Assignment 2

Kaggle ID: Songze Yang, u7192786

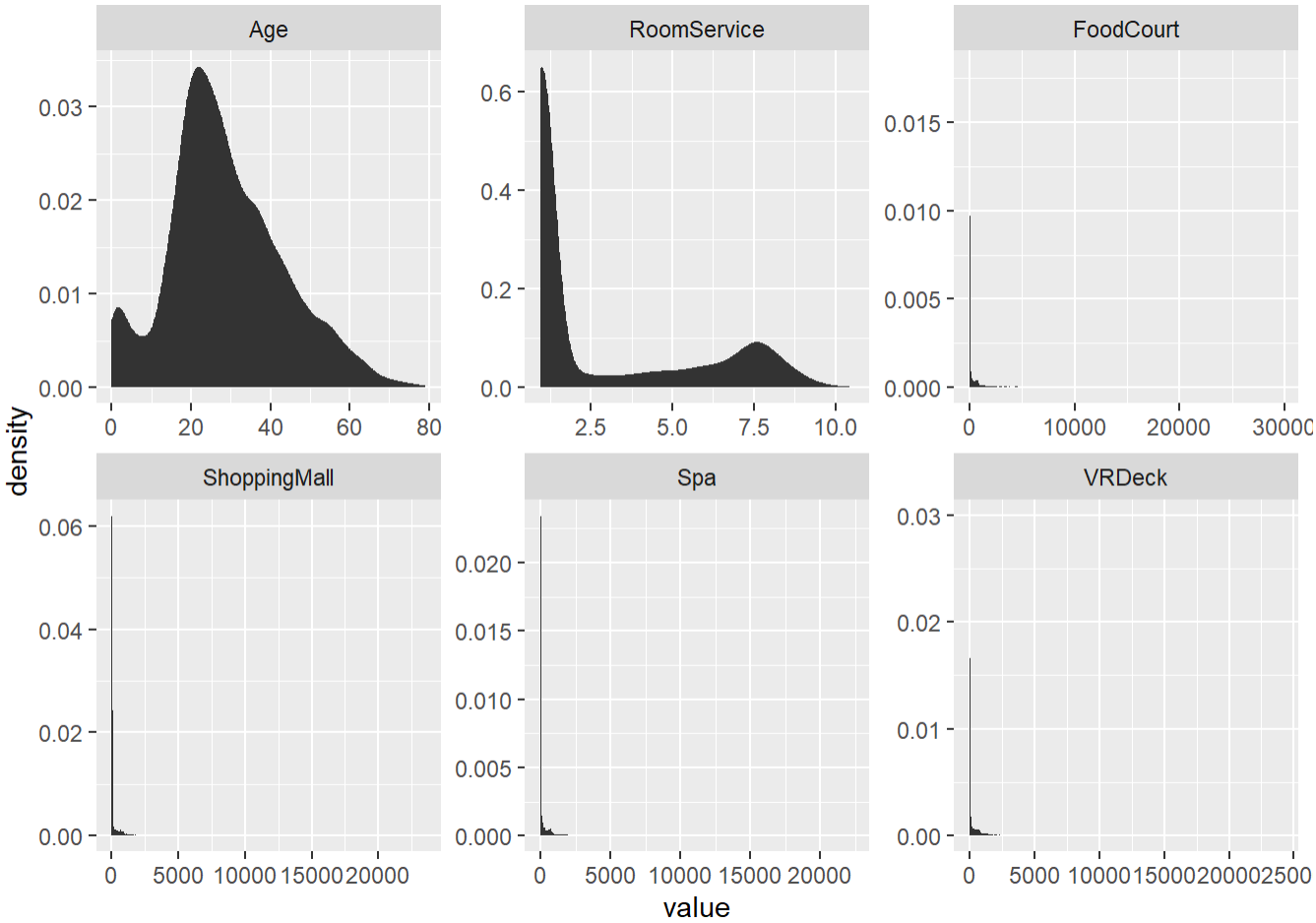
5/7/2022

The Spaceship Titanic is collided with a space time anomaly hidden within a dust cloud. It transmits us the recovered record data on the passengers. We will apply four different methods to predict whether the passengers have been transported successfully.

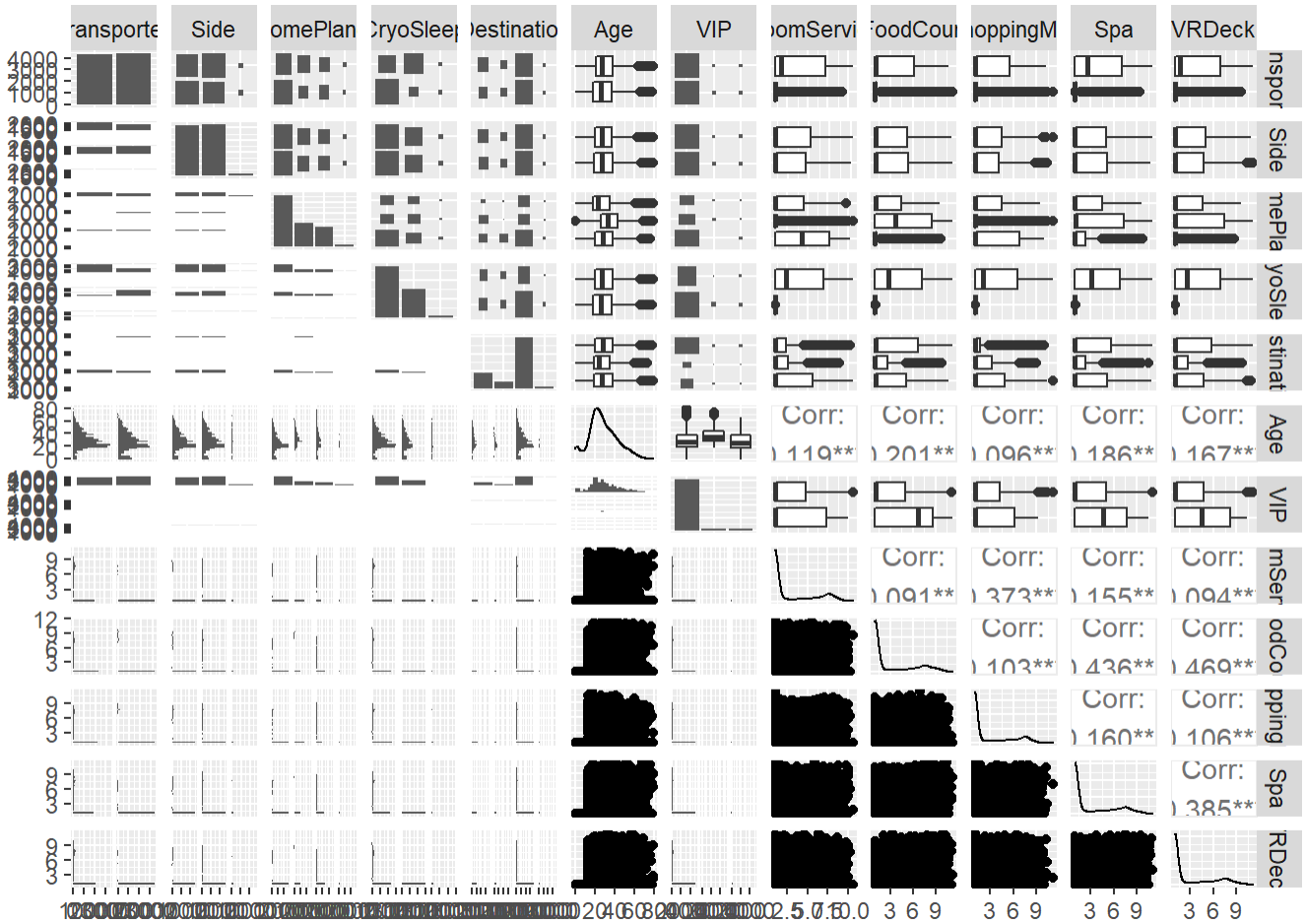
(c)

We will firstly conduct the exploratory data analysis on the data. From the summary statistics, we can see that the consumption variables, namely RoomService, FoodCourt, ShoppingMall, Spa and VRDeck, have a distinct right-skewed feature. A number of rather high expense on the right of the distribution can be regarded as outliers. We will consider applying a log transformation with an extra small random term to discriminate them to help us with the prediction later.

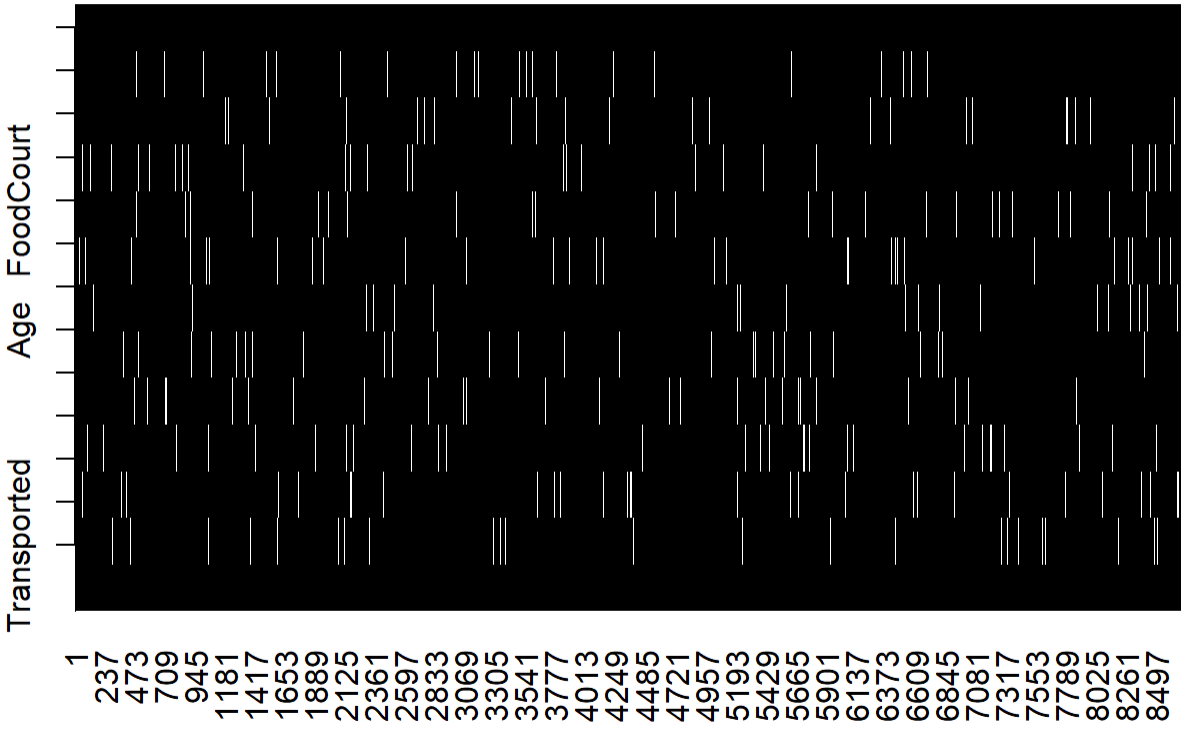
```
train$RoomService<-log(train$RoomService+1)+rnorm(nrow(train),1,0.01)
```



Notice from the scatter plot matrix that the CryoSleep variable is correlated with the consumption variables, as passengers in suspended animation are not able to expense (around 1 after we apply the transformation). This also explain the bimodal feature in the distribution of transformed consumption variables. With regard to the VIP group, it has some high expense points compared to non-VIP group.



Notice that our data contain the missing value. Sorted by the passenager's ID, the missing map has no particular pattern. However, we observe that the people in the same group tend to miss similar data as some of the missing value is quiet closed, indicating a correlation between group travellers.



Rearranged the percentage for the missing values, we clearly note that all the variables miss 2% of the data with the highest percentage of 2.5%. Also, the correlation between the numeric data has nothing higher than 0.5. Thus, we are safe to the apply the multiple imputation here.

```
##    CryoSleep ShoppingMall          VIP   HomePlanet         Side       VRDeck
##   0.02496261   0.02392730   0.02335212   0.02312205   0.02289198   0.02162660
##    FoodCourt          Spa  Destination  RoomService          Age
##   0.02105142   0.02105142   0.02093639   0.02082135   0.02059128
```

```
##              RoomService  FoodCourt ShoppingMall       Spa     VRDeck
## RoomService   1.00000000 0.09138476    0.3728472 0.1550462 0.09425638
## FoodCourt     0.09138476 1.00000000    0.1025941 0.4355119 0.46945028
## ShoppingMall  0.37284724 0.10259406    1.0000000 0.1599203 0.10562720
## Spa           0.15504623 0.43551185    0.1599203 1.0000000 0.38453973
## VRDeck        0.09425638 0.46945028    0.1056272 0.3845397 1.00000000
```

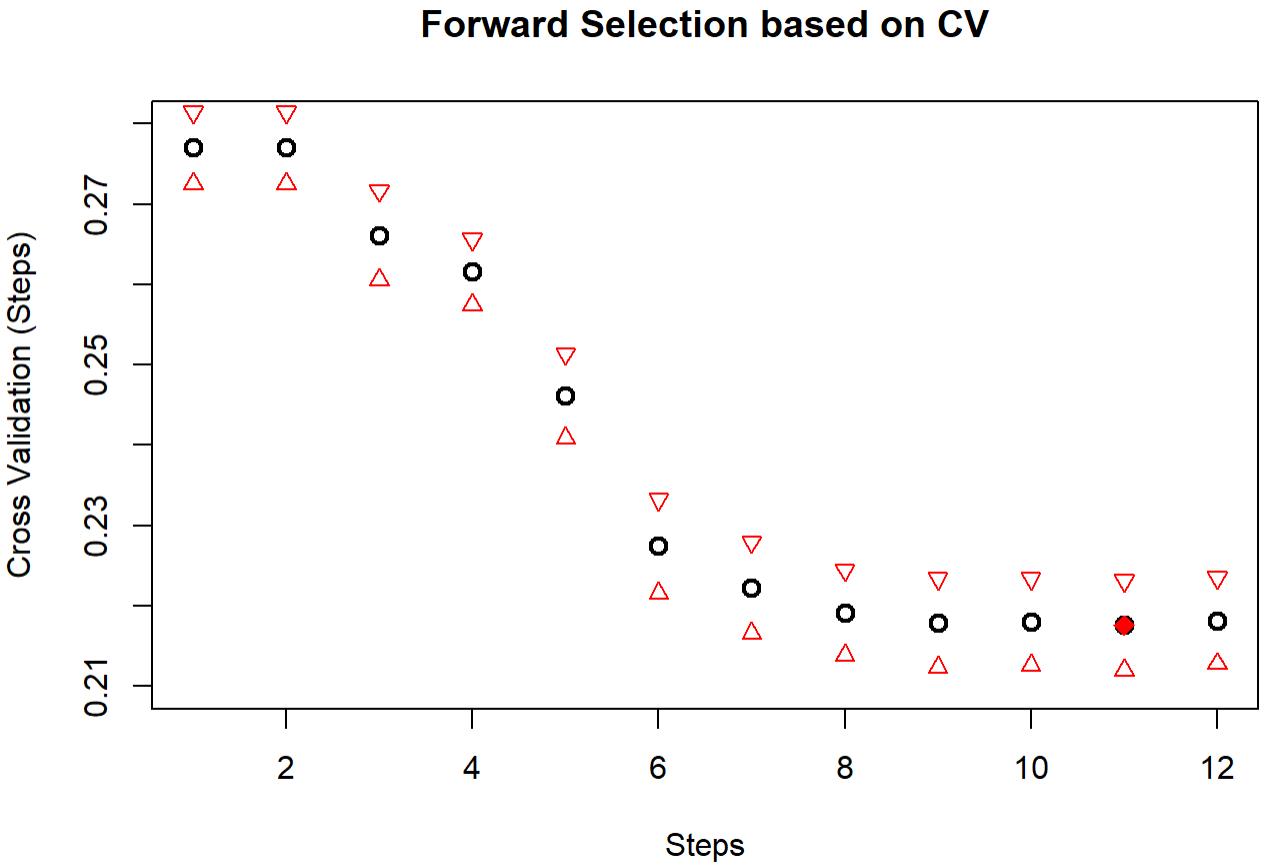
(d)

(i)

Here, we do the feature selection based a forward approach based on 10-fold cross validation. The result shown in graph indicates that the 11th model has lowest cross validation rate. The model is:

Transported ~ CryoSleep + Side + FoodCourt + VRDeck + RoomService + Spa + HomePlanet + ShoppingMall + Destination + VIP + PassengerGroup

```
##           CV(k) CV(k)-SE(CV(k)) CV(k)+SE(CV(k))
##  [1,] 0.2770000       0.2814541       0.2725459
##  [2,] 0.2770000       0.2814541       0.2725459
##  [3,] 0.2661429       0.2716759       0.2606098
##  [4,] 0.2615714       0.2656592       0.2574837
##  [5,] 0.2461429       0.2513768       0.2409089
##  [6,] 0.2274286       0.2332677       0.2215895
##  [7,] 0.2222857       0.2279410       0.2166305
##  [8,] 0.2191429       0.2244634       0.2138223
##  [9,] 0.2178571       0.2233828       0.2123315
## [10,] 0.2180000       0.2234009       0.2125991
## [11,] 0.2175714       0.2231249       0.2120179
## [12,] 0.2181429       0.2235010       0.2127847
```



(ii)

The “best” model from (i) is the 8th model as the model after it has no significant improvement on the error rate. Here, we will use the valid data set (the first 1693 of the train set) to examine the best model. The confusion matrix is provided below. As we increase the threshold of transported, the false positive rate decrease and the false negative rate increase, vice versa. The overall rate have not improved.

```
##
## glm.pred False True
##    False   638  143
##    True    245  667
```

```
##
## glm.pred.0.8 False True
##        False   797  436
##        True     86  374
```

```
##
## glm.pred.0.2 False True
##        False   342   50
##        True    541  760
```

```
##              Mean Error Rate False postitve rate False negative rate
## glm.pred           0.2291790          0.27746319           0.1765432
## glm.pred.0.8       0.3083284          0.09739524           0.5382716
## glm.pred.0.2       0.3490845          0.61268403           0.0617284
```

(iii)

From the summary table, we can calculate the upper 95% interval as:

```
##      (Intercept)    CryoSleepTrue            SideS        FoodCourt
##        0.5261342        1.2089652        0.8241649        0.1570741
##           VRDeck      RoomService              Spa HomePlanetEuropa
##       -0.2318066       -0.2010274       -0.2414680        1.8107253
##   HomePlanetMars     ShoppingMall
##        0.6536799        0.1247772
```

The lower 95% interval is:

```
##      (Intercept)    CryoSleepTrue            SideS        FoodCourt
##       0.08868945       0.81429060       0.58305537       0.10368616
##           VRDeck      RoomService              Spa HomePlanetEuropa
##      -0.28865559      -0.25660296      -0.29714742       1.42138174
##   HomePlanetMars     ShoppingMall
##       0.32468817       0.07301838
```

(iv)

The bootstrap method includes the recursive resampling from the original data set. We can calculate our coefficient based on every bootstrap sample and then compute the mean and stand error based on it. The result is shown below.

```
##    Coefficient Standard Error Upper 95% CI Lower 95% CI
## 1    0.3102201     0.11236225    0.5304501   0.08999013
## 2    1.0138541     0.10225554    1.2142750   0.81343327
## 3    0.7038853     0.06021053    0.8218979   0.58587263
## 4    0.1302792     0.01384365    0.1574128   0.10314569
## 5   -0.2609447     0.01452143   -0.2324827  -0.28940671
## 6   -0.2293323     0.01508251   -0.1997706  -0.25889405
## 7   -0.2701406     0.01465501   -0.2414168  -0.29886444
## 8    1.6240334     0.09316131    1.8066296   1.44143727
## 9    0.4901670     0.07832416    0.6436824   0.33665170
## 10   0.0992726     0.01344629    0.1256273   0.07291786
```

```
##                   Estimate Std. Error  z value  Pr(>|z|)
## (Intercept)       0.307412   0.111593   2.7548  0.005874
## CryoSleepTrue     1.011628   0.100682  10.0477 < 2.2e-16
## SideS             0.703610   0.061508  11.4394 < 2.2e-16
## FoodCourt         0.130380   0.013619   9.5731 < 2.2e-16
## VRDeck           -0.260231   0.014502 -17.9441 < 2.2e-16
## RoomService      -0.228815   0.014177 -16.1394 < 2.2e-16
## Spa              -0.269308   0.014204 -18.9601 < 2.2e-16
## HomePlanetEuropa  1.616054   0.099322  16.2708 < 2.2e-16
## HomePlanetMars    0.489184   0.083926   5.8287 5.585e-09
## ShoppingMall      0.098898   0.013204   7.4901 6.881e-14
##
## n = 7000 p = 10
## Deviance = 6670.71624 Null Deviance = 9701.41808 (Difference = 3030.70184)
```

The bootstrap estimates for coefficients and standard error are slight higher result compared to the summary table. This is because the bootstrap sample are correlated and thus inflate the standard error.

(v)

Here, we will apply the similar method like before. We will use the bootstrap to train the model and then predict based on the test set. We will get a large amount of the prediction and then we will take the average and standard deviation on the data. The results are shown below.

```
##   Est.prob on valid set Std of prob Upper 95% CI Lower 95% CI
## 1             0.8007469 0.022328837   0.84451142   0.75698237
## 2             0.0650464 0.006652798   0.07808589   0.05200692
## 3             0.2492757 0.019464662   0.28742644   0.21112496
## 4             0.5122128 0.024114759   0.55947769   0.46494783
## 5             0.1447675 0.013117907   0.17047862   0.11905643
```

```
##   Est.prob on test set Std of prob Upper 95% CI Lower 95% CI
## 1            0.8182706 0.011120670    0.8400671    0.7964741
## 2            0.2023072 0.019000001    0.2395472    0.1650672
## 3            0.9575829 0.003631276    0.9647002    0.9504656
## 4            0.5443594 0.021303823    0.5861148    0.5026039
## 5            0.6384951 0.019433180    0.6765841    0.6004061
```

(vi)

We have done all the works here. The submission results is a error rate of 0.78653. The rank is 850.

(e)

In this part onward, we will apply the linear discriminate analysis to our data. The logistic regression uses a link function to connect with the linear combination of the variables. Here, the interaction between those variables have not taken into account. As we have not taken the correlation between the CryoSleep and the consumption variables, seeing that the model from logistic, LDA and QDA all contains both the CryoSleep variable and part of the consumption variables, Our model will have room to improve. Adding interaction terms may be a solution to this problem.

For the two class question, one can show that the LDA has the same form as logistic model. Thus, one can expect that the result will be very similar to that in part d, with correlation not taking into account.

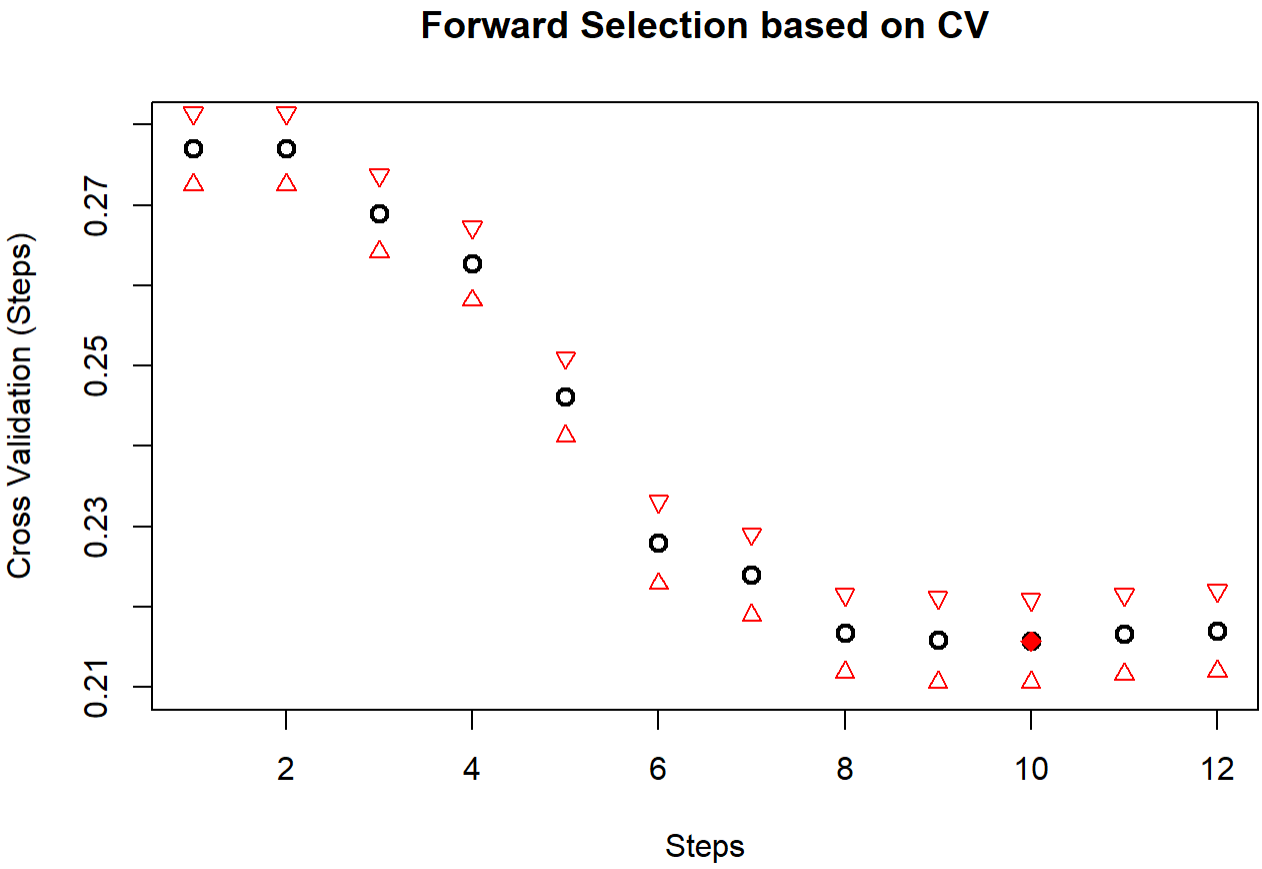
In addition, LDA assume that the covariance matrix in each class is the same, which is not the case here.

(i)

Using the 10-fold cross validation to forward select the best model, we can tell that the 10th model is the lowest in terms of error rate. Then, we our lowest cv model is:

Transported ~ CryoSleep + Side + FoodCourt + VRDeck + RoomService + Spa + HomePlanet + ShoppingMall + Destination + VIP

```
##           CV(k) CV(k)-SE(CV(k)) CV(k)+SE(CV(k))
##  [1,] 0.2770000       0.2814541       0.2725459
##  [2,] 0.2770000       0.2814541       0.2725459
##  [3,] 0.2690000       0.2737669       0.2642331
##  [4,] 0.2627143       0.2672889       0.2581396
##  [5,] 0.2461429       0.2509993       0.2412864
##  [6,] 0.2280000       0.2331296       0.2228704
##  [7,] 0.2240000       0.2290566       0.2189434
##  [8,] 0.2167143       0.2215661       0.2118625
##  [9,] 0.2158571       0.2211058       0.2106085
## [10,] 0.2157143       0.2208607       0.2105679
## [11,] 0.2165714       0.2215576       0.2115853
## [12,] 0.2170000       0.2220009       0.2119991
```



(ii)

The “best” model from (i) is the 8th model as the model after it has no significant improvement on the error rate. Here, we will use the valid data set (the first 1693 of the train set) to examine the best model. The confusion matrix is provided below. As we increase the threshold of transported, the false positive rate decrease and the false negative rate increase, vice versa. The overall rate have not improved.

```
## Transported ~ CryoSleep + Side + FoodCourt + VRDeck + RoomService +
##     Spa + HomePlanet + ShoppingMall
```

```
##
## lda.class False True
##     False   644  150
##     True    239  660
```

```
##
## lda.pred False True
##    False   796  378
##    True     87  432
```

```
##
## lda.pred False True
##    False   380   59
##    True    503  751
```

```
##                  Mean Error Rate False postitve rate False negative rate
## lda.pred.default       0.2297696          0.27066818          0.18518519
## lda.pred.0.8           0.2746604          0.09852775          0.46666667
## lda.pred.0.2           0.3319551          0.56964892          0.07283951
```

(v)

The similar method is applied as before, as the results shown below.

```
##     Est.prob Std of prob Upper 95% CI Lower 95% CI
## 1 0.77290776 0.023638052   0.81923834   0.72657718
## 2 0.05189041 0.005356737   0.06238961   0.04139121
## 3 0.20053284 0.018313105   0.23642652   0.16463915
## 4 0.45802807 0.026631333   0.51022548   0.40583066
## 5 0.12134324 0.012091650   0.14504288   0.09764361
```

```
##    Est.prob Std of prob Upper 95% CI Lower 95% CI
## 1 0.8646572 0.010816958    0.8858584    0.8434559
## 2 0.1940565 0.018578363    0.2304701    0.1576429
## 3 0.9639713 0.002909052    0.9696730    0.9582696
## 4 0.5120062 0.024607472    0.5602368    0.4637756
## 5 0.6145128 0.023327278    0.6602343    0.5687914
```

(vi)

We have done all the works here. The submission results is a error rate of 0.78653. The highest rank is 850. The error rate is same as the glm part.

(f)

In this part onward, we will apply the quadratic discriminate analysis to our data.

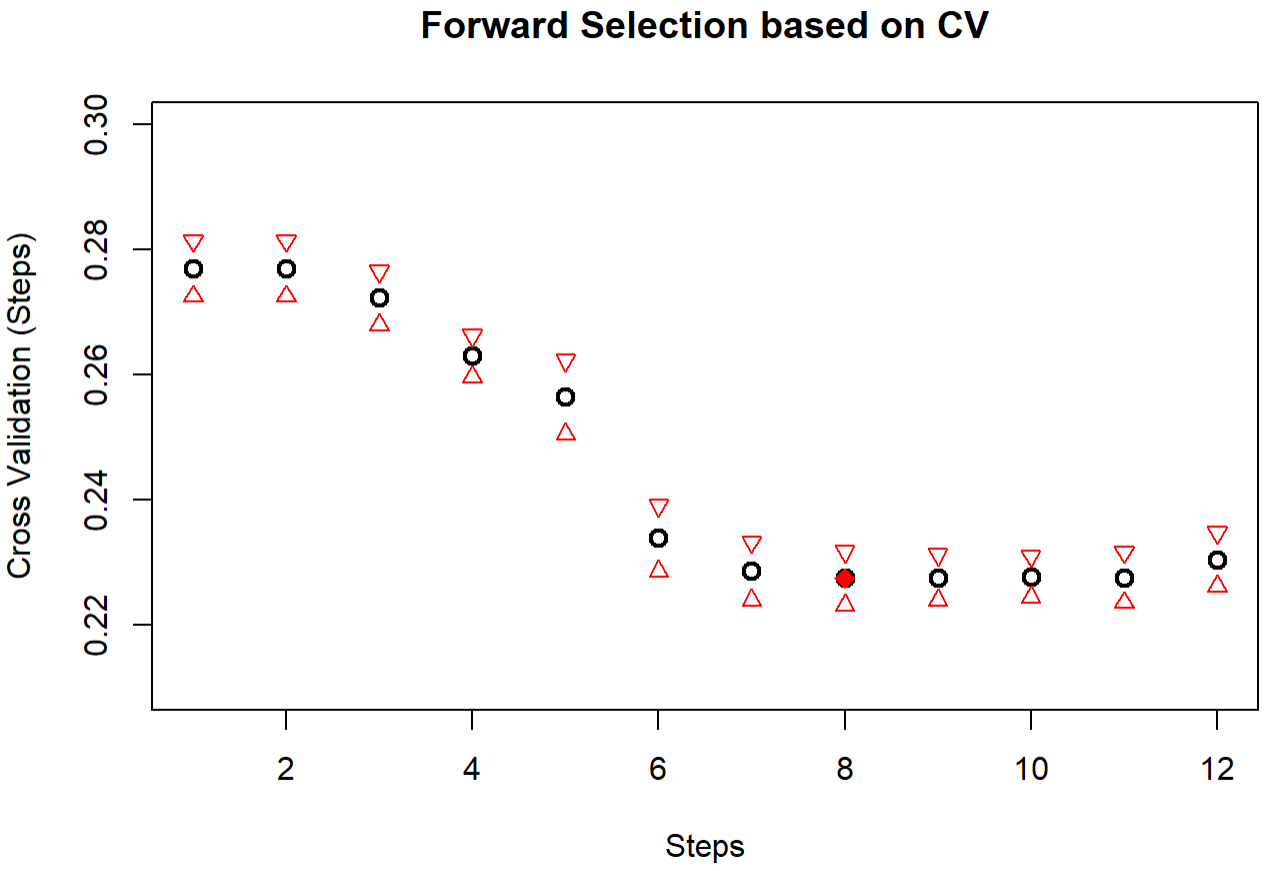
The QDA makes further assumption on the underlying distribution, namely the covariance matrix in each class is different, which is a better assumption here.

Here, the interaction between those variables have not taken into account. But the assumption helps model this as the covariance matrix varies

(i)

Using the 10-fold cross validation to forward select the best model, we can tell that the 8th model is the lowest in terms of error rate. Then, we our lowest cv model is:

Transported ~ CryoSleep + Side + Age + Spa + RoomService + VRDeck + HomePlanet + Destination



(ii)

The “best” model from (i) is the 7th model as the model after it has no significant improvement on the error rate. Here, we will use the valid data set (the first 1693 of the train set) to examine the best model. The confusion matrix is provided below. As we increase the threshold of transported, the false positive rate increase and the false negative rate decrease, vice versa. The overall rate have improved when we increase the threshold.

```
## Transported ~ CryoSleep + Side + Age + Spa + RoomService + VRDeck +
##     HomePlanet
```

```
##
## qda.class False True
##     False   671  188
##     True    212  622
```

```
##
## qda.class False True
##     False   671  188
##     True    212  622
```

```
##
## qda.pred False True
##    False   712  261
##    True    171  549
```

```
##
## qda.pred False True
##    False   640  130
##    True    243  680
```

```
##                  Mean Error Rate False postitve rate False negative rate
## qda.pred.default       0.2362670           0.2400906           0.2320988
## qda.pred.0.8           0.2203190           0.2751982           0.1604938
## qda.pred.0.2           0.2551683           0.1936580           0.3222222
```

(v)

The function applied here is the same as those in LDA. The result is shown below.

```
##      Est.prob Std of prob Upper 95% CI Lower 95% CI
## 1 0.993833809 0.002006596   0.99776674  0.989900881
## 2 0.024910687 0.006915498   0.03846506  0.011356312
## 3 0.009877534 0.003881987   0.01748623  0.002268840
## 4 0.010151970 0.003669367   0.01734393  0.002960010
## 5 0.005869021 0.002240845   0.01026108  0.001476965
```

```
##      Est.prob  Std of prob Upper 95% CI Lower 95% CI
## 1 0.972286848 5.562127e-03  0.983188617  0.961385079
## 2 0.001111976 5.153069e-04  0.002121977  0.000101974
## 3 0.999806591 5.138995e-05  0.999907315  0.999705867
## 4 0.160110527 3.183317e-02  0.222503534  0.097717520
## 5 0.697795044 2.459436e-02  0.745999995  0.649590093
```

(iv)

We have done all the works here. The submission results is a error rate of 0.77577. The highest rank is 850. The error rate is slightly lower than the LDA part.

(g)

In this part onward, we will apply the k-nearest neighbor to our data. The KNN method makes no particular assumption on the data. It simply uses the k-nearest neighbor in the train set the predict the test set. The no assumption is actually a better assumption on our data as the data do not appear to have normally distributed structure of any kind. However, the KNN method will easily collapse as there are too many points in the same distance. Our normalization of the consumption variables will help.

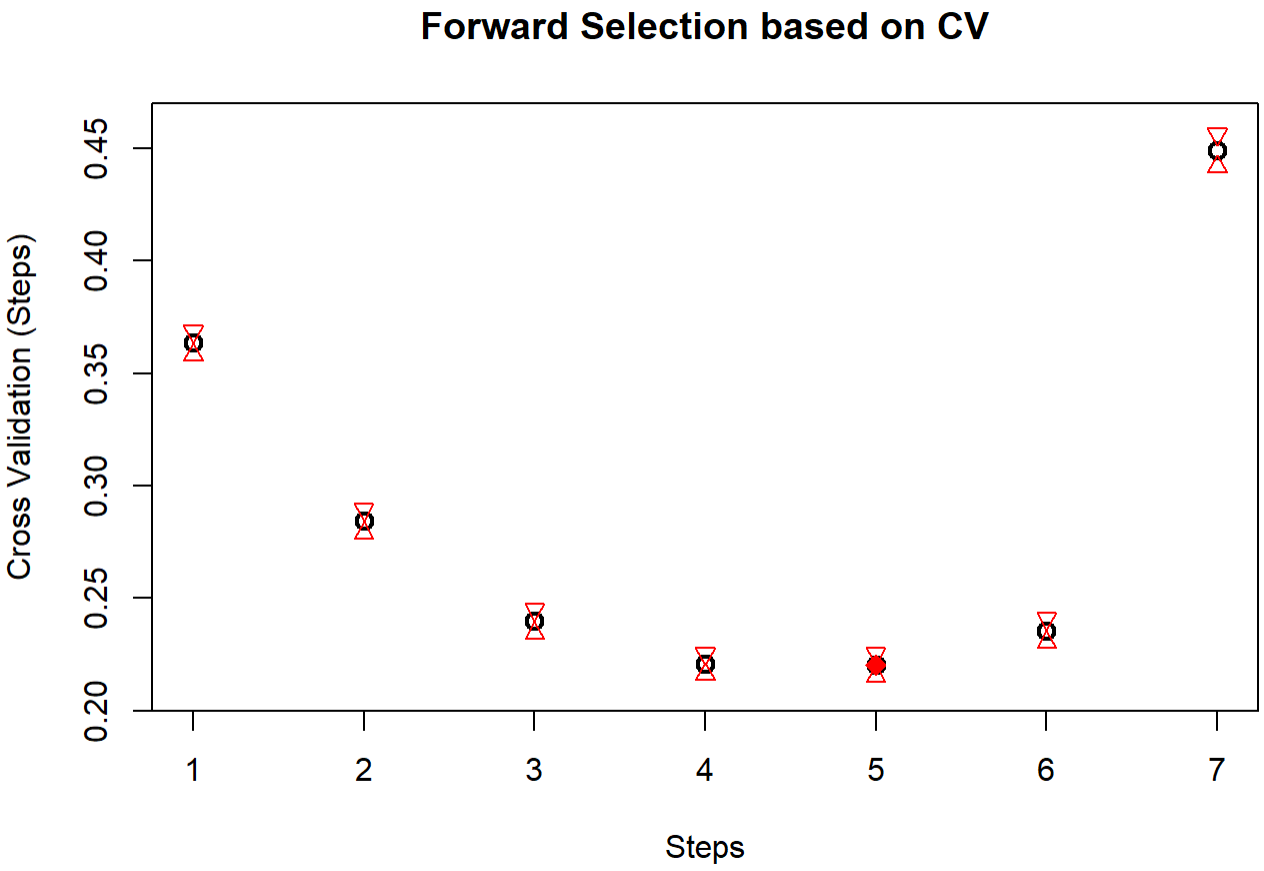
(i)

The k-nearest neighbor method make prediction on the k-nearest points in the train data. However, our data set contains many binomial variables, which will lead to the too many tires problem. Thus, to make a forward selection, we consider to use the shrink data with only the numeric variables.

Using the 10-fold cross validation to forward select the best model, we can tell that the 5th model is the lowest in terms of error rate. Then, our lowest cv model is:

Transported ~ RoomService + Spa + VRDeck + FoodCourt + ShoppingMall

```
##          CV(k) CV(k)-SE(CV(k)) CV(k)+SE(CV(k))
## [1,] 0.3634286       0.3686328       0.3582244
## [2,] 0.2842857       0.2895065       0.2790650
## [3,] 0.2397143       0.2450289       0.2343997
## [4,] 0.2208571       0.2255043       0.2162100
## [5,] 0.2204286       0.2254638       0.2153933
## [6,] 0.2355714       0.2408760       0.2302669
## [7,] 0.4491429       0.4563903       0.4418954
```



Further, if we are interested in getting the more complex model, the mlr3 package can be applied here to do the forward selection. The selected model has 8 selected variable and an error rate of 0.2322857 shown below. However, the model selected above only has an error rate of 0.2204286. Thus, the model selected by the package may have overfitting problem. We will stick with our model above.

(ii)

The “best” model from (i) is the 4th model as the model after it has no significant improvement on the error rate. Here, we will use the valid data set (the first 1693 of the train set) to examine the best model. The confusion matrix is provided below. As we increase the number of points taken into consideration, the false positive rate decrease and the false negative rate increase, vice versa. The overall rate have improved when we increase the k-nearest neighbor to 80.

```
##
## mod     False True
##   False   626  110
##   True    257  700
```

```
##
## mod     False True
##   False   605  159
##   True    278  651
```

```
##
## mod     False True
##   False   644  122
##   True    239  688
```

```
##        Mean Error Rate False postitve rate False negative rate
## knn.20       0.2167750           0.2910532           0.1358025
## knn.5        0.2581217           0.3148358           0.1962963
## knn.80       0.2132310           0.2706682           0.1506173
```

(v)

The function applied here is the same as those in previous question. The result is shown below.

```
##   Est.prob Std of prob Upper 95% CI Lower 95% CI
## 1     0.99         0.1        1.186        0.794
## 2     0.00         0.0        0.000        0.000
## 3     0.00         0.0        0.000        0.000
## 4     0.00         0.0        0.000        0.000
## 5     0.00         0.0        0.000        0.000
```

```
##   Est.prob Std of prob Upper 95% CI Lower 95% CI
## 1     0.95   0.2190429     1.379324   0.52067589
## 2     0.00   0.0000000     0.000000   0.00000000
## 3     1.00   0.0000000     1.000000   1.00000000
## 4     0.99   0.1000000     1.186000   0.79400000
## 5     0.80   0.4020151     1.587950   0.01205035
```

(vi)

We have done all the works here. The submission results is a error rate of 0.78255. The highest rank is 850. The error rate is slightly higher than the QDA part.

(h)

Logistic: Transported ~ CryoSleep + Side + FoodCourt + VRDeck + RoomService + Spa + HomePlanet + ShoppingMall ## Score: 0.78653 LDA: Transported ~ CryoSleep + Side + FoodCourt + VRDeck + RoomService + Spa + HomePlanet + ShoppingMall ## Score: 0.78653 QDA: Transported ~ CryoSleep + Side + Age + Spa + RoomService + VRDeck + HomePlanet ## Score: 0.77577 KNN: Transported ~ RoomService + Spa + VRDeck + FoodCourt ## Score: 0.78255

From the overall performance of our four model, we see that the logistic model and the LDA produce the same prediction results, followed by the KNN model and then the QDA model. Surprisingly, the KNN works quite good in the light of only four variables taken into account.

The transported rate is highly correlated with the CryoSleep variable, as in our forward selection process of logistic, LDA and QDA the CryoSleep is always the first to come, or in other word, most reduce our model error rate. Maybe when the collision happens, the people in sleeping are kept from harm.

Also, the side also matter, as it is significant in logistic, LDA and QDA model. The people at the Starboard have greater opportunity to be transported to the target planet. The effect of the home planet may also be useful but less.

Finally, the consumption variable may be helpful when predicting the people who are not in suspended animation in the interstellar travel as people in suspended animation have no expense. The KNN results are a good illustration, in which Room Service and Spa may be an better indicator than Age and Shopping Mall. It is possible that the individual room and Spa lounge are closer to the safe exits.

In conclusion, for the people who in the suspended animation and on starboard, their transported rates are higher. The passengers who is from Europa may benefit in the transportation process. Also, the information on who are awake and spend money on room service, spa, VR deck and food court may be relevent.

Appendix

Code used in this assignment can be found here.

```
################################## EDA

################################## Package loading

library(dplyr)
##install.packages("mice")
library(mice)
library(leaps)
library(MASS)
library (boot)
library(class)
library(reshape2)
library(GGally)

################################
getwd()
setwd("E:/ANU Sem 2/STAT3040STAT7040 - Statistical Learning/Assignment 2")
train <-
  read.csv(file="train.csv",header=TRUE,stringsAsFactors=F,na.strings = "")
test.raw<-
  read.csv(file="test.csv",header=TRUE,stringsAsFactors=F,na.strings = "")


################################  Create side variable for train

Side<-rep(NA, nrow(train))
Port<-grep("P",train$Cabin)
Starb<-grep("S",train$Cabin)
Side[Port]<-"P"
Side[Starb]<-"S"
Side<-as.factor(Side)
Transported<-train$Transported
train<-train[,1:13]
train<-data.frame(Transported,Side,train)

################################  Create side variable for test

Side<-rep(NA, nrow(test.raw))
Port<-grep("P",test.raw$Cabin)
Starb<-grep("S",test.raw$Cabin)
Side[Port]<-"P"
Side[Starb]<-"S"
Side<-as.factor(Side)
test.raw<-test.raw[,1:13]
test<-data.frame(Side,test.raw)

################################  Create Passenger Group variable train

train$PassengerGroup <-
  sapply(train$PassengerId,function(x) strsplit(x,'_')[[1]][1])

################################  Create Passenger Group variable test

test$PassengerGroup <-
```

```
  sapply(test$PassengerId,function(x) strsplit(x,'_')[[1]][1])

train <- train %>% dplyr::select(-PassengerId,-Cabin,-Name)

##(c)
summary(train)



require(reshape2)
melt.train <- melt(train)
ggplot(data = melt.train, aes(x = value)) +
  stat_density() +
  facet_wrap(~variable, scales = "free")

## apply transformation in the character variable

train$Transported<-as.factor(train$Transported)
train$Side<-as.factor(train$Side)
train$HomePlanet<-as.factor(train$HomePlanet)
train$CryoSleep<-as.factor(train$CryoSleep)
train$Destination<-as.factor(train$Destination)
train$VIP<-as.factor(train$VIP)
train$PassengerGroup<-as.factor(train$PassengerGroup)

##EDA discover the pattern of missing data

##suggest a transformation of the the consumption variable

## apply transformation in the Consumption variable

set.seed(1)
train$RoomService<-log(train$RoomService+1)+rnorm(nrow(train),1,0.01)
train$FoodCourt<-log(train$FoodCourt+1)+rnorm(nrow(train),1,0.01)
train$ShoppingMall<-log(train$ShoppingMall+1)+rnorm(nrow(train),1,0.0001)
train$Spa<-log(train$Spa+1)+rnorm(nrow(train),1,0.01)
train$VRDeck<-log(train$VRDeck+1)+rnorm(nrow(train),1,0.01)
summary(train)

melt.train <- melt(train)
ggplot(data = melt.train, aes(x = value)) +
  stat_density() +
  facet_wrap(~variable, scales = "free")

##scatter plot matrix
pairs(train, col = "blue")
ggpairs(train[,1:12])
#the CryoSleep and the consumption are correlated
#the Side and other qualitative var and the consumption are not correlated
## VIP may expense more on the boat

par(mfrow = c(2,3))
plot(train$Side,train$Age)
plot(train$CryoSleep,train$RoomService)
plot(train$CryoSleep,train$FoodCourt)
```

```r
plot(train$CryoSleep,train$ShoppingMall)
plot(train$CryoSleep,train$Spa)
plot(train$CryoSleep,train$VRDeck)

##Assignment 2
################################## Package loading

library(dplyr)
##install.packages("mice")
library(mice)
library(leaps)
library(MASS)
library (boot)
library(class)

##################################
getwd()
setwd("E:/ANU Sem 2/STAT3040STAT7040 - Statistical Learning/Assignment 2")
train <-
  read.csv(file="train.csv",header=TRUE,stringsAsFactors=F,na.strings = "")
test.raw<-
  read.csv(file="test.csv",header=TRUE,stringsAsFactors=F,na.strings = "")


################################## Create side variable for train

Side<-rep(NA, nrow(train))
Port<-grep("P",train$Cabin)
Starb<-grep("S",train$Cabin)
Side[Port]<-"P"
Side[Starb]<-"S"
Side<-as.factor(Side)
Transported<-train$Transported
train<-train[,1:13]
train<-data.frame(Transported,Side,train)



################################## Create Passenger Group variable train

train$PassengerGroup <-
  sapply(train$PassengerId,function(x) strsplit(x,'_')[[1]][1])

## apply transformation in the Consumption variable

set.seed(1)
train$RoomService<-log(train$RoomService+1)+rnorm(nrow(train),1,0.01)
train$FoodCourt<-log(train$FoodCourt+1)+rnorm(nrow(train),1,0.01)
train$ShoppingMall<-log(train$ShoppingMall+1)+rnorm(nrow(train),1,0.0001)
train$Spa<-log(train$Spa+1)+rnorm(nrow(train),1,0.01)
train$VRDeck<-log(train$VRDeck+1)+rnorm(nrow(train),1,0.01)
summary(train)

##manage variable
```

```
train <- train %>% dplyr::select(-PassengerId,-Cabin,-Name)

train$Transported<-as.factor(train$Transported)
train$Side<-as.factor(train$Side)
train$HomePlanet<-as.factor(train$HomePlanet)
train$CryoSleep<-as.factor(train$CryoSleep)
train$Destination<-as.factor(train$Destination)
train$VIP<-as.factor(train$VIP)
train$PassengerGroup<-as.factor(train$PassengerGroup)

################################  Detect the pattern of missing data train

##Missing data map
miss_map<-function(data){
  image(is.na(data), axes = FALSE, col = gray(0:1))
  axis(2, at = 1:ncol(data)/ncol(data), labels = colnames(data))
  axis(1, at = 1:nrow(data)/nrow(data), labels = row.names(data),
       las = 2,col = "white")
}
par(mfrow = c(1,1))
miss_map(train)
##people travel together tends to miss the similar data

t_missing <- unlist(lapply(train, function(x) sum(is.na(x))))/nrow(train)
sort(t_missing[t_missing > 0], decreasing = TRUE)
## All below the 25% threhold, ready to impute
cor(train[,8:12], use = "pairwise.complete.obs")
##pairs(train[,c(1,4,8:12)],col = "blue")
##no nigh proportional missing value
##no high correlated numeric value


################################  get train and valid data set

valid<-train[c(1:1693),]
train<-train[-c(1:1693),]

##exclude the passager group variable

imp <- mice(train, maxit=0)
predM <- imp$predictorMatrix
meth <- imp$method
predM[, c("PassengerGroup")] <- 0

# Specify a separate imputation model for variables of interest


# Ordered categorical variables
poly <- c("CryoSleep", "VIP","Transported")

# Dichotomous variable
log <- c("Side")

# Unordered categorical variable
```

```
poly2 <- c("HomePlanet","Destination","PassengerGroup")

meth[poly] <- "polr"
meth[log] <- "logreg"
meth[poly2] <- "polyreg"

train <- mice(train, maxit = 5,
              predictorMatrix = predM,
              method = meth, print =  FALSE)

train.comp <- mice::complete(train, 1)
train.comp$PassengerGroup<-as.factor(train.comp$PassengerGroup)
write.csv(train.comp,file="train_comp.csv",row.names=FALSE)
train.comp <-
  read.csv(file="train_comp.csv",header=TRUE,stringsAsFactors=T)
################################## Missing data check
par(mfrow = c(1,1))
miss_map(train.comp)




################################  Detect the pattern of missing data valid

##Missing data map
miss_map<-function(data){
  image(is.na(data), axes = FALSE, col = gray(0:1))
  axis(2, at = 1:ncol(data)/ncol(data), labels = colnames(data))
  axis(1, at = 1:nrow(data)/nrow(data), labels = row.names(data),
       las = 2,col = "white")
}
par(mfrow = c(1,1))
miss_map(valid)
##people travel together tends to miss the similar data

v_missing <- unlist(lapply(valid, function(x) sum(is.na(x))))/nrow(valid)
sort(v_missing[v_missing > 0], decreasing = TRUE)
## All below the 25% threhold, ready to impute
cor(valid[,8:12], use = "pairwise.complete.obs")
##pairs(train[,c(1,4,8:12)],col = "blue")
##no nigh proportional missing value
##no high correlated numeric value


##exclude the passager group variable

imp <- mice(valid, maxit=0)
predM <- imp$predictorMatrix
meth <- imp$method
predM[, c("PassengerGroup")] <- 0

# Specify a separate imputation model for variables of interest
```

```
# Ordered categorical variables
poly <- c("CryoSleep", "VIP","Transported")

# Dichotomous variable
log <- c("Side")

# Unordered categorical variable
poly2 <- c("HomePlanet","Destination","PassengerGroup")

meth[poly] <- "polr"
meth[log] <- "logreg"
meth[poly2] <- "polyreg"

valid <- mice(valid, maxit = 5,
              predictorMatrix = predM,
              method = meth, print =  FALSE)

valid<- mice::complete(valid, 1)
valid$PassengerGroup<-as.factor(valid$PassengerGroup)
write.csv(valid,file="valid_comp.csv",row.names=FALSE)
valid <-
  read.csv(file="valid_comp.csv",header=TRUE,stringsAsFactors=T)
################################## Missing data check
par(mfrow = c(1,1))
miss_map(valid)




################################  Create side variable for test

Side<-rep(NA, nrow(test.raw))
Port<-grep("P",test.raw$Cabin)
Starb<-grep("S",test.raw$Cabin)
Side[Port]<-"P"
Side[Starb]<-"S"
Side<-as.factor(Side)
test.raw<-test.raw[,1:13]
test.raw<-data.frame(Side,test.raw)

################################  Create Passenger Group variable test

test.raw$PassengerGroup <-
  sapply(test.raw$PassengerId,function(x) strsplit(x,'_')[[1]][1])


## apply transformation in the Consumption variable

set.seed(1)
test.raw$RoomService<-log(test.raw$RoomService+1)+rnorm(nrow(test.raw),1,0.01)
test.raw$FoodCourt<-log(test.raw$FoodCourt+1)+rnorm(nrow(test.raw),1,0.01)
test.raw$ShoppingMall<-log(test.raw$ShoppingMall+1)+rnorm(nrow(test.raw),1,0.0001)
test.raw$Spa<-log(test.raw$Spa+1)+rnorm(nrow(test.raw),1,0.01)
```

```
test.raw$VRDeck<-log(test.raw$VRDeck+1)+rnorm(nrow(test.raw),1,0.01)
summary(test.raw)

##manage variable
test.raw <- test.raw %>% dplyr::select(-PassengerId,-Cabin,-Name)

test.raw$Transported<-as.factor(test$Transported)
test.raw$Side<-as.factor(test.raw$Side)
test.raw$HomePlanet<-as.factor(test.raw$HomePlanet)
test.raw$CryoSleep<-as.factor(test.raw$CryoSleep)
test.raw$Destination<-as.factor(test.raw$Destination)
test.raw$VIP<-as.factor(test.raw$VIP)
test.raw$PassengerGroup<-as.factor(test.raw$PassengerGroup)



################################  Detect the pattern of missing data test

miss_map(test.raw)
test_missing <- unlist(lapply(test.raw, function(x) sum(is.na(x))))/nrow(test)
sort(test_missing[test_missing > 0], decreasing = TRUE)
test_numeric<-test.raw[,c(5,7:11)]
cor(test_numeric, use = "pairwise.complete.obs")
pairs(test_numeric,col = "blue")


test.raw <- test.raw %>% dplyr::select(-PassengerId,-Cabin,-Name)

test.raw$Side<-as.factor(test.raw$Side)
test.raw$HomePlanet<-as.factor(test.raw$HomePlanet)
test.raw$CryoSleep<-as.factor(test.raw$CryoSleep)
test.raw$Destination<-as.factor(test.raw$Destination)
test.raw$VIP<-as.factor(test.raw$VIP)


imp <- mice(test.raw, maxit=0)
predM <- imp$predictorMatrix
meth <- imp$method
predM[, c("PassengerGroup")] <- 0

# Ordered categorical variables
poly <- c("CryoSleep", "VIP")

# Dichotomous variable
log <- c("Side")

# Unordered categorical variable
poly2 <- c("HomePlanet","Destination","PassengerGroup")

meth[poly] <- "polr"
meth[log] <- "logreg"
meth[poly2] <- "polyreg"

test <- mice(test.raw, maxit = 5,
```

```
              predictorMatrix = predM,
              method = meth, print =  FALSE)

test <- mice::complete(test, 1)
test$PassengerGroup<-as.factor(test$PassengerGroup)
write.csv(test,file="test_comp.csv",row.names=FALSE)
test <-
  read.csv(file="test_comp.csv",header=TRUE,stringsAsFactors=T)


################################## Missing data check test
par(mfrow = c(1,1))
miss_map(test)

################################## GLM

################################## Package loading

library(dplyr)
##install.packages("mice")
library(mice)
library(leaps)
library(MASS)
library (boot)
library(class)

################################## data loading

setwd("E:/ANU Sem 2/STAT3040STAT7040 - Statistical Learning/Assignment 2")
train.comp <-
  read.csv(file="train_comp.csv",header=TRUE,stringsAsFactors=T)
valid<-
  read.csv(file="valid_comp.csv",header=TRUE,stringsAsFactors=T)
test<-
  read.csv(file="test_comp.csv",header=TRUE,stringsAsFactors=T)



##(d)

################################## i

full.mod<-as.formula(Transported~.)
null.mod<-as.formula(Transported~1)

glm.full<-glm(full.mod,data = train.comp,
              family = binomial)
glm.null <- glm(null.mod,
                data = train.comp,family = binomial
)

################################## CV

cv.err<-function(method,formula,data,K,knnfold){
```

```
  n<-nrow(data)
  set.seed(888)
  fold<-sample(rep(1:K,each = n/K))
  mse.out <- rep(0,K)

  ##
  for(k in 1:K){
    data.train <- data[fold!=k,]
    data.test <- data[fold==k,]

    if(method=="glm"){
      mod.train <- glm(formula, data = data.train,family = "binomial")
      pred.test <- predict(mod.train, newdata = data.test,type = "response")
      glm.pred<-rep("False",nrow(data.test))
      glm.pred[pred.test>0.5] = "True"
      mse.out[k] <- mean(data.test$Transported != glm.pred)
      }
    else if(method == "lda"){
      mod.train <- lda(formula, data = data.train)
      lda.class <- predict(mod.train, newdata = data.test)$class
      mse.out[k] <- mean(data.test$Transported != lda.class)
      }
    else if(method == "qda"){
      mod.train <- qda(formula, data = data.train)
      qda.class <- predict(mod.train, newdata = data.test)$class
      mse.out[k] <- mean(data.test$Transported != qda.class)
    }
    else if(method == "knn"){
      X.train <- model.matrix(formula, data =data.train)
      X.test <- model.matrix(formula, data = data.test)
      mod.out <- knn(X.train, X.test, data.train$Transported , k = knnfold)
      mse.out[k] <- mean(data.test$Transported != mod.out)
    }
  }
  mse.est <- mean(mse.out)

  sd.mse.est <- sd(mse.out)/sqrt(K)

  return(c(mse.est,sd.mse.est))
}

cv.err("glm",glm.null,train.comp,10)

################################## forward selection

method.fwd<-function(method,data,kfold){
  variable<-names(data[2:ncol(data)])
  max.steps<-length(variable)
  mod<-"Transported ~ "
  a<-1
  cv.out<-matrix(0,nrow = max.steps,ncol = 2)
  while(a<=max.steps){
    cv.list<-matrix(0,nrow = length(variable),ncol = 2)
    for(i in 1:length(variable)){
```

```
      if(a == 1){formula<-as.formula(paste(mod,variable[i]))}
      else if(a>1){formula<-as.formula(paste(mod,"+",variable[i]))}
      if(method=="glm")
        re<-cv.err("glm",formula,data,10)
      else if(method=="lda")
        re<-cv.err("lda",formula,data,10)
      else if(method == "qda")
        re<-cv.err("qda",formula,data,10)
      else if(method == "knn")
        re<-cv.err("knn",formula,data,10,knnfold = kfold)
      cv.list[i,1]<-re[1]
      cv.list[i,2]<-re[2]
    }
    selected.var<-variable[which.min(cv.list[,1])]
    cv.out[a,1]<-min(cv.list[,1])
    cv.out[a,2]<-cv.list[which.min(cv.list[,1]),2]
    variable<-variable[-which.min(cv.list[,1])]
    mod<-paste(mod, "+",selected.var)
    a <-a +1
  }
  return(list(cv.out,mod))
}

cv.out<-method.fwd("glm",train.comp)

plot(cv.out[[1]][,1],type = "p",lwd = 2,cex = 1.2,
     ylab = "Cross Validation (Steps)",
     xlab = "Steps",
     main = "Forward Selection based on CV",
     ylim = c(0.21,0.28))
points(which.min(cv.out[[1]][,1]),min(cv.out[[1]][,1]),
       pch = 18,col = "red",cex = 1.5)
points(1:(ncol(train.comp)-1),cv.out[[1]][,1]+cv.out[[1]][,2],
       pch = 25,col = "red",cex = 1)
points(1:(ncol(train.comp)-1),cv.out[[1]][,1]-cv.out[[1]][,2],
       pch = 24,col = "red",cex = 1)
mod.last<-cv.out[[2]]
mod.final<-as.formula(Transported ~CryoSleep + Side +
                        FoodCourt + VRDeck + RoomService +
                        Spa + HomePlanet + ShoppingMall)

################################ ii

mod.final<-as.formula(Transported ~CryoSleep + Side +
                        FoodCourt + VRDeck + RoomService +
                        Spa + HomePlanet + ShoppingMall)
best.mod<-glm(mod.final,data = train.comp,family = "binomial")

pred.valid <- predict(best.mod, newdata = valid,type = "response")
glm.pred<-rep("False",nrow(valid))
glm.pred[pred.valid>0.5] = "True"
c.table<-table(glm.pred,valid$Transported)
c.table
glm.pred<-c(mean(valid$Transported != glm.pred),
```

```
            c.table[2,1]/(c.table[1,1]+c.table[2,1]),
            c.table[1,2]/(c.table[1,2]+c.table[2,2]))

pred.valid <- predict(best.mod, newdata = valid,type = "response")
glm.pred.0.8<-rep("False",nrow(valid))
glm.pred.0.8[pred.valid>0.8] = "True"
c.table<-table(glm.pred.0.8,valid$Transported)
c.table
glm.pred.0.8<-c(mean(valid$Transported != glm.pred.0.8),
                c.table[2,1]/(c.table[1,1]+c.table[2,1]),
                c.table[1,2]/(c.table[1,2]+c.table[2,2]))

pred.valid <- predict(best.mod, newdata = valid,type = "response")
glm.pred.0.2<-rep("False",nrow(valid))
glm.pred.0.2[pred.valid>0.2] = "True"
c.table<-table(glm.pred.0.2,valid$Transported)
c.table
glm.pred.0.2<-c(mean(valid$Transported != glm.pred.0.2),
                c.table[2,1]/(c.table[1,1]+c.table[2,1]),
                c.table[1,2]/(c.table[1,2]+c.table[2,2]))
con.out<-rbind(glm.pred,glm.pred.0.8,glm.pred.0.2)
colnames(con.out) <-  c("Mean Error Rate","False postitve rate",
                        "False negative rate")
con.out


################################## iii

summary(best.mod)
summary (best.mod)$coef
summary (best.mod)$coef[,1]+1.96*summary (best.mod)$coef[,2]
summary (best.mod)$coef[,1]-1.96*summary (best.mod)$coef[,2]

################################## iv

boots<-function(best.mod,data, K){
  nvar<-length(best.mod$coefficients)
  boot.cof<-matrix(0,nvar,K)
  boot.se<-matrix(0,nvar,K)
  for(i in 1:K){
    set.seed(i+i^2+i*2022)
    resample <- data[sample(nrow(data), nrow(data),replace = TRUE), ]
    new.mod<-glm(best.mod$formula,data = resample,family = "binomial")
    boot.cof[,i]<-summary (new.mod)$coef[,1]
    boot.se[,i]<-summary (new.mod)$coef[,2]
  }
  boot.out<-matrix(0,nvar,2)
  for(i in 1:nvar){
    boot.out[i,1]<-mean(boot.cof[i,])
    boot.out[i,2]<-sd(boot.cof[i,])
  }
  return(boot.out)
}
boot.out<-as.data.frame(boots(best.mod,train.comp,2000))
```

```
boot.out<-cbind(boot.out,boot.out[,1]+1.96*boot.out[,2],boot.out[,1]-1.96*boot.out[,2])
names(boot.out)<-c("Coefficient","Standard Error","Upper 95% CI","Lower 95% CI")
boot.out
summary(best.mod)
#larger variance and the coefficient is closed

################################## v

boot.fn <- function (method, mod, K, data, test, index){
  boot.pred<-matrix(0,K,index)
  for(i in 1:K){
    set.seed(i+i^2+i*2022)
    resample <- data[sample(nrow(data), nrow(data),replace = TRUE), ]
    if(method == "glm"){
      new.mod<-glm(mod$formula,data = resample,family = "binomial")
      pred.test<-predict(new.mod, newdata = test[1:index,],type = "response")
    }
    else if(method == "lda"){
      new.mod<-lda(mod,data = resample)
      pred.test <- predict(new.mod, newdata = test[1:index,])$posterior[,2]
    }
    else if(method == "qda"){
      new.mod<-qda(mod,data = resample)
      pred.test <- predict(new.mod, newdata = test[1:index,])$posterior[,2]
    }
    else if(method=="knn"){
      X.train <- model.matrix(mod, data=resample)
      all.vars(mod)
      predictors<-as.formula(paste("~",all.vars(mod)[-1], collapse = "+"))
      X.test <- model.matrix(predictors, data=test[1:index,])
      Y.train <- as.factor(resample$Transported)
      pred.test <- knn(X.train, X.test, Y.train , k = 20)
      pred.test<-ifelse(pred.test=="True",1,0)
    }
    for(a in 1:index){
        boot.pred[i,a]<-pred.test[a]
    }
  }
  out<-matrix(0,index,2)
  out[,1]<-apply(boot.pred,2,mean)
  out[,2]<-apply(boot.pred,2,sd)
  return(out)
}

boot.pred.valid<-as.data.frame(boot.fn("glm",best.mod,100,train.comp,valid,5))
boot.pred.valid<-cbind(boot.pred.valid,boot.pred.valid[,1]+
                         1.96*boot.pred.valid[,2],boot.pred.valid[,1]-1.96*boot.pred.valid[,2])
names(boot.pred.valid)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
boot.pred.valid

boot.pred.test<-as.data.frame(boot.fn("glm",best.mod,100,train.comp,test,5))
boot.pred.test<-cbind(boot.pred.test,boot.pred.test[,1]+1.96*boot.pred.test[,2],boot.pred.test[,1]-1.96*boo
t.pred.test[,2])
names(boot.pred.test)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
```

```
boot.pred.test

################################## vi

pred.test <- predict(best.mod, newdata = test,type = "response")
glm.pred<-rep("False",nrow(test))
glm.pred[pred.test>0.5] = "True"
submiss.d<-data.frame(test.raw$PassengerId,glm.pred)
names(submiss.d)<-c("PassengerId","Transported")
write.csv(submiss.d,file="submission d.csv",row.names=FALSE)
## Ranking: 844 Songze Yang
## Score: 0.78653


################################## LDA

################################## Package loading

library(dplyr)
##install.packages("mice")
library(mice)
library(leaps)
library(MASS)
library (boot)
library(class)

################################## data loading

setwd("E:/ANU Sem 2/STAT3040STAT7040 - Statistical Learning/Assignment 2")
train.comp <-
  read.csv(file="train_comp.csv",header=TRUE,stringsAsFactors=T)
valid<-
  read.csv(file="valid_comp.csv",header=TRUE,stringsAsFactors=T)
test<-
  read.csv(file="test_comp.csv",header=TRUE,stringsAsFactors=T)



##(e)

################################## i

full.mod<-as.formula(Transported~.)
null.mod<-as.formula(Transported~1)

cv.out<-method.fwd("lda",train.comp)

plot(cv.out[[1]][,1],type = "p",lwd = 2,cex = 1.2,
     ylab = "Cross Validation (Steps)",
     xlab = "Steps",
     main = "Forward Selection based on CV",
     ylim = c(0.21,0.28)
     )
points(which.min(cv.out[[1]][,1]),min(cv.out[[1]][,1]),
```

```
       pch = 18,col = "red",cex = 1.5)
points(1:(ncol(train.comp)-1),cv.out[[1]][,1]+cv.out[[1]][,2],
       pch = 25,col = "red",cex = 1)
points(1:(ncol(train.comp)-1),cv.out[[1]][,1]-cv.out[[1]][,2],
       pch = 24,col = "red",cex = 1)
mod.lowcv<-cv.out[[2]]
mod.final<-as.formula(Transported ~  CryoSleep + Side +
                        FoodCourt + VRDeck + RoomService +
                        Spa + HomePlanet + ShoppingMall)

################################## ii

best.mod<-lda(mod.final,data = train.comp)
pred.valid <- predict(best.mod, newdata = valid)
lda.class<-pred.valid$class
mean(valid$Transported != lda.class)
c.table<-table (lda.class,valid$Transported)
c.table
lda.pred.default<-c(mean(valid$Transported != lda.class),
            c.table[2,1]/(c.table[1,1]+c.table[2,1]),
            c.table[1,2]/(c.table[1,2]+c.table[2,2]))

lda.pred<-rep("False",nrow(valid))
lda.pred[pred.valid$posterior[, 2] > .8] = "True"
mean(valid$Transported != lda.pred)
c.table<-table(lda.pred,valid$Transported)
c.table
lda.pred.0.8<-c(mean(valid$Transported != lda.pred),
            c.table[2,1]/(c.table[1,1]+c.table[2,1]),
            c.table[1,2]/(c.table[1,2]+c.table[2,2]))

lda.pred<-rep("False",nrow(valid))
lda.pred[pred.valid$posterior[, 2] > .2] = "True"
mean(valid$Transported != lda.pred)
table(lda.pred,valid$Transported)
c.table
lda.pred.0.8<-c(mean(valid$Transported != lda.pred),
                c.table[2,1]/(c.table[1,1]+c.table[2,1]),
                c.table[1,2]/(c.table[1,2]+c.table[2,2]))
con.out<-rbind(lda.pred.default,lda.pred.0.8,lda.pred.0.8)
colnames(con.out) <-  c("Mean Error Rate","False postitve rate",
                        "False negative rate")
con.out

################################## v

boot.pred.valid<-as.data.frame(boot.fn("lda",mod.final,100,train.comp,valid,5))
boot.pred.valid<-cbind(boot.pred.valid,boot.pred.valid[,1]+1.96*boot.pred.valid[,2],boot.pred.valid[,1]-1.9
6*boot.pred.valid[,2])
names(boot.pred.valid)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
View(boot.pred.valid)

boot.pred.test<-as.data.frame(boot.fn("lda",mod.final,100,train.comp,test,5))
boot.pred.test<-cbind(boot.pred.test,boot.pred.test[,1]+1.96*boot.pred.test[,2],boot.pred.test[,1]-1.96*boo
```

```
t.pred.test[,2])
names(boot.pred.test)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
View(boot.pred.test)


################################## iv

pred.test <- predict(best.mod, newdata = test)$class
submiss.e<-data.frame(test.raw$PassengerId,pred.test)
names(submiss.e)<-c("PassengerId","Transported")
write.csv(submiss.e,file="submission e.csv",row.names=FALSE)
## Score: 0.78653


################################## QDA

################################## Package loading

library(dplyr)
##install.packages("mice")
library(mice)
library(leaps)
library(MASS)
library (boot)
library(class)

################################## data loading

setwd("E:/ANU Sem 2/STAT3040STAT7040 - Statistical Learning/Assignment 2")
train.comp <-
  read.csv(file="train_comp.csv",header=TRUE,stringsAsFactors=T)
valid<-
  read.csv(file="valid_comp.csv",header=TRUE,stringsAsFactors=T)
test<-
  read.csv(file="test_comp.csv",header=TRUE,stringsAsFactors=T)


##(f)

################################## i

full.mod<-as.formula(Transported~.)
null.mod<-as.formula(Transported~1)

cv.out<-method.fwd("qda",train.comp)

plot(cv.out[[1]][,1],type = "p",lwd = 2,cex = 1.2,
     ylab = "Cross Validation (Steps)",
     xlab = "Steps",
     main = "Forward Selection based on CV",
     ylim = c(0.21,0.30)
)
points(which.min(cv.out[[1]][,1]),min(cv.out[[1]][,1]),
       pch = 18,col = "red",cex = 1.5)
```

```
points(1:(ncol(train.comp)-1),cv.out[[1]][,1]+cv.out[[1]][,2],
       pch = 25,col = "red",cex = 1)
points(1:(ncol(train.comp)-1),cv.out[[1]][,1]-cv.out[[1]][,2],
       pch = 24,col = "red",cex = 1)
mod.lowcv<-cv.out[[2]]
mod.final<-as.formula(Transported ~  CryoSleep + Side + Age
                      + Spa + RoomService + VRDeck
                      + HomePlanet)

################################## ii

best.mod<-qda(mod.final,data = train.comp)
pred.valid <- predict(best.mod, newdata = valid)
qda.class<-pred.valid$class
mean(valid$Transported != qda.class)
table (qda.class,valid$Transported)
pred.valid$posterior[1:20, 1]
sum (pred.valid$posterior[, 2] > .9)


qda.pred<-rep("False",nrow(valid))
qda.pred[pred.valid$posterior[, 2] > .9] = "True"
mean(valid$Transported != qda.pred)
table(qda.pred,valid$Transported)


qda.pred<-rep("False",nrow(valid))
qda.pred[pred.valid$posterior[, 2] > .1] = "True"
mean(valid$Transported != qda.pred)
table(qda.pred,valid$Transported)

################################## v

boot.pred.valid<-as.data.frame(boot.fn("qda",mod.final,100,train.comp,valid,5))
boot.pred.valid<-cbind(boot.pred.valid,boot.pred.valid[,1]+1.96*boot.pred.valid[,2],boot.pred.valid[,1]-1.9
6*boot.pred.valid[,2])
names(boot.pred.valid)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
View(boot.pred.valid)

boot.pred.test<-as.data.frame(boot.fn("qda",mod.final,100,train.comp,test,5))
boot.pred.test<-cbind(boot.pred.test,boot.pred.test[,1]+1.96*boot.pred.test[,2],boot.pred.test[,1]-1.96*boo
t.pred.test[,2])
names(boot.pred.test)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
View(boot.pred.test)


################################## iv

pred.test <- predict(best.mod, newdata = test)$class
submiss.f<-data.frame(test.raw$PassengerId,pred.test)
names(submiss.f)<-c("PassengerId","Transported")
write.csv(submiss.f,file="submission f.csv",row.names=FALSE)
## Score: 0.77577
```

```
################################## KNN

################################## Package loading

library(dplyr)
##install.packages("mice")
library(mice)
library(leaps)
library(MASS)
library (boot)
library(class)
#install.packages("mlr3")
library(mlr)
library(mlr3)
#install.packages("mlr3viz")
library(mlr3viz)
#install.packages("GGally")
library(GGally)
#install.packages("kknn")
library(kknn)
#install.packages("mlr3verse")
library(mlr3verse)
#install.packages("FSelector")
library(FSelector)
library(mlr3fselect)



################################## data loading

setwd("E:/ANU Sem 2/STAT3040STAT7040 - Statistical Learning/Assignment 2")
train.comp <-
  read.csv(file="train_comp.csv",header=TRUE,stringsAsFactors=T)
valid<-
  read.csv(file="valid_comp.csv",header=TRUE,stringsAsFactors=T)
test<-
  read.csv(file="test_comp.csv",header=TRUE,stringsAsFactors=T)


##(g)

################################## i

shrink.data<-train.comp[,c(1,6,8:13)]

cv.out<-method.fwd("knn",shrink.data,10)
cv.out[[2]]

plot(cv.out[[1]][,1],type = "p",lwd = 2,cex = 1.2,
     ylab = "Cross Validation (Steps)",
     xlab = "Steps",
```

```
     main = "Forward Selection based on CV",
     ylim = c(0.21,0.46)
)
points(which.min(cv.out[[1]][,1]),min(cv.out[[1]][,1]),
       pch = 18,col = "red",cex = 1.5)
points(1:(ncol(shrink.data)-1),cv.out[[1]][,1]+cv.out[[1]][,2],
       pch = 25,col = "red",cex = 1)
points(1:(ncol(shrink.data)-1),cv.out[[1]][,1]-cv.out[[1]][,2],
       pch = 24,col = "red",cex = 1)
mod.lowcv<-cv.out[[2]]
mod.final<-as.formula(Transported ~  RoomService + Spa +
                        VRDeck + FoodCourt)

################################## Forward Selection machine learning

task = as_task_classif(train.comp, target = "Transported", positive = "True")
knn_learner <- lrn("classif.kknn")
resampling = rsmp("cv", folds = 10)
measure = msr("classif.ce")
resampling$instantiate(task)

terminator = trm("stagnation", iters = 50)
instance = FSelectInstanceSingleCrit$new(
  task = task,
  learner = knn_learner,
  resampling = resampling,
  measure = measure,
  terminator = terminator)

fselector = fs("sequential")
fselector$optimize(instance)
fselector$optimization_path(instance)
as.data.table(instance$archive, exclude_columns = c("runtime_learners", "timestamp", "batch_nr", "resample_
result", "uhash"))
##Only the Destination is FALSE var and is not significant in our case



################################## ii

X.train <- model.matrix(mod.final, data=shrink.data)
X.test <- model.matrix(mod.final, data=valid)
Y.train <- as.factor(shrink.data$Transported)

mod <- knn(X.train, X.test, Y.train , k = 20)
mean(valid$Transported != mod)
table (mod,valid$Transported)

mod <- knn(X.train, X.test, Y.train , k = 5)
mean(valid$Transported != mod)
table (mod,valid$Transported)

mod <- knn(X.train, X.test, Y.train , k = 80)
mean(valid$Transported != mod)
```

```
table (mod,valid$Transported)

################################# v

boot.pred.valid<-as.data.frame(boot.fn("knn",mod.final,100,shrink.data,valid,5))
boot.pred.valid<-cbind(boot.pred.valid,boot.pred.valid[,1]+1.96*boot.pred.valid[,2],boot.pred.valid[,1]-1.9
6*boot.pred.valid[,2])
names(boot.pred.valid)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
boot.pred.valid

boot.pred.test<-as.data.frame(boot.fn("knn",mod.final,100,shrink.data,test,5))
boot.pred.test<-cbind(boot.pred.test,boot.pred.test[,1]+1.96*boot.pred.test[,2],boot.pred.test[,1]-1.96*boo
t.pred.test[,2])
names(boot.pred.test)<-c("Est.prob","Std of prob","Upper 95% CI","Lower 95% CI")
boot.pred.test



################################# iv
X.train <- model.matrix(mod.final, data=shrink.data)
X.test <- model.matrix(mod.final, data=valid)
Y.train <- as.factor(shrink.data$Transported)
err.knn<-rep(0,100)
for(i in 1:100){
  mod <- knn(X.train, X.test, Y.train , k = i)
  err.knn[i]<-mean(valid$Transported != mod)
  }
plot(err.knn)
points(which.min(err.knn),min(err.knn),
       pch = 18,col = "red",cex = 1.5)
which.min(err.knn)

predictors<-as.formula(paste("~",all.vars(mod.final)[-1], collapse = "+"))
X.test <- model.matrix(predictors, data=test)
pred.test <- knn(X.train, X.test, Y.train , k = 61)
submiss.g<-data.frame(test.raw$PassengerId,pred.test)
names(submiss.g)<-c("PassengerId","Transported")
write.csv(submiss.g,file="submission g.csv",row.names=FALSE)
```