**Predictive Analytics**

# ALY6020 Module 1 Assignment

# Faculty: Prof. Na Yu

# 2020 Spring CPS Quarter

# Image result for neu cps

Shraddha Gopalakrishnan [NUID: 001376444]

Submitted On: 04/11/2020

**Introduction**

In this era where we are surrounded by data, it is important for students in STEM programs to know what goes beyond this “BIG DATA”. Machine learning, a branch of computer science has done wonders to the world i.e., has contributed towards making human’s life simpler. Predictive analytics being one sub-category within ML amalgamates mining of information and machine learning using various statistical techniques and tools. Computer algorithms are major catalysts to train a given model to facilitate the process of artificial intelligence and this is where KNN comes into picture. KNN (K-Nearest Neighbor) is a supervised machine learning algorithm used for classifying datasets into most suitable clusters. Let’s dig into the implementation of KNN for few data abstracts.

**Analysis**

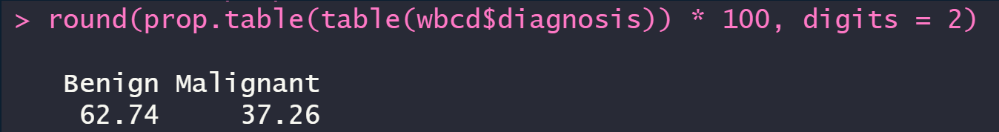
*Data:*

We have two parts in our homework and in both the parts, the steps followed will be same for R programming only difference being the datasets. We make use of RStudio, an open source software to develop data science models easily using pre-defined packages.

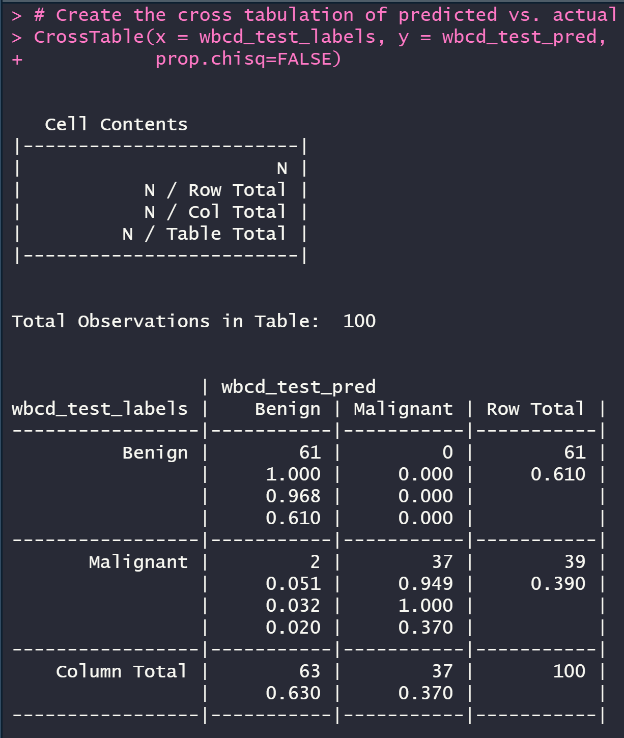
*Part 1:*

Here, we have a dataset based on breast cancer samples from a hospital in Wisconsin. In the datasheet, there are columns like ID and Diagnosis with 30 other real-valued features and 569 measurements are obtained from the nuclei of breast samples. Dr. Wolberg did study of cells and extracted features using which we will make predictions with the help of KNN.

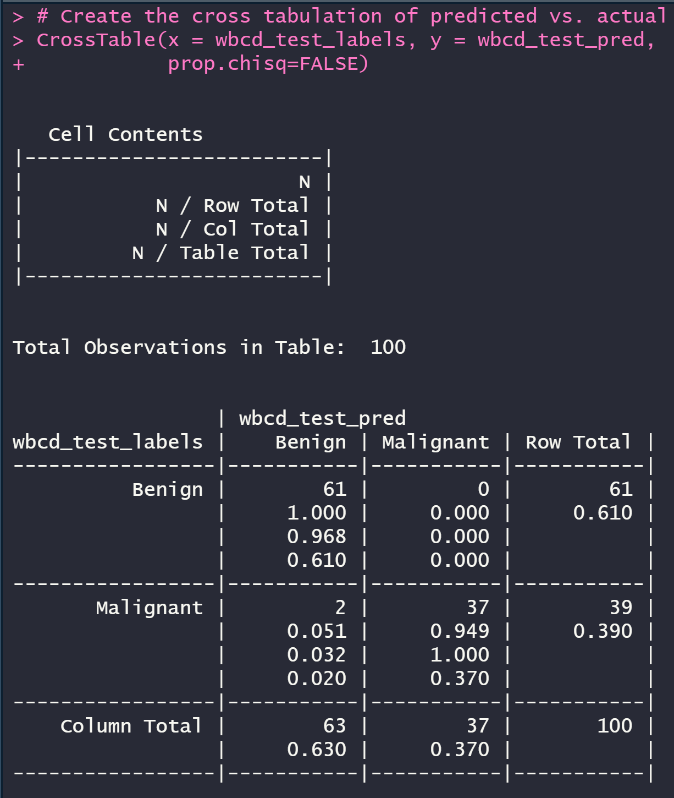
First, we load the data into our RStudio and view its structure. Looking at the same, we find that there is 1-integer datatype, 1-character datatype on which we classify and predict, and the rest are numerical variables. The id column seems irrelevant and so we get rid of it. Now our data has 569 observations and 30 variables. We replace the values in diagnosis column to factors by specifying the levels. We use the following code to just check the ratios of both factors. The proportion obtained is 62.74% Benign and 37.26% Malignant (decimal places mentioned as 2).



As the values in different features have different ranges, we normalize the dataset to bring all the values between 0 and 1. This is done to maintain smoothness in the overall trend. Remember while normalizing, do not include the character variable column as lapply () function only works for numerical not categorical variables. R libraries like “class” for knn function, “gmodels” for cross-table function and model fitting are utilized.



The Cross Table shows values and ratios for predicted and actual values. The cell contents are Row and Column in a tabular form. Here, we keep the proportion test as FALSE as we don’t need the chi-square values to be considered for measurement.

Since our training set contains 100 observations, our Cross Table shows scaled values totaling to a 100. We read the table as follows:

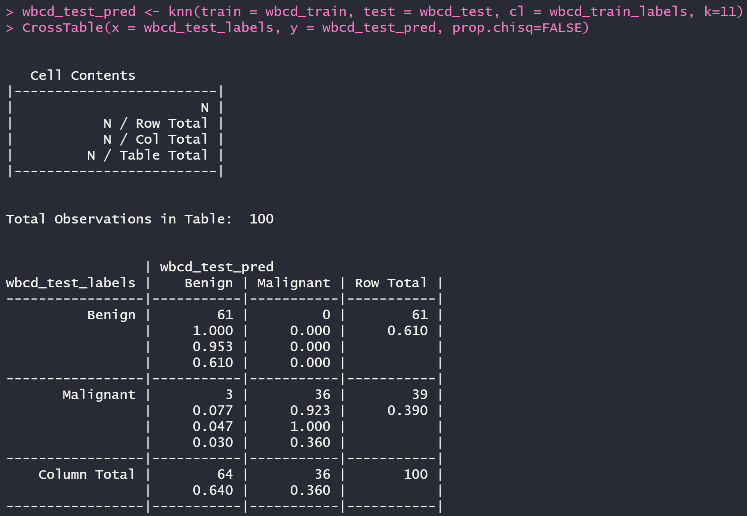
**True Negative**: 61 values correctly identified Benign and hence show Breast Cancer as negative.

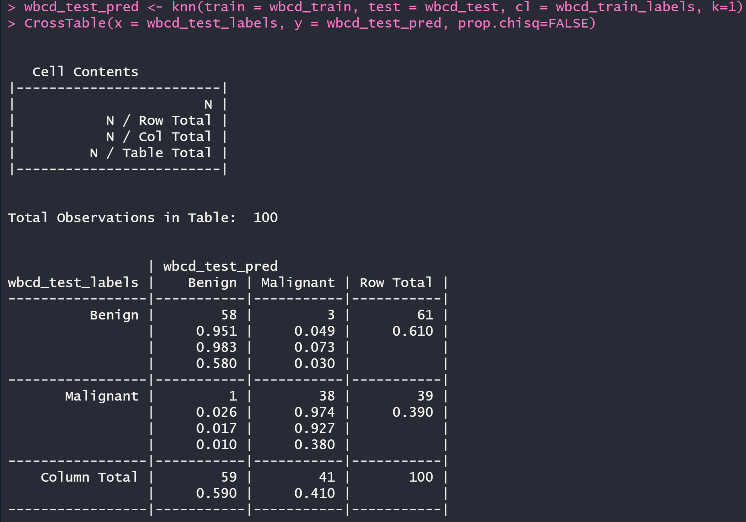
**True Positive**: 37 values correctly classified as Malignant testing positive for breast cancer.

The bottom-left value 63 is **False Negative** i.e., 63 malign masses are incorrectly classified as benign. This stat is crucial as it means that a person has cancer, but the tests came out negative.

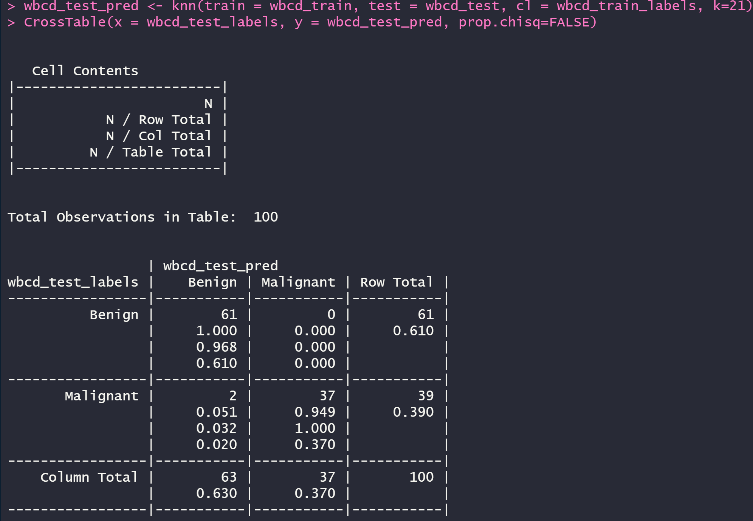
To improve accuracy of this model z-score standardization is taken into account. Normalization might not always yield best of results and so we try a proxy method. Table obtained after z-score scaling our data frame shows **66 TN** and **34 TP** values.

Next we try different values of k to get new tables.

*k=1 k=11*



*k=21*

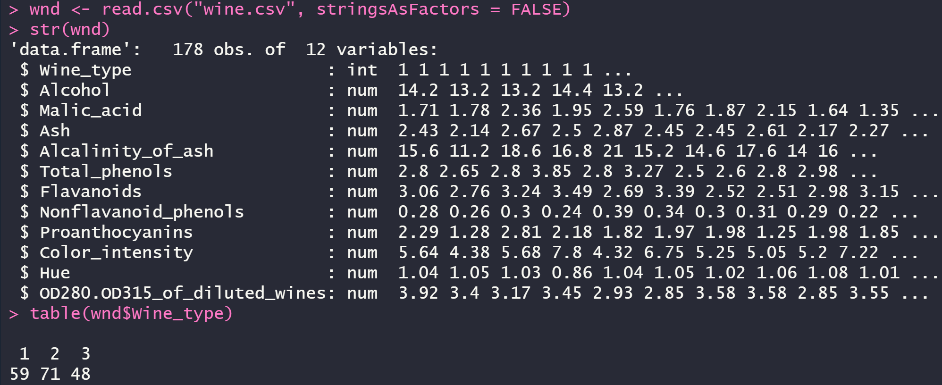
**

We see variations with different values of k. My perspective of choosing **k=21** for our model because it it’s the one which shows least percentage of False Negative. Only 2% cells are incorrectly classified as Benign and hence is the optimal solution for our KNN problem.

Thus, our analysis for *Part 1* states that we found 98% of cells anticipated correctly with our Nearest Neighbor value **k=21**.

*Part 2:*

This part is a simulation of the above KNN process with a different dataset. For this segment, I tried to find unique datasets and executed them but not all the datasets fit knn algorithms which brought me back to the simple yet informative **Wine Data set**. It contains measurements of a chemical analysis for 3 different kinds of cultivated wines in the Tuscan region of Italy. Wine data has 1-integer attribute and 11-numerical attributes. The use of KNN here is to correctly classify the different kinds of Wines.

We load our data on to RStudio, view the structure and see that there are 178 observations and 12 variables including **Wine\_type**.

To see the classification of types of wine we use table function. Looking at the image besides, we have these stats:

Wine\_type 1 -> 59

Wine\_type 2 -> 71

Wine\_type 3 -> 48.

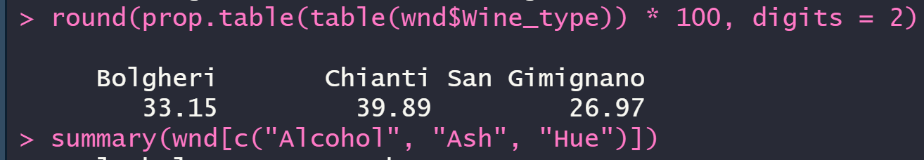
Since the name of wine types as 1,2,3 is very vague, I researched a bit on them and based on their Alcohol content (“1” <12.5%, “2” 12.5-13.5%, “3” >13.5%) categorized them as-

Wine\_type 1 -> Bolgheri

Wine\_type 2 -> Chianti

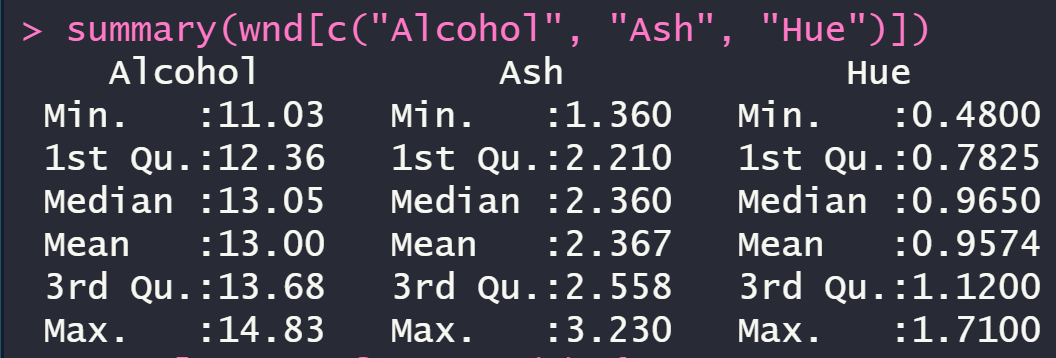
Wine\_type 3 -> San Gimingnano

The 11 variables are chemical statistics such as Alcohol content, Malic\_acid, Ash, Hue, etc and we will use these as predictors for our classification.



Here, we notice the proportions of Wine types as **33.15%**, **39.89%**, and **26.97%** for **Bolgheri**, **Chianti**, and **San Gimignano** respectively.

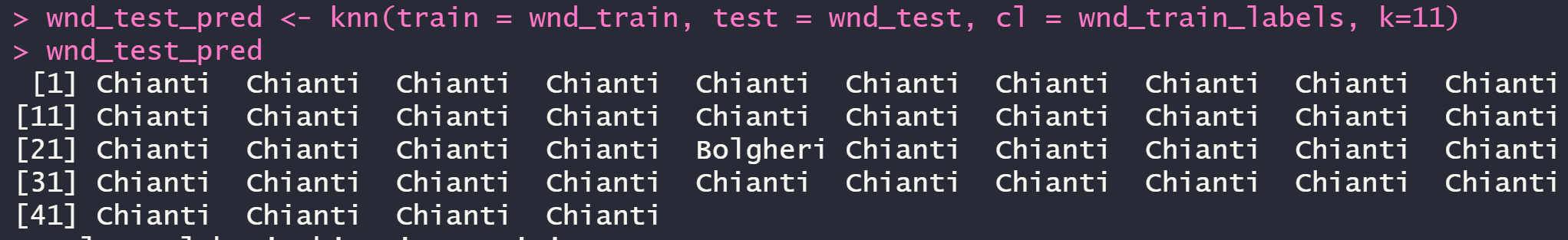
Next we move on to checking the summary of 3 numerical features Alcohol, Ash and Hue.



It’s clear that the values have a set of varying ranges and hence to keep a uniform pattern and bring the range between 0 and 1 we normalize the data frame excluding the first column of Wine\_type.

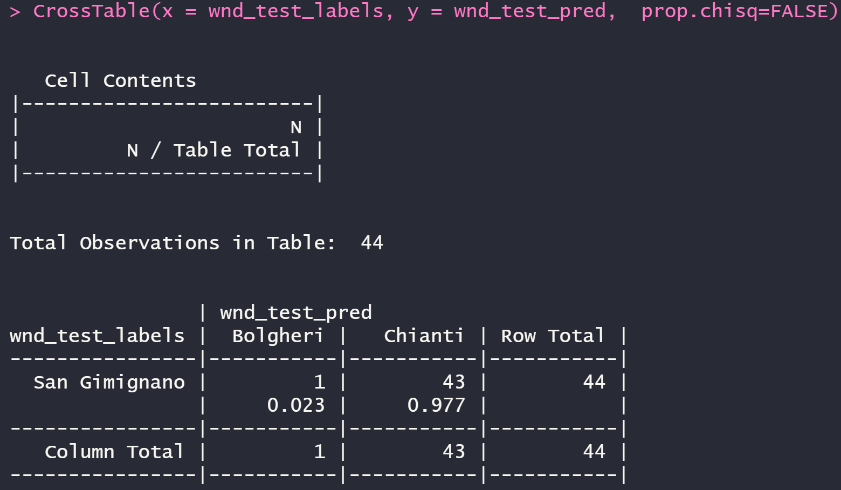
I always prefer dividing the training and testing datasets in the 75:25 ratio for machine learning algorithms. Training for implementing the KNN algorithm and testing to find the accuracy. Along with this we create train-test labels for our types of wine column.We install packages “class” and “gmodels” for our computations.

The test prediction dataset is:



There are 44 observations in the prediction test dataset. Function knn () is used with k=11 as it is best to keep k value appropriate to an odd number closer to the square root of count of training set.

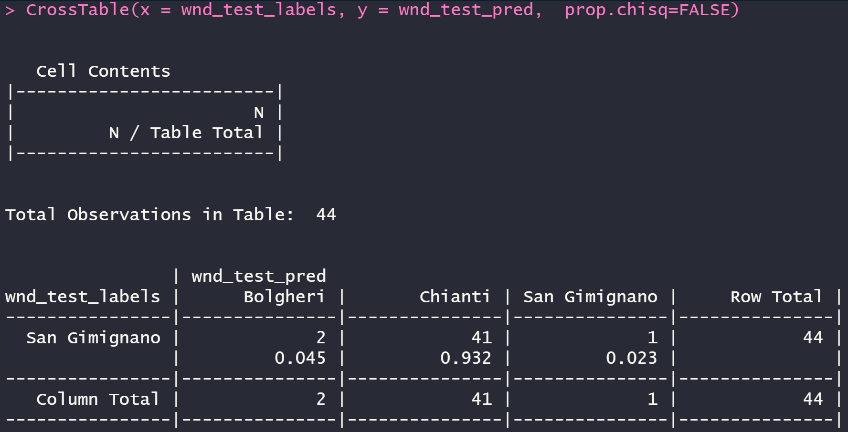
Scaling helps speed up the knn process. Before scaling, the cross-table output looks like:



The problem with CrossTab () function is it’s made for viewing maximum 2-way cross tabulations. Here I have 3 kinds of wine and hence the function automatically wraps into a single row header.

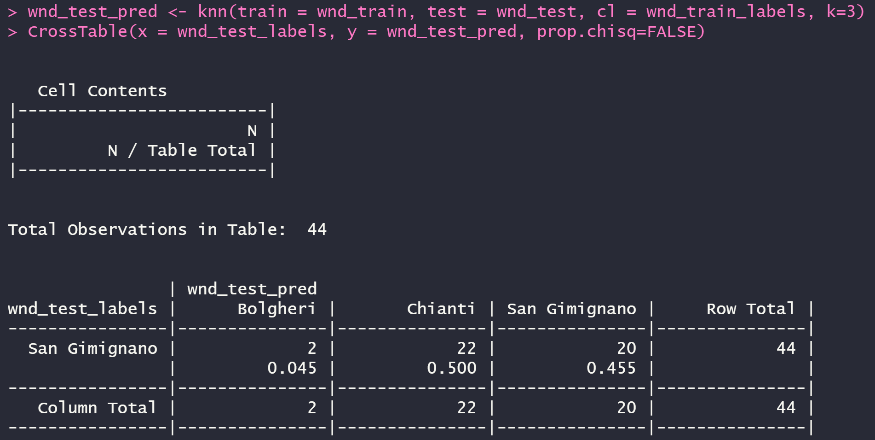
It is estimated **23%** **San Gimignano** wine is wrongly classified as **Bolgheri** and **97.7%** as **Chianti**. There is nothing about San Gimignano classified as San Gimignano.

Let us try the same after scaling using z-score.



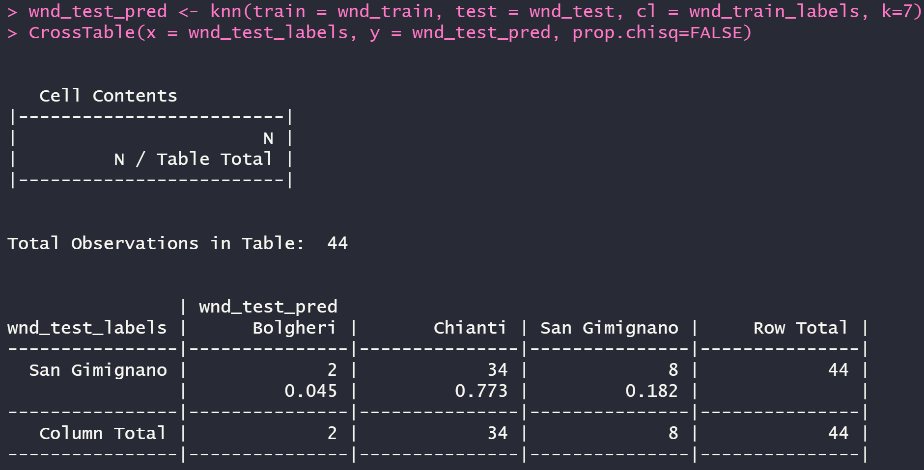
Now we see a more accurate result as there is an estimation for all the 3 Wines. San Gimignano wine is **4.5%,** **93.2%, 2.3%** incorrectly classified as **Bolgheri**, **Chianti**, **San Gimignano**.

The count values in the same order are 2, 41 and 1 out of 44 test predictions.

Then we examine how our KNN model performs for different values of k. Since cross tabulations is only with respect to one row it can be confusing for interpreting. In simple words we find the **k value** such that at least one of the Wines is 0% incorrectly classified.

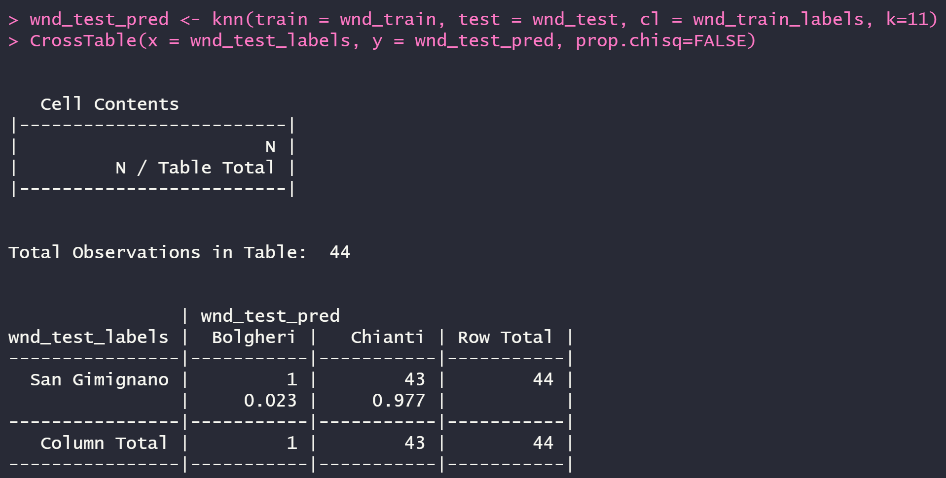
The right image is for *k=3* showing **4.5 : 50 : 45.5** incorrect proportions for 3 wines.

*k=7*



Here the ratio is **4.5 : 77.3 : 18.2**.

*k=11*

**

The percentage ratio becomes

**2.3 : 97.7 : 0**.

This is where we stop and finalize the k-value.

The result turned same for even *k=13*.

Looking at all the 3 tabulations, it is clear that the percentage of incorrect classification went 0 for last table. Even though the percentage of incorrect classification for **Chianti** is super high, the model is better with minimal errors. K=3 and k=7 show incorrect partings for all 3 wines but only k=11 tells us that **there were no San Gimignano wines wrongly predicted**. Meaning, if you look in the tables, the San Gimignano column values are 20, then reduce to 18 and finally 0 for *k=11*.

This also portrays our KNN model has learnt through repeated computations and experience. This is nothing but Machine Learning and we have successfully obtained our KNN model with true and best predictions.

**Conclusion**

Even though the classifiers for Part 1 and 2 were not perfect, we were able to avoid false negatives of one or more categorical variables. Moreover, we tailored our models on test data which in real-world scenarios has a greater number of surveillance than the datasets we worked on.

We are now well-versed with the k-nearest neighbor classification algorithm of predictive analytics. We learnt that KNN unlike others, does not do any learning. It just stores the training samples and when test set is executed the unlabeled records are matched via distance function and are assigned the labels closest in distance to that datapoint.

Using R codes, the practical turned out to be less complex and more accurate in making predictions.

**References**

[1] Lantz, B. (2015). Machine learning with R: learn how to use R to apply powerful machine learning methods and gain an insight into real-world applications. Birmingham: Packt Publ.

[2] (n.d.). Retrieved from <http://archive.ics.uci.edu/ml/datasets/wine>