Homework 3: Bayesian Methods

"Advanced Topics in Data Science II"
"Harvard University, Spring 2017"

Tim Hagmann

February 23, 2017

Contents

Problem 1: Authorship Attribution

In this problem, the task is to build a model that can predict if a given news article was written by one of two authors. We will explore two different probabilistic models for this binary classification task. Your model will be evaluated based on its classification accuracy.

Pre-Processing Details

The data is a subset of the Reuter's 50×50 data set in the UCI repository. We provide you with pre-processed versions of these data sets `dataset1_train_processed_subset.txt` and `dataset1_test_processed_subset.txt`.

The text articles have been converted into numerical feature representations. We use the *bag-of-words* feature encoding, where each article is represented by a vector of word counts. More specifically, we construct a dictionary of K frequent words in the corpus, and represent each article using a K-length vector: the i-th entry in this vector contains the number of times the dictionary word i occurs in the article.

We then further preprocessed the data to include the **100 words** that are distinctive to each author to improve our predictions. The dictionary of words used for this representation have been provided in the file `words_preprocessed.txt`.

Hint: Make use of the `header` argument in either `read.csv` or `read.table`.

We begin with a simple Naive Bayes classifier for this task, and will then explore a fully Bayesian modeling approach.

Initialize

In the following code chunk all the necessary setup for the modelling environment is done.

```
## Options
options(scipen = 10)                              # Disable scientific notation
update_package <- FALSE                           # Use old status of packages

## Init files (always execute, eta: 10s)
source("scripts/01_init.R")                       # Helper functions to load packages
source("scripts/02_packages.R")                   # Load all necessary packages
source("scripts/03_functions.R")                  # Load project specific functions
```

Load the data

```
## Read data
df_train <- data.frame(read_csv("data/dataset1_train_processed_subset.txt",
                                col_names=FALSE))
df_test <- data.frame(read_csv("data/dataset1_test_processed_subset.txt",
                               col_names=FALSE))
words <- read_csv("data/words_preprocessed.txt", col_names=FALSE, col_types = ("c"))$X1
```

Preprocess data

```
# Renaming the corpus
names(df_train) <- c("Author", words)
names(df_test) <- c("Author", words)
```

Part 1a: Naive Bayes Classifier

Fit a Naive Bayes classification model to the training set and report its classification accuracy on the test set. The input to this model is the word count encoding for an article, and the output is a prediction of the author identity.

Overall accuracy

Using 0-1 loss what is the overall accuracy?

Calculate Naive Bayes model

```
# Model
df_train$Author <- factor(df_train$Author)
fit_nb <- naiveBayes(Author ~ ., data=df_train)

# Prediction
prd_nb_train <- predict(fit_nb, newdata=df_train, type="class")
prd_nb_test <- predict(fit_nb, newdata=df_test, type="class")

# Confusion matrix
table_nb_train <- table(prd_nb_train, df_train$Author)
table_nb_test <- table(prd_nb_test, df_test$Author)
```

```
# Accuracy
acc_nb_train <- sum(diag(table_nb_train)) / sum(table_nb_train)
acc_nb_test <- sum(diag(table_nb_test)) / sum(table_nb_test)

pander((table_nb_test))
```

	AaronPressman	AlanCrosby
AaronPressman	49	23
AlanCrosby	1	27

The Naive Bayes classification training accuracy is at: 76 %. Furthermore, the classification of Aaron Pressman articles appear to be more accurate than the one for Alan Crosby. That means that Only one articles is wrongly classified as Alan Crosby. However, the algorithm performs worse for Alan Crosby. There are only 27 out of the 50 articles correctly classified. It appears that the algorithm has a bias towards classifying articles as Aaron Pressman. ### Problem with Naive Bayes The principal assumption of the Naive Bayes algorithm is, that the occurence of one word is independent on the occurence of another word, i.e., that there is no correlation/dependence on the occurence of another word if we already know that the article is written by Aaron Presson. The problem with the task at hand is, that some words pears are not independet of each other. A simplified example of this would be "United States of America". The word "United" correlates with "States" and the word pair "United States" correlates with "of America" etc.

Part 1b: Dirichlet-Multinomial Model

Let us consider an alternate Bayesian approach for authorship inference. We recommend a Dirichlet-Multinomial probabilistic model for articles by each author. The author identity for an article can be predicted by computing the posterior predictive probability under this model. This is similar to the approach described in class for the Beatles musical authorship inference example, except that we shall use word features in place of transition couplets.

Probability model: Let $(y_1^A, y_2^A, \ldots, y_K^A)$ denote the total counts of the K dictionary words across the articles by author A, and $(y_1^B, y_2^B, \ldots, y_K^B)$ denote the total counts of the K dictionary words across the articles by author B. We assume the following *multinomial model*:

$$p(y_1^A, y_2^A, \ldots, y_K^A) \propto (\theta_1^A)^{y_1^A} (\theta_2^A)^{y_2^A} \ldots (\theta_K^A)^{y_K^A}$$

$$p(y_1^B, y_2^B, \ldots, y_K^B) \propto (\theta_1^B)^{y_1^B} (\theta_2^B)^{y_2^B} \ldots (\theta_K^B)^{y_K^B}.$$

The model parameters $(\theta_1^A, \ldots, \theta_K^A)$ and $(\theta_1^B, \ldots, \theta_K^B)$ are assumed to follow a *Dirichlet* prior with parameter α.

Posterior probabilty function

In order to solve the problem with a Dirichlet-Multinomial probabilistic model (DMM) one can use the the *posterior_pA* function (see scripts/03_functions.R) to infer authorship for a given test article by predicting author A if $p_i = p(A \,|\, y_i, data) > 0.5$ and author B otherwise.

Preprocess data

```
# Split dataframe into a list of two groupings (AaronPressman & AlanCrosby)
lst_train <- split(df_train[-1], df_train[1])
lst_test <- split(df_test[-1], df_test[1])
```

```r
# Sum word occurances according to different authors (AaronPressman & AlanCrosby)
y_til_train <- lapply(lst_train, function(x) {
  apply(x, 2, sum)
})
y_til_test <- lapply(lst_test, function(x) {
  apply(x, 2, sum)
})
```

Calculate posterior probabilty for training and test data

```r
# Calculate posterior probabilty
post_pA_train <- apply(df_train[-1], 1, function(x) {
  posterior_pA(alpha=1,
               yA=y_til_train[[1]],
               yB=y_til_train[[2]],
               y_til=x)
})

post_pA_test <- apply(df_test[-1], 1, function(x) {
  posterior_pA(alpha=1,
               yA=y_til_test[[1]],
               yB=y_til_test[[2]],
               y_til=x)
})


# Prediction
prd_pA_train <- factor(post_pA_train > 0.5, labels=c("AaronPressman", "AlanCrosby"))
prd_pA_test <- factor(post_pA_test > 0.5, labels=c("AaronPressman", "AlanCrosby"))

# Confusion matrix
table_pA_train <- table(prd_pA_train, df_train$Author)
table_pA_test <- table(prd_pA_train, df_train$Author)

pander((table_nb_train))
```

	AaronPressman	AlanCrosby
AaronPressman	50	17
AlanCrosby	0	33

```r
pander((table_nb_test))
```

	AaronPressman	AlanCrosby
AaronPressman	49	23
AlanCrosby	1	27

Loss function

Evaluate the classification accuracy that you get on the test set using the log-loss function

$$\sum_i y_i \log(p_i) + (1 - y_i) \log(1 - p_i),$$

where y_i is the binary author identity in the test set for article i and p_i is the posterior mean probability that article i is written by author A. The necessary loss function following the above design can be found in scripts/03_functions.R.

The following choices of the Dirichlet parameter have to be made:

- α is set to 1
- α is tuned by cross-validation on the training set under log-loss.

What does setting $\alpha = 1$ imply?

An Alpha of 1 assumes a uniform prior for the two authors. This implies, that the probability of finding a particular keyword is equal to the probability of finding some other keyword. When the alpha value is small (e.g., zero) the prior distribution for each observation consist of only a few groups. As alpha increases, the observation distribution is represented more and more equally by each group.

Perform opimization

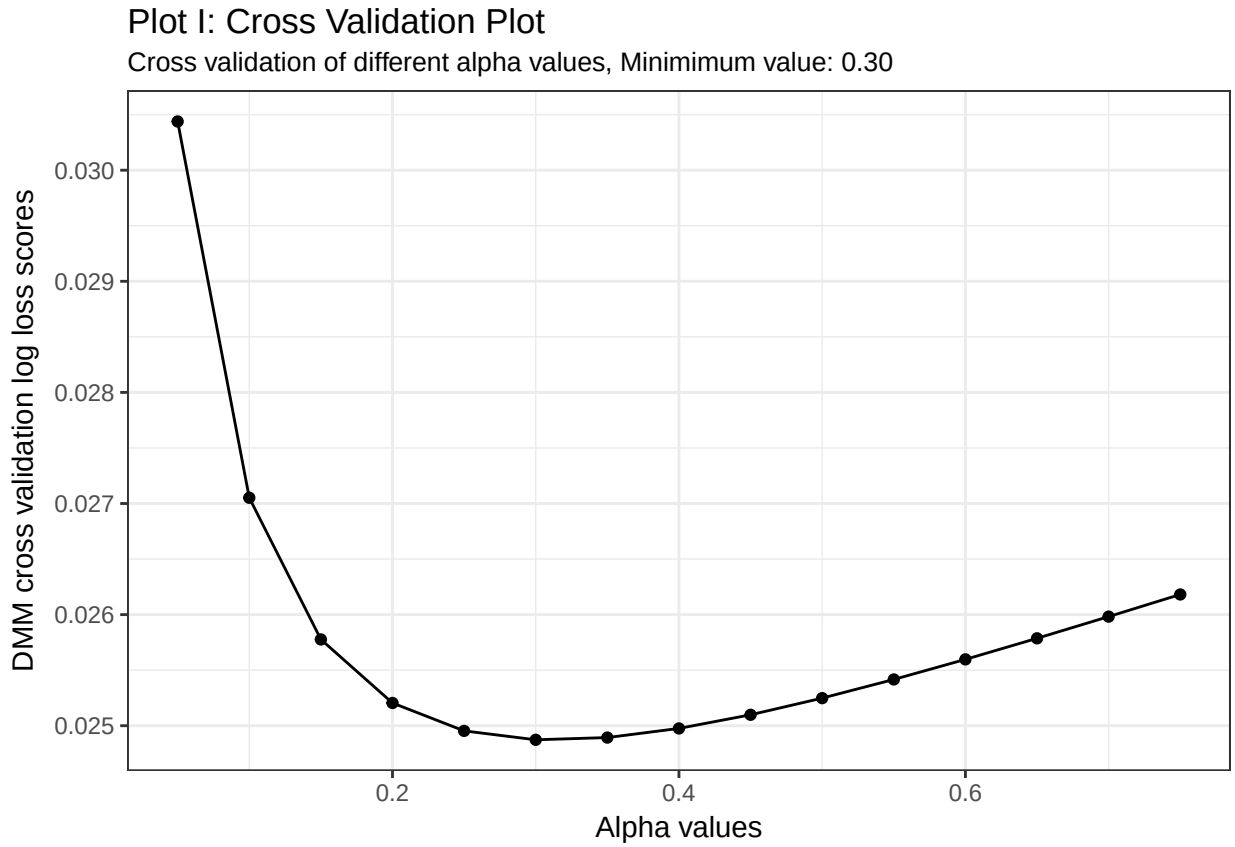
What is the optimal value of α found through cross-validation? What is your final log-loss prediction error?

Note: Out of computational reasons only a 2-fold crossvalidation is performed. It would be advisable to increase this value for more reliable results.

```
# Generate alpha values
alpha_values <- c(1:15)/20

# Calculate log losses
set.seed(123) # Set seed for random number generator
log_losses <- cross_val_fun(df_train, alpha_values, k=2) # 2 fold crossvalidation

# Get minimum alpha value
alpha_min_parameter <- alpha_values[which.min(log_losses)]
log_loss_min_parameter <- log_losses[alpha_values == alpha_min_parameter]
```

Visualize the cross validation parameters

```
ggplot(data=data.frame(alpha_values, log_losses),
       aes(x=alpha_values, y=log_losses, group=1)) +
       geom_line() +
       geom_point() +
       labs(title="Plot I: Cross Validation Plot",
       subtitle=sprintf("Cross validation of different alpha values, Minimimum value: %.2f",
                        alpha_min_parameter)) +
       theme_bw() +
       ylab("DMM cross validation log loss scores") +
       xlab("Alpha values")
```



The above cross validation plot shows the plot of alpha against the cross validation log loss score. The optimal alpha value is 0.3000 with a log loss score of 0.0249.

Comparison with Naive Bayes

For the optimal value of α, how do the accuracies (based on $0-1$ loss) obtained compare with the previous Naive Bayes classifier?

In order to do this we have to calculate the posterior probabilities with the optimal alpha as well as with an $\alpha = 1$.

```
# Calculate posterior probability
pA_alpha_cv <- calc_class_probability(df_train[-1], df_test[-1], df_train[1],
                                      alpha=alpha_min_parameter)
pA_alpha_1 <- calc_class_probability(df_train[-1], df_test[-1], df_train[1], alpha=1)

# Prediction
prd_pA_cv <- factor(pA_alpha_cv < 0.5, labels=c("AaronPressman", "AlanCrosby"))
prd_pA_1 <- factor(pA_alpha_1 < 0.5, labels=c("AaronPressman", "AlanCrosby"))

# Confusion matrix
table_pA_cv <- table(prd_pA_cv, df_test$Author)
table_pA_1 <- table(prd_pA_1, df_test$Author)
pander((table_pA_cv))
```

	AaronPressman	AlanCrosby
AaronPressman	46	6
AlanCrosby	4	44

```
pander((table_pA_1))
```

	AaronPressman	AlanCrosby
AaronPressman	46	8
AlanCrosby	4	42

```
# Accuracy
acc_dmm_cv <- sum(diag(table_pA_cv)) / sum(table_pA_cv)
acc_dmm_1 <- sum(diag(table_pA_1)) / sum(table_pA_1)
```

The tables above show, that the tuned DMM performs much better than the Naive Bayes model. The model with a strict alpha of 1 has an accuracy rate of 88 % compared to the naive bayes model with an accuracy rate of 76. The best model according to the overall accuracy rate is the cv optimized model with an overall accuracy rate of 88.

Part 1c: Monte-Carlo Posterior Predictive Inference

In the **posterior_pA** function, the posterior predictive distribution was computed using a closed-form numerical expression. We can compare the analytic results to those resulting from Monte Carlo simulation. In order to do this an alternate `R` function (`approx_posterior_pA`) can be used to calculate posterior predictive probabilities of authorship using Monte-Carlo simulation. The code takes the number of simulation trials as an additional input.

Consider the situation in Part 1b, using the Monte-Carlo approximation in `approx_posterior_pA` numbers of simulation trials (you may set α to the value chosen by cross-validation in part 1b).

MCMC accuracy over different number of simulations

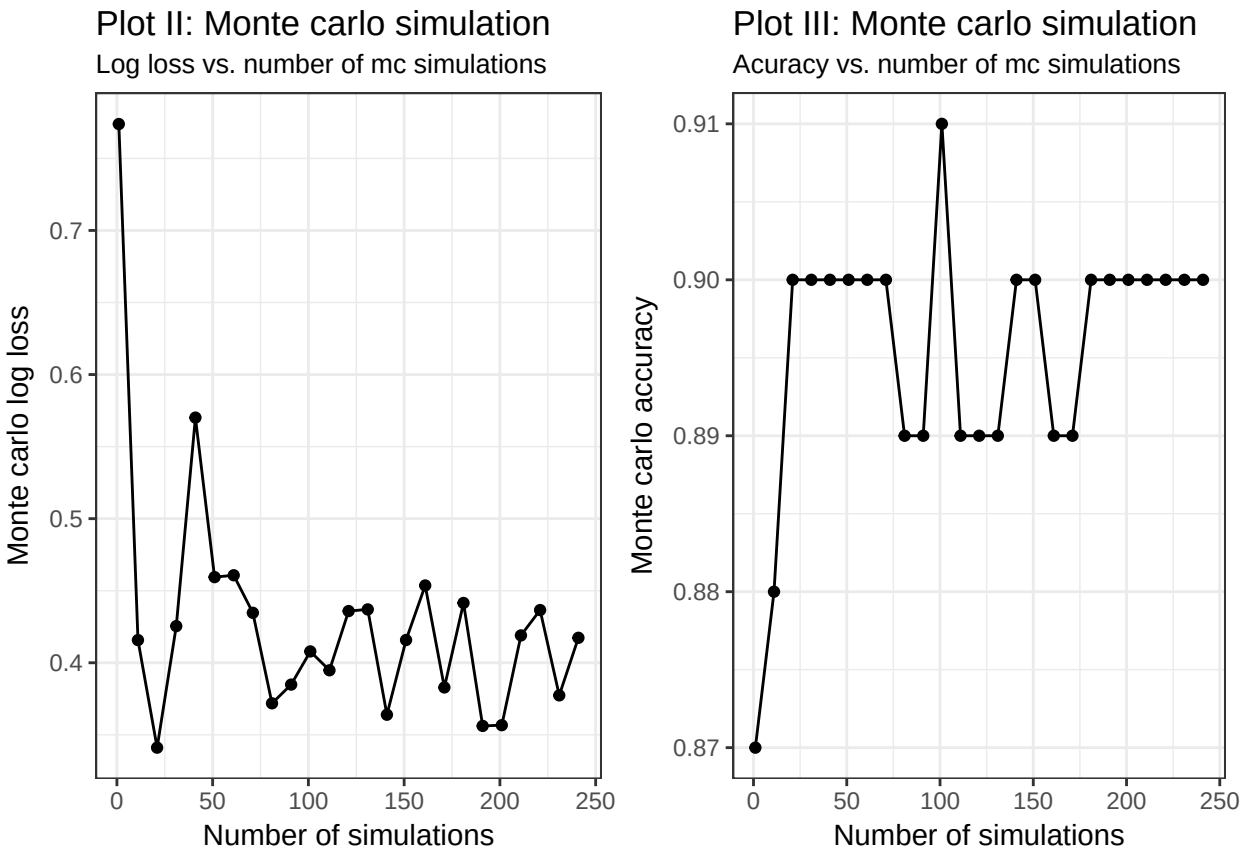
- At what point does doing more simulations not give more accuracy in terms of prediction error?
- Report on the number of simulations you need to match the test accuracy in part 1b. Does increasing the number of simulations have a noticeable effect on the model's predictive accuracy? Why or why not?

```
# Set number of simulations
set.seed(123)
n_sim <- seq(1, 250, by=10)

## Create emply vectors
loss_mc <- c()
acc_mc <- c()

## iterate using a for-loop
for(n in n_sim) {

  pa_value <- calc_class_probability_mc(df_train[-1],
                                        df_test[-1],
```

```
                                    df_train[[1]],
                                    alpha=alpha_min_parameter,
                                    n_sim=n)

  # Transform to numeric predictor variable
  y_test <- as.numeric(factor(df_test$Author))
  y_test <- replace(y_test, y_test == 2, 0)

  # Prediction
  prd_vals <- factor(pa_value < 0.5, labels=c("AaronPressman", "AlanCrosby"))
  table_mc <- table(prd_vals, df_test$Author)
  acc_value <- sum(diag(table_mc)) / sum(table_mc)

  # Append
  loss_mc <- append(loss_mc, MultiLogLoss(y_test, pa_value))
  acc_mc <- append(acc_mc, acc_value)
}
```

Visualize

```
g1 <- ggplot(mapping=aes(x=n_sim, y=loss_mc)) +
  geom_line() +
  geom_point() +
  labs(title="Plot II: Monte carlo simulation",
       subtitle="Log loss vs. number of mc simulations") +
  theme_bw() +
  ylab("Monte carlo log loss") +
  xlab("Number of simulations")

g2 <- ggplot(mapping=aes(x=n_sim, y=acc_mc)) +
  geom_line() +
  geom_point() +
  labs(title="Plot III: Monte carlo simulation",
       subtitle="Acuracy vs. number of mc simulations") +
  theme_bw() +
  ylab("Monte carlo accuracy") +
  xlab("Number of simulations")

grid.arrange(g1, g2, nrow=1, ncol=2)
```



The above plots show, that after more or less 25 simulations, the log loss score reaches a minimum and stabilizes after that. A similar picture appears in plot III where the accuracy rate reaches a high rate also at 25.

There appears to be a small spike at 100 simulations. However, the improvement only amounts to one percent and is probably just some random variation. A probable reason why the MCMC model doesn't improve by much after 25 simulations is that the model has found a good posterior distribution and the remaining predictions can't improve anymore.

Part 1d: Author vocabulary analysis

The prescribed Bayesian model can also be used to analyze words that are most useful for inferring authorship. One way to do this is to compute or approximate the posterior distribution of the ratio of multinomial model parameters (relative ratio) for one author relative to the other, and identify the words that receive high values of this ratio. More specifically, we can calculate this ratio of the posterior parameter values

$R_k = \theta_k^A/(\theta_k^A + \theta_k^B), k = 1, ..., K$

and return a Monte-Carlo approximation of $\mathbb{E}[R_k \,|\, data]$. The largest R_k this would indicate high relative usage of a word for author A while the smaller values would indicate the same instead for author B.

We again provide you with the relevant `R` code. The input to the code is the Dirichlet parameter α, the number of MC draws `n.sim` for the approximation and the total word counts $(y_1^A, y_2^A, \ldots, y_K^A)$ and $(y_1^B, y_2^B, \ldots, y_K^B)$ from the training set articles written by author A. The output is a vector containing the approximate values of $\mathbb{E}[R_k \,|\, data]$.

Using the `posterior_mean_R` function and the word dictionary `words.txt`, list the top 25 words that are indicative of each author's writing style (you may set α and `n.sim` the values chosen in part 1b and 1c

respectively).

Using the posterior mean function

```
# Apply the function
post_mean <- posterior_mean_R(alpha=0.3, y_til_train[[1]], y_til_train[[2]], n.sim=25)

# Get the probabilty values
prob_words <- data.frame(post_mean, words)
prob_words_author_1 <- prob_words[order(-post_mean), ]
prob_words_author_2 <- prob_words[order(post_mean), ]
```

Top 25 words for Aaron Pressman (largest 25 values of E[R])

```
as.character(prob_words_author_1$words[1:25])
```

```
##  [1] "proposal"        "unions"          "clinton"
##  [4] "communications"  "consumer"        "names"
##  [7] "software"        "corp"            "businesses"
## [10] "disputes"        "policies"        "codes"
## [13] "jim"             "business"        "latest"
## [16] "treasury"        "union"           "allow"
## [19] "hill"            "delivery"        "1933"
## [22] "directly"        "recommendations" "school"
## [25] "people"
```

Top 25 words for Alan Crosby (smallest 25 values of E[R])

```
as.character(prob_words_author_2$words[1:25])
```

```
##  [1] "henman"         "spt"            "banka"          "minister"
##  [5] "czechs"         "bourse"         "1996"           "party"
##  [9] "turnout"        "becker"         "ods"            "team"
## [13] "14"             "ferreira"       "korda"          "seed"
## [17] "seats"          "investor"       "ing"            "warsaw"
## [21] "earnings"       "zagreb"         "constituencies" "situation"
## [25] "devised"
```

How can we interpret $E[R_k]$?

$E[R_k]$ can be interpreted as the representation of the writing style or the specific preference for certain topics of each author. That means that the higher the respective value, the more used the word is by one author over the relative usage by the other author.

Differences in vocabulary

Do you find visible differences in the authors choice of vocabulary?

It looks like Crosby used more words associated with central/eastern Europe, names of cities, and also foreign names. Pressman on the other hand used more words associated with politics like unions and businesses.

Problem 2: Bayesian Logisitic Regression

You are provided with data sets `dataset_2_train.txt` and `dataset_2_test.txt` containing details of contraceptive usage by 1934 Bangladeshi women. There are 4 attributes for each woman, along with a label indicating if she uses contraceptives. The attributes include:

- `district`: identifying code for the district the woman lives in
- `urban`: type of region of residence
- `living.children`: number of living children
- `age_mean`: age of women (in years, centered around mean)

The women are grouped into 60 districts. The task is to build a classification model that can predict if a given woman uses contraceptives.

Load the data

```
# Clean workspace
rm(list=ls())

# Read data
df_train <- read_csv("data/dataset_2_train.txt")
df_test <- read_csv("data/dataset_2_test.txt")
```

Preprocess data

```
# Renaming the variables (replace . with _)
names(df_train) <- gsub("[.]", "_", names(df_train))
names(df_test) <- gsub("[.]", "_", names(df_test))

# Create factor variables
df_train$district <- factor(df_train$district)
df_test$district <- factor(df_test$district)

df_train$contraceptive_use <- factor(df_train$contraceptive_use, labels=c("No", "Yes"))
df_test$contraceptive_use <- factor(df_test$contraceptive_use, labels=c("No", "Yes"))

# Aggregate data on a districe level
df_dist1 <- aggregate(contraceptive_use ~ district, data=df_train, FUN=length)
df_dist2 <- aggregate((as.numeric(df_train$contraceptive_use) - 1) ~ district,
                      data=df_train, FUN=sum)

# Left join data
df_dist <- merge(df_dist1, df_dist2, by="district", all=TRUE)
names(df_dist) <- c("distric", "n_people", "sum_contra_use")

# Calculations
df_dist$perc_contra <- round((df_dist$sum_contra_use / df_dist$n_people) * 100, 2)

# Change order of the districts
df_dist$distric <- factor(df_dist$distric,
                          levels=df_dist$distric[
                            order(df_dist$perc_contra, decreasing=FALSE)])
```

```
df_dist <- df_dist[order(df_dist$perc_contra, decreasing=FALSE), ]

# Add number of women in the district to the district
levels(df_dist$distric) <- paste0("D",
                                  sprintf("%.2d",
                                  as.numeric(levels(df_dist$distric))),
                                  " (n=", sprintf("%.2d", df_dist$n_people), ")")
rm(df_dist1, df_dist2)
```

Differences in the districts (visualization)

Use simple visualizations to check if there are differences in contraceptive usage among women across the districts. If so, would we benefit by fitting a separate classification model for each district? To answer this question, you may fit the following classification models to the training set and compare their accuracy on the test set:

Table I: Count of contraceptive usage

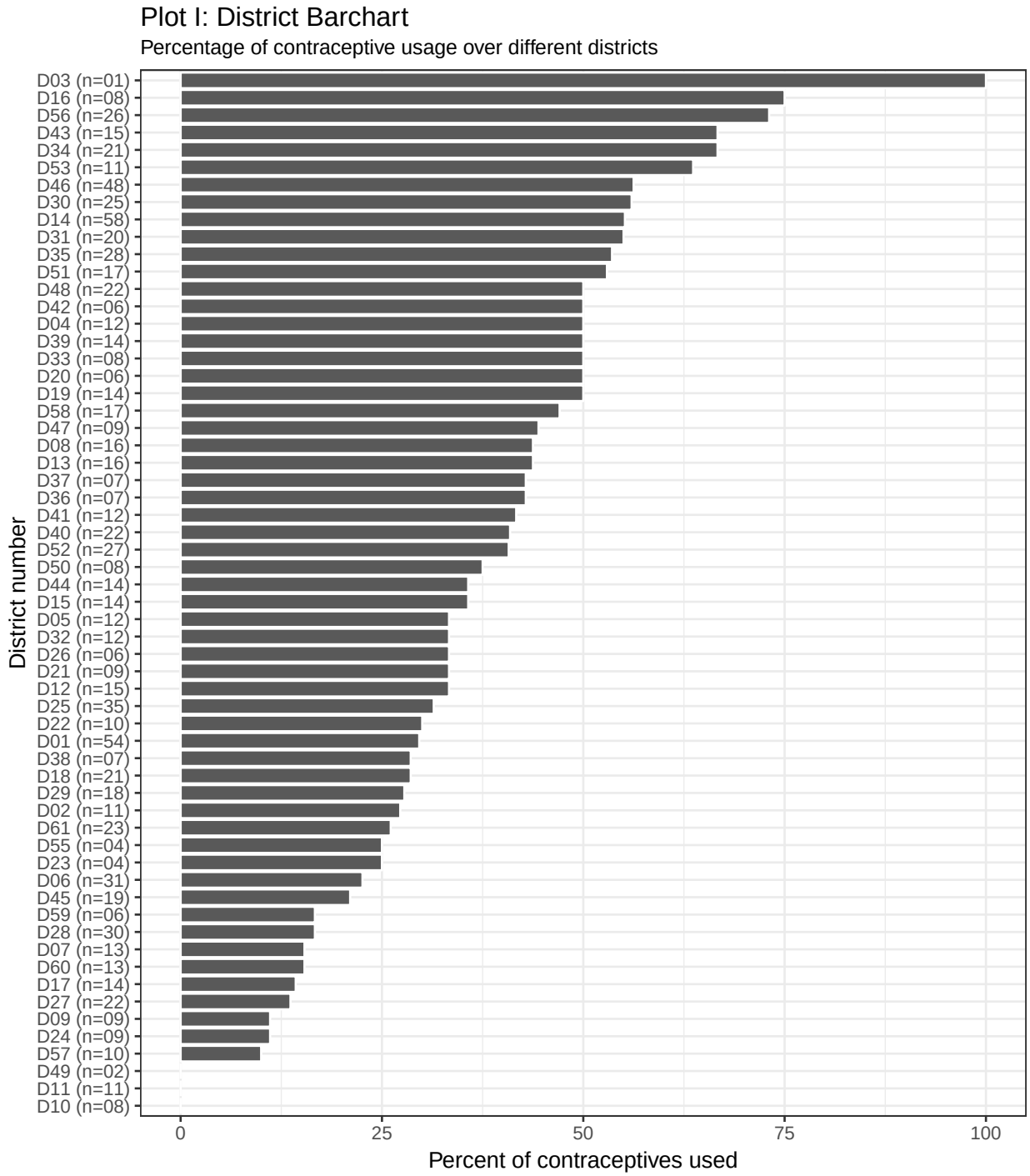
```
contrac <- df_train$contraceptive_use
urban <- df_train$urban
xtab <- table(contrac, urban)
pander(ftable(xtab))
```

	"urban"	"0"	"1"
"contrac"			
"No"		445	145
"Yes"		225	152

Table I shows, that the underlying dataset is not balanced, i.e., more women are not-using contraceptives compared to women using contraceptives.

Visualize data

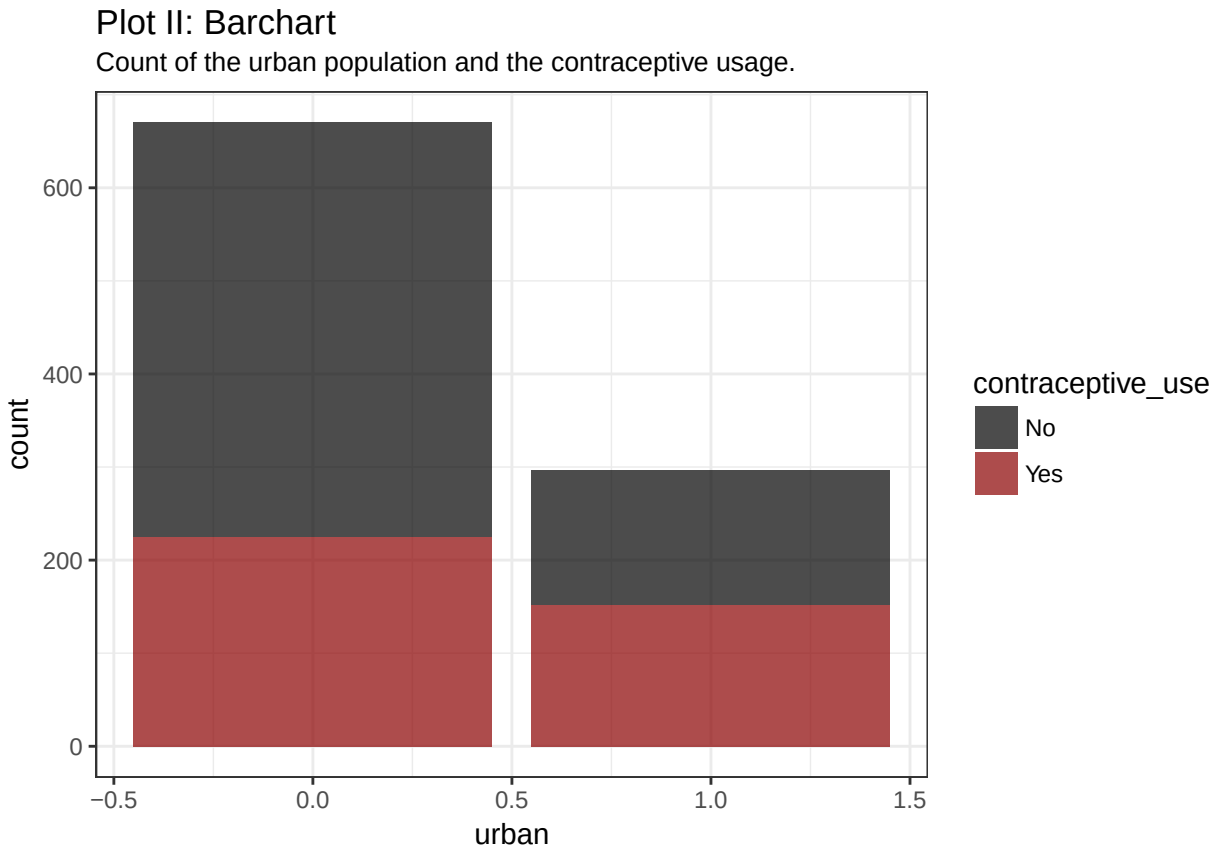
```
ggplot(df_dist, aes(x=distric, y=perc_contra)) +
    labs(title="Plot I: District Barchart",
       subtitle="Percentage of contraceptive usage over different districts") +
  geom_bar(stat="identity", colour="white") +
  theme_bw() +
  ylab("Percent of contraceptives used") +
  xlab("District number") +
  coord_flip()
```



The above plot I shows, that the percentage of contraceptives used in the different districts varies heavily from zero percent in district 10, 11 and 49 to 100% in the district 3. However, we have to be a bit careful with the interpretation as the sum of people in the different districts varies heavily. This can for example be seen in district 3 which counts only 1 women. The 100% contraceptive usage in that district is therefore to be taken with a certain amount of caution.

```
ggplot(df_train, aes(x=urban, fill=contraceptive_use))  +
  geom_bar() +
  labs(title="Plot II: Barchart",
```

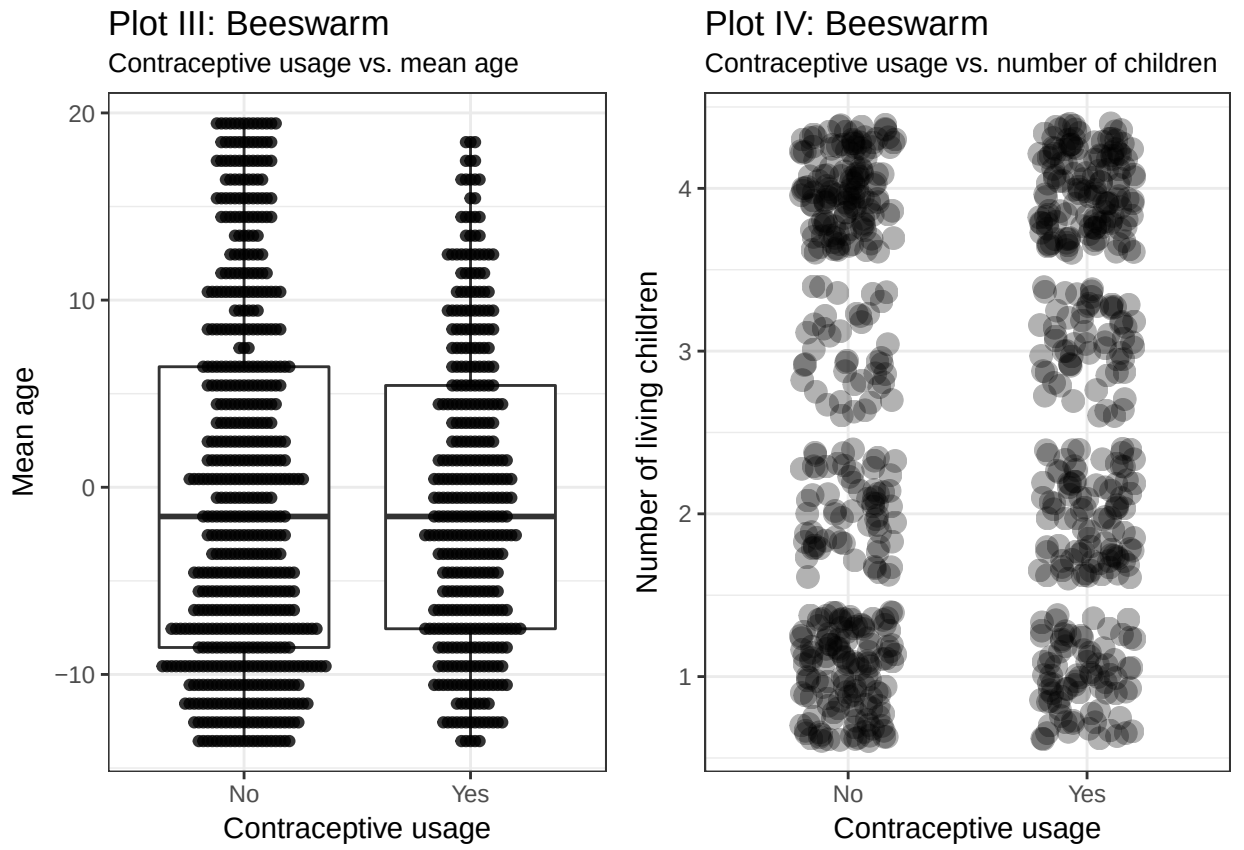
```
  subtitle="Count of the urban population and the contraceptive usage.") +
  scale_fill_manual(values=alpha(c("black", "darkred"), 0.7)) +
  theme_bw()
```



Plot II shows that as a percentage of the population, there is greater contraceptive usage in urban rather than rural districts. In rural districts, the majority of the women in the data, do not use contraception.

```
g1 <- ggplot(df_train, aes(y=age_mean, x=factor(contraceptive_use))) +
  labs(title="Plot III: Beeswarm",
       subtitle="Contraceptive usage vs. mean age") +
  geom_boxplot() +
  geom_beeswarm(color="black", alpha = 0.8) +
  xlab("Contraceptive usage") +
  ylab("Mean age") +
  theme_bw()

# Change class balance
df_yes <- df_train[df_train$contraceptive_use == "Yes", ]
df_no <- df_train[df_train$contraceptive_use == "No", ]
set.seed(123)
row_ids <- sample(row.names(df_no), size=nrow(df_yes), replace=FALSE)
df_no <- df_no[row.names(df_no) %in% row_ids, ]
df <- rbind(df_yes, df_no)

g2 <- ggplot(df, aes(y=living_children, x=factor(contraceptive_use))) +
  labs(title="Plot IV: Beeswarm",
```

```
      subtitle="Contraceptive usage vs. number of children") +
  geom_jitter(alpha = 0.3, width = 0.2, shape = 21, colour="black",
              fill = "black", size = 4, stroke = 0) +
  xlab("Contraceptive usage") +
  ylab("Number of living children") +
  theme_bw()

grid.arrange(g1, g2, nrow=1, ncol=2)
```




```
rm(df_yes, df_no, row_ids, df)
```

Plot III shows a beeswarm plot with a boxplot superimposed on it. The underlying data for the plot is the the mean age vs the contraceptive usage. It can be seen, that there is a very slight difference in the median of the mean age. Whereby the median mean age of the women using a contraceptive appears to be higher. However, the difference might not be significant.

Plot IV on the other hand gives an indication that the number of children could also be related to the contraceptive usage as the amount in each category varies.

A separate logistic regression model for each district

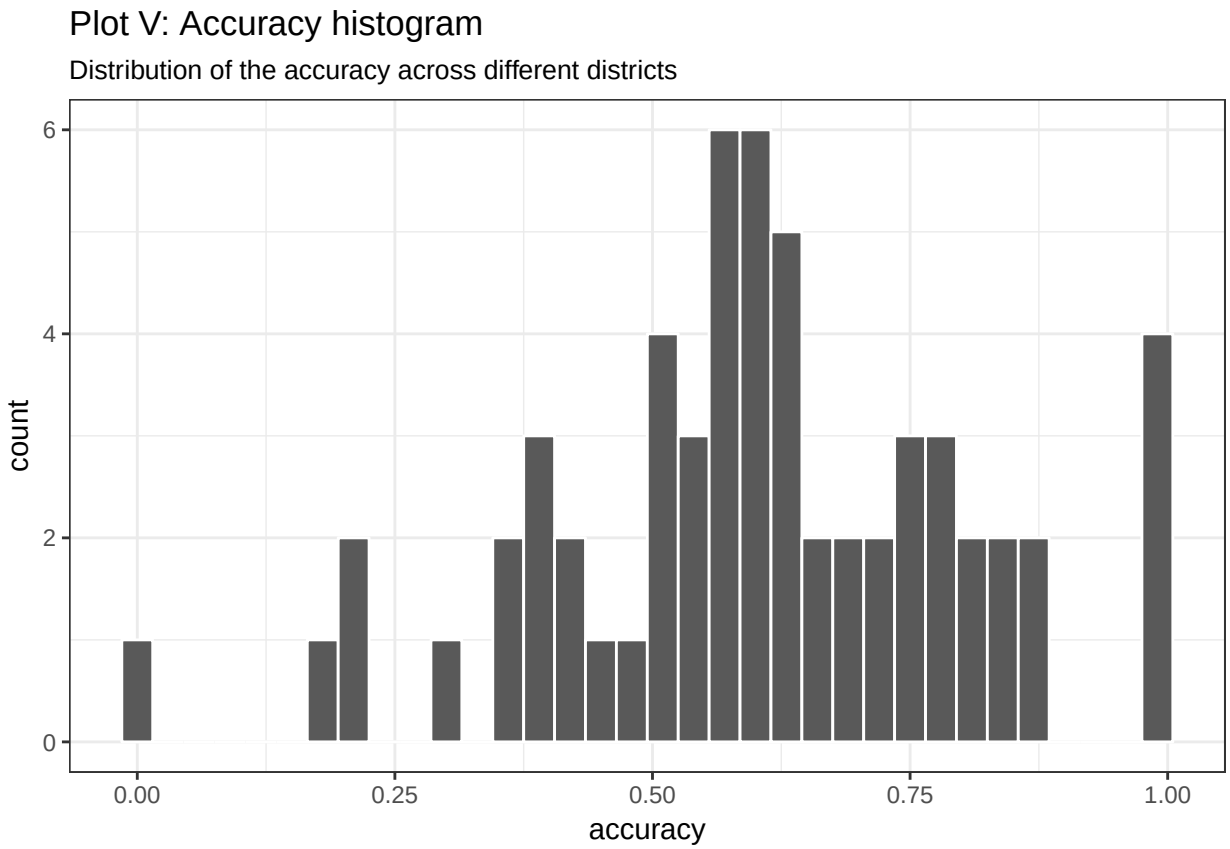
Calculation

```
# Get unique districts
districts <- as.character(unique(df_train$district))
```

```
# Create empty dataframe
df_dist_acc <- data.frame(district=numeric(), accuracy=numeric())

# Iterate over the models
for (i in 1:length(districts)){

  d <- districts[i]
  # Create subsets
  subset_train <- df_train[df_train$district == d, ]
  subset_test <- df_test[df_test$district == d, ]

  # Remove district
  subset_train$district <- NULL
  subset_test$district <- NULL


  # Fit model
  subset_fit <- glm(contraceptive_use ~ .,
                    data=subset_train,
                    family=binomial(link="logit"))

  # Predict
  prt_dist <- round(predict(subset_fit,
                            newdata=subset_test,
                            type="response"))

  # Confusion matrix
  table_log_test <- table(prt_dist, subset_test$contraceptive_use)

  # Accuracy
  acc_log_test <- sum(diag(table_log_test)) / sum(table_log_test)

  # Add to dataframe
  df_dist_acc[i, "district"] <- as.numeric(d)
  df_dist_acc[i, "accuracy"] <- acc_log_test
}
```

Visualizating the accuracy rates

```
ggplot(df_dist_acc, aes(x=accuracy))  +
  geom_histogram(binwidth=0.03, colour="white") +
  labs(title="Plot V: Accuracy histogram",
  subtitle="Distribution of the accuracy across different districts") +
  theme_bw()
```



The above plot shows, that the accuracy rate varies heavily across the different districts. The overall mean average lies at 60.0 %. The problem with this numbe is, that each district accuracy rate has the same weight in the overall average. This can be corrected using a weighted average.

Weighted accuracy

```
df_dist_acc <- merge(df_dist_acc,
                     data.frame(table(df_train$district)),
                     by.x="district", by.y="Var1")

weighted_acc <- sum((df_dist_acc$Freq/sum(df_dist_acc$Freq) * df_dist_acc$accuracy))
```

The weighted average shows that the accuracy of individual models for each district is at around sprintf("%.1f", weighted_acc * 100)‘ %. In the next steps we can compare this number with a single logistic regression model.

A single logistic regression model using the entire training set

```
# Prepare the data
df_model_train <- df_train
df_model_test <- df_test
df_model_train$district <- NULL
df_model_test$district <- NULL
```

```r
# Fit model
fit_log <- glm(contraceptive_use ~ .,
                    data=df_model_train,
                    family=binomial(link="logit"))

# Predict
prt_log <- round(predict(fit_log,
                         newdata=df_model_test,
                         type="response"))

# Confusion matrix
table_log_test <- table(prt_log, df_model_test$contraceptive_use)

# Accuracy
acc_log_test <- sum(diag(table_log_test)) / sum(table_log_test)
```

The mean accuracy of the single logistic regression model is with an accuracy rate of 64.7 % higher than the individual models for each district. The reason for this is most likely that due to the small sample sizes in some of the districts the individual model heavily overfit the training data and therefore can't reach the same accuracy rate.