Midterm Exam 1

Advanced Topics in Data Science II
Harvard University, Spring 2017

Tim Hagmann

February 25, 2017

Contents

The set of questions below address the task of predicting the merit of a restaurant on Yelp. Each restaurant is described by a set of business attributes, and is accompanied by a set of text reviews from customers. For the purpose of the problems below, the average rating (originally on a scale from 0 to 5) was converted into a binary variable depending on whether the average was above 3.5, in which case it is considered "good" (labeled 1), or below 3.5 in which case it is considered "bad" (labeled 0). The overall goal is to predict these binary ratings from the information provided.

The data are split into a training and test set and are in the files `dataset_1_train.txt` and `dataset_1_test.txt` respectively. The first column contains the rating for the restaurant (0 or 1), columns 2-21 contain the business attributes, and columns 22-121 contain text features extracted from the customer reviews for the restaurant. The details about the business attributes are provided in the file `dataset_1_description.txt`.

We use the bag-of-words encoding to generate the text features, where the set of reviews for a restaurant are represented by a vector of word counts. More specifically, we construct a dictionary of 100 frequent words in the customer reviews, and include 100 text features for each restaurant: the i-th feature contains the number of times the dictionary word i occurs in customer reviews for the restaurant. For example, a text feature 'fantastic' with value 18 for a restaurant indicates that the word 'fantastic' was used a total of 18 times in customer reviews for that restaurant.

Preparation

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

Initialize

```
## 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 data

```
## Read data
df_train <- read.csv("data/dataset_1_train.txt", stringsAsFactors=TRUE)
df_test <- read.csv("data/dataset_1_test.txt", stringsAsFactors=TRUE)
```

Problem 1 [20 points]

Does the location of a restaurant relate to its rating? Construct a compelling visualization to address this question, and write a brief (under 300 words) summary that clearly explains your analysis and conclusions to someone without a data science background.

Preprocess data

To determine if the location of a restaurant is a contributing factor to it's rating, a different number of visualization can be applied on the training data. There are 5 geograpic variables present in the data: **latitude**, **longitude**, **postal code**, **city**, **state**. For the following analysis the most granular data with the **latitude** and **longitude** coordinates as well as the state information is beeing.

```
# Extract words
words <- names(df_train[22:length(df_train)])

# Create factor variables
df_train$rating <- factor(df_train$rating, labels=c("bad", "good"))
df_test$rating <- factor(df_test$rating, labels=c("bad", "good"))
```

Aggregate

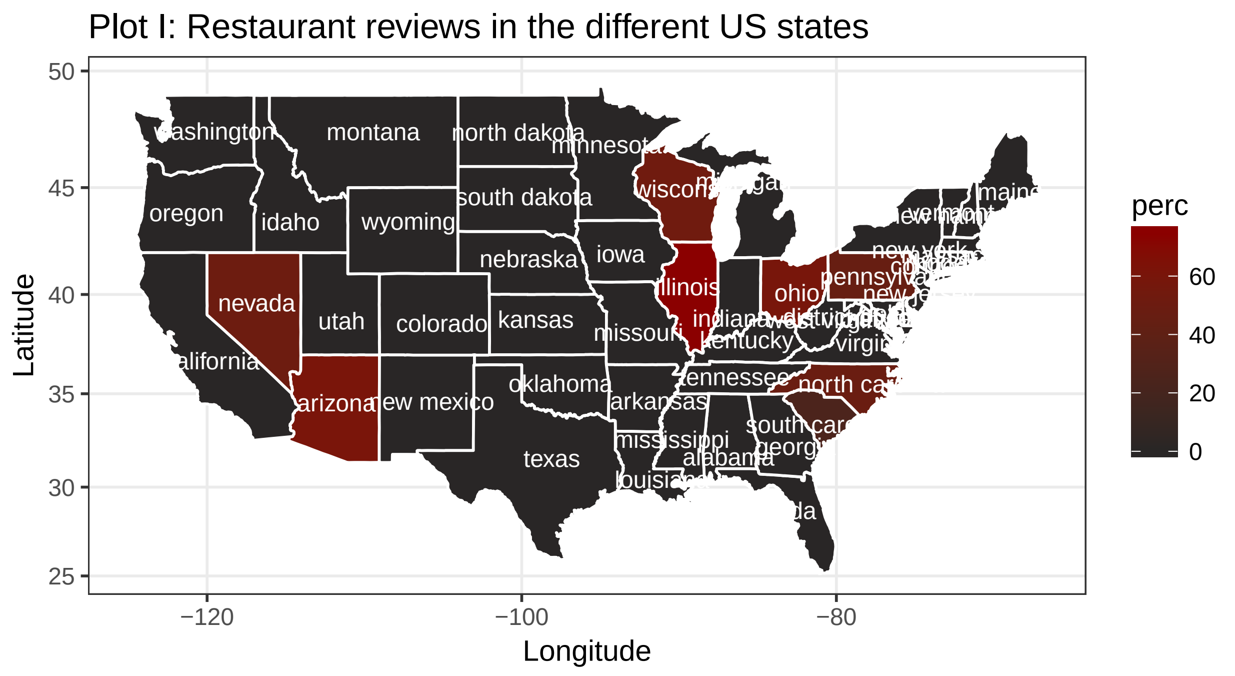
```
# Aggregate data on a districe level
df_state1 <- aggregate(rating ~ state, data=df_train, FUN=length)
df_state2 <- aggregate((as.numeric(df_train$rating) - 1) ~ state,
                       data=df_train, FUN=sum)

# Left join data
df_state <- merge(df_state1, df_state2, by="state", all=TRUE)
names(df_state) <- c("state", "n_review", "sum_good_reviews")

# Rename states
```

```r
df_state$id <- df_state$state
df_state$state_name <- df_state$state
levels(df_state$state_name) <- c("arizona", "illinois", "north carolina", "nevada",
                                  "ohio", "pennsylvania", "south carolina", "wisconsin")

# Calculations
df_state$perc <- round((df_state$sum_good_reviews / df_state$n_review) * 100, 2)

# Change order of the states
df_state$state <- factor(df_state$state,
                          levels=df_state$state[
                            order(df_state$perc, decreasing=FALSE)])

df_state <- df_state[order(df_state$perc, decreasing=FALSE), ]

# Add number reviews on a state level (n=xxx) for the visualization below
levels(df_state$state) <- paste0("State:", levels(df_state$state),
                                  " (n=", sprintf("%.2d", df_state$n_review), ")")
rm(df_state1, df_state2)
```

Geovisualization

The first step in the analyis is to visualize the data on a state level.

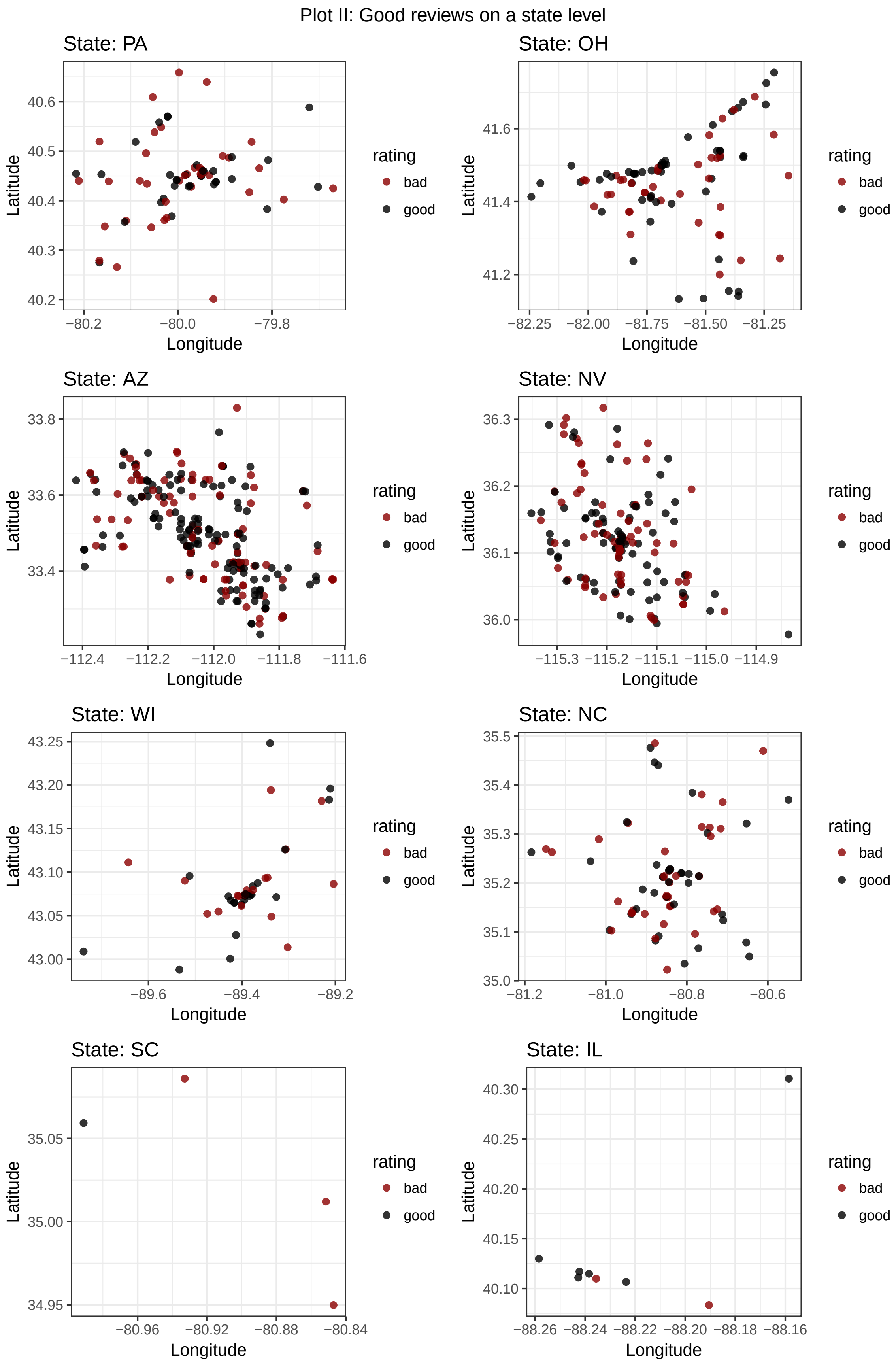
```r
plot_map(df_state)
```



The above plot shows the distribution of the good reviews on a state level. At zero percent no data is available. There is only review data for the following states available: Nevada, Arizona, Wisconsin, Illinois, Ohio, Pennsylvania, North Carolina and South Carolina. The lighter the red bar, the better are the reviews. There appears to be a difference on the state level. The next step is to visualize the data on a state level.

Inside the States

```r
# Prepare for visualization
p <- c()
states <- unique(df_train$state)

# Iterate trough the states
for(i in 1:length(states)){
  p[[i]] <- ggplot(df_train[df_train$state == states[i], ],
                 aes(x=longitude, y=latitude, color=rating)) +
    geom_point(stroke=1, size=1, alpha=0.8) +
    theme_bw() +
    labs(title=paste("State:", states[i])) +
    ylab("Latitude") +
    scale_color_manual(values=c("darkred", "black")) +
    xlab("Longitude")
}

do.call(grid.arrange, c(p, list(ncol=2, top="Plot II: Good reviews on a state level")))
```

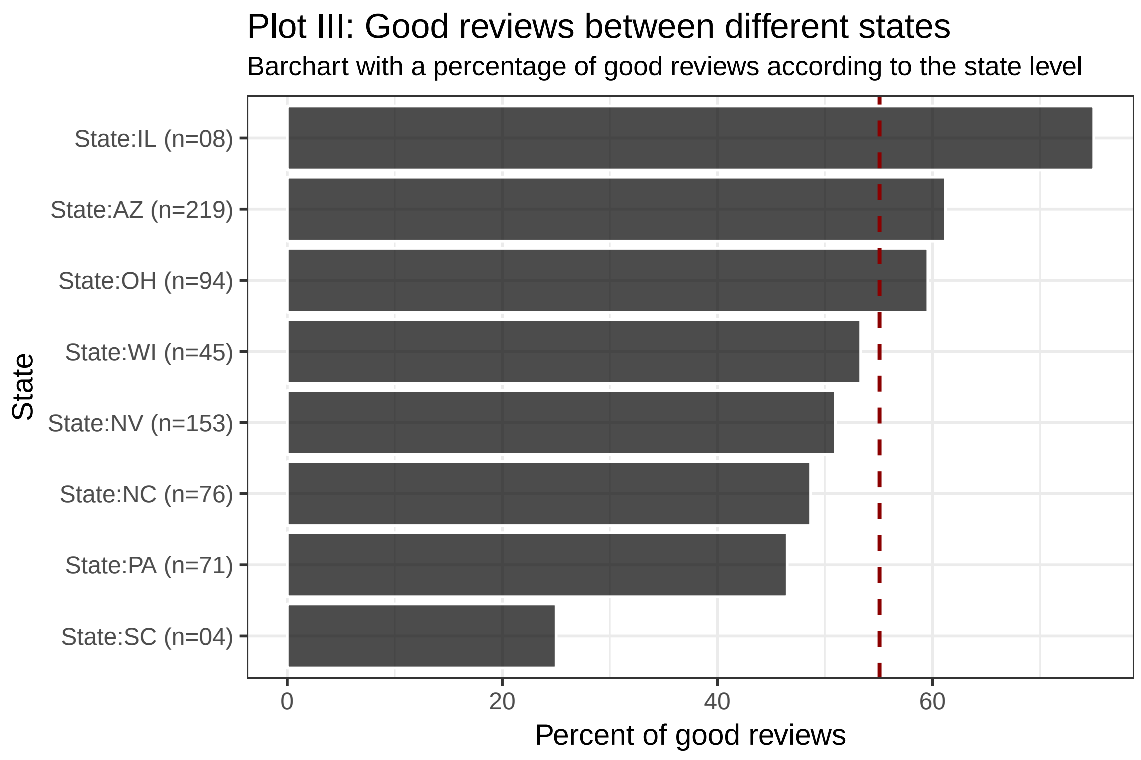


Plot II: Good reviews on a state level
State: PA
Latitude
40.6
40.5
40.4
40.3
40.2
−80.2
−80.0
−79.8
Longitude
rating
bad
good
State: OH
41.6
41.4
41.2
−82.25
−82.00
−81.75
−81.50
−81.25
State: AZ
33.8
33.6
33.4
−112.4
−112.2
−112.0
−111.8
−111.6
State: NV
36.3
36.2
36.1
36.0
−115.3
−115.2
−115.1
−115.0
−114.9
State: WI
43.25
43.20
43.15
43.10
43.05
43.00
−89.6
−89.4
−89.2
State: NC
35.5
35.4
35.3
35.2
35.1
35.0
−81.2
−81.0
−80.8
−80.6
State: SC
35.05
35.00
34.95
−80.96
−80.92
−80.88
−80.84
State: IL
40.30
40.25
40.20
40.15
40.10
−88.26
−88.24
−88.22
−88.20
−88.18
−88.16

The above plot II shows, that there appears to be no regional difference on a state level, i.e., the good and bad reviews appear to be evenly spread among the different states. That means that there is no visible in-state regional difference among good and bad reviews. It appears that there is a between-state difference but no in-state difference. Next we're having a closer look at the difference at the between-state difference.

Between-state difference

```
# Calculate the mean review percentage
mean_review <- mean(as.numeric(df_train$rating)-1) * 100

# Visualize
ggplot(df_state, aes(x=state, y=perc)) +
    labs(title="Plot III: Good reviews between different states",
       subtitle=
         "Barchart with a percentage of good reviews according to the state level") +
  geom_bar(stat="identity", colour="white", fill="black", alpha=0.7) +
  theme_bw() +
  ylab("Percent of good reviews") +
  xlab("State") +
  geom_hline(yintercept=mean_review, linetype="dashed",
  color="darkred", size=0.7) +
  coord_flip()
```



Report

As already noted, there appears to be a slight difference concerning the percentage of good and bad reviews between the states. On average, reviewers give in 55.1% of the cases a good review (noted by the **red line**).

The above plot shows, that Illinois, Arizona and Ohio have an above average of good reviews while the other states are below average.

Illinois has the highest amount of good reviews while South Carolina has the highest amount of bad reviews. However, there were only 8 reviews in Illinois and 4 in South Carolina, which puts there rank on a shaky foundation. Only one good respectively bad review and they switch there rank with another state.

Looking at the state with more reviews shows, that Arizona is in the first place concerning good reviews (60%) and Pennsylvania is on last place with only around 45% good reviews. Looking at the overall variability in the data, it shows, that the states with sufficiently many reviews the amount of good and bad reviews don't vary much around the overall average.

Overall, the location of a restaurant appears to have only a small effect on its review. This was also visible in Plot II where there was no in-state connection between location and review visible. This analysis suggests that predicting the rating of a restaurant based on its location alone isn't be advisable.

Problem 2 [35 points]

This problem is concerned with predicting a restaurant's rating based on text features. We'll consider the Multinomial-Dirichlet Bayesian model to describe the distribution of text features and finally to predict the binary ratings.

Probability model: Let $(y^g_1, y^g_2, \ldots, y^g_{100})$ denote the total counts of the 100 dictionary words across the reviews for "good" restaurants, and $(y^b_1, y^b_2, \ldots, y^b_{100})$ denote the total counts of the 100 dictionary words across the reviews for "bad" restaurants. We assume the following *multinomial* likelihood model:

$$p(y^g_1, y^g_2, \ldots, y^g_{100} \,|\, \theta^g_1, \ldots, \theta^g_{100}) \propto (\theta^g_1)^{y^g_1}(\theta^g_2)^{y^g_2}\ldots(\theta^g_{100})^{y^g_{100}}$$

$$p(y^b_1, y^b_2, \ldots, y^b_{100} \,|\, \theta^b_1, \ldots, \theta^b_{100}) \propto (\theta^b_1)^{y^b_1}(\theta^b_2)^{y^b_2}\ldots(\theta^b_{100})^{y^b_{100}}.$$

The model parameters $(\theta^g_1, \ldots, \theta^g_{100})$ and $(\theta^b_1, \ldots, \theta^b_{100})$ are assumed to follow a *Dirichlet* prior distribution with parameter α. That is

$$p(\theta^g_1, \ldots, \theta^g_{100}) \propto (\theta^g_1)^{\alpha}\ldots(\theta^g_{100})^{\alpha}$$
$$p(\theta^b_1, \ldots, \theta^b_{100}) \propto (\theta^b_1)^{\alpha}\ldots(\theta^b_{100})^{\alpha}.$$

Hence we can interpret, for example, θ^g_5 as the probability the word "perfect" is observed once in a review of "good" restaurants. For the purposes of this problem, set $\alpha = 2$.

(a) Dirichlet posterior distribution

Describe briefly in words why the posterior distribution formed from a Dirichlet prior distribution and a multinomial likelihood is a Dirichlet posterior distribution? What are the parameters for the Dirichlet posterior distribution? [5 points]

A Dirichlet distribution is the conjugate prior for the multinomial likelihood distribution. This is the case because the posterior Dirichlet distribution is in the same family as the prior probability distribution. In other words, a posterior distribution formed from a Dirichlet prior and Multinomial likelihood will follow a Dirichlet distribution (even though with different parameters).

The Multinomial distribution is the probability to observe every possible outcome c_i X_i times in a sequence of n trials **(1/0)**. Which is parameterised by a vector of occurence counts **(N)**. The Dirichlet distribution is a density distribution over **n** positive numbers parameterised by a vector α. The Dirichlet Multinomial posterior distribution has the parameters **N** and α.

(b) Monte Carlo Simulation

From a Monte Carlo simulation of the Dirichlet posterior distribution for “good” restaurants, what is the posterior mean probability that the word “chocolate” is used? From a Monte Carlo simulation of the Dirichlet posterior distribution for bad restaurants, what is the posterior mean probability that the word “chocolate” is used? [15 points]

MC Simulation

```
# the total word counts from all articles by author A in the training set
yA <- as.numeric(apply(df_train[df_train$rating == 'good', words], 2, sum))
yB <- as.numeric(apply(df_train[df_train$rating == 'bad', words], 2, sum))

# Apply a posterior mean function
set.seed(123)
ER_A <- posterior_mean(alpha=2, y=yA, n_sim=3000)
ER_B <- posterior_mean(alpha=2, y=yB, n_sim=3000)

# Get the probabilty values
df_mean_values <- data.frame("word"=words, "prob_good"=ER_A, "prob_bad"=ER_B)

rm(ER_A, ER_B)
```

The posterior mean porbabilty that for a good restaurant review the word **chocolate** is used is: 1.72%. This compared to: 1.72% for a bad restaurant.

(c) Calculate the Dirichlet-Multinomial model

For the restaurants in the test data set, estimate the probability based on the results of the Dirichlet-Multinomial model that each is good versus bad. Create a visual summary relating the estimated probabilities and the actual binary ratings in the test data. [15 points]

We can use the posterior_pA function (**scripts/functions.R**) to infer the rating for a given restaurant review.

Note: The Dirichlet parameter *alpha* is set to 2.

Calculation

```
prob_dirichlet <- rep(NA, nrow(df_test))
for(i in 1:nrow(df_test)){
  y_til = as.numeric(as.character(df_test[i, words]))
  prob_dirichlet[i] = posterior_pA(alpha=2, yA=yA, yB=yB, y_til=y_til)
}

df_pred <- data.frame(df_test$rating, prob_dirichlet)
names(df_pred) <- c("actual", "prob_good")
miss_class_rate <- mclust::classError(df_pred$prob_good > 0.5,
                                      df_pred$actual)$errorRate * 100
```

Confusion matrix

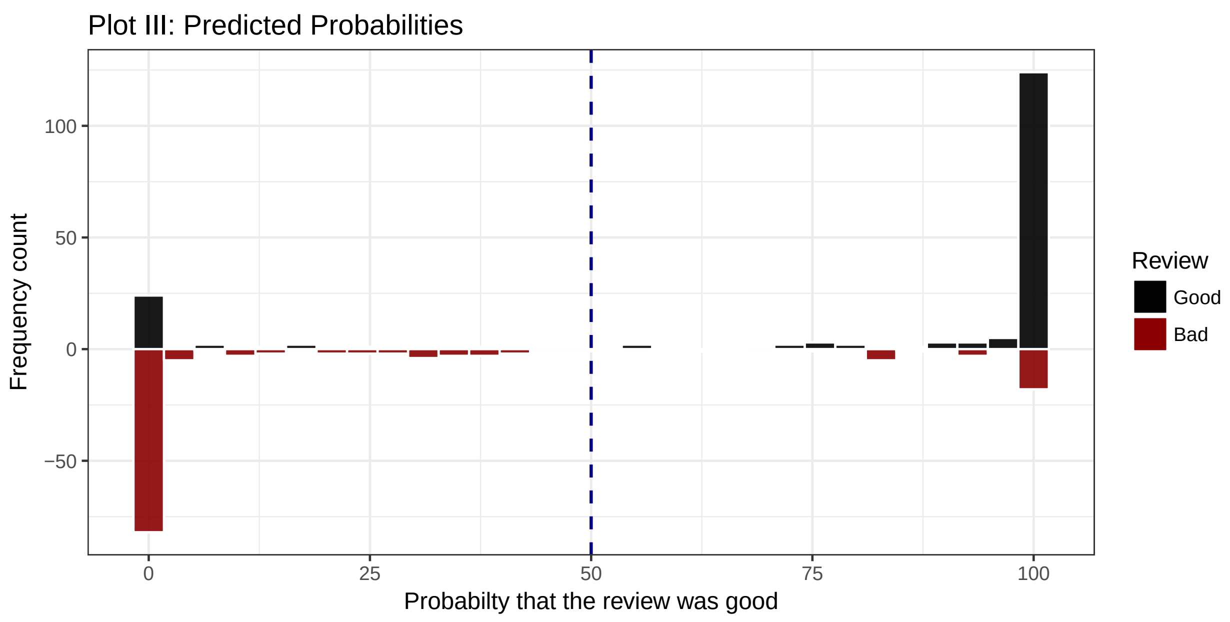
```
# Confusion matrix
pander(table(df_pred$actual, ifelse(df_pred$prob_good > 0.5, 1, 0)))
```

	0	1
bad	111	35
good	34	148

The confusion matrix shows that some good restaurants are misclassified as bad and vice verca. This can be also shown in a visualization. The overall missclassification rate is at: 21.04%.

Visualization

```
df_pred$prob_good <- df_pred$prob_good * 100
ggplot() +
  labs(title="Plot III: Predicted Probabilities") +
  geom_histogram(data=df_pred[df_pred$actual == "good", ],
                 aes(prob_good, y = ..count.., fill="b", color="b"),
                 alpha=0.9, bins=30) +
  geom_histogram(data=df_pred[df_pred$actual == "bad", ],
                 aes(prob_good, y = -..count.., fill="r", color="r"),
                 alpha=0.9, bins=30) +
  scale_colour_manual(name="Review", values=c("r"="white", "b"="white"),
                      labels=c("b"="Good", "r"="Bad")) +
  scale_fill_manual(name="Review", values=c("r"="darkred", "b"="black"),
                    labels=c("b"="Good", "r"="Bad")) +
  geom_vline(xintercept=50, linetype="dashed",
             color="darkblue", size=0.7) +
  ylab("Frequency count") +
  xlab("Probabilty that the review was good") +
  theme_bw()
```



The histogram above shows, that for the good restaurant reviews **(black)** some restaurants are missclassified as bad, i.e., the one that are below the 50% mark. The same is true for the bad restaurant reviews **(red)**, where some bad reviews are wrongly labled as good.

Problem 3 [45 points]

This problem is concerned with modeling a restaurant's rating on factors other than word occurrences.

(a) Model based on business attributes

Construct a model for the probability a restaurant is rated "good" as a function of latitude and longitude, average word count, and business attributes. Include quantitative predictor variables as smoothed terms as you see appropriate. You may use default tuning parameter. Summarize the results of the model. Does evidence exist that the smoothed terms are significantly non-linear? Produce visual displays to support your conclusions. [20 points]

In order to model the different business attributes a GAM model with smoothing splines for the quantitative factors can be used. Before starting to model we do some preprocessing and data cleaning.

Preprocess data

```r
# Define business attributes from 'data/dataset_1_description.txt'
business_attributes <- c("rating","latitude", "longitude", "cuisine",
                         "WheelchairAccessible", "WiFi", "BusinessAcceptsCreditCards",
                         "Alcohol", "NoiseLevel", "RestaurantsPriceRange2",
                         "RestaurantsAttire", "Smoking", "RestaurantsReservations",
                         "OutdoorSeating", "GoodForKids", "avg_word_count")

# Create new dataframes
df_train2 <- df_train[, business_attributes]
```

```r
df_test2 <- df_test[, business_attributes]

# Normalize data
df_train2$longitude <- as.numeric(scale(df_train2$longitude))
df_train2$latitude <- as.numeric(scale(df_train2$latitude))
df_test2$longitude <- as.numeric(scale(df_test2$longitude))
df_test2$latitude <- as.numeric(scale(df_test2$latitude))
df_train2$avg_word_count <- as.numeric(scale(df_train2$avg_word_count))
df_test2$avg_word_count <- as.numeric(scale(df_test2$avg_word_count))
```

Cleanup

```r
# Remove non zero values
nzv <- c("WheelchairAccessible", "BusinessAcceptsCreditCards", "RestaurantsAttire")
pander(summary(df_train2[, nzv]))
```

WheelchairAccessible	BusinessAcceptsCreditCards	RestaurantsAttire
no : 31	no : 9	casual:665
yes:639	yes:661	dressy: 4
NA	NA	formal: 1

```r
df_train2 <- df_train2[, !names(df_train2) %in% nzv]
df_test2 <- df_test2[, !names(df_train2) %in% nzv]
rm(nzv)
```

Values which have a near zero variance are removed, as they are prown to overfitting and furthermore lead to issues in crossvalidation.

In order to find the best spar parameter value a 5-fold crossvalidation is beeing performed.

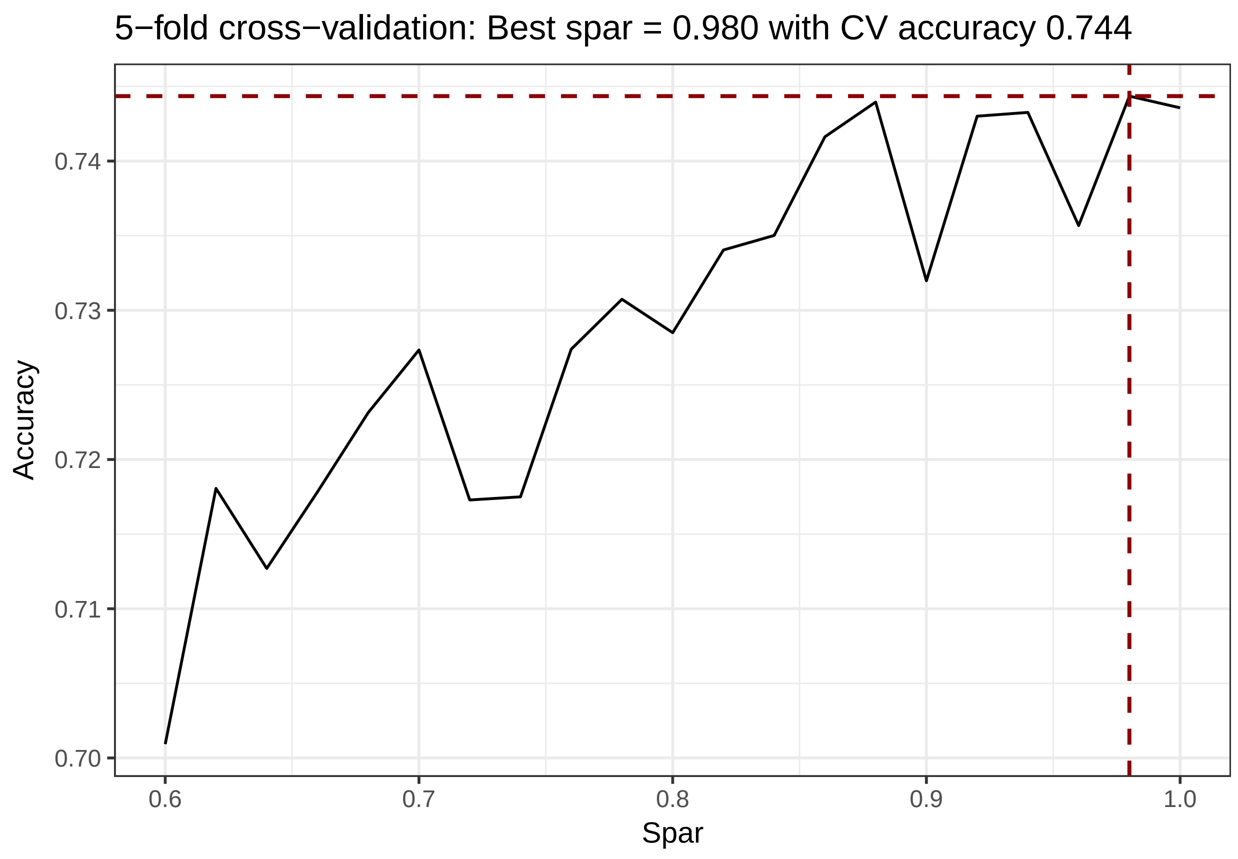
Find best spar value trough 5-fold crossvalidation

```r
# Crossvalidate best spar value
cv_gam_spar <- function(spars, k_folds, data){
  acc <- rep(NA, length(spars))
  for (i in 1:length(spars)) {
    gam_formula <- as.formula(paste0("rating ~ s(longitude, spar=",spars[i],") +
                                     s(latitude, spar=",spars[i],") +
                                     s(avg_word_count, spar=",spars[i],") +
                                     cuisine + RestaurantsPriceRange2 +
                                     Smoking + RestaurantsReservations +
                                     WiFi + OutdoorSeating + GoodForKids"))
    model_gam <- gam(gam_formula, data=data, family=binomial(link = "logit"))
    acc[i] <- 1 - boot::cv.glm(data, model_gam, K=k_folds)$delta[1]
  }

  df_spars <- data.frame("spar_val"=spars, "accuracy"=acc)
  df_spars
}
```

```r
set.seed(123)
df_cv_spars <- cv_gam_spar(spars=seq(0.6, 1, 0.02), k_folds=5, data=df_train2)
```

Visualize value

```r
# Find spar with highest CV accuracy
df_best <- df_cv_spars[df_cv_spars$accuracy == max(df_cv_spars$accuracy), ]

# Plot - Classification accuracy as a function of Spar values
ggplot(df_cv_spars, aes(x=spar_val, y=accuracy)) +
  geom_line() +
  labs(x="Spar",
       y="Accuracy",
       title=sprintf("5-fold cross-validation: Best spar = %.3f with CV accuracy %.3f",
                     df_best$spar_val, df_best$accuracy)) +
  geom_hline(yintercept=df_best$accuracy, linetype="dashed",
             color="darkred", size=0.7) +
  geom_vline(xintercept=df_best$spar_val, linetype="dashed",
             color="darkred", size=0.7) +
  theme_bw()
```



The optimal value appears to be at a spar of 0.98. This value is used for the modeling. In order to have a benchmark three modells are beeing build, a Null model, a full model without splines and a model with splines.

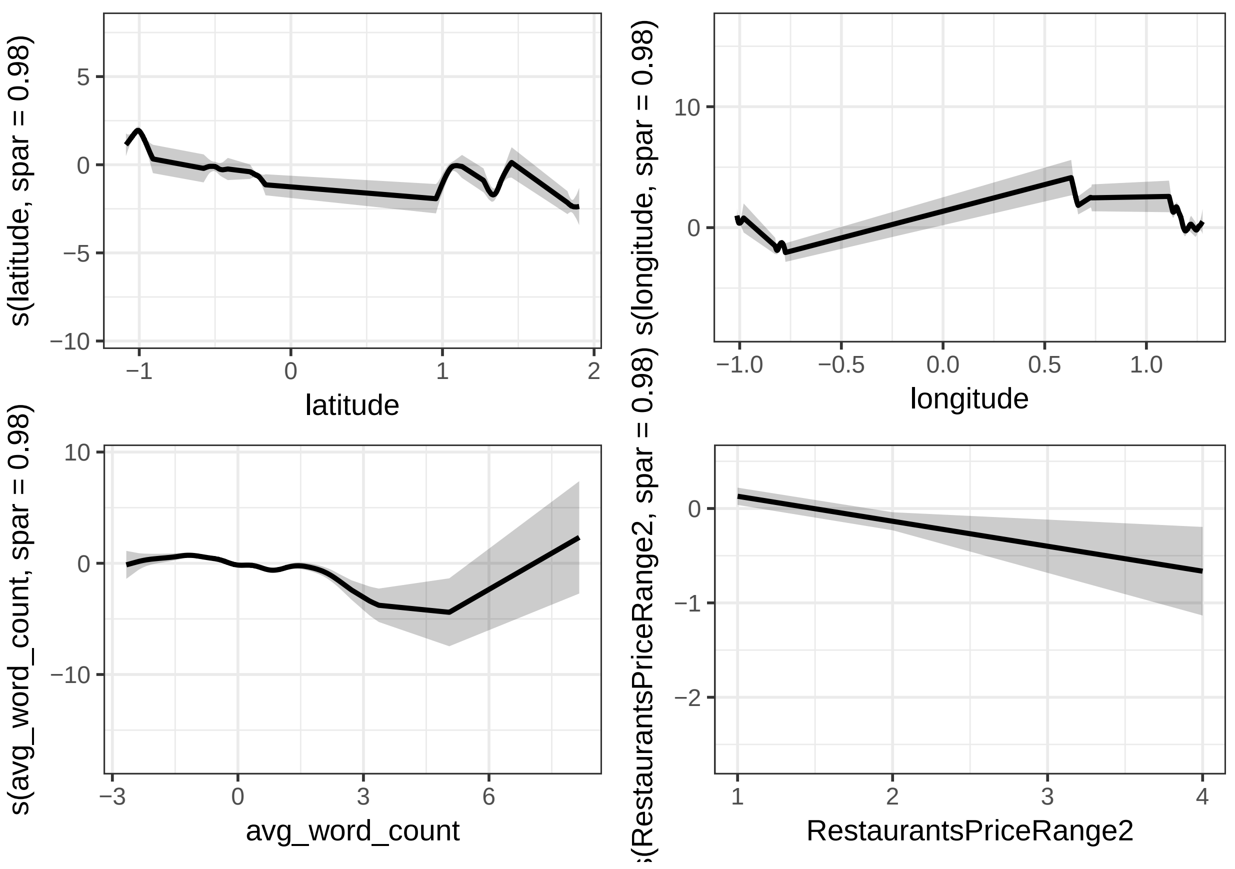
Fit different models

```r
# GAM with only the intercept
fit_gam_intercept <- gam(rating ~ 1, family=binomial(link="logit"),
                         data=df_train2)

# GAM full model
fit_gam_lin <- gam(rating ~ ., family=binomial(link="logit"),
                   data=df_train2)

# GAM with splines
fit_gam_spl <- gam(rating ~ s(latitude, spar=0.98) +
                     s(longitude, spar=0.98) +
                     s(avg_word_count, spar=0.98) +
                     s(RestaurantsPriceRange2, spar=0.98) +
                     cuisine + WiFi + Alcohol + NoiseLevel +
                     Smoking + RestaurantsReservations +
                     OutdoorSeating + GoodForKids,
                   family=binomial(link="logit"),
                   data=df_train2)
```

Looking a the smooth of the predictors can indicate if the smoothed term are linear or non-linear.

Visualize

```r
g1 <- plot_splines(model=fit_gam_spl, term=names(preplot(fit_gam_spl))[1])
g2 <- plot_splines(model=fit_gam_spl, term=names(preplot(fit_gam_spl))[2])
g3 <- plot_splines(model=fit_gam_spl, term=names(preplot(fit_gam_spl))[3])
g4 <- plot_splines(model=fit_gam_spl, term=names(preplot(fit_gam_spl))[4])

grid.arrange(g1, g2, g3, g4, nrow=2, ncol=2)
```



The plot above shows, that there is some non-linear behavior present. The **longitude** and **latitude** variables show non linearity as their values are being concentrated in a few specific regions (see question 1). The avg_word_count also appears to have a non-linear relationship with the response. However, at a standartized word count bigger than 5 there are few data points and the confidence intervals is large. The RestaurantsPriceRange2 on the other hand appears to be linear.

Analyse the parameters

```r
# GAM with splines
pander(summary(fit_gam_spl)[[4]],
       add.significance.stars=FALSE, style='rmarkdown', caption="",
       split.table=Inf, digits=2, justify='center')
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
s(latitude, spar = 0.98)	1	71	71	68	1.1e-15
s(longitude, spar = 0.98)	1	25	25	25	0.00000096
s(avg_word_count, spar = 0.98)	1	10	10	9.8	0.0018
s(RestaurantsPriceRange2, spar = 0.98)	1	0.11	0.11	0.11	0.74
cuisine	10	8.1	0.81	0.78	0.65
WiFi	2	2.5	1.3	1.2	0.3
Alcohol	2	7	3.5	3.4	0.036
NoiseLevel	3	13	4.3	4.1	0.0065
Smoking	2	1.2	0.62	0.6	0.55
RestaurantsReservations	1	5.6	5.6	5.4	0.021
OutdoorSeating	1	0.77	0.77	0.74	0.39

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
GoodForKids	1	3.3	3.3	3.2	0.074
Residuals	605	629	1	NA	NA

The above table shows, that the variable **cuisine** is highly insignificant. In the next steps we're excluding this variable.

Optimization

```
fit_gam_lin2 <- gam(rating ~ latitude + longitude +
                      avg_word_count + RestaurantsPriceRange2 +
                      WiFi + Alcohol + NoiseLevel +
                      Smoking + RestaurantsReservations +
                      OutdoorSeating + GoodForKids,
                    family=binomial(link="logit"),
                    data=df_train2)

fit_gam_spl2 <- gam(rating ~ s(latitude, spar=0.98) +
                      s(longitude, spar=0.98) +
                      s(avg_word_count, spar=0.98) +
                      s(RestaurantsPriceRange2, spar=0.98) +
                      WiFi + Alcohol + NoiseLevel +
                      Smoking + RestaurantsReservations +
                      OutdoorSeating + GoodForKids,
                    family=binomial(link="logit"),
                    data=df_train2)
```

Next we're looking at the missclassification rates of the different models.

(b) Missclassification rate

For your model in part (a), summarize the predictive ability by computing a misclassification rate. [10 points]

Prediction

```
# Prediction
pred_gam_intercept <- predict(fit_gam_intercept, newdata=df_test2, type="response")
pred_gam_lin <- predict(fit_gam_lin, newdata=df_test2, type="response")
pred_gam_lin2 <- predict(fit_gam_lin2, newdata=df_test2, type="response")
pred_gam_spl <- predict(fit_gam_spl, newdata=df_test2, type="response")
pred_gam_spl2 <- predict(fit_gam_spl2, newdata=df_test2, type="response")

# Confusion matrix
cm_gam_intercept <- table(Actual=df_test2$rating, Predicted=pred_gam_intercept > 0.5)
cm_gam_lin <- table(Actual=df_test2$rating, Predicted=pred_gam_lin > 0.5)
cm_gam_lin2 <- table(Actual=df_test2$rating, Predicted=pred_gam_lin2 > 0.5)
cm_gam_spl <- table(Actual=df_test2$rating, Predicted=pred_gam_spl > 0.5)
cm_gam_spl2 <- table(Actual=df_test2$rating, Predicted=pred_gam_spl2 > 0.5)
```

```
# Accuracy (TP+TN)/total
acc_gam_intercept <- sum(diag(cm_gam_intercept)) / sum(cm_gam_intercept)
acc_gam_lin <- sum(diag(cm_gam_lin)) / sum(cm_gam_lin)
acc_gam_lin2 <- sum(diag(cm_gam_lin2)) / sum(cm_gam_lin2)
acc_gam_spl <- sum(diag(cm_gam_spl)) / sum(cm_gam_spl)
acc_gam_spl2 <- sum(diag(cm_gam_spl2)) / sum(cm_gam_spl2)
accuracy <- c(acc_gam_intercept, acc_gam_lin, acc_gam_lin2, acc_gam_spl, acc_gam_spl2)

# Misslassification rate (FP+FN)/total or 1-accuracy
mcr_gam_intercept <- 1 - acc_gam_intercept
mcr_gam_lin <- 1 - acc_gam_lin
mcr_gam_lin2 <- 1 - acc_gam_lin2
mcr_gam_spl <- 1 - acc_gam_spl
mcr_gam_spl2 <- 1 - acc_gam_spl2
missclass <- c(mcr_gam_intercept, mcr_gam_lin, mcr_gam_lin2, mcr_gam_spl, mcr_gam_spl2)
```

Accuracy an missclassification rate on teh test data

```
# Names
model_names <- c("Null Model", "Full wo splines", "Optimized wo splines",
                 "Full GAM w splines", "Optimized GAM w splines")
# Dataframe
pander(data.frame("Model"=model_names, "Accuracy"=accuracy,
                  "Missclassification"=missclass))
```

Model	Accuracy	Missclassification
Null Model	0.4451	0.5549
Full wo splines	0.5762	0.4238
Optimized wo splines	0.5854	0.4146
Full GAM w splines	0.5427	0.4573
Optimized GAM w splines	0.5457	0.4543

The above table shows, that the model with the lowest missclassification rate on the test set is the model without splines and with the insignificant variable **cuisine** excluded. Overall it can also be said that the misslassification rate for all the models is above 40% which is almost double the rate for the model with the word occurance. This makes intuitivly sense as the words express the customers fillings and opinions while the business attributes are much more indirectly related.

(c) Likelihood ratio test

Consider a version of model (a) that does not include the `cuisine` predictor variable. Explain briefly how you would test in your model whether `cuisine` is an important predictor of the probability of a good restaurant rating. Perform the test, and explain your conclusions. [15 points]

During the above analysis we already saw two indicators that the variable **cuisine** is not an important variable for predicting the rating of a review. The first was that the value is highly insignificant, the second that excluding the variable from the model actually increases the accuracy of the prediction on the testing set. A third way is to perform a a **likelihood ratio test**. The test can be used to compare the goodness of fit of two models, one of which (the null model) is a special case of the other (the alternative model). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one model than the other.

Likelihood ratio test

```r
anova_tbl <- anova(fit_gam_lin, fit_gam_lin2, test="Chi")
pander(anova_tbl,
       add.significance.stars=TRUE, style='rmarkdown', caption="",
       split.table=Inf, digits=2, justify='center')
```

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
643	860	NA	NA	NA
653	869	-10	-9.8	

Signif. codes: 0 ‘’ ***0.001*** ’’ *0.01* ” 0.05 ‘.’ 0.1 ‘’ 1

The above table shows, that the difference in performance between the two models (i.e., with and without the variable **cuisine**) is not statistically significant.