Gender Prediction from Name

Zhao, Jiahao

5/17/2020

1. Introduction

In this report, we discuss multiple machine learning classification methods, such as logistic regression and decision trees. We utilized these methods to predict a dataset with only two variables: first name and gender. Although the basic packages behind gender prediction were developed by statistians and/or hobbists beginning 14 years ago (Blevins and Mullen 2015, Wais 2006), the dataset used were mostly based from U.S populations. This is regrettable because there are many other languages that names can come from. There are also cultural differences that may differ between the U.S and other countries that lead to inaccurate predictions. We demonstrate that RandomForest has higher accuracy when compared to other methods such as Naive Bayes and regular decision trees.

This paper is organized as follows. Section 2 presents the code used to manipulate, clean, visualize, and explore data along with the reasoning behind them. In Section 3, we briefly review the following methods used in this report: Naive Bayes classifier, logistic regression classifier, Decision Tree classifier, and RandomForest classifier. In Section 4, we present the modeling results and discusses the model performance. Section 5 concludes with a brief summary of the report, its potential impact, its limitations, and future work.

2. Data Manipulation and Splitting Dataset

The following code is used to download and import the dataset into R.

```
# Download dataset
download.file(
  "https://github.com/nickzhao99/genderpred/raw/master/data_masked.csv", "~/df.csv")

# Import csv file as df
df <- read_csv("~/df.csv")
# Set seed as 1 to ensure same results across all iterations
set.seed(1)
```

Now that we have imported the dataset, let's take a look at it.

Name	Gender
KHAWAJA	Male
ALLAHDIN	Male
ABU	Male
MARGARET	Female
TAJ	Male
MALIK	Male

The dataset consists of 246959 records. Each with a Pakistani first name and his/her respective gender. Most other variables have been deleted due to privacy concerns.

What is the most frequent name?

Name	n
MUHAMMAD	24896
SYED	5551
ABDUL	4584
MOHAMMAD	2885
AYESHA	2246
SANA	1597

What is the distribution of female and male?

Gender	n
Female	117627
Male	129332

However, there is no way to analyze the names without seperating them by letter and conver them to numbers. The gender variable also isn't a dummy. This can cause issues in other programming languages. Here's the code to resolve these issues:

```
# Creates a dummy variable for female
df <- df %>% mutate(female = case_when(Gender =="Female" ~1, Gender =="Male"~0))
# Converts female to a factor variable.
df$female <- as.factor(df$female)

# Converts names to lower case
df$Name <- tolower(df$Name)
```

The match function matches alphabets to numbers.

```
# demonstration of match function, which converts alphabet to letters
match("a", letters)
```

```
## [1] 1
```

```
match("b", letters)
```

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

```
match("c", letters)
```

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

```
# Creates a new column for each letter in the Name column, up to eight.
# The letters are then converted to numbers for future possible Naive Bayes
# algorithms or decision rees.
df$one <- substr(df$Name, 1,1)
df$one <- match(df$one, letters)

df$two <- substr(df$Name, 2,2)
df$two <- match(df$two, letters)

df$three <- substr(df$Name, 3,3)
df$three <- match(df$three, letters)

df$four <- substr(df$Name, 4,4)
df$four <- match(df$four, letters)
```

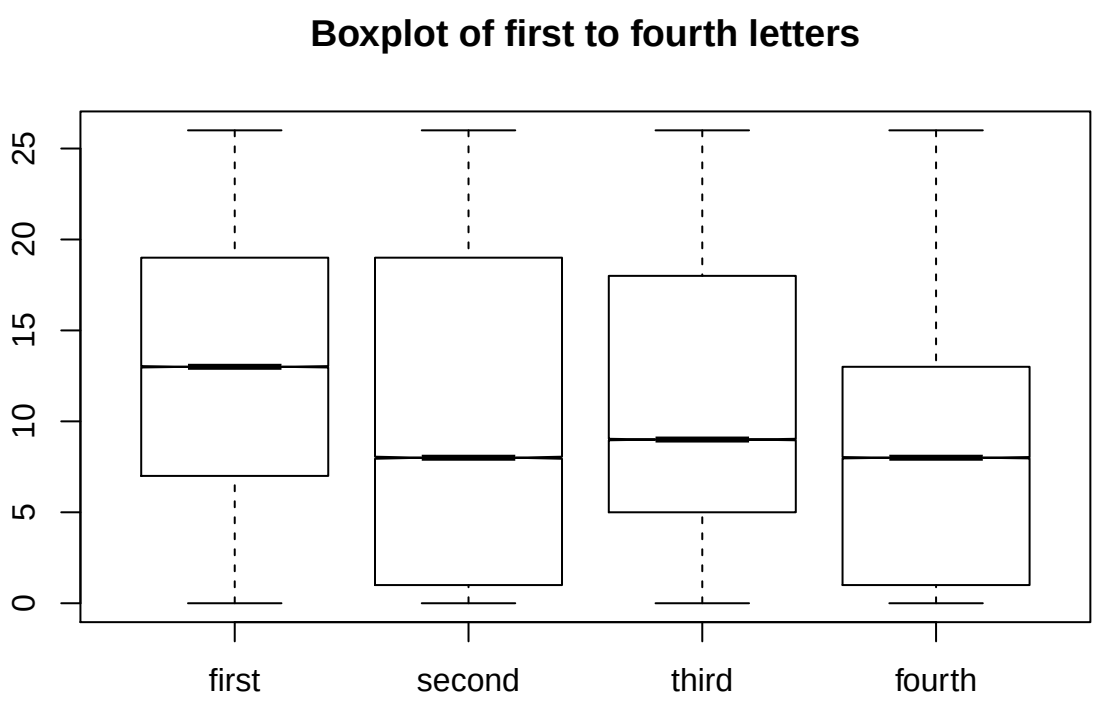
```
df$five <- substr(df$Name, 5,5)
df$five <- match(df$five, letters)

df$six <- substr(df$Name, 6,6)
df$six <- match(df$six, letters)

df$seven <- substr(df$Name, 7,7)
df$seven <- match(df$seven, letters)

df$eight <- substr(df$Name, 8,8)
df$eight <- match(df$eight, letters)

# Fills NA values with 0 in cases where number of letters in a name < 8
df[is.na(df)] <- 0
```

Let's check the data again.

Name	Gender	female	one	two	three	four	five	six	seven	eight
khawaja	Male	0	11	8	1	23	1	10	1	0
allahdin	Male	0	1	12	12	1	8	4	9	14
abu	Male	0	1	2	21	0	0	0	0	0
margaret	Female	1	13	1	18	7	1	18	5	20
taj	Male	0	20	1	10	0	0	0	0	0
malik	Male	0	13	1	12	9	11	0	0	0

Looks like the data is ready. We will create testing and training sets. We utilized 80% training and 20% testing. The reason is as follows: there are only 246959 total records, which means that we need more to verify if our algorithms performs well out of sample. We There are no substantial data to train upon, and I believe 80/20 is a good ratio.

```
#split data into training and test data sets
indxTrain <- createDataPartition(y=df$female, p = 0.8, list = FALSE)
training <- df[indxTrain,]
testing <- df[-indxTrain,]
```



This corresponds to the most frequent name, which is muhammad.

3. Brief Introduction of Classifiers Used

I) Naive Bayes classifier

Naive Bayes classifiers are probabilistic classifiers based on Baye's theorem. It assumes independence between the variables and a normal distrubiton of variables (Rish). Although independence is not achieved here, we will still attempt to apply it onto our dataset. Why is independence not achieved? Think of the most common name, muhammad. If the first letter of a name is m, there is a high chance of it that the second letter is u. Hence, the next letter of a name often depends on the previous letter of the name. Normality of variables also is not achieved in this dataset. We do not believe that this classifier will perform well.

How can we improve? We decided to use an alternative version, which "removes the requirement for the variables to be normally distributed, but retains the essential structure and other underlying assumptions of the method." (Soria,Garibaldi,Ambrogi,Biganzoli, and Ellis 2011) This is the non-parametric form of the naive bayes classifier.

II) Decision Tree

According to Mayur Kulkarni: "Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split." (Kulkarni 2017)

We will use decision tree because we believe the dependent nature of each letters in names will lead to

satisfactory results for our gender prediction system. We will build a classification tree to predict the gender.

III) Logistic Regression

Logistic regression a type of regression analysis to predict binary outcomes. We will fit the data set and predict using our logistic equation.

IV) Random Forest

Random forest is an ensemble of learning methods by combining multiple decision trees and outputting its classification. "The essence of the method is to build multiple treas in randomly selected subspaces of the feature space. Trees in different subspaces generalize their classification in complementary ways." (Ho 1995)

4. Modeling Results

We create x and y as a global matrix/vector.

```
# create x, a matrix consisting of first letter - eighth letter
x = as.matrix(training[,c(4:11)])
# create outcome y column
y = training$female
```

```
# our first model will be fit using naive bayes
model.nb1 <- naivebayes::naive_bayes(x,y)
# fits model using naive bayes with x as input, y as output
Predict <- predict(model.nb1,newdata = x)
# used to predict the training set
confusionMatrix(Predict, training$female)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 41963 17232
##          1 61503 76870
##
##                Accuracy : 0.6015
##                  95% CI : (0.5993, 0.6036)
##     No Information Rate : 0.5237
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2178
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4056
##             Specificity : 0.8169
##          Pos Pred Value : 0.7089
##          Neg Pred Value : 0.5555
##              Prevalence : 0.5237
##          Detection Rate : 0.2124
##    Detection Prevalence : 0.2996
##       Balanced Accuracy : 0.6112
##
##        'Positive' Class : 0
```

```
##
# creates confusion matrix for the training set for the given algorithm.
```

With naive bayes classifier, The accuracy is only 60.15%, a little better than just guessing.

```
model.nb2 = naivebayes::nonparametric_naive_bayes(x,y)
# fits model
Predict <- predict(model.nb2,newdata = x)
# used to predict the training set
confusionMatrix(Predict, training$female)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 76989 31867
##          1 26477 62235
##
##                Accuracy : 0.7047
##                  95% CI : (0.7027, 0.7067)
##     No Information Rate : 0.5237
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4065
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7441
##             Specificity : 0.6614
##          Pos Pred Value : 0.7073
##          Neg Pred Value : 0.7015
##              Prevalence : 0.5237
##          Detection Rate : 0.3897
##    Detection Prevalence : 0.5510
##       Balanced Accuracy : 0.7027
##
##        'Positive' Class : 0
##
```

```
# creates confusion matrix for the training set for the given algorithm.
```

With non-parametric naive bayes classifier, our accuracy increased to 70.47%. But, can we do any better?

```
model.logit <- glmnet(x, y, family = "binomial") #apply logistic equation
Predict <- predict(model.logit,newx = x) # apply predict function
predicted.classes <- ifelse(Predict > 0.5, "1", "0") #if p > 0.5, female, else male
observed.classes <- training$female # checks observed classes
mean(predicted.classes == observed.classes)# calculates accuracy
```

```
## [1] 0.5761555
```

```
rm(predicted.classes,observed.classes) # clears memory
```

Logistic regression only has an accuracy of 57.61%! Time to move on.

```
# fit model using classification tree
model.ct <- rpart(female ~ one + two + three + four + five + six + seven + eight, data=training, method
```

```r
# make prediction on training dataset
Predict <- predict(model.ct, newdata= training, type = 'class')
mean(Predict == training$female) # calculates accuracy
```

```
## [1] 0.8286261
```

With classification tree, our accuracy increased to 82.86%.

```r
#model.rf = randomForest(x,y)
# fits model using randomForest
Predict <- predict(model.rf,newdata = x)
# used to predict the training set
confusionMatrix(Predict, training$female)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 101092   3778
##          1   2374  90324
##
##                Accuracy : 0.9689
##                  95% CI : (0.9681, 0.9696)
##     No Information Rate : 0.5237
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9375
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9771
##             Specificity : 0.9599
##          Pos Pred Value : 0.9640
##          Neg Pred Value : 0.9744
##              Prevalence : 0.5237
##          Detection Rate : 0.5117
##    Detection Prevalence : 0.5308
##       Balanced Accuracy : 0.9685
##
##        'Positive' Class : 0
##
```

```r
# creates confusion matrix for the training set for the given algorithm.
```

With RandomForest, our accuracy increased to 96.77%! We will apply this to the testing dataset.

```r
# apply alg to testing dataset
Predict <- predict(model.rf,newdata = as.matrix(testing[,c(4:11)]))
# create confusion matrix on testing dataset
confusionMatrix(Predict, testing$female)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 25115  1120
##          1   751 22405
```

```
##
##                Accuracy : 0.9621
##                  95% CI : (0.9604, 0.9638)
##     No Information Rate : 0.5237
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.924
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9710
##             Specificity : 0.9524
##          Pos Pred Value : 0.9573
##          Neg Pred Value : 0.9676
##              Prevalence : 0.5237
##          Detection Rate : 0.5085
##    Detection Prevalence : 0.5312
##       Balanced Accuracy : 0.9617
##
##        'Positive' Class : 0
##
```

As we can see, our final accuracy is 96.2%. This shows that we did not overtrain our algorithm.

5. Conclusion

By testing different methods, we managed to obtain a final accuracy of 96% with only the first eight letters of a name! The accuracy would even be better with all the hidden variables included. We could also try to combine this model with other machine learning algorithms. Other limitations of this project includes that it only has Pakistani names. Due to privacy concerns, more variables could not be included. It was a great project and a great introduction of the field of data analytics.