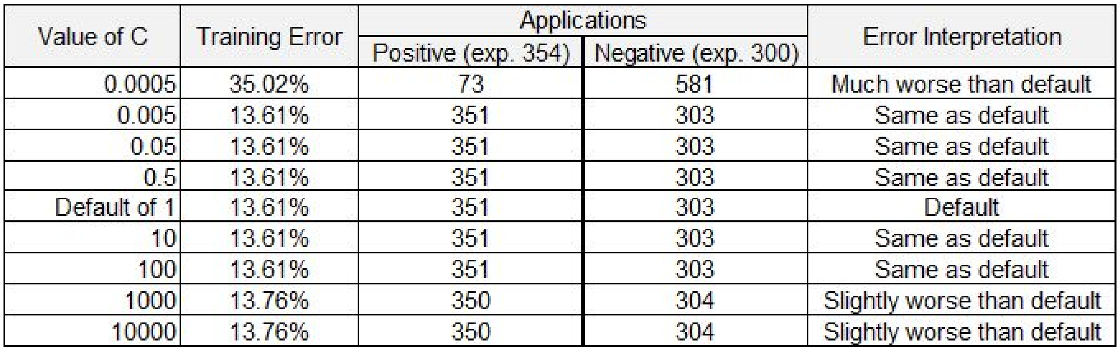
Question 1:

A situation that would benefit from a classification model is, “Should I mow my lawn or not?” Predictors for this model would include the height of the grass, the probability of rain, time of day, my energy level, and air temperature.

Question 2.1:

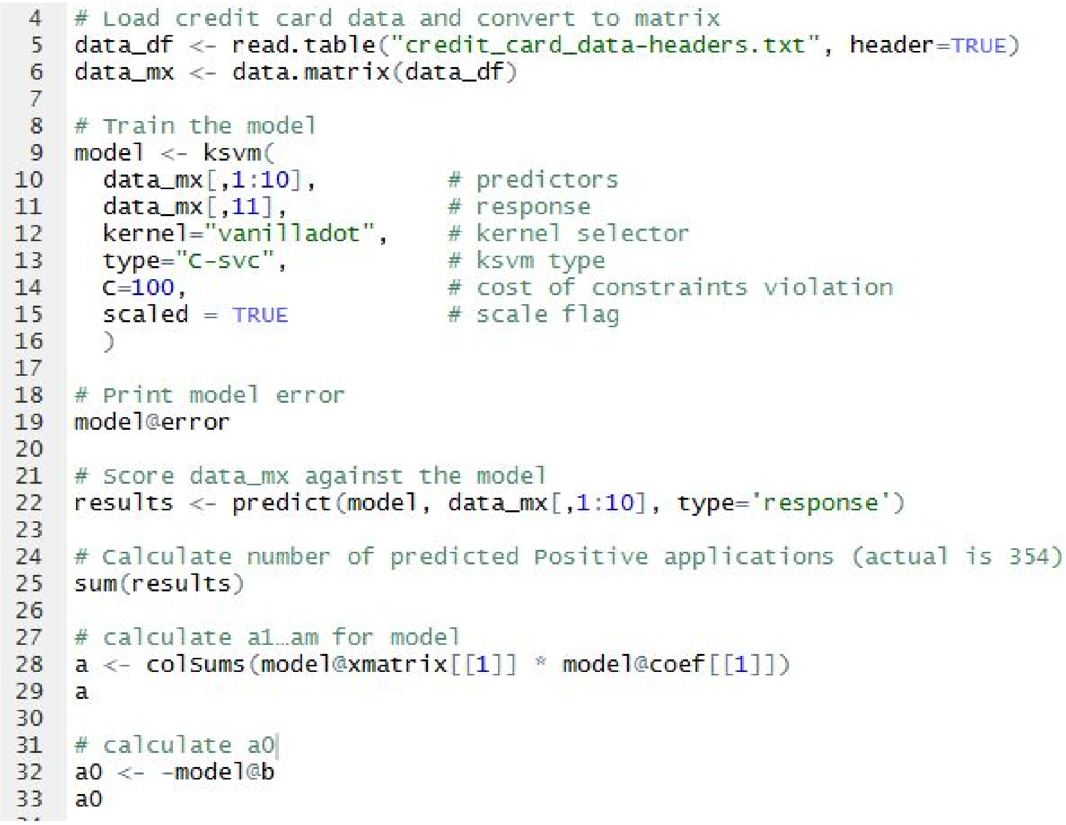


Value of C	Training Error	Applications		Error Interpretation
		Positive (exp. 354)	Negative (exp. 300)	
0.0005	35.02%	73	581	Much worse than default
0.005	13.61%	351	303	Same as default
0.05	13.61%	351	303	Same as default
0.5	13.61%	351	303	Same as default
Default of 1	13.61%	351	303	Default
10	13.61%	351	303	Same as default
100	13.61%	351	303	Same as default
1000	13.76%	350	304	Slightly worse than default
10000	13.76%	350	304	Slightly worse than default

Model ←
-0.0010065348*A1 -0.0011729048*A2 -0.0016261967*A3 + 0.0030064203*A8 +
1.0049405641*A9 -0.0028259432*A10 + 0.0002600295*A11 -0.0005349551*A12
-0.0012283758*A14 + 0.1063633995*A15 + 0.08158492

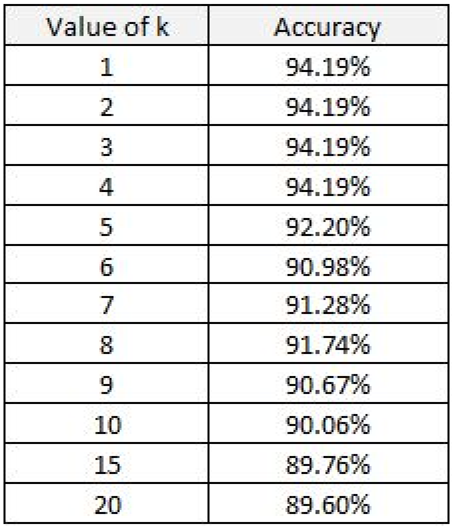
Given that we're analyzing whether to extend credit to a person; I assumed that the potential loss of someone defaulting was greater than the marginal revenue of extending them credit. With this assumption, I think that since the training error is the same for values of C between 0.005 and 100, we should place more importance on not extending credit to a potential defaulter. As a result, I think that C = 100 is the optimal choice.

R-script is also attached and named question-2.1.R



```r
# Load credit card data and convert to matrix
data_df <- read.table("credit_card_data-headers.txt", header=TRUE)
data_mx <- data.matrix(data_df)

# Train the model
model <- ksvm(
  data_mx[,1:10],         # predictors
  data_mx[,11],           # response
  kernel="vanilladot",    # kernel selector
  type="C-svc",           # ksvm type
  C=100,                  # cost of constraints violation
  scaled = TRUE           # scale flag
  )

# Print model error
model@error

# Score data_mx against the model
results <- predict(model, data_mx[,1:10], type='response')

# Calculate number of predicted Positive applications (actual is 354)
sum(results)

# calculate a1…am for model
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a

# calculate a0
a0 <- -model@b
a0
```

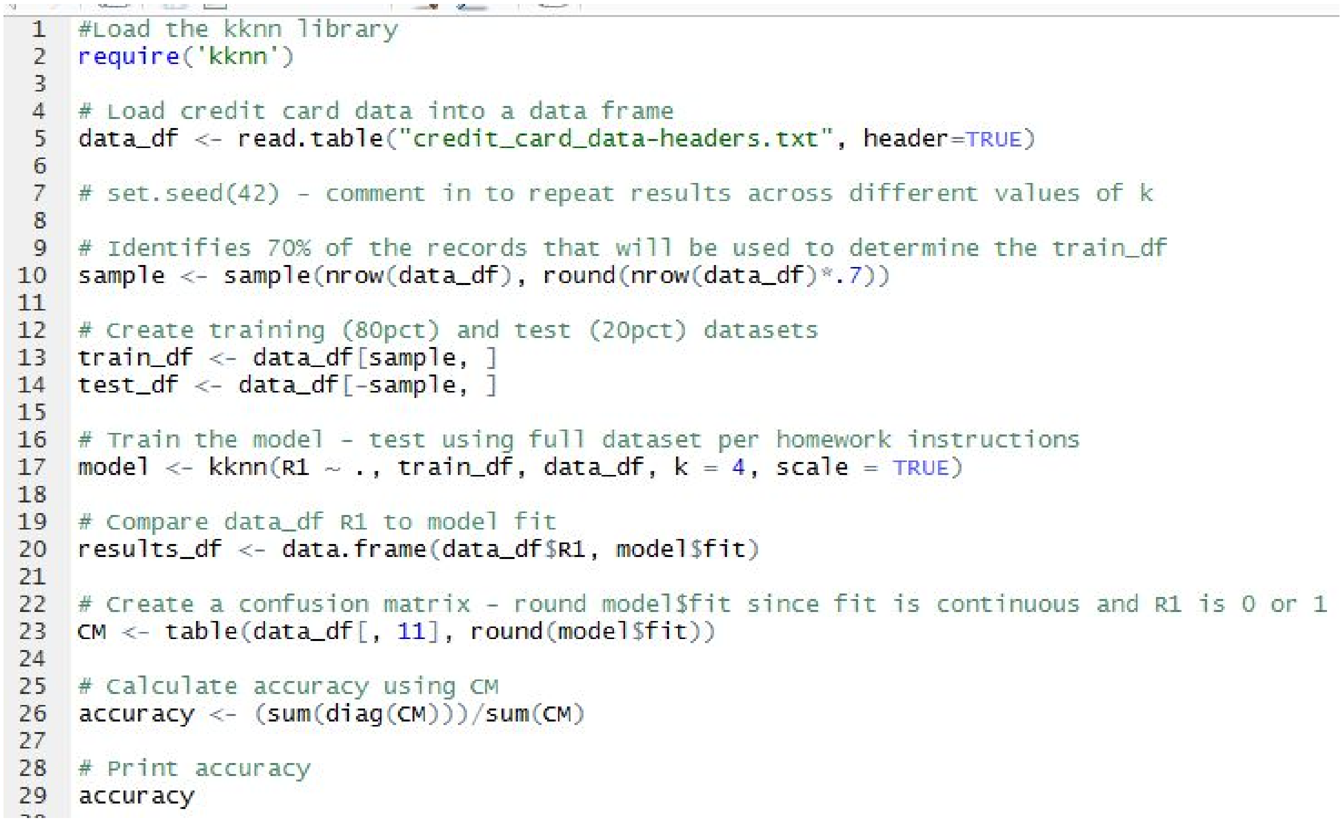
Question 2.2



Value of k	Accuracy
1	94.19%
2	94.19%
3	94.19%
4	94.19%
5	92.20%
6	90.98%
7	91.28%
8	91.74%
9	90.67%
10	90.06%
15	89.76%
20	89.60%

For k values between one and four, the model's accuracy held constant. Increases in the value of k past four resulted in accuracy losses. As a result, I believe that four would be an optimal choice for k. This k value choice is due to the belief that when applying the model to a new data point, comparing it to four neighbors instead of one, two, or three, the likelihood of misclassifying the new data point will be lessened.

R-script is also attached and named question-2.2.R



```r
#Load the kknn library
require('kknn')

# Load credit card data into a data frame
data_df <- read.table("credit_card_data-headers.txt", header=TRUE)

# set.seed(42) - comment in to repeat results across different values of k

# Identifies 70% of the records that will be used to determine the train_df
sample <- sample(nrow(data_df), round(nrow(data_df)*.7))

# Create training (80pct) and test (20pct) datasets
train_df <- data_df[sample, ]
test_df <- data_df[-sample, ]

# Train the model - test using full dataset per homework instructions
model <- kknn(R1 ~ ., train_df, data_df, k = 4, scale = TRUE)

# Compare data_df R1 to model fit
results_df <- data.frame(data_df$R1, model$fit)

# Create a confusion matrix - round model$fit since fit is continuous and R1 is 0 or 1
CM <- table(data_df[, 11], round(model$fit))

# Calculate accuracy using CM
accuracy <- (sum(diag(CM)))/sum(CM)

# Print accuracy
accuracy
```