HarvardX PH125.9x - Capstone Project Wheat seeds classifier

Dario Abadie

January 2020

Contents

1 Introduction

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Examples are assigning a given email to the "spam" or "non-spam" class, and assigning a diagnosis to a given patient based on observed characteristics of the patient (sex, blood pressure, presence or absence of certain symptoms, etc.). An algorithm that implements classification, especially in a concrete implementation, is known as a classifier [1].

In this project a classifier for wheat seeds is developed based on the *Seeds* dataset. In order to achieve this goal, 6 models are proposed and evaluated, one of them being the ensemble of the other 5.

2 Objetive

The objetive of the project consists on developing a a classifier for wheat seeds based on the *Seeds* dataset, provided by the Center for Machine Learning and Intelligent Systems of UCI.

This work is part of the *Data Science: Capstone* course of the *HarvardX Data Science Professional Program*.

3 Methodology

The following task were performed in order to develop the recommender system:

- Data load: Data is extractd from the original source and loaded in our work environment.
- Data processing: Transformations are applied on data in order to obtain the features that will be used in the following sections of the study.
- Data exploration: Different visualization techniques are used to obtain insights about the predictors and their behaviour.
- Classification algorithms: Six classifiers are evaluated in order to identify the most accurate one.

3.1 Data load

Before start working on the dataset, it is necessary to add the required libraries.

```
set.seed(7)
library(matrixStats)
library(tidyverse)
library(caret)
library(dslabs)
library(hopach)
library(ggfortify)
library(ggplot2)
library(gridExtra)
library(grid)
```

The dataset must be previously downloaded from the follwing link https://archive.ics.uci.edu/ml/datasets/seeds and then copied in the project directory.

```
seeds <- read.delim("seeds_dataset.txt")
```

```
head(seeds) %>%
  print.data.frame()
```

```
##   X15.26 X14.84 X0.871 X5.763 X3.312 X2.221 X5.22 X1
## 1  14.88  14.57 0.8811  5.554  3.333  1.018 4.956  1
## 2  14.29  14.09 0.9050  5.291  3.337  2.699 4.825  1
## 3  13.84  13.94 0.8955  5.324  3.379  2.259 4.805  1
## 4  16.14  14.99 0.9034  5.658  3.562  1.355 5.175  1
## 5  14.38  14.21 0.8951  5.386  3.312  2.462 4.956  1
## 6  14.69  14.49 0.8799  5.563  3.259  3.586 5.219  1
```

These dataset contains seven predictors and the class of each sample. According to the dataset documentation, the predictors are: Area, Perimeter, Compactness, Kernel length, Kernel width, Asymmetry, and Kernel groove, whereas the classes are Kama, Rosa and Canadian.

We additionally perform a s summary of these subsets in order to check if there are missing values.

```
summary(seeds)
```

```
##      X15.26          X14.84          X0.871            X5.763
##  Min.   : 1.00   Min.   : 1.00   Min.   :0.8081   Min.   :0.8189
##  1st Qu.:12.11   1st Qu.:13.43   1st Qu.:0.8576   1st Qu.:5.2430
##  Median :14.11   Median :14.28   Median :0.8740   Median :5.5160
##  Mean   :14.29   Mean   :14.43   Mean   :0.8713   Mean   :5.5630
##  3rd Qu.:17.10   3rd Qu.:15.70   3rd Qu.:0.8878   3rd Qu.:5.9800
##  Max.   :21.18   Max.   :17.25   Max.   :0.9183   Max.   :6.6750
```

```
##  NA's   :1       NA's   :9       NA's   :14       NA's   :11
##      X3.312          X2.221           X5.22             X1
##  Min.   :2.630   Min.   :0.7651   Min.   :3.485   Min.   :1.000
##  1st Qu.:2.955   1st Qu.:2.6400   1st Qu.:5.045   1st Qu.:1.000
##  Median :3.244   Median :3.6000   Median :5.228   Median :2.000
##  Mean   :3.281   Mean   :3.7006   Mean   :5.408   Mean   :2.089
##  3rd Qu.:3.568   3rd Qu.:4.7730   3rd Qu.:5.879   3rd Qu.:3.000
##  Max.   :5.325   Max.   :8.4560   Max.   :6.735   Max.   :5.439
##  NA's   :12      NA's   :11      NA's   :15      NA's   :15
```

3.2 Data processing

As result of the *summary* performed in the last section, we observe the presence of missing values in the dataset. It is necessary to delete them with the next line of code.

```
seeds <- na.omit(seeds)
```

Now we proceed to add the names of each column according to their meaning.

```
names(seeds) <- c("Area","Perimeter","Compactness","Kernel_length","Kernel_width",
                  "Asymmetry","kernel_groove","Class")
```

Finally we replace the class (1, 2 and 3) with their names too (Kama, Rosa and Canadian).

```
y <- seeds$Class # We save the original values since they will be useful later

seeds$Class <- replace(seeds$Class, seeds$Class==1.00, "Kama") %>% replace(
  seeds$Class==2.00, "Rosa") %>% replace(seeds$Class==3.00, "Canadian")

head(seeds) %>%
  print.data.frame()
```

```
##    Area Perimeter Compactness Kernel_length Kernel_width Asymmetry
## 1 14.88     14.57      0.8811         5.554        3.333     1.018
## 2 14.29     14.09      0.9050         5.291        3.337     2.699
## 3 13.84     13.94      0.8955         5.324        3.379     2.259
## 4 16.14     14.99      0.9034         5.658        3.562     1.355
## 5 14.38     14.21      0.8951         5.386        3.312     2.462
## 6 14.69     14.49      0.8799         5.563        3.259     3.586
##   kernel_groove Class
## 1         4.956  Kama
## 2         4.825  Kama
## 3         4.805  Kama
## 4         5.175  Kama
## 5         4.956  Kama
## 6         5.219  Kama
```

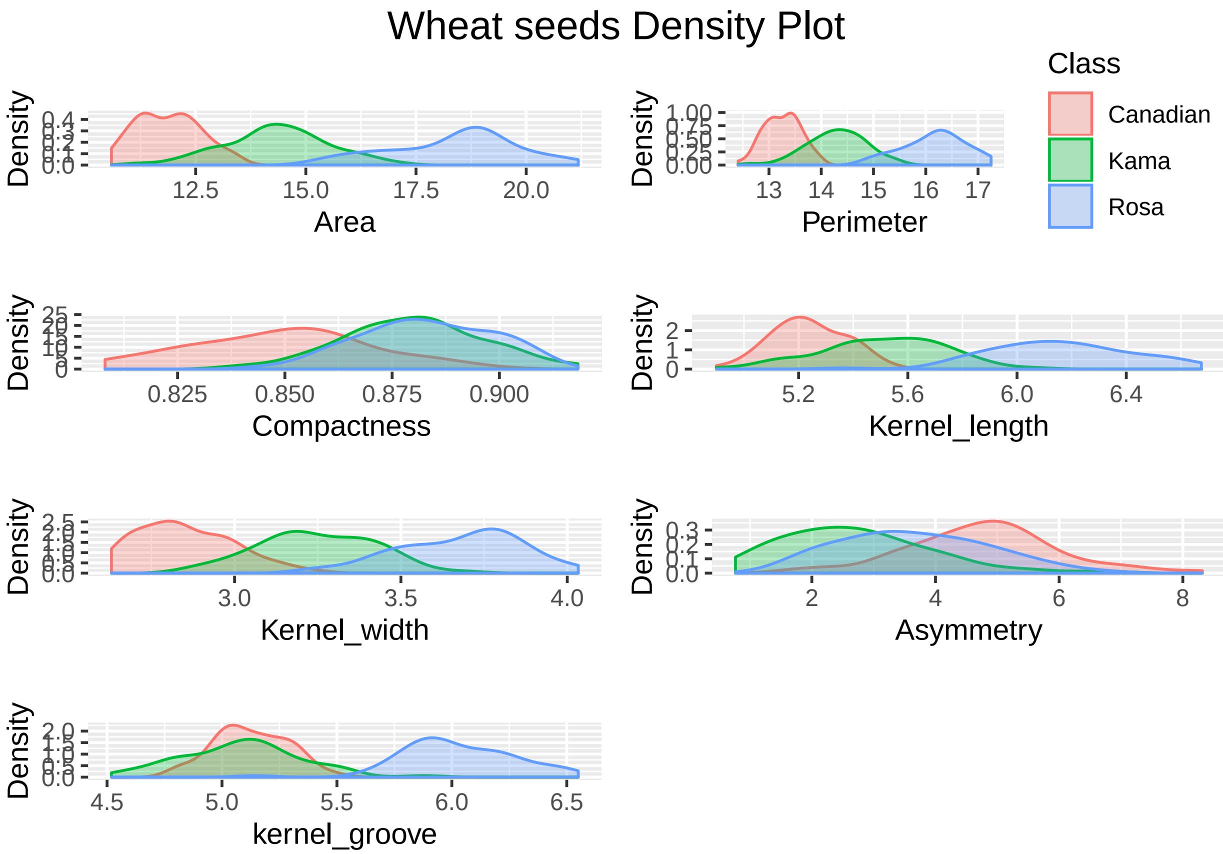
3.3 Data exploration

In this section we analyze the *seeds* object. We generate histograms for each feature in order to visualize how these predictors behave for each class.

```
D1 <-    ggplot(seeds, aes(x=Area, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("Area") +
```

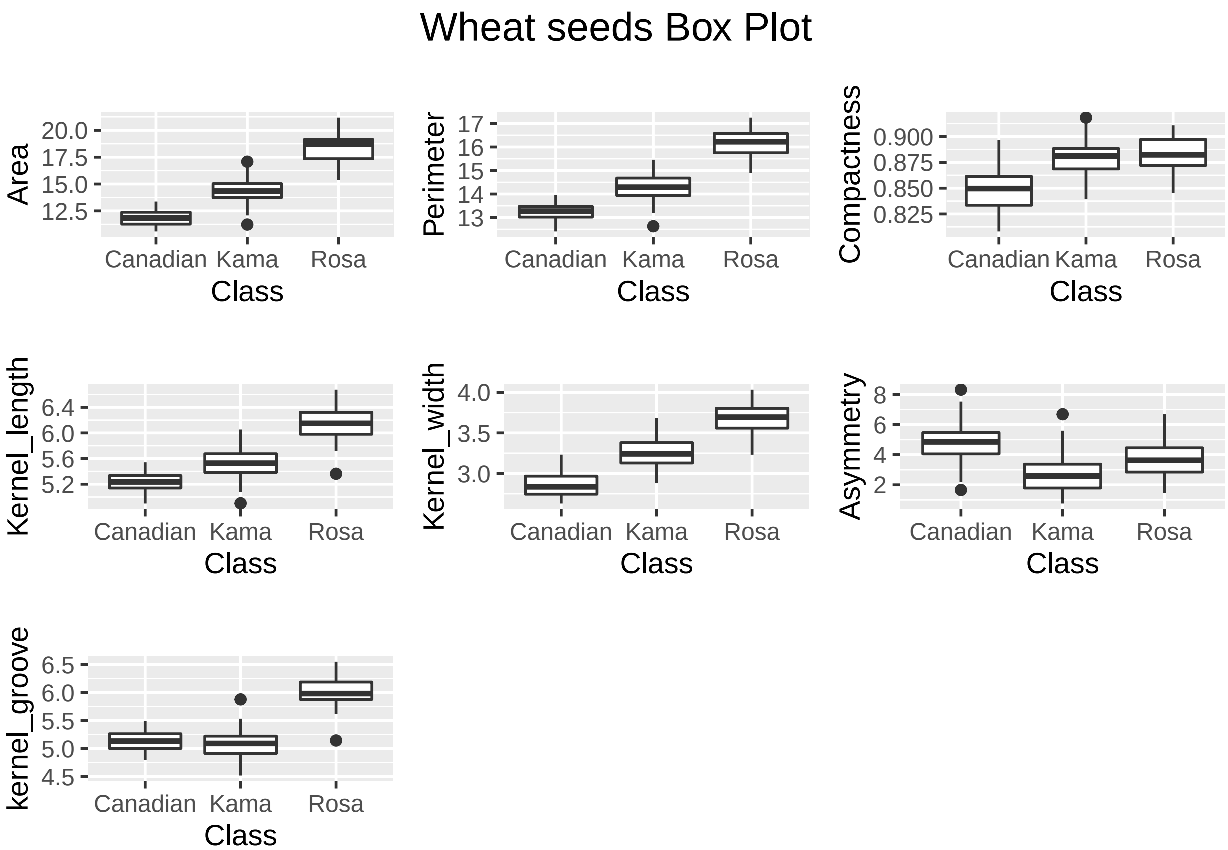
```r
  ylab("Density")+
  theme(legend.position="none")

D2 <- ggplot(seeds, aes(x=Perimeter, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("Perimeter") +
  ylab("Density")



D3 <- ggplot(seeds, aes(x=Compactness, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("Compactness") +
  ylab("Density")+
  theme(legend.position="none")


D4 <- ggplot(seeds, aes(x=Kernel_length, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("Kernel_length") +
  ylab("Density")+
  theme(legend.position="none")

D5 <- ggplot(seeds, aes(x=Kernel_width, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("Kernel_width") +
  ylab("Density")+
  theme(legend.position="none")

D6 <- ggplot(seeds, aes(x=Asymmetry, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("Asymmetry") +
  ylab("Density")+
  theme(legend.position="none")

D7 <- ggplot(seeds, aes(x=kernel_groove, colour=Class, fill=Class)) +
  geom_density(alpha=.3) +
  xlab("kernel_groove") +
  ylab("Density")+
  theme(legend.position="none")


# Plot all density visualizations
grid.arrange(D1 + ggtitle(""),
             D2  + ggtitle(""),
             D3 + ggtitle(""),
             D4 + ggtitle(""),
             D5  + ggtitle(""),
             D6 + ggtitle(""),
             D7 + ggtitle(""),
             nrow = 4,
             top = textGrob("Wheat seeds Density Plot",
                            gp=gpar(fontsize=15)))
```

```
)
```



Additionally we create *Box plots* for each feature to complement the analysis.

```
grid.arrange(
  ggplot(seeds, aes(Class, Area)) + geom_boxplot() + ggtitle(""),
  ggplot(seeds, aes(Class, Perimeter)) + geom_boxplot() + ggtitle(""),
  ggplot(seeds, aes(Class, Compactness)) + geom_boxplot() + ggtitle(""),
  ggplot(seeds, aes(Class, Kernel_length)) + geom_boxplot() + ggtitle(""),
  ggplot(seeds, aes(Class, Kernel_width)) + geom_boxplot() + ggtitle(""),
  ggplot(seeds, aes(Class, Asymmetry)) + geom_boxplot() + ggtitle(""),
  ggplot(seeds, aes(Class, kernel_groove)) + geom_boxplot() + ggtitle(""),

             top = textGrob("Wheat seeds Box Plot",
                            gp=gpar(fontsize=15)))
```



Although we can appreciate some patters in the previous histograms, the classification between classes is not evident. As consequence, we need to apply machine learning algorithms called *Classifiers*.

3.4 Classification system

As we stated, the objetive consists on developong an algorith to classify wheat seeds in 3 different classes using the features provided in the *Seeds* dataset.

We evaluate 6 different models and finally we select the best one. As evaluation criteria, we use *Accuracy*, which is the degree of closeness of measurements of a quantity to that quantity's true value [2].

3.4.1 Normalization

Before proceeding with the classifiers, it is necessary to normalize data. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values [3].

To achieve that, we substract the mean and divide by the standard deviation every value X of each feature i.

$$x = \frac{X - \mu_i}{\sigma i}$$

The following code computes this operation.

```
seeds_x <- data.frame(seeds[,1:7]) # Separate predictors and the target variable
seeds_y <- data.frame(seeds[,8])
```

```
df_means=t(apply(seeds_x,2,mean))
df_sds=t(apply(seeds_x,2,sd))
df=sweep(sweep(seeds_x,2,df_means,"-"),2,df_sds,"/")

x_scaled <- df
```

3.4.2 Training and testing sets

In order to validate the performance of the algorithm, it is required to split the dataset into training and testing sets. All the classifiers will be developed using the training set and later evaluated on the testing set. In this study we assign 80% of the dataset for training and 20% for testing.

```
test_index <- createDataPartition(seeds_x$Perimeter, times = 1, p = 0.2,
                                  list = FALSE) # Create the partition

test_x <- x_scaled[test_index,] # Testing set
test_y <- y[test_index]

train_x <- x_scaled[-test_index,] # Training set
train_y <- seeds_y[-test_index]
train_df <- data.frame(train_x )
train_df["y"] <- train_y
```

3.4.3 Classification algorithms

In this project we use the following classification algorighms:

- Linear discriminant analysis (LDA)[4]
- Quadratic discriminant analysis (QDA) [5]
- Locally estimated scatterplot smoothing (LOESS) [6]
- Random forest (RF) [7]
- K nearest neighbours (Knn) [8]

First we use the LDA algorithm to make predictions.

```
# LDA
fit <- train(y ~ ., method = "lda", data = train_df) # Creation of model
lda_preds <- predict(fit,test_x) # Predictions using the model
lda_preds
```

```
##  [1] Kama     Kama     Kama     Kama     Kama     Kama     Kama     Kama
##  [9] Kama     Kama     Kama     Kama     Kama     Rosa     Rosa     Rosa
## [17] Rosa     Rosa     Rosa     Rosa     Rosa     Rosa     Rosa     Rosa
## [25] Rosa     Rosa     Rosa     Rosa     Rosa     Canadian Canadian Canadian
## [33] Canadian Canadian Canadian Canadian Canadian Canadian Kama     Canadian
## [41] Canadian
## Levels: Canadian Kama Rosa
```

Now we replace the factors in the *lda_preds* for numbers according to the data pre processing we performed at the begining (Kama = 1, Rosa = 2, Canadian = 3).

```
levels(lda_preds) <- c(3,1,2) # Change levels according to data pre processing.
lda_preds
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3
```

```
## [39] 1 3 3
## Levels: 3 1 2
```

Finally we calculate the accuracy and save it a table.

```
lda_accuracy <- mean(lda_preds == test_y) # LDA accuracy

results <- data_frame(Algorithm = "LDA", Accuracy = lda_accuracy ) # Table with results
results %>% knitr::kable()
```

Algorithm	Accuracy
LDA	0.9756098

We repeat the process for the rest of the algorithms.

```
# QDA
fit <- train(y ~ ., method = "qda", data = train_df)
qda_preds <- predict(fit,test_x)
levels(qda_preds) <- c(3,1,2)
qda_accuracy <- mean(qda_preds == test_y)
results <- bind_rows(results, data_frame(Algorithm = "QDA", Accuracy = qda_accuracy ))

# LOESS
fit <- train(y ~ ., method = 'gamLoess', data = train_df)
loess_preds <- predict(fit,test_x)
levels(loess_preds) <- c(3,1,2)
loess_accuracy <- mean(loess_preds == test_y)
results <- bind_rows(results, data_frame(Algorithm = "LOESS", Accuracy = loess_accuracy ))

# Random Forest
fit <- train(y ~ ., method = 'rf', data = train_df,metric = "Accuracy",tuneGrid =
               expand.grid(.mtry=c(3,5,7,9)))
rf_preds <- predict(fit,test_x)
levels(rf_preds) <- c(3,1,2)
rf_accuracy <- mean(rf_preds == test_y)
results <- bind_rows(results, data_frame(Algorithm = "RF", Accuracy = rf_accuracy ))

# K nearest neighbours
fit <- train(y ~ ., data = train_df, method = "knn", tuneLength = seq(3,21,2))
knn_preds <- predict(fit,test_x)
levels(knn_preds) <- c(3,1,2)
knn_accuracy <- mean(knn_preds == test_y)
results <- bind_rows(results, data_frame(Algorithm = "Knn", Accuracy = knn_accuracy ))
```

Additionally we perform and ensemble of this algorithms in order to obtain a system which makes its decisions based on majority vote.

```
# We create a dataframe containing the results of each algorithm.
#It is necessary to convert every prediction from factor to numeric.
ensemble <- cbind(lda =as.numeric(as.character(lda_preds)),
                  qda =  as.numeric(as.character(qda_preds)),
                  rf = as.numeric(as.character(rf_preds)),
                  loess = as.numeric(as.character(loess_preds)),
                  knn = as.numeric(as.character(knn_preds)))
```

```r
# Now we calculate the majority vote for each sample
ensemble_preds  <- apply(ensemble[,-1], 1, function(idx) {
  which(tabulate(idx) == max(tabulate(idx)))
})
sapply(ensemble_preds, paste, sep="", collapse = "")
```

```
##  [1] "1"  "1"  "1"  "1"  "3"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "1"  "2"  "2"
## [16] "2"  "2"  "2"  "2"  "2"  "2"  "2"  "2"  "1"  "12" "2"  "2"  "1"  "2"  "3"
## [31] "3"  "3"  "3"  "3"  "3"  "3"  "3"  "3"  "1"  "3"  "3"
```

```r
# Accuracy is calculated
ensemple_accuracy <- mean(na.omit(as.numeric(as.character(ensemble_preds)) == test_y))

results <- bind_rows(results, data_frame(Algorithm = "Ensemble",
                                         Accuracy = ensemple_accuracy )) # Save results
```

4 Results

The accuracy of each the model are located in the following table:

Algorithm	Accuracy
LDA	0.9756098
QDA	0.9024390
LOESS	0.5853659
RF	0.9268293
Knn	0.9024390
Ensemble	0.9000000

We therefore conclude that the highest accuracy was 0.9756098 using the *LDA* algorithm. Except for *LOESS* algorithm, we can affirm that the rest of the classifiers obtained decent results too.

5 Conclusions

We can succesfully state that we built a machine learning algorithm to classify wheat seeds based on the Seeds dataset. The *LDA* algorithm proved to be the most accurate classifier in the present project, obtaining an accuracy of 0.9756098.

6 Future works

We have obtained a recommender system that provides accurate results for wheat seed classification. This algorithm was developed following data exploration analysis and machine learning algorithms. It would be convenient for future works to add more classification algorithms to the ensemble in order to improve accuracy and generate a more robust system.

7 References

- [1] - Wikipedia - Statistical classification - https://en.wikipedia.org/wiki/Statistical_classification

- [2] - JCGM 200:2008 International vocabulary of metrology — Basic and general concepts and associated terms (VIM).
- [3] - Medium - Why Data Normalization is necessary for Machine Learning models - https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029
- [4] - Wikipedia - Linear discriminant analysis - https://en.wikipedia.org/wiki/Linear_discriminant_analysis
- [5] - Wikipedia - Quadratic classifier - https://en.wikipedia.org/wiki/Quadratic_classifier
- [6] - Wikipedia - Local regression - https://en.wikipedia.org/wiki/Local_regression
- [7] - Wikipedia - Random forest - https://en.wikipedia.org/wiki/Random_forest
- [8] - Wikipedia - K nearest neightbours algorithm - https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm