Exam 1

Sam

2022-10-28

Question 1

1(a)

States:

State 1 = C. = Cold
State 2 = V.C. = Very Cold
State 3 = E.C. = Extremely Cold

1(b)

The transition matrix is:

$$\begin{array}{cccc} - & C. & V.C. & E.C. \\ C. & 0.7 & 0.2 & 0.1 \\ V.C. & 0.4 & 0.3 & 0.3 \\ E.C. & 0.2 & 0.4 & 0.4 \end{array}$$

1(c)

The probabilities of each state after two days are as follows:

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{bmatrix} \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{bmatrix} =$$

$$-------------------------$$

$$\begin{bmatrix} 0.4 & 0.3 & 0.3 \end{bmatrix} \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{bmatrix} = \begin{bmatrix} 0.46 & 0.29 & 0.25 \end{bmatrix}$$

1(d)

The code for calculating the state probability can be seen below:

```
c <- c(0.7, 0.2, 0.1)
vc <- c(0.4, 0.3, 0.3)
ec <- c(0.2, 0.4, 0.4)


m <- matrix(c(c, vc, ec), ncol = 3, byrow = TRUE)
result = matrix(c(0, 1, 0), ncol = 3, byrow = TRUE)
result = result %*% m %*% m
print(result)
```

```
##      [,1] [,2] [,3]
## [1,] 0.46 0.29 0.25
```

Question 2

2(a)

SVM optimization problem:

$$min(a,b) a^T \begin{bmatrix} 2 & 231 & 6 \end{bmatrix} + b^3 + 1 * \begin{bmatrix} 5 & 52 & 12 \end{bmatrix}$$
$$Subject\ to\ 3 * a \geq 0$$

2(b)

Import data

```
d1 = c(0, 2, 231, 6)
d2 = c(1, 5, 52, 12)
d3 = c(0, 1, 129, 0)
data_lst = matrix(c(d1, d2, d3), ncol = 4, byrow = TRUE)

df_svm <- data.frame(matrix(ncol = 4, nrow = 0))
colnames(df_svm) <- c('y', 'f1', 'f2', 'f3')
count = 1
for (i in 1:3) {
  for (j in 1:4){
      df_svm[i, j] = data_lst[i, j]
  }
}
```

Calculate hyperplane

```
library(e1071)

svm = svm(factor(y) ~ f1 + f2 + f3, kernel = "linear", data = df_svm)

beta = t(svm$coefs)%*%svm$SV
beta0 = svm$rho
```

The equation of the hyperplane is y = -0.447966 * f_1 + 0.5012494 * f_2 + -0.3546845 * f_3 + -0.3333333

2(c)

Calculating the prediction here:

```
x_4 = c(2, 230, 0)

f1_m = mean(df_svm$f1)
f1_s = sd(df_svm$f1)
f2_m = mean(df_svm$f2)
f2_s = sd(df_svm$f2)
f3_m = mean(df_svm$f3)
f3_s = sd(df_svm$f3)

x_4[1] = (x_4[1]-f1_m)/f1_s
x_4[2] = (x_4[2]-f2_m)/f2_s
x_4[3] = (x_4[3]-f3_m)/f3_s

pred = beta[1]*x_4[1] + beta[2]*x_4[2] + beta[3]*x_4[3] + beta0

print(pred)
```

```
## [1] 0.6821206
```

We can see that the prediction is above 0. Since the prediction is above 0, we can predict 0 a.k.a. no churn.

2(d)

```
library(gstat)
pred_values = list(predict(svm, df_svm[2:4]))
real_values = list(df_svm[1])
print(pred_values)
```

```
## [[1]]
## 1 2 3
## 0 1 0
## Levels: 0 1
```

```
print(real_values)
```

```
## [[1]]
##   y
## 1 0
## 2 1
## 3 0
```

Since the ground truth is the same as the values predicted when predicting on the training data, we can say the data is linearly separable. The equation is as follows:

$$arg\ max(w,b)\ \frac{1}{||w||}\ s.t.\ y^i(w^\top x_i + b) - 1 \geq 0, \forall\ i$$

Question 3

3(a)

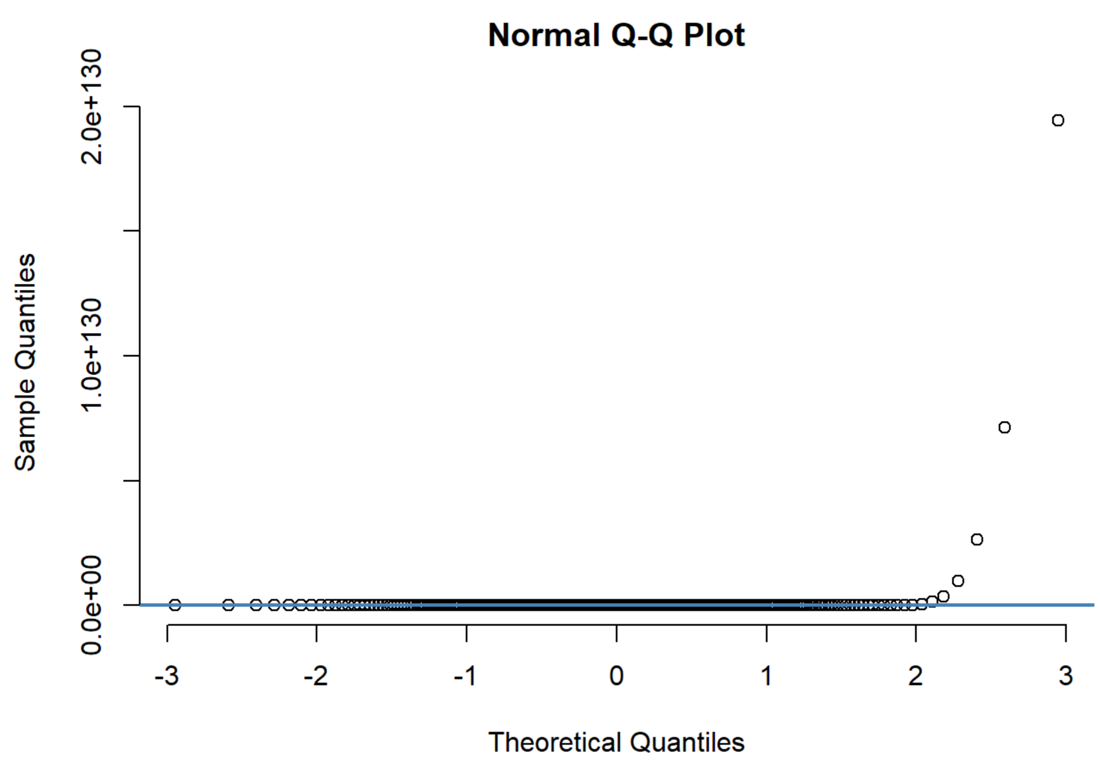
We need to normalize the data.

We can see from the qq plots below that the data is not normal. Thus, this is why we are normalizing instead of standardizing.

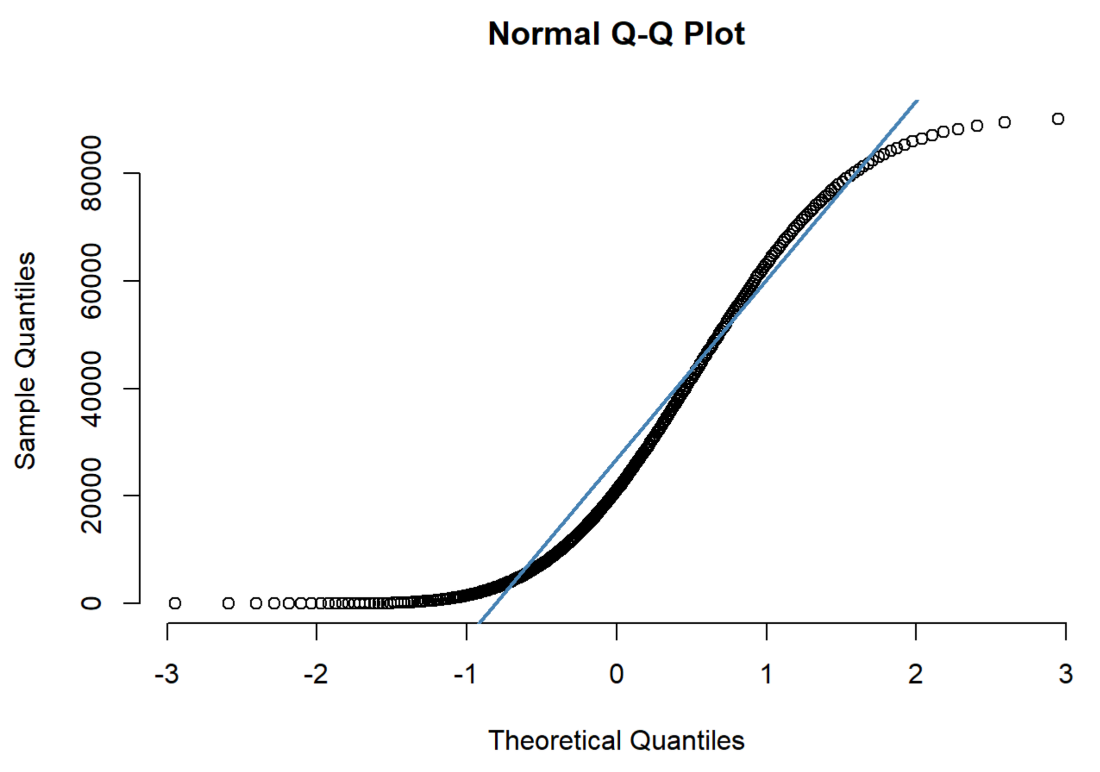
```
ground_truth = c()
i_values = c()
e_values = c()
for (i in -10:300) {
  i_values = append(i_values, i^2)
  e_values = append(e_values, exp(i))
  if (i %% 2 == 0) {
  ground_truth = append(ground_truth, 0)
  } else {
  ground_truth = append(ground_truth, 1)
  }
}

df_q3 <- data.frame(ground_truth, e_values, i_values)
colnames(df_q3) <- c('ground_truth', 'e_values', 'i_values')
```

```
qqnorm(df_q3$e_values, pch = 1, frame = FALSE)
qqline(df_q3$e_values, col = "steelblue", lwd = 2)
```




```
qqnorm(df_q3$i_values, pch = 1, frame = FALSE)
qqline(df_q3$i_values, col = "steelblue", lwd = 2)
```



To normalize the data we need to apply the formula below to each column:

$$\frac{(x_i - min(x))}{(max(x) - min(x))} * 1 = Normalization$$

3(b)

```
i_values_norm = (i_values-min(i_values))/(max(i_values)-min(i_values))
e_values_norm = (e_values-min(e_values))/(max(e_values)-min(e_values))

df_q3_norm <- data.frame(ground_truth,
                         i_values_norm,
                         e_values_norm)
colnames(df_q3_norm) <- c('ground_truth',
                          'e_values_norm',
                          'i_values_norm')
head(df_q3_norm)
```

```
##   ground_truth e_values_norm i_values_norm
## 1            0  0.0011111111  0.000000e+00
## 2            1  0.0009000000 4.016105e-135
## 3            0  0.0007111111 1.493301e-134
## 4            1  0.0005444444 4.460823e-134
## 5            0  0.0004000000 1.252738e-133
## 6            1  0.0002777778 3.445457e-133
```

Question 4

```
truth = c(0, 0, 0, 0, 1, 0, 1)
pred = c(0, 0, 0, 1, 0, 0, 1)
tab = table(truth, pred)
print(tab)
```

```
##      pred
## truth 0 1
##     0 4 1
##     1 1 1
```

```
TN = tab[1, 1]
TP = tab[2, 2]
FN = tab[2, 1]
FP = tab[1, 2]
```

4(a)

The accuracy of the model is:

```
acc = (TP+TN) / (TP+TN+FP+FN)
print(acc)
```

```
## [1] 0.7142857
```

4(b)

```
pre = TP / (TP+FP)
print(pre)
```

```
## [1] 0.5
```

4(c)

```
recall = (TP) / (TP+FN)
print(recall)
```

```
## [1] 0.5
```

4(d)

```
f1 = (TP) / (TP+0.5*(FP+FN))
print(f1)
```

```
## [1] 0.5
```

4(e)

We should use F1 score here. Since we have unbalanced data (30 70 split) accuracy will not be a good measure of the model performance. If we just guess 0 for every value we will already at the accuracy score calculated for this problem, 0.7142857.