Question 2.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a classification model would be appropriate. List some (up to 5) predictors that you might use.

Answer:
A person who goes to the hospital for treatment due to a certain disease, whether he will be cured or not will depend on multiple dimensions of the predictors:

1. The level of professional title of the doctor
2. The availability of medical equipment in the hospital
3. The patient's attitude
4. The patient's cooperation with the treatment
5. The basic condition of the patient's health condition

Question 2.2

The files credit_card_data.txt (without headers) and credit_card_data-headers.txt (with headers) contain a dataset with 654 data points, 6 continuous and 4 binary predictor variables. It has anonymized credit card applications with a binary response variable (last column) indicating if the application was positive or negative. The dataset is the "Credit Approval Data Set" from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Credit+Approval) without the categorical variables and without data points that have missing values.

1. Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data. Show the equation of your classifier, and how well it classifies the data points in the full data set. (Don't worry about test/validation data yet; we'll cover that topic soon.)

   Notes on ksvm
   - You can use scaled=TRUE to get ksvm to scale the data as part of calculating a classifier.
   - The term λ we used in the SVM lesson to trade off the two components of correctness and margin is called C in ksvm. One of the challenges of this homework is to find a value of C that works well; for many values of C, almost all predictions will be "yes" or almost all predictions will be "no".
   - ksvm does not directly return the coefficients a_0 and $a_1 \ldots a_m$. Instead, you need to do the last step of the calculation yourself. Here's an example of the steps to take (assuming your data is stored in a matrix called data):[1]

```
# call ksvm.  Vanilladot is a simple linear kernel.
model <- ksvm(data[,1:10],data[,11],type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
# calculate a1…am
```

[1] I know I said I wouldn't give you exact R code to copy, because I want you to learn for yourself. In general, that's definitely true – but in this case, because it's your first R assignment and because the ksvm function leaves you in the middle of a mathematical calculation that we haven't gotten into in this course, I'm giving you the code.

```
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
# calculate a0

a0 <- –model@b
a0
# see what the model predicts

pred <- predict(model,data[,1:10])
pred
# see what fraction of the model's predictions match the actual classification

sum(pred == data[,11]) / nrow(data)
```

Hint: You might want to view the predictions your model makes; if C is too large or too small, they'll almost all be the same (all zero or all one) and the predictive value of the model will be poor. Even finding the right order of magnitude for C might take a little trial-and-error.

Note: If you get the error "Error in vanilladot(length = 4, lambda = 0.5) : unused arguments (length = 4, lambda = 0.5)", it means you need to convert data into matrix format:

```
model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)
```

Answer:

The aim of this task is to find a suitable classifier for the credit card dataset and to evaluate the model's performance on the whole dataset.

```
# Set working directory
 setwd("D:/A_R/ISYE 6501")
 # Read the data
 data <- read.table("D:/A_R/ISYE 6501/R_HW1/data 2.2/credit_card_data-headers.txt",
sep = "\t", header = TRUE)
 # Load necessary libraries
 library(kernlab)
 library(dplyr)
 library(tidyverse)
 library(caret)
 library(kknn)
 library(modelr)
 library(ggplot2)
 # Prepare features and target
 features <- as.matrix(data[, 1:10])
 target <- as.factor(data[, 11])
 # Define the range of C values
 m <- 100  # Starting value of C
 n <- 500  # Ending value of C
 step <- 10 # Step size for C
 # Initialize a data frame to store the results
 results <- data.frame(C_value = numeric(), Accuracy = numeric())
```

```
 # Loop over the range of C values
 for (C_value in seq(m, n, by = step)) {
   # Train the model with the current C value
   model <- ksvm(features, target, type = "C-svc", kernel = "vanilladot", C = C_value,
scaled = TRUE)

   # Calculate coefficients a1...am
   a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
   a0 <- -model@b

   # Make predictions
   pred <- predict(model, features)

   # Calculate the accuracy
   accuracy <- sum(pred == target) / nrow(data)

   # Store the results
   results <- rbind(results, data.frame(C_value = C_value, Accuracy = accuracy))
```

Methodology of the code:

We use the Support Vector Machine (SVM) function ksvm from the kernlab package and try different values of C for the classification task on the credit card dataset. First, to ensure smooth training of the model, we extracted the 11th column of data as the dependent variable *target* and used the first 10 columns of data as the independent variable *features*.

Second, we decided to try different ranges of C-values (minimum value of 100, maximum value of 500, and step size of 10).

Third, we used the ksvm function to train a linear kernel (vanilladot) support vector machine model with a binary classification task specified using C-svc.

Fourth, we used a linear combination of support vectors and their coefficients to compute the model coefficients a. a0 was used to extract the bias term (taking a negative value) to get the model's intercept.

Finally, we use the trained model to make predictions on the entire dataset and match it with the actual results to evaluate the accuracy.

Answer Discussion:

During model training, we optimized the model by using the C-svc category in SVM and continuously adjusted the regularization parameter C. We performed round-robin tests for different values of C and recorded the accuracy of the model each time (see Fig.1).

#	C	Accuracy		#	C	Accuracy
1	100	0.863914		21	300	0.862385
2	110	0.863914		22	310	0.862385
3	120	0.863914		23	320	0.863914
4	130	0.863914		24	330	0.863914
5	140	0.863914		25	340	0.862385
6	150	0.863914		26	350	0.863914
7	160	0.863914		27	360	0.863914
8	170	0.863914		28	370	0.863914
9	180	0.863914		29	380	0.862385
10	190	0.863914		30	390	0.862385
11	200	0.863914		31	400	0.862385
12	210	0.863914		32	410	0.863914
13	220	0.863914		33	420	0.862385
14	230	0.863914		34	430	0.862385
15	240	0.862385		35	440	0.863914
16	250	0.863914		36	450	0.862385
17	260	0.862385		37	460	0.862385
18	270	0.863914		38	470	0.863914
19	280	0.863914		39	480	0.863914
20	290	0.863914		40	490	0.863914
				41	500	0.863914

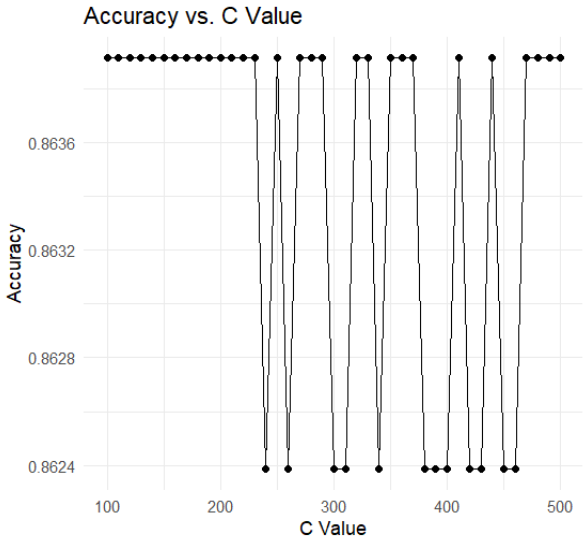

Figure 1. The value of C and corresponding accuracy rate

The results show that the maximum value of the model's accuracy is 0.8639144 and the minimum value is 0.8623853 for different values of C. **Therefore, the optimal C value(C=100) corresponds to a model accuracy of 0.8639144 during our initial evaluation of the model's performance.**

The function is:
Y=(-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+0.0815849217

Although we obtained preliminary optimal results by simple loop attempts, the performance of the model may be unstable due to the size and complexity of the dataset. Therefore, as a next step, we propose to use

cross-validation to further optimize the model to obtain more robust results and avoid overfitting or underfitting problems.

Question 2.3

2.2.3 Using the k-nearest-neighbors classification function kknn contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don't forget to scale the data (scale=TRUE in kknn).

Notes on kknn

- You need to be a little careful. If you give it the whole data set to find the closest points to i, it'll use i itself (which is in the data set) as one of the nearest neighbors. A helpful feature of R is the index –i, which means "all indices except i". For example, data[-i,] is all the data except for the ith data point. For our data file where the first 10 columns are predictors and the 11th column is the response, data[-i,11] is the response for all but the ith data point, and data[-i,1:10] are the predictors for all but the ith data point.
  (There are other, easier ways to get around this problem, but I want you to get practice doing some basic data manipulation and extraction, and maybe some looping too.)
- **Note** that kknn will read the responses as continuous, and return the fraction of the k closest responses that are 1 (rather than the most common response, 1 or 0).

Answer:

The aim of the task is to find a suitable k value for the credit card dataset based on the k-nearest-neighbors classification function.

```
setwd("D:/A_R/ISYE 6501")

data <- read.table("D:/A_R/ISYE 6501/R_HW1/data 2.2/credit_card_data-headers.txt",
sep = "\t", header = TRUE)

library(kknn)
library(dplyr)
library(tidyverse)
library(caret)
library(kernlab)
library(modelr)

features <- as.matrix(data[, 1:10])
target <- as.factor(data[, 11])

k_values <- seq(1, 300, by = 10)
```

```
results <- data.frame(k = integer(), Accuracy = numeric())

for (k in k_values) {

  model <- kknn(formula = target ~ ., train = data, test = data, k = k, scale = TRUE)

  pred <- fitted(model)

  accuracy <- sum(pred == data[, 11]) / nrow(data)

  results <- rbind(results, data.frame(k = k, Accuracy = accuracy))
}

print(results)

best_k <- results[which.max(results$Accuracy), ]
print(paste("Best k value:", best_k$k))
print(paste("Best accuracy:", best_k$Accuracy))

library(ggplot2)
ggplot(results, aes(x = k, y = Accuracy)) +
  geom_line() +
  geom_point() +
  labs(title = "Accuracy vs. k Value", x = "k Value", y = "Accuracy") +
  theme_minimal()
```

Methodology of the code

We use the K-Nearest-Neighbors(KNN) function from the the kknn package and try difference values of K for the classification task on the credit card dataset.

First we prepared the features and target variable, extracted the 11th column from the ‘data’ and converts it into a factor, then used the first 10 columns from the data frame and converts them into a matrix.

Second, we defined the range of k values (minimum value of 2, maximum value of 20, and step size of 1).

Third, to restore the results of the model of each k value, we initialized an empty data frame with columns ‘k’ and ‘accuracy’, then we began to loop over k values.

Fourth, we iterated over each value of k we define, then we used the function to train a k-nearest neighbors model with the k value and extracted the prediction results.

Finally, we calculated the accuracy with the 11th column and match it with prediction results. Then we can find the optimal k value.

	k	Accuracy			k	Accuracy
1	2	1		16	152	0.96789
2	12	0.989297		17	162	0.96789
3	22	0.986239		18	172	0.96789
4	32	0.977064		19	182	0.96789
5	42	0.972477		20	192	0.96789
6	52	0.96789		21	202	0.96789
7	62	0.96789		22	212	0.96789
8	72	0.96789		23	222	0.96789
9	82	0.96789		24	232	0.96789
10	92	0.96789		25	242	0.96789
11	102	0.966361		26	252	0.96789
12	112	0.966361		27	262	0.96789
13	122	0.966361		28	272	0.966361
14	132	0.966361		29	282	0.966361
15	142	0.96789		30	292	0.966361

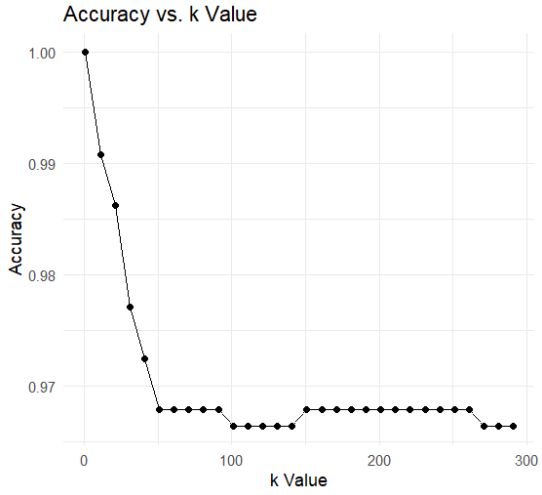

Figure 2. The value of K and corresponding accuracy rate（k=2-292）

	k	Accuracy			k	Accuracy
1	2	1		15	16	0.987768
2	3	1		16	17	0.987768
3	4	1		17	18	0.987768
4	5	0.995413		18	19	0.986239
5	6	0.993884		19	20	0.986239
6	7	0.990826		20	21	0.986239
7	8	0.990826		21	22	0.986239
8	9	0.990826		22	23	0.986239
9	10	0.990826		23	24	0.98471
10	11	0.990826		24	25	0.981651
11	12	0.989297		25	26	0.978593
12	13	0.989297		26	27	0.978593
13	14	0.987768		27	28	0.978593
14	15	0.987768		28	29	0.977064
				29	30	0.977064

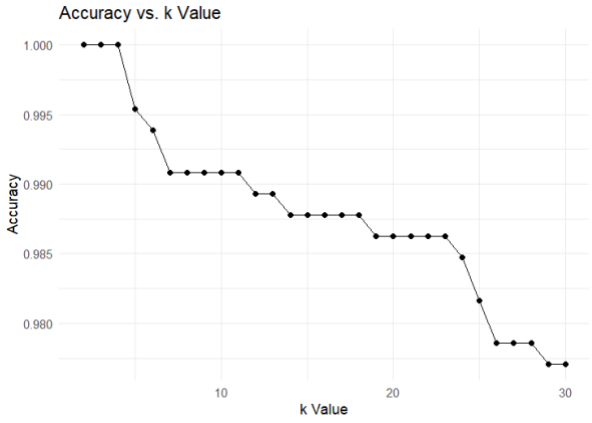

Figure 3. The value of K and corresponding accuracy rate（k=2-30）

"Best k value: 2"
"Best accuracy: 1"
We begin with k=2 and try a space of 10 to observe the trend between 2 to 292. There is a noticeable trend that the accuracy started to decrease from 12. Thus, we chose to use a space of 1 to examine between k=2 and k=29, finding that K=2, 3, and 4 are the highest in accuracy since they all 100% fit the data. However, since 100% accuracy is too idealized for a model to fit all situations and outliers, k=2 is usually too tiny for KNN. Thus, **we pick k=5 for our conclusion for the best outcome of accuracy (0.995413) and tolerance to new data**.