COP6930
Introduction to Data Mining
Fall 2017
PROJECT 1 - CLASSIFICATION

MUGDHA MATHKAR
UFID:5414-7979

TABLE OF CONTENTS

1.INTRODUCTION

Classification is used in data mining to assign objects to a class label from a group of categories. The goal of this project was to implement classification models such as KNN, SVM, RIPPER and C4.5 on a life expectancy data set [1] and do an analysis of the results. In this report I have described the way in which I prepared the datasets and trained them for each model. The efficiency of the models was calculated using some accuracy measures as explained later.

2.DATA PREPARATION

Data preparation is a critical task in the mining process. This step involved transforming the raw data into appropriate format and adding relevant features(Continents) and class labels, removing noise and selecting records and features that were needed for mining the dataset. I did this by adding the class label -Continent to the dataset with values: Africa, Australia, Asia, South America, North America. The input dataset was then converted to a csv file format and was stored on the disk.

Sampling:

I used simple random sampling to get training and test set samples from the original data set. In this method, every item has an equal probability of getting selected. A subset of data was selected randomly to divide it into training and test sets. These data sets were provided to the classification model.

Preprocessing was carried out on the data to ensure its quality. I checked if the data contained any missing values by using anyNA() method. Also, the summarized details of the dataset were checked to get an idea of the basic range of the attributes.

Summary of the dataset:

```
> summary(Data_csv_Original)
      Rank              Continent  Overall.life.expectancy.at.birth Male.life.expectancy.at.birth
 Min.   :  1.0   Africa       :56   Min.   :50.20                     Min.   :48.60
 1st Qu.: 56.5   Asia         :50   1st Qu.:67.45                     1st Qu.:64.50
 Median :112.0   Australia    :16   Median :74.90                     Median :72.20
 Mean   :112.0   Europe       :55   Mean   :72.49                     Mean   :70.04
 3rd Qu.:167.5   North America:33   3rd Qu.:78.60                     3rd Qu.:75.85
 Max.   :223.0   South America:13   Max.   :89.50                     Max.   :85.60
 Female.life.expectancy.at.birth
 Min.   :51.00
 1st Qu.:69.75
 Median :77.90
 Mean   :75.02
 3rd Qu.:81.60
 Max.   :93.50
```

Data Slicing:

In this step, I divided the data into training and test sets. The training data would be used for building the classifier model and the test set would be used to predict the class labels. The following method was used to create training and test set samples:

createDataPartition(y=Data_csv$Continent, p= 0.8, list = FALSE)

Here, the **createDataPartition()** method ensured that the training set has 80% of the data while the remaining 20% data is in the test set. This ensures that there is no mixing up between the two datasets. Thus, the training set has 180 observations and the test set has remaining 43 observations. 6 sets of training and test data were created to find the average accuracy of the classifiers.

Class Label: The target variable in this dataset consists of 6 values:

Africa, Australia, Asia, South America, North America.

I converted these to categorical variables using the following

training[["Continent"]] = factor(training[["Continent"]])

3. EVALUATION MEASURES

Following evaluation measures were used for evaluating the model accuracies during the test prediction phase:

Confusion Matrix: I am using a confusion matrix to show a tabular summary of the predicted and actual class labels.

Accuracy: Overall classification accuracy was calculated for each of the classifier model as follows:

```
accuracy = sum(diagonal) / n
```

F1 Measure: F1 score is the harmonic mean of precision and recall.

```
f1 = 2 * precision * recall / (precision + recall)
```

Precision: the fraction of correct predictions for a certain class

```
precision = diag / colsums
```

Recall: the fraction of instances of a class that were correctly predicted

```
recall = diag / rowsums
```

Average Accuracy: the average accuracy is defined as the fraction of correctly classified instances in the sum of one-vs-all matrices matrix.

```
avgAccuracy = sum(diag(s)) / sum(s)
```

4.CLASSIFICATION METHODS

I used kNN, Support Vector Machine, RIPPER and C4.5 classification methods for this project. In the following sections I have described the training and test prediction steps for each of the classification method.

1. k-Nearest Neighbor (kNN)

k Nearest Neighbors algorithm is a non-parametric method used for classification. It is necessary to find the optimal value of k for increasing the classification efficiency.

Package Used: Caret

This package provides train() method for implementing the kNN model and has many tuning parameters.

1.TRAINING

This package has a **trainControl()** method that is used for controlling the computational nuances of the classifier.

The package also provides a **train()** method for training the kNN classifier. This method takes the following arguments in my code:

train(Continent ~., data = training, method = "knn", trControl=trctrl,

preProcess = c("center", "scale"), tuneLength = 10)

Parameters Provided:

method: I passed **knn** as a parameter to this method.

target variable: The data has to be classified based on the class label Continent. This is provided as the target variable.

data: Training dataset that had been created earlier is given here.

preProcess: This parameter is used for preprocessing the training dataset. I'm passing center and scale as the values to this parameter. This will ensure that the data is centered and scaled properly. method = "center" subtracts the mean of the predictor's data (again from the data in x) from the predictor values while method = "scale" divides by the standard deviation.

tuneLength: An integer denoting the amount of granularity in the tuning parameter grid. This argument is the number of levels for each tuning parameter that should be generated by train(). I have given this value as 10.

trControl: This parameter takes the result from trainControl() method. This has values that determine how the function should behave.

I provided the following parameters to the **trainControl()** function:

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

Arguments:

- **method**: This can take parameter values that are used for resampling, such as **repeatedcv**, I am using repeated k-fold cross validations for kNN to obtain better accuracy.
- **number**: This is the number of resampling iterations. Thus, the repeatedcv will be done for 10 times. This ensures that every class label is fairly attributed for in the training set.
- **repeats**: This is the number of times the cross validation has to be repeated.

Here is a snapshot of the Knn classifier that is obtained after using the train() method:

```
k-Nearest Neighbors

180 samples
  4 predictor
  6 classes: 'Africa', 'Asia', 'Australia', 'Europe', 'North America', 'South America'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 163, 164, 162, 160, 162, 162, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.5047231  0.3773531
   7  0.4959453  0.3596214
   9  0.4901756  0.3498015
  11  0.5035369  0.3622984
  13  0.5032177  0.3618785
  15  0.5109501  0.3705228
  17  0.5101234  0.3671529
  19  0.5184404  0.3761080
  21  0.5203664  0.3757540
  23  0.5269945  0.3833175

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 23.
```

Figure 1 Result of train() on KNN

Out of the results across the tuning parameters, k=23 has the maximum accuracy of 53%. Hence, this value of k is selected for the final classifier model. The following graph is generated by the training classifier for accuracy versus number of neighbors.

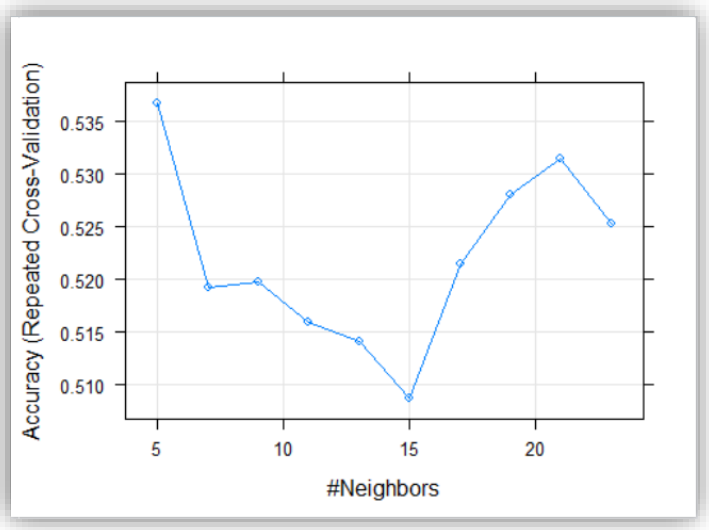
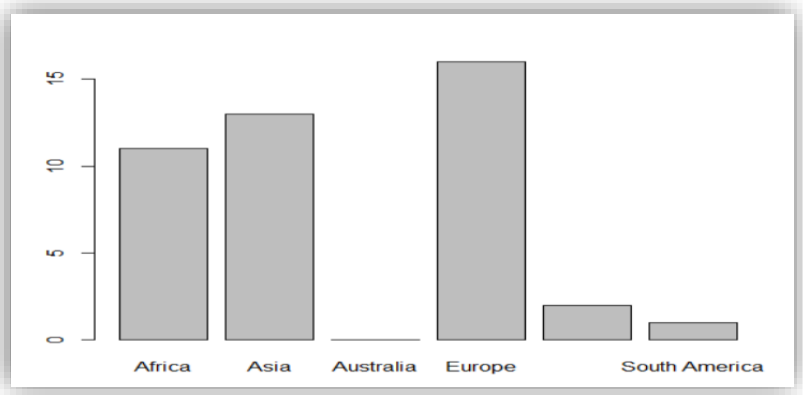


Figure 2 Selecting the optimal value of k

2.TEST PREDICTION

There were 43 random observations in the test set for each group of the data set.

predict() method provided by the caret package was used for test set prediction. It takes the test data as the value for data parameter and the training classifier model that was created for kNN.The result of the predict() was displayed as a plot



The following accuracy measures were obtained for a test set and the corresponding graphs were plotted:

Confusion Matrix:

```
[1] "Model3"
[1] 0.5581395
Confusion Matrix and Statistics

               Reference
Prediction      Africa Asia Australia Europe North America South America
  Africa            10    1         0      0             0             0
  Asia               0    6         2      2             3             1
  Australia          0    0         0      0             0             0
  Europe             1    2         0      7             2             1
  North America      0    1         1      1             1             0
  South America      0    0         0      1             0             0

Overall Statistics

               Accuracy : 0.5581
                 95% CI : (0.3988, 0.7092)
    No Information Rate : 0.2558
    P-Value [Acc > NIR] : 2.413e-05

                  Kappa : 0.4242
 Mcnemar's Test P-Value : NA
```

Figure 3 Confusion Matrix generated for KNN(Model 3)

The confusion matrix shows the number of correctly predicted class labels and the incorrectly identified ones. The results of the six groups were taken and the accuracy measures were applied on them. These measures are displayed as follows:

```
[1] "Accuracy Final:"
[1] "Best Accuracy obtained using KNN :"
[1] 0.5581395
[1] "KNN Accuracy Measures(Recall,Precision,F1) :"
[1] "Key- 0:Africa,1:Asia,2:Australia,3:Europe,4:North America,5:South America"
[1] "Accuracy Measures for first group"
        acc f1Measure    recall      prec   microf1 x
1 0.5348837 0.8181818 0.8181818 0.8181818 0.5348837 0
2 0.5348837 0.4545455 0.4166667 0.5000000 0.5348837 0
3 0.5348837 0.0000000 0.0000000 0.0000000 0.5348837 0
4 0.5348837 0.6923077 0.6000000 0.8181818 0.5348837 0
5 0.5348837 0.0000000 0.0000000 0.0000000 0.5348837 0
6 0.5348837 0.0000000 0.0000000 0.0000000 0.5348837 0
```

1. Support Vector Machine

The principle behind SVM is to find a separating hyperplane for different class labels. It is important that we maximize the distance between the hyperplane and the nearest data point of another class.

Package Used: e1071

The following steps were used for training the SVM classifier:

1.Tuning

Tuning is performed on the training data set to find the best cost and gamma parameters.

tune(svm,Continent~.,data=training,ranges=list(epsilon=seq(0,1,0.01),cost=2^(2:7)))

Parameters Provided:

method: The function to be tuned, svm,in this case

train.x: The class label Continent on which prediction is to be done

data: The dataset to be tuned. Here ,it is the training set

ranges: This has a named list of parameter vectors spanning the sampling space.

epsilon: will have values from 0 to 1 with increments of 0.01. This will give us the gamma value.

cost : captures cost of constraint violation and the default value is 1.

If the cost is too high, it will imply a penalty for non-separable points leading to over fitting. On the other hand, if the cost is too low, it will lead to under fitting. Thus, I have provided wide range of cost values in the range 2^2 to 2^7.This gives us 5 different cost values and thus helps in finding the optimal cost value.

SVM Radial:

I am using radial SVM as it gives better results as compared to other SVM models.

mymodel1 <- tmodel$best.model

The model is trained on the complete training set using the best model parameter combination.

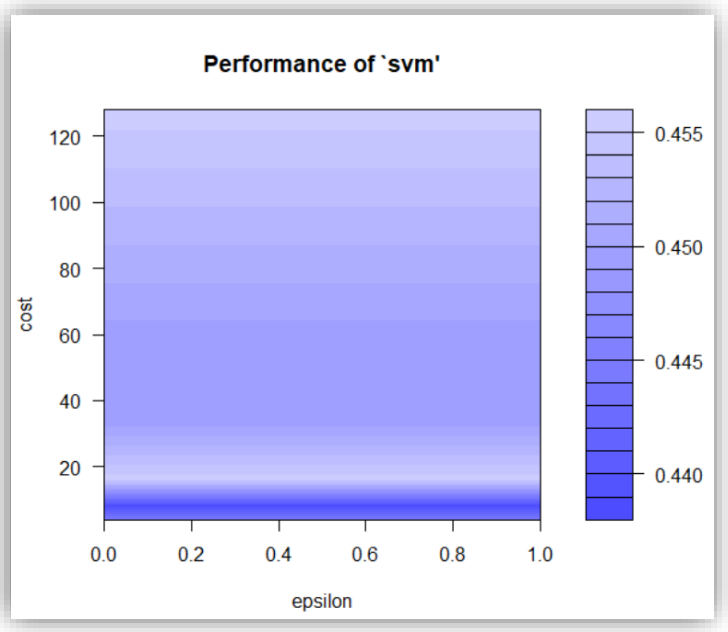


Figure 4 Finding the ideal cost and epsilon value for svm classifier.

Classification Plot

The following classification plot was obtained for the training set.

plot(mymodel1,data=training,Male.life.expectancy.at.birth~Female.life.expectancy.at.birth, slice = list(Overall.life.expectancy.at.birth=60))

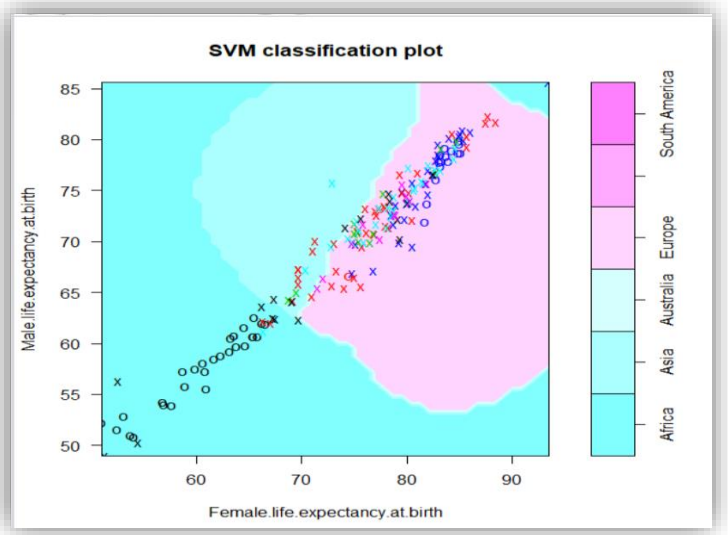


Figure 5 SVM classification plot

2.TEST PREDICTION

There were 43 observations in the test set for each group of data set.

predict() method provided by the caret package was used for test set prediction. It takes the test data as the value for data parameter and the training classifier model that was created for SVM.

The following accuracy measures were obtained for a test set and the corresponding graphs were plotted:

Confusion Matrix for SVM:

The following confusion matrix was obtained for the SVM model:

```
Confusion Matrix and Statistics

               Reference
Prediction      Africa Asia Australia Europe North America South America
  Africa            10    2         0      0             0             0
  Asia               1    6         3      2             2             1
  Australia          0    0         0      0             0             0
  Europe             0    1         0      9             4             1
  North America      0    1         0      0             0             0
  South America      0    0         0      0             0             0

Overall Statistics

               Accuracy : 0.5814
                 95% CI : (0.4213, 0.7299)
    No Information Rate : 0.2558
    P-Value [Acc > NIR] : 6.124e-06

                  Kappa : 0.4456
 Mcnemar's Test P-Value : NA
```

Figure 6 Confusion matrix for SVM

Accuracy measures were calculated and displayed as follows:

```
[1] "SVM Accuracy for the dataset groups:"
[1] "Best Accuracy obtained using SVM :"
[1] 0.5813953
[1] "SVM Accuracy Measures(Recall,Precision,F1) :"
[1] "Key- 0:Africa,1:Asia,2:Australia,3:Europe,4:North America,5:South America"
[1] "Accuracy Measures for first group"
        acc f1Measure    recall      prec   microf1 x
1 0.5581395 0.8181818 0.8181818 0.8181818 0.5581395 0
2 0.5581395 0.4000000 0.3333333 0.5000000 0.5581395 0
3 0.5581395 0.0000000 0.0000000 0.0000000 0.5581395 0
4 0.5581395 0.7142857 0.5882353 0.9090909 0.5581395 0
5 0.5581395 0.0000000 0.0000000 0.0000000 0.5581395 0
6 0.5581395 0.0000000 0.0000000 0.0000000 0.5581395 0
```

Figure 7 Accuracy measures for SVM

It was observed that class label Africa had the highest recall, precision and F1 measures followed by Europe and Asia.

2. RIPPER CLASSIFICATION MODEL

Ripper is a classification technique that is based on rule based induction. It works well with multiclass datasets having imbalanced class distributions. As observed during the preprocessing step, the Life Expectancy data is highly imbalanced with Africa being the continent with maximum data points and Australia being the least. Another advantage of using this model is that it is highly suitable for noisy datasets as it uses the validation set to prevent overfitting.

RIPPER classification is carried out in the following two steps:

- **Rule Growing:**

  RIPPER builds rules using the general to specific strategy and chooses the best conjunct using FOIL's information gain measure.

- **Building the Rule Set:**

  After generating a rule, all the positive and negative examples covered by the rule are eliminated. The rule is then added into the rule set as long as it does not violate the stopping condition, which is based on the minimum description length principle.

1.Training:

I used JRip() method provided in RWeka package for implementing the RIPPER algorithm.

JRip(formula, data, subset, na.action,control = Weka_control(), options = NULL)

I have given the following values as arguments to the JRip() methods:

JRip(Continent ~ ., data = training ,control = Weka_control(O=50,F=10))

Parameters Provided:

formula: This describes the model to be fit. We are classifying based on class label Continent.

data: The dataset provided to the classifier. The training dataset is given as a value to this parameter in my program.

control: This is an object of class Weka_control. I have given the following options to this parameter:

-O This is the number of runs that will be carried out. I trained the classifier on several values of O and concluded that O=50 gives the best accuracy.

-F This is the number of folds. One fold is used for pruning. The default value is 3.I gave this value as 10 based on the results from several runs.

The following 6 rules were obtained after training the JRIP classifier:

```
JRIP rules:
===========

(Female.life.expectancy.at.birth <= 78) and (Male.life.expectancy.at.birth >= 64.6) and (Rank >= 149) => Continent=Asia (15.0/4.0)
(Rank <= 119) and (Female.life.expectancy.at.birth <= 78) => Continent=Asia (11.0/2.0)
(Rank <= 7) => Continent=Asia (5.0/1.0)
(Female.life.expectancy.at.birth >= 80.9) => Continent=Europe (45.0/16.0)
(Female.life.expectancy.at.birth >= 76.4) and (Male.life.expectancy.at.birth <= 69.9) => Continent=Europe (5.0/0.0)
 => Continent=Africa (99.0/55.0)

Number of Rules : 6
```

Figure 8 JRIP rules generated by classifier

2.Test Prediction

The test prediction was done on test sets having 43 observations. A confusion matrix is generated for each model group. It shows the positive and negative predictions.I also calculated the precision,recall,F1 measure and average of the results.

Confusion Matrix:

Table 1 Confusion Matrix for RIPPER

```
Confusion Matrix and Statistics

               Reference
Prediction      Africa Asia Australia Europe North America South America
  Africa            10    1         0      0             0             0
  Asia               0    6         2      2             3             1
  Australia          0    0         0      0             0             0
  Europe             1    2         0      7             2             1
  North America      0    1         1      1             1             0
  South America      0    0         0      1             0             0
```

Following evaluation measures were calculated for the model :

```
[1] "Best Accuracy obtained using RIPPER :"
[1] 0.6046512
[1] "RIPPER Accuracy Measures(Recall,Precision,F1) :"
[1] "Key- 0:Africa,1:Asia,2:Australia,3:Europe,4:North America,5:South America"
[1] "Accuracy Measures for first group"
        acc f1Measure    recall      prec   microf1 x
1 0.4883721 0.6923077 0.6000000 0.8181818 0.4883721 0
2 0.4883721 0.2857143 0.5000000 0.2000000 0.4883721 0
3 0.4883721 0.0000000 0.0000000 0.0000000 0.4883721 0
4 0.4883721 0.6060606 0.4545455 0.9090909 0.4883721 0
5 0.4883721 0.0000000 0.0000000 0.0000000 0.4883721 0
6 0.4883721 0.0000000 0.0000000 0.0000000 0.4883721 0
```

Figure 9 Accuracy measures for RIPPER

3. C4.5 Classification Model

C4.5 is an algorithm used to generate decision trees that can be used for classification.I trained the model using J48() method which generated a decision tree.

Package Used: RWeka

1.Training:

In the training phase, I gave the following parameters to the J48() classifier function:

tree_j48 <- J48(Continent ~ ., data = training,options=parse_Weka_digraph)

Parameters Provided:

The J48() method takes the following arguments:

formula: symbolic description of the model to be fit; Continent~. in this case.

control: Object of class Weka_control() that takes the following options:

-M: Minimum number of instances per leaf

-R: Reduced pruning option

-N: Number of folds in reduced pruning

-C: pruning confidence

options: list of options. I have given parse_Weka_digraph option that parses the graph and returns a list of nodes and edges as shown in the graph().

data: The training dataset to be provided.

The result of this train function is a decision tree that is generated as shown below for one of the data groups:

```
J48 pruned tree
------------------

Overall.life.expectancy.at.birth <= 65.9: Africa (40.0/4.0)
Overall.life.expectancy.at.birth > 65.9
|   Female.life.expectancy.at.birth <= 78.4
|   |   Overall.life.expectancy.at.birth <= 75.9
|   |   |   Overall.life.expectancy.at.birth <= 73.9
|   |   |   |   Overall.life.expectancy.at.birth <= 67.2: Australia (6.0/2.0)
|   |   |   |   Overall.life.expectancy.at.birth > 67.2
|   |   |   |   |   Overall.life.expectancy.at.birth <= 71.4: Asia (16.0/4.0)
|   |   |   |   |   Overall.life.expectancy.at.birth > 71.4
|   |   |   |   |   |   Overall.life.expectancy.at.birth <= 72.4
|   |   |   |   |   |   |   Male.life.expectancy.at.birth <= 69.7: Europe (2.0)
|   |   |   |   |   |   |   Male.life.expectancy.at.birth > 69.7: North America (3.0/1.0)
|   |   |   |   |   |   Overall.life.expectancy.at.birth > 72.4
|   |   |   |   |   |   |   Overall.life.expectancy.at.birth <= 73.4: Australia (9.0/5.0)
|   |   |   |   |   |   |   Overall.life.expectancy.at.birth > 73.4: South America (4.0/2.0)
|   |   |   Overall.life.expectancy.at.birth > 73.9
|   |   |   |   Overall.life.expectancy.at.birth <= 74.5: North America (2.0)
|   |   |   |   Overall.life.expectancy.at.birth > 74.5: Asia (14.0/4.0)
|   |   Overall.life.expectancy.at.birth > 75.9: Africa (4.0/1.0)
|   Female.life.expectancy.at.birth > 78.4
|   |   Male.life.expectancy.at.birth <= 78.2
|   |   |   Overall.life.expectancy.at.birth <= 75.6
|   |   |   |   Male.life.expectancy.at.birth <= 70: Europe (3.0)
|   |   |   |   Male.life.expectancy.at.birth > 70
|   |   |   |   |   Female.life.expectancy.at.birth <= 79: Europe (2.0)
|   |   |   |   |   Female.life.expectancy.at.birth > 79: Africa (2.0)
|   |   |   Overall.life.expectancy.at.birth > 75.6
|   |   |   |   Male.life.expectancy.at.birth <= 75.7
|   |   |   |   |   Female.life.expectancy.at.birth <= 78.8: North America (3.0/1.0)
|   |   |   |   |   Female.life.expectancy.at.birth > 78.8
|   |   |   |   |   |   Female.life.expectancy.at.birth <= 80.6
|   |   |   |   |   |   |   Female.life.expectancy.at.birth <= 80.1
|   |   |   |   |   |   |   |   Overall.life.expectancy.at.birth <= 75.8: South America (2.0)
|   |   |   |   |   |   |   |   Overall.life.expectancy.at.birth > 75.8
|   |   |   |   |   |   |   |   |   Overall.life.expectancy.at.birth <= 76.8: Europe (4.0)
|   |   |   |   |   |   |   |   |   Overall.life.expectancy.at.birth > 76.8: South America (3.0/1.0)
```

Figure 10 C4.5 decision tree

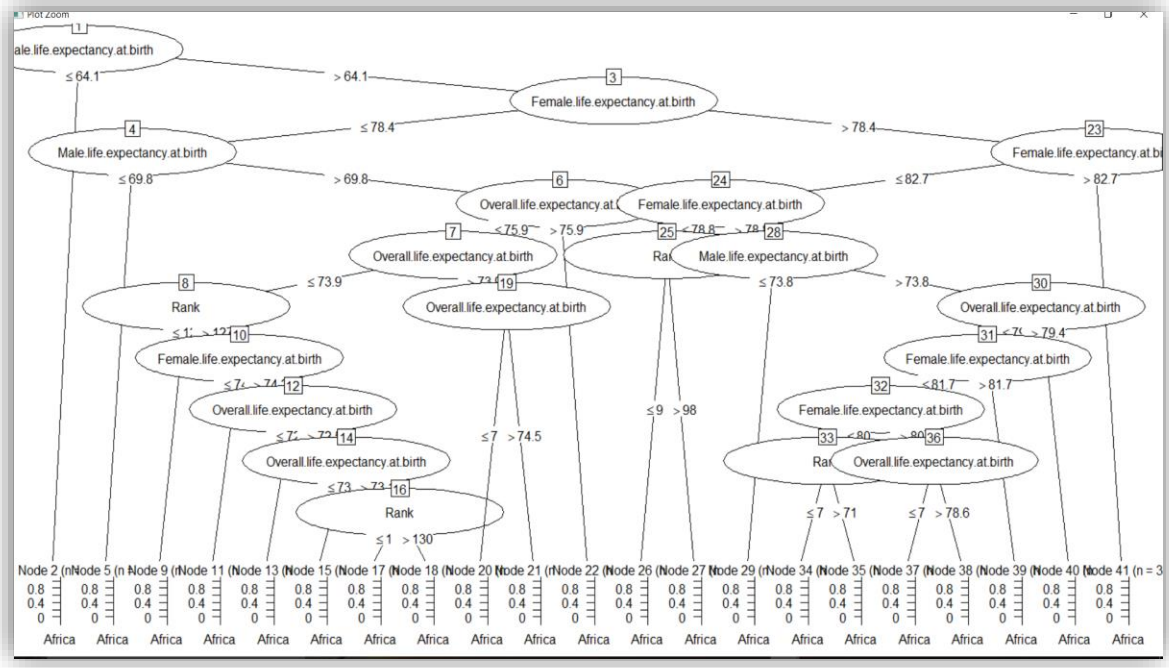


Figure 11 Plotting the Decision Tree generated by C4.5

2.TEST PREDICTION

The test prediction is done using the predict() function in the caret package. Confusion matrix was generated for the data set as follows:

```
Confusion Matrix and Statistics

               Reference
Prediction      Africa Asia Australia Europe North America South America
  Africa             9    0         0      0             1             0
  Asia               0    3         1      2             0             0
  Australia          1    1         1      0             0             0
  Europe             0    4         0      6             3             1
  North America      0    1         0      3             0             1
  South America      1    1         1      0             2             0
```

The result obtained was provided to the functions calculating the evaluation measures.I got the following values for the accuracy(best among 6 datasets),F1,precision,recall and micro F1 measures:

```
[1] "Best Accuracy obtained using C4.5 :"
[1] 0.5813953
[1] "C4.5 Accuracy Measures(Recall,Precision,F1) :"
[1] "Accuracy Measures for first group"
        acc f1Measure    recall      prec   microf1 x
1 0.5116279 0.8695652 0.8333333 0.9090909 0.5116279 0
2 0.5116279 0.2500000 0.3333333 0.2000000 0.5116279 0
3 0.5116279 0.0000000 0.0000000 0.0000000 0.5116279 0
4 0.5116279 0.7407407 0.6250000 0.9090909 0.5116279 0
5 0.5116279 0.0000000 0.0000000 0.0000000 0.5116279 0
6 0.5116279 0.0000000 0.0000000 0.0000000 0.5116279 0
```

Figure 12 Accuracy measures for C4.5

```
[1] 0.5813953
[1] "Recall(Positive Prediction Value)"
[1] 0.9000000 0.4615385 0.0000000 0.6666667 0.0000000 0.0000000
[1] "Precision Value(Sensitivity)"
[1] 0.8181818 0.6000000 0.0000000 0.9090909 0.0000000 0.0000000
[1] "F1 Measure"
[1] 0.8571429 0.5217391 0.0000000 0.7692308 0.0000000 0.0000000
[1] "microF1 Average "
[1] 0.5813953
```

Figure 13 Evaluation measures

4. CONCLUSION

In the end, the accuracy values from each classifier were finally displayed to show a comparison of the performance of each classifier on the test set. On comparing the accuracies of the 4 models, I found that RIPPER had the maximum accuracy of 60.48% followed by C4.5 and SVM.

Here is the result obtained:

```
[1] "Comparison of Accuracies for the models:"
        KNN       SVM       C45    RIPPER
1 0.5581395 0.5813953 0.5813953 0.6046512
[1] "**************************"
```

These classification algorithms are highly dependent on the way in which the training data is prepared. The quality of the results is based on how well the classifier is trained. I used several values for the tuning parameters to come up with the ones that worked well for this data set.

K Nearest Neighbors classifies based on the value of k and Support Vector Machine is dependent on the cost and epsilon values for finding the best separating hyperplane. RIPPER and C4.5 are decision tree classifiers which are based on rule generation.RIPPER works well for multiclass problems and hence gave a better accuracy as compared to other classifiers that I tried.

5. References:

[1] List by CIA,
https://en.wikipedia.org/wiki/List_of_countries_by_life_expectancy#List_by_the_CIA_.282016.29

[2] http://dataaspirant.com/2017/01/09/knn-implementation-r-using-caret-package/

[3] Introduction to Data Mining, Pang Ning Tang

[4] https://github.com/Azure/Azure-MachineLearning-DataScience/blob/master/Utilities/R/MultiClassEvaluation/multi_class_measure.R

[5] https://machinelearningmastery.com/machine-learning-in-r-step-by-step/

[6] http://rischanlab.github.io/SVM.html