ISyE 6501 HOMEWORK 1

Group Members:
Jingyu, Li | alanli@gatech.edu | GTID: 903520148
Jinyu, Huang | jhuang472@gatech.edu | GTID: 903522245
Chengqi, Huang | chengqihuang@gatech.edu | GTID: 903534690
Yuefan, Hu | yuefanhu@gatech.edu | GTID:903543027

Question 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:

For Amazon, when the 6-month free trial of Prime Membership for the college student account goes to the end, whether or not student users will continue to pay for the Prime Membership within one month is a business issue, for which the classification model would be appropriate. By classifying the "continue using" user group and "stop using" user group, company can allocate different resources and apply different operation strategy to maintain each user group on the platform.

The dependent variable of the model is whether the student users subscribe Prime Membership within one month after the 6-month free trial ends (Yes = 1, No = -1). And the features we propose are in *Table a* as following:

Table a Features in the Classification Model

Name	Definition	Data Type	Reason
Total Consumption	Total amount of money an account used within the 6-month free trial	Ratio	Total consumption reflects the users' activity on the platform and the degree of their online shopping need. If the need for online shopping is stronger, users are more likely to pay for Prime. On the other hand, it also indicates the general economic status of the users. With higher affordability, they have higher potential to pay for Prime.
Average Online Time	Average active time per week of an account within the 6-month free trial	Ratio	Online time is another indicator of the degree of activity for the users. The longer they spend on the platform, the higher probability they may need extra services.

Prime Exclusive Deals Consumption	Total amount of money an account used within the 6-month free trial to buy Prime Exclusive Deals	Ratio	This feature indicates the need for Prime exclusive service of the users. If the need is strong and users get used to using Prime, users may continue to use after the free trial ends.
Age Group	under 23 years old; 23 or upper	Ordinal	Generally, college students in different age group have different affordability. Students under 23 years old normally are undergraduates, who are not that economic independent. Students with 23 years old or upper normally are graduate students, who have higher living budget.

Question 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.**

2.1 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.)

Answer:

We tried different C values in the model setting in R as follows:

model <-
ksvm(data[,1:10],data[,11],type='C-svc',kernel='vanilladot',C=c_value,scaled=TRUE)

And the results of what fraction of the model's predictions match the actual classification are in *Table b*. According to the results we can see, if C is too large or too small in the model, the predictive value of the model will be poor (e.g. Matching percentages are just around 55% when we set c_value as 0.00001 or 10,000,000). On the other hand, if C is in the proper range (e.g. 0.01 to 100,000), the performance of matching percentage is better and all are around 86%. Besides, because we use the training data to evaluate the predictive performance of the model, we can see that the matching percentages vary a little when C is changed from 0.01 to 100,000.

Table b Percentage of Matching Points

c_value	0.00001	0.0001	0.001	0.01	0.1	1	100
matching_percentage(%)	54.74	54.74	83.79	86.39	86.39	86.39	86.39
C_value	500	1000	5000	10000	100000	1000000	10000000
matching_percentage(%)	86.39	86.24	86.24	86.24	86.39	62.54	54.59

In total, we assume that we choose the model when C=100, and the classifier is:

* source code please refer to *hw1-Q2.2-1.R* and detailed model results with different c values please refer to *ksvm_linear.csv*

2.2 You are welcome, but not required, to try other (nonlinear) kernels as well; we're not covering them in this course, but they can sometimes be useful and might provide better predictions than *vanilladot*.

Answer:
We write a function with "kernel" as parameter to try out how the model fits the given data set when the kernel functions are changed. To be specific, the following 8 kernel functions have been applied to our ksvm model (C=100): 'rbfdot', 'polydot', 'vanilladot', tanhdot', 'lapdacedot', 'besseldot', 'anovadot', 'splinedot'.

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

According to our observation, the fitting rate of our model varies when different kernel functions are plugged in. Fitting rate reaches beyond 0.9 for 5 kernel functions, namely 'anovadot'(0.91), 'besseldot'(0.93), 'rbfdot'(0.95), 'splinedot'(0.98), 'laplacedot'(1). Fitting rate of the other 3 kernel functions, 'tanhdot', 'polydot' and 'vanilladot', return to '0.72', '0.86', '0.86'. Statistically, ksvm models with 'polydot' and 'vanilladot' as kernel functions fit to the data set in the same degree, when other parameters of ksvm model remain the same.
Judging from the results, we have 2 interesting findings. For one thing, non-linear kernel functions in general have better performance than the linear ones. For another, the fitting rates of our models can be extremely high, represented by the one with 'laplacedot' kernel function (fitting rate comes up to 100%). In attempt to explain the 2 findings, we consider the following possibilities. On the one hand, the distribution of the data points may be more non-linear by nature, making it more difficult for a linear function to classify than a non-linear one. On the other hand, the train data we

use in this case is the entire data set, which can be responsible for overfit when we test the model with the same data set.

Here directly follows up with a question: which kernel function is the best to choose? To answer this question, we may not jump to conclusion based on the fitting rates we have so far. We need to split the data set into 3 groups and carry out validation, followed by a test.

Table c Fitting rate (given different kernel functions)

	Kernel_Type	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a0	Fitting
1	rbfdot	-19.0825454904208	-37.5173185586883	-8.6732263173343	56.227347158581 7	50.063061890 3967	-24.23039 1947126	15.113388 378474	-24.01142 92894912	-58.11716 58810896	51.4140 5812993 63	0.79439 9172433 416	0.95259 9388379 205
2	polydot	-0.00109297047444185	-0.00124257405317896	-0.00156281568711708	0.0027739329134 0031	1.0051 7814022689	-0.00269010764388622	-0.00019355115223 6571	-0.00052703569614 8093	-0.00145836980542 17	0.10639 9744274 522	0.08157 7161598 8521	0.86391 4373088 685
3	vanilladot	-0.00100653481057611	-0.00117290480611665	-0.00162619672236963	0.0030064202649194	1.0049 4056410556	-0.00282594323043472	0.0002600 29507016313	-0.00053495514349 4997	-0.00122837582291523	0.10636 3399527 188	0.08158 4921659 538	0.86391 4373088 685
4	tanhdot	1609.692 54456138	-606.8929 05450411	-74.03734 60550279	869.1337 30240431	3916.0 9986736614	0	2096.2774 5176997	-1092.973 82822181	269.4265 52504159	1836.96 4315624 19	80.0872 6905374 78	0.72171 2538226 3
5	laplacedot	-3.782973 7410594	-13.85836 98340374	-5.398231 1992979	20.74241 71524196	38.161 5271823421	-10.64435 3221191	15.105446 7850571	-9.285377 30517221	-27.55941 71210435	37.8607 0183224 84	0.31575 9789732 433	1
6	besseldot	-468.8852 25990485	-1977.851 30508313	-1693.069 16166974	492.4782 96508027	83.806 050186 295	223.7565 6298308	-150.4764 38675035	-444.9798 39070019	-135.2223 32399023	310.304 5340735 8	19.4810 8555306 87	0.92507 6452599 388
7	anovadot	0.018837 239605196	-22.49799 70644974	-28.05112 5052939	-2.35838 910572144	2.5358 1364103984	-1.122327 94482793	-3.114825 38723148	-0.045905 027575216 1	-16.44537 42797092	14.8248 1721627 6	1.17407 4331035 67	0.90672 7828746 177
8	splinedot	6.035428 00894268	439.9866 68591817	-103.5800 59560676	-23.0075 1968548 86	-16.089 700460 988	60.72145 29801023	-17.64691 78684267	107.81532 8596042	-233.0894 45707365	7.75704 0354550 7	273.893 2770722 73	0.97859 3272171 254

* source code please refer to *hw1-Q2.2-2.r*

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*).

Answer:

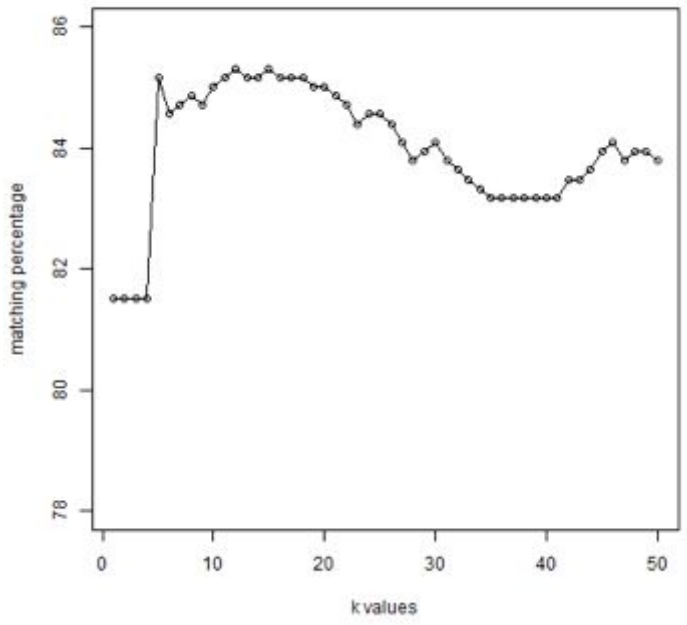
The main solution frame for this question is that we traverse each row of the dataset and select it as the test data for the model based on the entire data excluding the test row. Then we calculate the percentage of matching points in a certain model setting. Source code for the function is:

Figure a kknn Function

```
kknn_model <- function(dis_type,kernel_type,k_value){
  pred <- vector()
  for (i in seq(1:nrow(data))){
    model <- kknn(R1~.,data[-i,],data[i,],distance=dis_type,kernel=kernel_type,k=k_value,scale=TRUE)
    if (fitted(model)>0.5){pred[i] <- 1}
    else {pred[i] <- 0}
  }
  matching_percentage <- sum(pred==data[,11])/nrow(data)*100
  return (matching_percentage)
}
```

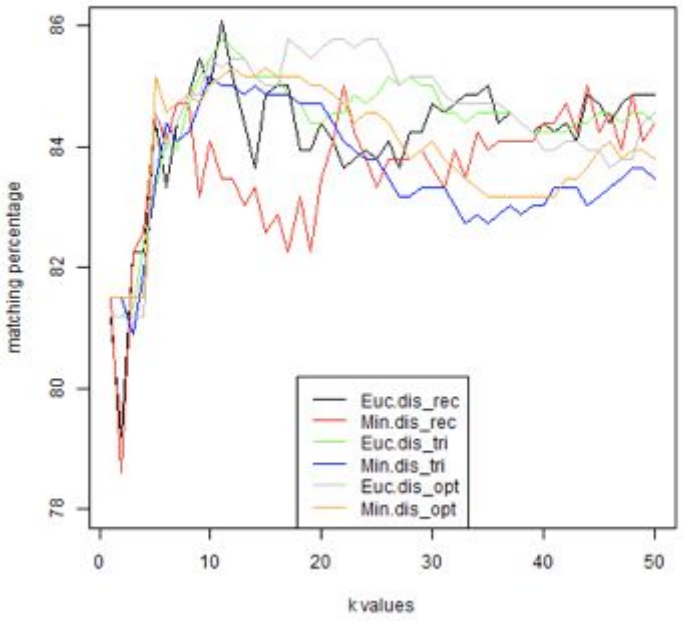
In the first step, we use the default setting for distance (which is 2) and kernel (which is 'optimal'), and we calculate the matching percentage among different k values (from 1 to 50). The result shows that the matching percentage is the highest (around 85.3%) when k=12.

Figure b Matching Percentage with Different k Values (distance=1, kernel='optimal')



Then we run some trials with different combination of distance (1, Euclidean Distance or 2, Minkowski Distance) and kernel ('rectangular'(standard unweighted), 'triangular', or 'optimal'). The matching percentage results are shown in *Figure c*. In general, the variance of matching percentage is not large and most of the data locate between 82% to 86%. As we can see, for most of the models, the best matching results appear around k=10. And among the trials we run, we get the largest matching percentage (86.1%) when k=11, distance=1(Euclidean Distance), kernel='rectangular'.

Figure c Matching Percentage with Different Model Parameters



* source code please refer to *hw1-Q2.2-3(1).R*

Aside from the method of choosing training data and test data defined by the home work instruction, we also tried the sample function to determine the 2 parts of data. With the sample function, we randomly assign 70% of the data points to the training group and the rest 30% to the test group.
After plug all the parameters into kknn model (distance = 1, kernel = "optimal", k = 7, details as below), we can see that 83.08% of the results this model predicts from the test data fit well. Compared to the fitting rate retrieved from model illustrated beforehand, which takes 1 data point out of the whole data set each time until every data point is tested, there is slight difference.

```
pre <- kknn(R1~., traindata, testdata, distance = 1, kernel = "optimal", k = 7, scale = TRUE)
```

* source code please refer to *hw1-Q2.2-3(2).R*

Considering the limit of time and our startup stage, we may put a pause to our exploration of this homework for now. Hopefully, we will update some or many of the methods and observations mentioned above when we trace back with future endeavors in this course. Also, we are open to further discussion if there is any concern regarding the content of our homework.