Lab 1

Thomas Haase

August 25, 2025

Table of contents

Part 1 - Bernie Sanders and Donald Trump tweets

Task 1.1

```
data <- fread(file = './trumpbernie.csv')

data[1:5,20:25]
```

```
     abe   abl abort absolut absurd  abus
   <int> <int> <int>   <int>  <int> <int>
1:     0     0     0       0      0     0
2:     0     0     0       0      0     0
3:     0     0     0       0      0     0
4:     0     0     0       0      0     0
5:     0     0     0       0      0     0
```

```r
nrow(data)
```

```
[1] 1003
```

```r
ncol(data)
```

```
[1] 1496
```

High-dimensionality describes a dataset where the number of variables is large relative to the number of observations. Since the columncount of the data is larger than the number of observations the dataset is highdimensional. A logistic regression would produce a perfect fit, which means that each observation can be completely explained. Since in this case also the "noise" () is part of the data modeling, the model becomes too flexible. This is also called overfitting.

Task 1.2

a)

```r
glm <- glm(trump_tweet ~ .,
           data = data,
           family = "binomial")

coef(glm)[1000:1050]
```

```
      poor      popul    popular    possibl       post    potenti    poverti
 149.95472  349.33975 -771.91771  -30.26993   25.75712 -380.91305   59.38731
     power    practic     prayer     prefer    premium     prepar       pres
        NA         NA         NA         NA         NA         NA         NA
 prescript    present     presid presidenti      press    pressur    pretti
        NA         NA         NA         NA         NA         NA         NA
   prevent   previous      price    primari      prime   prioriti     prison
        NA         NA         NA         NA         NA         NA         NA
    privat   privileg        pro    probabl    problem    process   proclaim
        NA         NA         NA         NA         NA         NA         NA
    produc    product     profit    program   progress    project     promis
        NA         NA         NA         NA         NA         NA         NA
```

```
    proper     propos    protect    protest      proud     provid     public
        NA         NA         NA         NA         NA         NA         NA
    puerto       pull
        NA         NA
```

A lot of coefficients are NA/not defined values. The model only fits the parameters it needs to reach a perfect fit because of the high dimensionality and the rest are set to NA. The model tries to improve the fit until it can explain every observation.

b)

```r
# Extract predictions on training data & observed values
comparison_df <- data.frame(train_predictions=glm$fitted.values,
observed=glm$y)
# Apply prediction threshold
comparison_df$train_predictions<-ifelse(comparison_df$train_predictions>=0.5,
yes = 1,
no = 0)
# Compute accuracy (scale: 0-1, 0=0%, 1=100%)
nrow(comparison_df[comparison_df$train_predictions==comparison_df$observed,]) /
nrow(comparison_df)
```

```
[1] 1
```

As theorized in a) the model reached a perfect fit.

Task 1.3

```r
tc <- trainControl(method = 'cv', number = 3)
glm_cv <- caret::train(as.factor(trump_tweet) ~ .,
                       data = data,
                       method = "glm",
                       family = "binomial",
                       trControl = tc)

glm_cv$results
```

```
  parameter  Accuracy        Kappa AccuracySD    KappaSD
1      none 0.4985164 -0.002957549 0.02472487 0.04944414
```

The displayed accuracy is the mean accuracy of the 3 calculated models. The function takes the best model with the highest singular accuracy as the final one. The accuracy measures the amount of in/correctly predicted predictions, which for the model using crossvalidation is ~50%. Before the model reached a perfect fit, but now we extended the previous try through crossvalidation. The model still overfits, but the accuracy displayed now is the out of sample prediction.

Task 1.4

```
glm_ridge <- cv.glmnet(x = as.matrix(data[,-"trump_tweet"]),
                       y = data$trump_tweet,
                       nfolds = 5,                # Number CV-folds
                       standardize = TRUE,        # Standardize X
                       family = "binomial",       # Outcome binary --> logit/binomial
                       alpha = 0,                 # alpha=0 --> ridge, for Lasso set it to 1
                       type.measure = "class",    # measure performance in terms of accuracy
                       keep = T)

glm_ridge
```

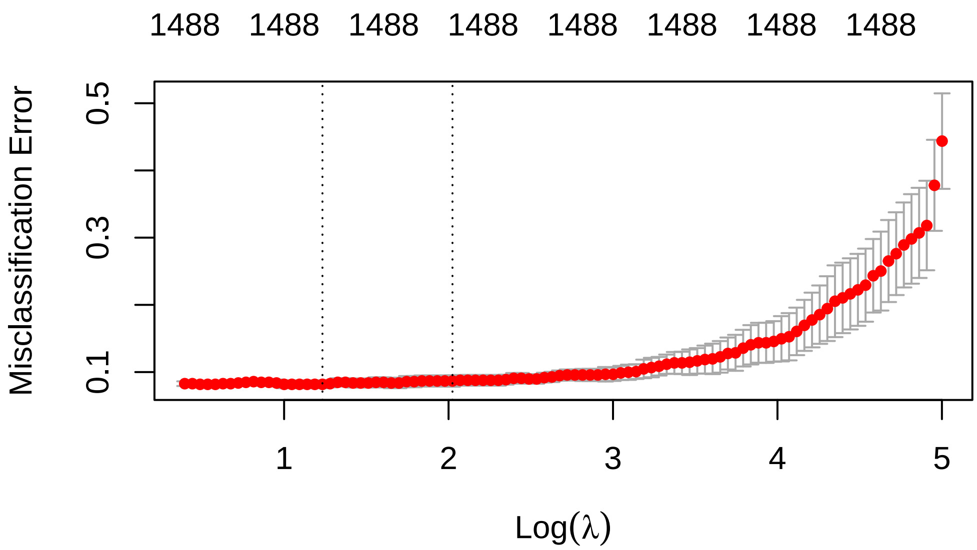
```
Call:  cv.glmnet(x = as.matrix(data[, -"trump_tweet"]), y = data$trump_tweet,      type.measure =

Measure: Misclassification Error

    Lambda Index Measure       SE Nonzero
min  3.432    82 0.08175 0.005693    1488
1se  7.568    65 0.08674 0.008513    1488
```

The accuracy of the resulting model is 0.92 for Lambda 3.43.

The first model (Task 1.2) was overfitting which typically happens with models that are too flexible. The standard Logit-model seems to be too flexibel for our task. To reduce the flexibility ridge regression adds a penaltyterm to punish a model using too many variables as predictors. The resulting model is way more accurate than the logistic regression model from before.

Task 1.5

```
plot(glm_ridge, sign.lambda = 1)
```



The plot shows the misclassification error of the model, which is plotted against the λ. λ is determining how strong the model punishes the addition of new variables.
The plot shows that as λ increases the misclassification error monotonously rises. Large rise is visible from $\text{Log}(\lambda) = 3$ on upward.

A model with a high λ punishes extra variables a lot. Therefore a model with a high λ converges towards the most simple model with one predictor. A model like this is very biased, extremely inflexible and can not account for the complexities of the observed data very well. The plot shows that a high λ is also connected to a big misclassification error.

A model with a low λ does not punishes extra variables at all. A model with a low λ converges towards the most complex model with infinite predictors. A model like this extremely flexible and inherits very low bias. It accounts for every little notion in the data - even the ones we are not interested in. The plot shows that a low λ is also connected to a very low misclassification error.

The best model lies somewhere in between. The lines help identifying the "perfect" model. The most left line marks the model with the lowest misclassification error. The model marked by the line to the right is only a little less predictive, more detailed: It lies one standard deviation away from the model with the best predicting *lambda* and is therefore simpler/has less predictors than the best predicting model but still has a sufficient accuracy.

Task 1.6

```
print(
  data.table(word = rownames(coef(glm_ridge, s = "lambda.min")),
             coef = coef(glm_ridge, s = "lambda.min")[,1])[order(coef,decreasing = T)],
  topn = 15
)
```

```
           word       coef
         <char>      <num>
   1:   atlanta  0.1428543
```

```
   2:         Npme  0.1364964
   3:      patriot  0.1315325
   4:     colorado  0.1293414
   5:     sacrific  0.1243933
   6:       confid  0.1241375
   7:        shape  0.1238952
   8:      liberti  0.1223073
   9:       plenti  0.1208467
  10:       prayer  0.1198828
  11:       extend  0.1189702
  12:       talent  0.1183770
  13:       regard  0.1177688
  14:       spirit  0.1169570
  15:         fool  0.1167845
  ---
1482:     xenophob -0.1180616
1483:     overturn -0.1186338
1484:       caucus -0.1186744
1485:     franklin -0.1187431
1486:           dr -0.1189617
1487:         born -0.1210366
1488:      largest -0.1212463
1489:        forum -0.1215589
1490:         youv -0.1218376
1491:        petit -0.1219183
1492:       vulner -0.1264633
1493:         view -0.1270916
1494:        visit -0.1275855
1495:       volunt -0.1282115
1496: (Intercept) -0.1346252
```

When words with a large positive coefficient occur in a tweet it is more likely that the tweet was published by Trumps twitter account. Words with large negative coefficient are also strong predictors, but through the mechanism that Trump is avoiding those words and Sanders is using them instead. This is backed by “patriot”, “prayer” and american cities/states being part of the positive coefficients. Trump avoids words with stems like “xenophob”, “vulner” and “volunt” which are used by Sanders instead.

Part 2

```r
rm(list = ls())
gc()
```

```
          used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells 2423405 129.5    4730707 252.7  3514559 187.7
Vcells 4235016  32.4   46984017 358.5 58729548 448.1
```

Task 2.1

```r
data <- fread(file = "./Kaggle_Social_Network_Ads.csv")
data$Purchased <- data$Purchased |> factor()
```

Task 2.2

```r
tc <- trainControl(method = 'cv', number = 5)

set.seed(12345)
glm <- caret::train(Purchased ~ Age + Salary + Gender,
                    data = data,
                    method = "glm",
                    family = "binomial",
                    trControl = tc)

glm$results
```

```
  parameter  Accuracy     Kappa AccuracySD   KappaSD
1      none 0.8452715 0.6501309 0.06200742 0.1436548
```

Task 2.3

```r
set.seed(12345)
gam2 <- caret::train(Purchased ~ ns(Age,2) + ns(Salary,2) + Gender,
                     data = data,
                     method = "glm",
                     family = "binomial",
                     trControl = tc)

set.seed(12345)
```

```r
gam3 <- caret::train(Purchased ~ ns(Age,3) + ns(Salary,3) + Gender,
                     data = data,
                     method = "glm",
                     family = "binomial",
                     trControl = tc)

set.seed(12345)
gam4 <- caret::train(Purchased ~ ns(Age,4) + ns(Salary,4) + Gender,
                     data = data,
                     method = "glm",
                     family = "binomial",
                     trControl = tc)

rbind(gam2$results,gam3$results,gam4$results)
```

```
  parameter  Accuracy     Kappa AccuracySD    KappaSD
1      none 0.8974621 0.7732889 0.03905348 0.08999520
2      none 0.9049930 0.7900486 0.03139541 0.07105771
3      none 0.9025246 0.7866560 0.02387829 0.05550475
```

The accuracy of all GAM models is higher than the accuracy of the standard logistic regression. A better accuracy suggests higher flexibility, because the model is representing the data better. A higher flexibility is always associated with lower bias. The logistic regression is therefore more biased because it assumes linearity which makes it less flexible as the GAMs. The logistic regression underfits because of that

I prefer the gam with two degrees of freedom most, because it has the big increase in accuracy compared to the logistic regression just as the other GAMs. When comparing the increase in accuracy between the GAMs with different degrees of freedom, the rise in accuracy is neglegible considering the high flexibility added through another degree of freedom on two variables!

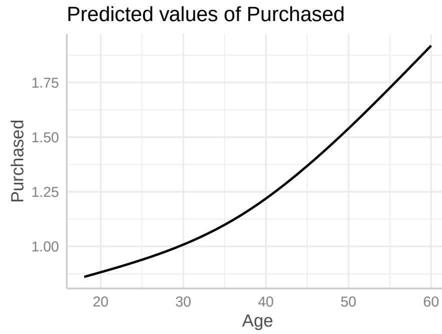
Task 2.4

```r
final_model <- lm(Purchased ~ ns(Age,2) + ns(Salary,2) + Gender,
                  data = data)

final_model_predictions <- ggpredict(final_model)

plot(final_model_predictions)
```

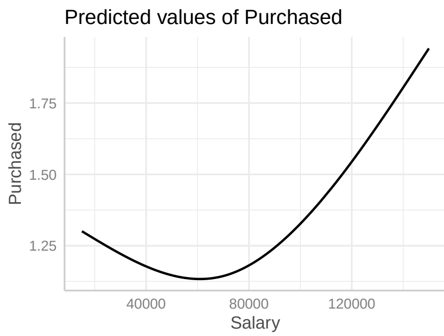
The plot of the "Predicted values of Purchased" vs. Age suggests that the relationship between the Age and the probability of purchasing shows a nonlinear relationship. The older a person gets, the higher gets the steepness of the curve and therefore the rise in purchasing probability.

The plot of the "Predicted values of Purchased" vs. Salary suggests another nonlinear relationship. This relationship is more complex, since the customer's purchase probability follows a U-shaped curve. It starts high at low salaries, drops to its minimum around $60,000-80,000, then rises sharply at higher salary levels.

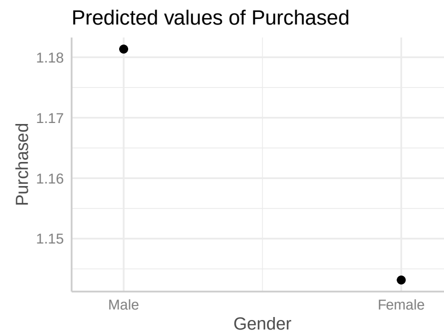
$Age



$Salary



$Gender



Task 2.5

The GAMs addressed the nonlinear nature of the data well. This was the main underlying reason, why the fit could be improved. The ridge- or lasso-regression extends the original model by a penalty term that punishes high amounts of added variables. Since the data is not high dimensional and only consists of 3 possible predictors it would not improve the fit significantly compared to the standard logistic regression.

Part 3

```
rm(list = ls())
gc()
```

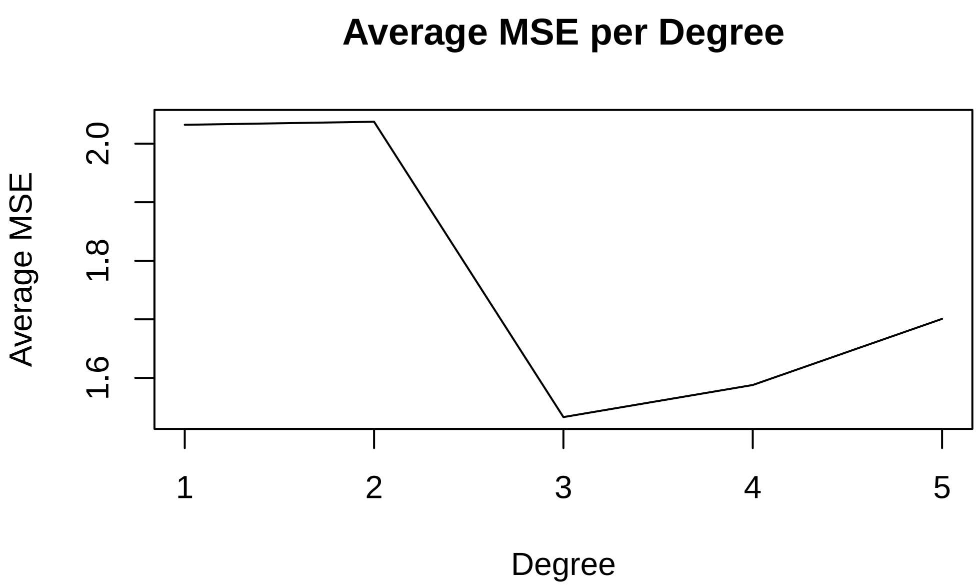
```
          used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells 2659313 142.1    4730707 252.7  4730707 252.7
Vcells 4667392  35.7   37587214 286.8 58729548 448.1
```

```
dt <- readRDS("dt.rds")




cv_model <- function(dt, k = 3, d = 5){
 DATA <- dt
 k <- k
 n <- nrow(DATA)
```

```r
d <- d # polynomialdegree
 set_ids <- cut(seq_len(n), breaks = k, labels = FALSE)

 folds <- setNames(
   lapply(1:k, function(i) DATA[set_ids == i, ]),
   paste0("fold_", 1:k)
 )

 models <- setNames(
   lapply(1:k, function(fold_idx) {
     setNames(
       lapply(1:d, function(deg) lm(Y ~ poly(X, deg), data = folds[[fold_idx]])),
       paste0("deg_", 1:d)
     )
   }),
   paste0("fold_", 1:k)
 )

 predictions <- setNames(
   lapply(1:k, function(fold) {
     other_folds <- setdiff(1:k, fold)

     setNames(
       lapply(1:d, function(deg) {
         setNames(
           lapply(other_folds, function(test_fold) {
             predict(models[[fold]][[deg]], newdata = folds[[test_fold]])
           }),
           paste0("test_fold_", other_folds)
         )
       }),
       paste0("deg_", 1:d)
     )
   }),
   paste0("fold_", 1:k)
 )

 # Calculate MSE using the predictions object
 mse <- lapply(names(predictions), function(fold_name) {
   lapply(names(predictions[[fold_name]]), function(deg_name) {
     sapply(names(predictions[[fold_name]][[deg_name]]), function(test_fold_name) {
       pred <- predictions[[fold_name]][[deg_name]][[test_fold_name]]
       true <- folds[[sub("test_fold_", "fold_", test_fold_name)]]$Y
       mean((pred - true)^2)
     }, USE.NAMES = TRUE)
   })
 })
```

```r
  names(mse) <- names(predictions)
  for(fold in names(mse)) names(mse[[fold]]) <- names(predictions[[fold]])

  avg_mse <- sapply(1:d, function(deg) {
    all_deg_mse <- unlist(lapply(mse, function(fold) fold[[paste0("deg_", deg)]]))
    mean(all_deg_mse)
  })
  names(avg_mse) <- paste0("deg_", 1:d)

  cat("The average MSE per degree are:\n")
  print(as.data.frame(avg_mse))
  cat("\n\n----------\nDegree with minimum MSE: ", names(avg_mse)[which.min(avg_mse)], "\n")
  plot(as.numeric(sub("deg_", "", names(avg_mse))),avg_mse, type = "l",
       main = "Average MSE per Degree",xlab = "Degree",ylab = "Average MSE")
  return(avg_mse)
}

cv_model(dt = dt, k = 2, d = 5)
```

```
The average MSE per degree are:
        avg_mse
deg_1 2.032259
deg_2 2.037449
deg_3 1.533011
deg_4 1.587824
deg_5 1.700770


----------
Degree with minimum MSE:  deg_3
```



```
   deg_1    deg_2    deg_3    deg_4    deg_5
2.032259 2.037449 1.533011 1.587824 1.700770
```

Quiz Wrap up

1

- Predicting grades on a 1-100 scale is a regression problem. Since we are interested in understanding this is a problem of inference.
- Understanding the relation between the centrality of a twitter user (proxied by the number of followers) and the number of retweets sounds like a regression problem. Since we are interested in understanding this is an inference problem.
- Predicting wether a student will drop out or not is a classification problem. Since we are interested in prediction this is a predicting problem.

2

- a:i
- b:ii
- c:iii