Advanced Analytics

Project 1

Business report

By Kirill Samaray

“Diagnosing Breast Cancer”

The purpose of the project was to use machine learning algorithms in helping patients and clinics to identify cancerous cells. By automating the process of detection, it may reduce the time spent on examining the cells by a doctor and potentially could provide a better accuracy due to eliminating a human factor since some healthcare specialists might potentially make mistakes while diagnosing. The algorithm chosen for this project was kNN. With the help of this algorithm the computer studied the biopsied cells from both of the genders in order to identify cancer.

At first, we collected the data from wis_bc_data.csv provided by the University of Wisconsin. It included the data needed for the algorithm to perform its functions. The data set consisted of 569 examples of cancer biopsies with 32 features each. Features included ID, diagnosis, and other important metrics such as: radius, texture, perimeter, etc.

Next, we explored and confirmed the data by checking its structure using str() command. We could see that the data consisted of various types of data such as integers, characters, and numeric values. Since we didn’t need the ID data to be used with our algorithm, we removed it. Since the first thing we wanted to extract was the information about the diagnosis provided by the doctors, we created a table with diagnosis values and then labeled them properly indicating the percentage for each of the cases.

As a matter of fact, kNN algorithm needs a distance function that would determine the similarity between several features of the dataset. In order to identify similarities, we need to have a standard range for all the features in the dataset before using the algorithm. Otherwise, the distance will be skewed in case some features are very different rom each other in terms of its values. That was exactly the risk we were running into with our project because such values of the features as points_mean, perimeter_mean, texture_mean and some others varied significantly. We had to utilize normalization to bring the values to a standard range of values.

```
> summary(project_data[c("points_mean","perimeter_mean","texture_mean")])
  points_mean      perimeter_mean    texture_mean
 Min.   :0.00000   Min.   : 43.79   Min.   : 9.71
 1st Qu.:0.02031   1st Qu.: 75.17   1st Qu.:16.17
 Median :0.03350   Median : 86.24   Median :18.84
 Mean   :0.04892   Mean   : 91.97   Mean   :19.29
 3rd Qu.:0.07400   3rd Qu.:104.10   3rd Qu.:21.80
 Max.   :0.20120   Max.   :188.50   Max.   :39.28
```

Thus, with the help of normalize() function we could bring the values to the same range and apply this function to our numeric elements in the data set. We also used lapply() function to apply the normalization function to all the elements of our data set at once.

The next step was creating training and test datasets to see how the efficiency of our prediction model could be compared to the traditional method of diagnosing patients by doctors. We divided our dataset with 80% (455) being allocated for the training and 20% (114) for test parts including diagnosis information as our target variable.

As a matter of fact, kNN algorithm is known to be a lazy algorithm hence it didn't build any model but rather kept the data in a specific format. We used knn() function to implement kNN algorithm and to distinguish the closest distance to either of the k-Nearest Neighbors. We selected k=21 since as a rule of thumb we could select a square root of the number of items in the training set. This number represented the number of neighbors.

The next step for us was to assess the performance of the model by comparing a vector with predicted values to a vector with predetermined values. With the help of CrossTable() function we could easily compare both of the tables.

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  114

                         | project_data_test_pred
project_data_test_labels |    Benign | Malignant | Row Total |
-------------------------|-----------|-----------|-----------|
                  Benign |        70 |         0 |        70 |
                         |     1.000 |     0.000 |     0.614 |
                         |     0.972 |     0.000 |           |
                         |     0.614 |     0.000 |           |
-------------------------|-----------|-----------|-----------|
               Malignant |         2 |        42 |        44 |
                         |     0.045 |     0.955 |     0.386 |
                         |     0.028 |     1.000 |           |
                         |     0.018 |     0.368 |           |
-------------------------|-----------|-----------|-----------|
            Column Total |        72 |        42 |       114 |
                         |     0.632 |     0.368 |           |
-------------------------|-----------|-----------|-----------|
```

TN FP FN TP

As shown on the picture, we labeled all 4 categories with TN(True Negative), TP(True Positive), FP(False Positive) and FN(False Negative) respectively. TN and TP values represent the cases when both the

algorithm and clinical results match with the mass being either malignant or benign out of all 114 cases. TN and TP values in this case constituted 61.4% and TP 36.8% respectively. The other values in the cells (FP and FN) represent the cases where the kNN algorithm didn't match with the real data. FN values represent the cases when Benign label was predicted by the algorithm although the clinical results showed a malignant case. On the contrary, FP cell demonstrates the cases where Malignant tumor was predicted although the true label was diagnosed as benign. In our examples there were no FP values with only 2 FN values. Therefore, we could achieve the accuracy of 98.2%.

After reaching a certain degree of accuracy we decided to adjust the algorithm in order to check whether a better accuracy could be achieved. Even though 98.2% could be treated as a high level of percentage, considering a sensitive nature of the cases tested, we tried to improve the model. First, different types of normalization were applied. Previously used Range Normalization was substituted with z-score one. Due to the nature of some tumor specifications, some outliers may play a more important role in distance calculation. We used scale() function with our data frame and came up with a z-score version of our standardized data. Other steps were identical to our Range Normalization procedure with dividing our data set into training and test ones and then applying knn() function.

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  114


                         | project_data_test_pred
project_data_test_labels |    Benign | Malignant | Row Total |
-------------------------|-----------|-----------|-----------|
                  Benign |        70 |         0 |        70 |
                         |     1.000 |     0.000 |     0.614 |
                         |     0.933 |     0.000 |           |
                         |     0.614 |     0.000 |           |
-------------------------|-----------|-----------|-----------|
               Malignant |         5 |        39 |        44 |
                         |     0.114 |     0.886 |     0.386 |
                         |     0.067 |     1.000 |           |
                         |     0.044 |     0.342 |           |
-------------------------|-----------|-----------|-----------|
            Column Total |        75 |        39 |       114 |
                         |     0.658 |     0.342 |           |
-------------------------|-----------|-----------|-----------|
```

As shown on the picture above, the outcome of the new approach showed a reduced accuracy – 95.6%. The takeaway from that was that we didn't reach a desired accuracy by classifying our FN cases.

Another attempt to improve the accuracy was made with the help of adjusting the values of k. We tried using the same dataset with 114 values. Each one of them showed a worse accuracy than the one we achieved earlier with k value of 21. Our attempts to avoid FN cases were successful, however, it resulted in generating FP cases which should be also avoided.

K value	#(FN) false negatives	# (FP) false positives	Percent classified Incorrectly
3	2	2	3.5
7	3	1	3.5
14	3	1	3.5
25	5	0	4.4

We came to a conclusion that with the help of kNN algorithm, we managed to reach 98.2% accuracy with our method of diagnosing patients. We properly collected, explored, prepared, and transformed the data. Furthermore, we trained the model and evaluated its performance. After determining some of the model's imperfections we tried to improve the model, however, a better level of accuracy was not achieved. We learned that with a simple code we were able to eliminate some human error and automate the process of tumor diagnosing. In future the outcomes of our project could be used to apply in real application domains with potential adjustments and improvements of the model in order to achieve 100% accuracy.

Coding explanation:

Step 1 **Collecting data**

```
> install.packages("class")

> library(class)

> project_data = read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
```

Step 2 **Exploring and preparing the data**

```
> str(project_data)

> project_data = project_data[-1]

> project_data$diagnosis = factor(project_data$diagnosis, levels = c("B","M"),labels =
c("Benign","Malignant"))

> round(prop.table(table(project_data$diagnosis))*100,digits = 1)

> summary(project_data[c("points_mean","texture_mean","perimeter_mean")])

> norm = function(x){

+ return ((x-min(x))/(max(x) - min(x)))}
```

```
> project_dataN = as.data.frame(lapply(project_data[2:31],norm))

> project_data_train = project_dataN[1:455, ]

> project_data_test = project_dataN[456:569, ]

> project_data_train_labels = project_data[1:455, 1]

> project_data_test_labels = project_data[456:569, 1]
```

Step 3 **Training a model**

```
> project_data_test_pred = knn(train = project_data_train, test = project_data_test, cl =
project_data_train_labels, k = 21)
```

Step 4 **Evaluating performance**

```
> install.packages("gmodels")

> library(gmodels)
```

Step 5 **Changing normalization method**

```
> project_data_train = project_data_z[1:455, ]

> project_data_test = project_data_z[456:569, ]

> project_data_train_labels = project_data[1:455, 1]

> project_data_test_labels = project_data[456:569, 1]

> project_data_test_pred = knn(train = project_data_train, test = project_data_test, cl=
project_data_train_labels, k = 21)

> CrossTable(x = project_data_test_labels, y = project_data_test_pred, prop.chisq = FALSE)
```

Attempts to change k-value

```
> project_data_test_pred = knn(train = project_data_train, test = project_data_test, cl=
project_data_train_labels, k = 3)

> CrossTable(x = project_data_test_labels, y = project_data_test_pred, prop.chisq = FALSE)


> project_data_test_pred = knn(train = project_data_train, test = project_data_test, cl=
project_data_train_labels, k = 7)
```

```
> CrossTable(x = project_data_test_labels, y = project_data_test_pred, prop.chisq = FALSE)
```

```
> project_data_test_pred = knn(train = project_data_train, test = project_data_test, cl= project_data_train_labels, k = 14)
> CrossTable(x = project_data_test_labels, y = project_data_test_pred, prop.chisq = FALSE)
```

```
> project_data_test_pred = knn(train = project_data_train, test = project_data_test, cl= project_data_train_labels, k = 25)
> CrossTable(x = project_data_test_labels, y = project_data_test_pred, prop.chisq = FALSE)
```