Wine Type Prediction

Sujatha Rajarathnam

11/12/2019

Multi Class Classification for Wine Dataset

Overview:

We would like to identify the wine types (the cultivars) based on the 13 chemical components. The 13 components are 1. Alcohol 2. Malic 3. Ash
4. Alcalinity 5. Magnesium 6. Phenols 7. Flavanoids 8. Nonflavanoids
9. Proanthocyanins 10.Color 11.Hue 12.Dilution 13.Proline

The net output is the wine type that are classified as 1,2 or 3

The goal is to build a model that would predict the wine type with a minium of 68% accuracy . Which is a pretty good estimate for this type of data. I have used k nearest neighbor model to build the model and achieve the accuracy of 68% or above.

About the data set: The rattle package contains the data for wines grown in specific area of Italy The wine data was obtained from UCI Machine Learning Repository. Lets include the following packages for our analysis: Tidyverse and Caret

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Warning: package 'dplyr' was built under R version 3.6.1
```

```
# If caret package does not exist, load from the following repo and install it
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

You can obtain the data set by installing the "rattle" package

```
if(!require(rattle)) install.packages("rattle")
```

```
## Loading required package: rattle
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

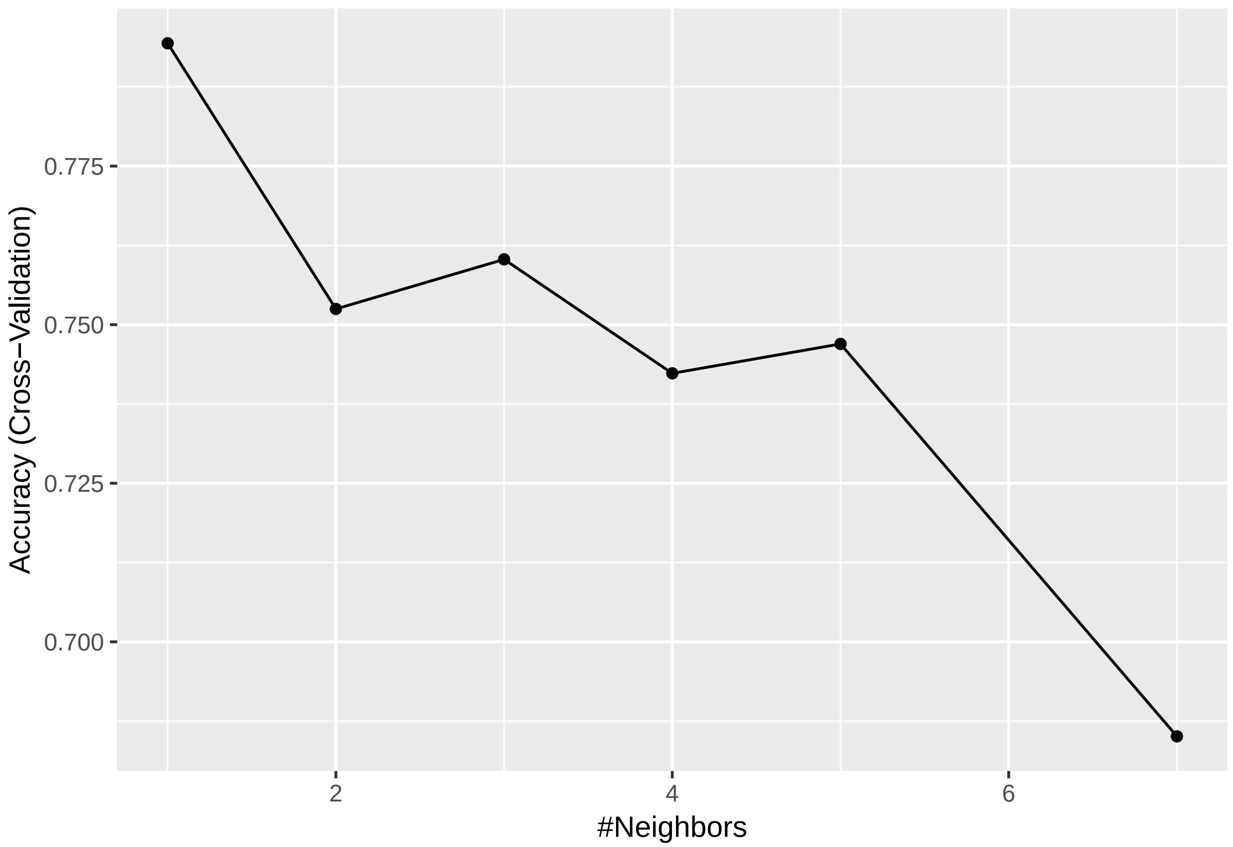
```
# The following command will list all the datasets in a package, we are interested
# in the wine dataset.
data(package="rattle")

#Loading the wine dataset
data("wine")
```

Created Data Partition so that we can have train dataset and test dataset. We are randomly picking 90% of the data to train and we will use the 10% of the data to validate our model. The code is as below

```
set.seed(66)
wine_test_index <- createDataPartition(y = wine$Type, times = 1, p = 0.1, list = FALSE)

############################ Training set ######################################
wine_train_set <- wine[-wine_test_index,]
# Store train input features in a separate variable
train_input_features <- wine_train_set %>% select(-Type)

# Store train output in a separate variable
train_output <- wine_train_set %>% select(Type) %>% .$Type

############################ Test set ######################################
wine_test_set <- wine[wine_test_index,]
# Store test input features in a separate variable
test_input_features <- wine_test_set %>% select(-Type)

# Store test output in a separate variable
test_output <- wine_test_set %>% select(Type) %>% .$Type
```

Method and Analysis

To build the K Nearest Neighbor model, I have decided to train the model using some tuning parameters. I have decided to use cross validation with 10 resampling with 90% of the data. I then decided to train the model with k value of 1,3,4,5 and 7. This is to determine the ideal k value that has relatively good accuracy. The plan is then to use the k value to create the final model. AS you can see from the plot knn= 5 and 7 seems to produce an acceptable accuracy to build our model



Method and Analysis - Contd

From the graph above we will use k value that would give the expected accuracy as per our goal (above 60%)

```r
# Now lets create a model with the ideal K
knn_fit <- knn3(wine_train_set,train_output,k=3)

# now lets run the knn_fit model against the test data to get the predicted Y
y_hat_knn <- predict(knn_fit,wine_test_set,type="class")
```

Results

As you can see from the below output that achieved an accuracy of above 60%

```r
# lets see how accurate it is by executing the confusion matrix: Predicted output versus actual output
# in the test data
cm <- confusionMatrix(y_hat_knn,test_output)

# Lets determine the accuracy
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.6315789
```

Conclusion

Through the K nearest neighbor model we were able to predict the output classification i.e the wine type 1, 2 or 3 with over 60% accuracy. Which was our goal to begin with. This exercise also proves the usefulness of the K nearest neighbor when the target output is more than 2 classifiers (in this case its 3.) Although this dataset is very little in size, this simple model did a pretty decent job in classifying the output (3 values) from multiple features (13 inputs.)