code predicts the class of that tuple taken from test set.

When k is odd either of the number of neighbor positive classes will have at least one more entry than the number of neighbor negative classes or vice versa, the predicted class will be the one which has highest entries. When k is even there may be a case when the number of neighbor positive classes is equal to number of neighbor negative classes. In such case the code chooses the closes neighbor class as predicted class.

Posterior probability of a class is determined by the number of neighbors in majority class / total number of neighbors considered. In case of a tie, posterior probability is always 0.5.

2.4 CONFUSION MATRIX AND ERROR RATES:

2.4.1 CONFUSION MATRIX

A confusion matrix is used to determine the efficiency of any classifier. It has the number of true positives, true negatives, false positives and false negatives in a table manner. Iris dataset deals with 3 classes, so its confusion matrix has 3 rows and 3 columns. Income dataset has 2 rows and 2 columns.

Confusion Matrix for Iris dataset k=1

Actual \Predicted class	Setosa	Versicolor	Virginica
Setosa	20	0	0
Versicolor	0	22	7
Virginica	0	2	18

Confusion Matrix for Iris dataset k=1

Actual \Predicted class	Postive <=50k	Negative >50k
Postive <=50k	197	24
Negative >50k	47	20

2.4.2 ERROR RATES:

Accuracy = (True positive + True negative) / Total number of rows
How many rows were classified right by the classifier

Error = (False positive + False negative) / Total number of rows = 1- Accuracy
How many rows were classified wrong by the classifier

True Positive Rate = True positive / (True positive + False negative)
Number of rows truly classified as positive. Should be high or 1.

True Negative Rate = True negative / (True negative + False positive)
Number of rows truly classified as negative. Should be high or 1.

False Positive Rate = False positive / (False positive + True negative) = 1 – TNR
Number of rows falsely classified as positive. Should be low or 0.

False Negative Rate = False negative / (False negative + True positive) = 1 – TPR
Number of rows falsely classified as negative. Should be low or 0.

Precision = True positive / (True positive + False positive)
Recall = True positive / (True positive + False negative)

Eg: -------- Income dataset Implementing knn for k = 14----------
TPR, FNR, FPR, TNR Rates

Actual \Predicted class	Positive <=50k	Negative >50k
Positive <=50k	TPR= 0. 9185520	FNR= 0. 0.08144796
Negative >50k	FPR= 0.6716418	TNR= 0.32835821

Accuracy rate: 0.781250
Error rate: 0.218750
Precision: 0.818548
Recall: 0.918552
F measure: 0.432836

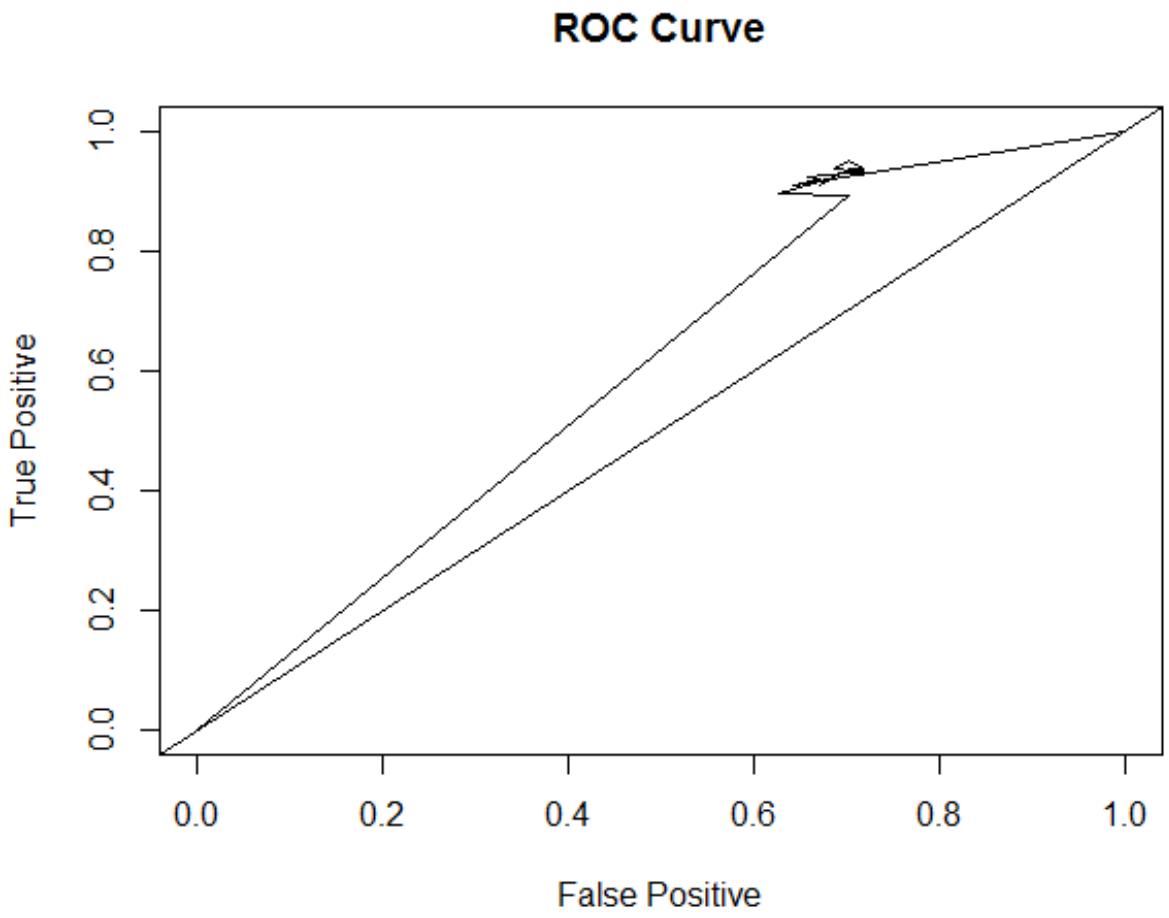
2.5 ROC CURVE

ROC curve is used to analyze the performance of any classifier. It is a function that plots true positive rate on y-axis against false positive rate on x-axis. The curve repeats this for different values of k. the goal is to achieve a (1, 0) condition, i.e. true positive rate = 1 and false positive rate = 0. A diagonal line is drawn which represents random guessing. Lift is the improvement in performance.

Values used to plot ROC of income dataset:

"Values on ROC curve:"

K	1	2	3	4	5	6	7	8	9	10
TPR	0.891	0.891	0.895	0.923	0.909	0.936	0.936	0.950	0.936	0.932
FPR	0.701	0.701	0.626	0.686	0.671	0.701	0.716	0.701	0.686	0.731



3. ANALYSIS:

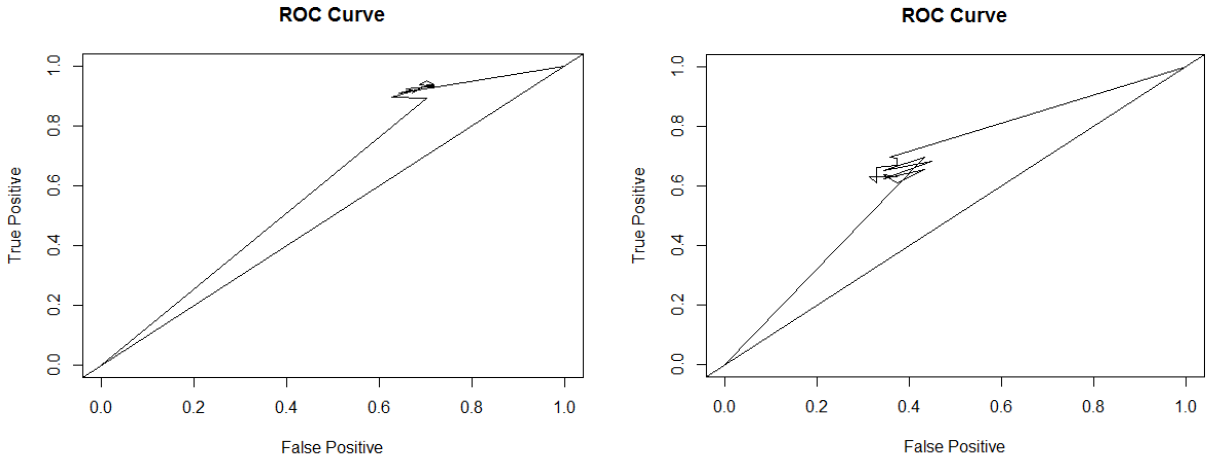
The time taken by knn classifier mainly depends on the size of the training set. No optimizations can be made to the knn classifier as it always has to calculate the distance for all rows in the training data. The accuracy increases with increase in k up to a certain point and thereby drops again. The point where the lift is maximum corresponds to the most suitable value of k for the knn classifier. Generally the False positive rates go down with increase in K.

These results are printed by the program in console

4. MODIFICATIONS:

4.1 CHANGING DISTANCE METRIC:

Manhattan distance method is now used by the knn classifier and the results are as below. It seems that Manhattan distance doesn't perform well from the ROC curve. It might due to the fact that Euclidean uses RootMeanSquare value while Manhattan doesn't.



4.2 VARIATION OF K

There is a significant change in the TPR and TNR values when N is changed. **Generally the False positive rates go down with increase in K**. On testing with different values for k from 1 to 15, for k = 15 Iris dataset gave very high accuracy rate = 0.87. For income dataset k=9 performed well with Accuracy rate = 0.791.

4.3 VARIATION OF TRAINING DATASET:

It is observed that the size of training dataset affects the performance of the classifier in two ways. Increasing the size of training dataset increases the time taken to classify but in most cases produced better results. The INCOME_NEW dataset was used to infer these results. Income_new contains 3256 rows which is almost 8 times the size of the previous training dataset.

For k =2

INCOME SMALLER TRAINING SET: Accuracy rate: 0.753472, Error rate: 0.246528

INCOME LARGER TRAINING SET: Accuracy rate: 0.756944, Error rate: 0.243056

As the values of n increases, this effect is profound.

For k = 8

INCOME SMALLER TRAINING SET: Accuracy rate: 0.725694, Error rate: 0.274306

INCOME LARGER TRAINING SET: Accuracy rate: 0.774306, Error rate: 0.225694

5. CHANGES FROM ASSIGNMENT 1

The usage of dataframes in Assignment 1 increased the time complexity of the code. Conversion of dataframes into matrices in Assignment 2 has reduced the time taken to run the code exponentially.