

Lab 1

Thomas Haase

August 22, 2025

Table of contents

1	Part 1 - Bernie Sanders and Donald Trump tweets	1
1.1	Task 1.1	1
1.2	Task 1.2	2
1.2.1	a)	2
1.2.2	b)	3
1.3	Task 1.3	3
1.4	Task 1.4	4
1.5	Task 1.5	5
1.6	Task 1.6	6
2	Part 2	7
2.1	Task 2.1	7
2.2	Task 2.2	7
2.3	Task 2.4	8
2.4	Task 2.5	9
3	Part 3	9

1 Part 1 - Bernie Sanders and Donald Trump tweets

1.1 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

```
nrow(data)
```

```
[1] 1003
```

```
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.

1.2 Task 1.2

1.2.1 a)

```
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.

1.2.2 b)

```
# 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.

1.3 Task 1.3

```
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 does not reach a perfect fit anymore, since it is not trained on the full data and optimized on out of sample predictions. Sadly the resulting model is very bad and still only as good as a coinflip in classifying the data. The model in task 2 was definitely overfitting while this one is underfitting a lot.

1.4 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. By using crossvalidation it becomes possible to create a model which is optimized in regards of out-of-sample prediction.

The first model (Task 1.2) was overfitting which typically happens with models that are too flexible. Often models with higher flexibility have higher variance. Variance here means small changes in training data lead to large changes in the model. Models with higher flexibility therefore are more sensitive to different training sets - when we changed the estimation procedure and implemented crossvalidation the accuracy dropped to 50%. This shows that just taking a sample/fold to estimate the model affected it a lot already. The standard Logit-model seems to be too flexible for our task. To reduce the flexibility ridge regression adds a penalty term to punish a model using too many variables as predictors. The resulting model is way more accurate than the logistic regression model from before.

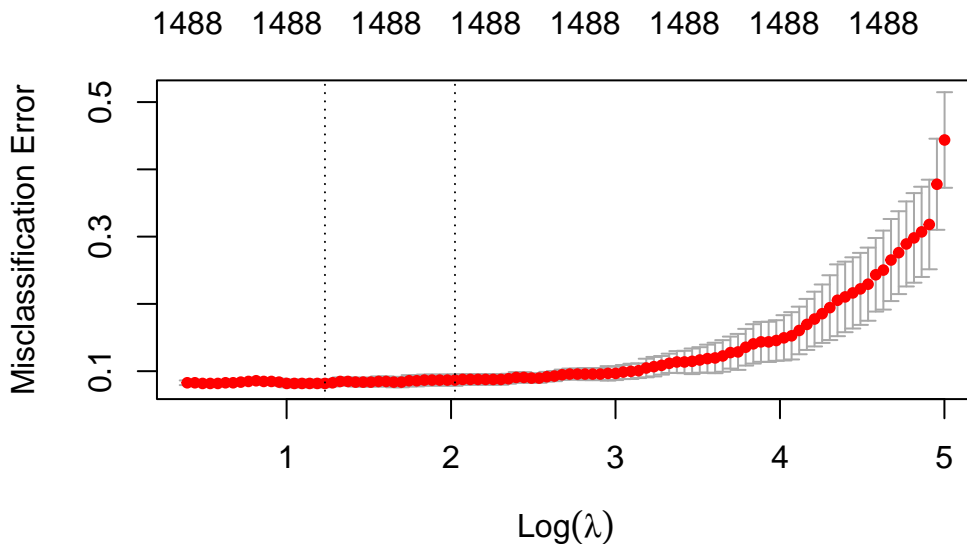
```
tapply(ifelse(glm_ridge$fit.preval[, which(glm_ridge$lambda==glm_ridge$lambda.min)]>0.5, 1, 0) == data,
  sd())
```

```
[1] 0.02708674
```

As we can see the standard deviation of the accuracy across different folds is much higher for the ridge regression compared to the logistic regression. This is a direct indicator for a lower flexibility.

1.5 Task 1.5

```
plot(glm_ride, 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 punish 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.

1.6 Task 1.6

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

	word <char>	coef <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. 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”.

2 Part 2

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

	used	(Mb)	gc trigger	(Mb)	max used	(Mb)
Ncells	2423458	129.5	4730251	252.7	3514342	187.7
Vcells	4234375	32.4	46982700	358.5	58728175	448.1

2.1 Task 2.1

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

2.2 Task 2.2

```
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

```
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)
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 and less flexible as the GAMs. It is probably underfitting the data.

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 negligible considering the high flexibility added through another degree of freedom on two variables!

2.3 Task 2.4

```

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

final_model_predictions <- ggpredict(final_model)

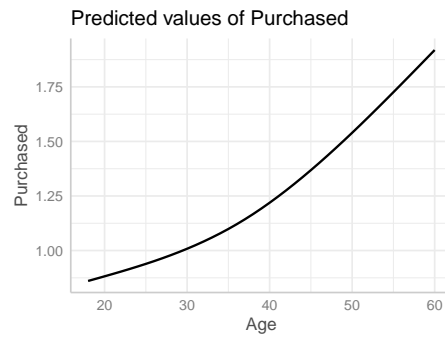
plot(final_model_predictions)

```

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



2.4 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.

3 Part 3