HW4

Vrunda Shah, Hansika Karkera

30/03/2020

#Promlem 1

```
library(readxl)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
u <- read_excel("UniversalBank.xlsx", sheet = "Data")
u$ID <- NULL
u$`ZIP Code`<- NULL
u[,c("Education1","Education2","Education3")]<-model.matrix(~Education-1,data = u)
u$Education<-NULL
u <- u[,c(1,2,3,4,5,12,13,14,6,8,9,10,11,7)]


indices= sample(nrow(u), 0.6*nrow(u))
train = u[indices, ]
val = u[-indices, ]
new_df <- data.frame(Age=40,Experience=10,Income=84,Family=2,CCAvg=2,Education2=1,Education3=0,Mortgage=
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
train <- as.data.frame(lapply(train, normalize))
val <- as.data.frame(lapply(val, normalize))

train_l <- train[,14]
val_l <- val[,14]

pred<-class::knn(train=train[,-14],test=val[,-14],cl=train_l,k=1)
head(pred)
```

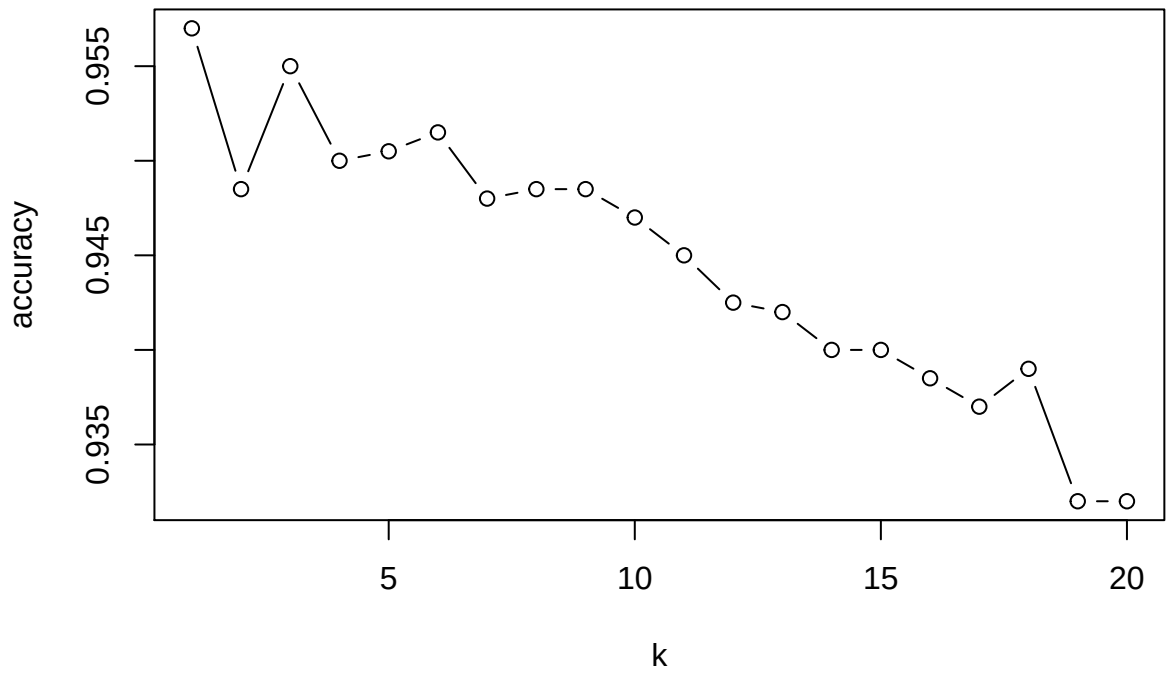
```
## [1] 0 0 0 0 1 0
## Levels: 0 1
```

Thus, from above we can say that customer will not take a personal loan.

```r
## TASK 1(B)
accuracy_df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))
for(i in 1:20) {
  knn.pred <- class::knn(train[,-14], val[,-14], cl = train_l, k=i)
  accuracy_df[i, 2] <- confusionMatrix(knn.pred, as.factor(val_l))$overall[1]
}
accuracy_df
```

```
##     k accuracy
## 1   1   0.9570
## 2   2   0.9485
## 3   3   0.9550
## 4   4   0.9500
## 5   5   0.9505
## 6   6   0.9515
## 7   7   0.9480
## 8   8   0.9485
## 9   9   0.9485
## 10 10   0.9470
## 11 11   0.9450
## 12 12   0.9425
## 13 13   0.9420
## 14 14   0.9400
## 15 15   0.9400
## 16 16   0.9385
## 17 17   0.9370
## 18 18   0.9390
## 19 19   0.9320
## 20 20   0.9320
```

```r
plot(accuracy_df,type='b')
```



Thus the best choice of k is 5 as balance between overfitting and ignoring the predictors

```
##TASK 1(C)
knn1 <- class::knn(train[,-14], val[,-14], cl = train_l, k=5)
table(val_l,knn1)
```

```
##      knn1
## val_l    0    1
##     0 1785   17
##     1   82  116
```

```
##TASK 1(D)
train$Education1<- NULL
knn2<- class::knn(train=train[,-13],test=new_df,cl=train_l,k=5)
knn2
```

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

```
## TASK 1(E)
#dividing the data
train_set<-sample(row.names(u),0.5*dim(u)[1])
valid_set<-sample(setdiff(row.names(u),train_set),0.3*dim(u)[1])
test_set<-setdiff(row.names(u),union(train_set,valid_set))
train_df<-u[train_set,]
```

```
valid_df<-u[valid_set,]
test_df<-u[test_set,]

train_df2 <- as.data.frame(lapply(train_df, normalize))
val_df2 <- as.data.frame(lapply(valid_df, normalize))
test_df2 <- as.data.frame(lapply(test_df, normalize))

train_l2 <- train_df2[,14]
val_l2 <- val_df2[,14]
test_l2 <- test_df2[,14]

#classification matrix for test data set
knn.test<-class::knn(train_df2[,-14], test_df2[,-14], cl = train_l2, k=5)
table(test_l2,knn.test)
```

```
##        knn.test
## test_l2   0   1
##       0 904   3
##       1  55  38
```

```
#classification matrix for training set
knn.train<-class::knn(train_df2[,-14], train_df2[,-14], cl = train_l2, k=5)
table(train_l2,knn.train)
```

```
##         knn.train
## train_l2    0    1
##        0 2260    6
##        1   69  165
```

```
#classification matrix for validation
knn.valid<-class::knn(train_df2[,-14], val_df2[,-14], cl = train_l2, k=5)
table(val_l2,knn.valid)
```

```
##       knn.valid
## val_l2    0    1
##      0 1344    3
##      1   81   72
```

From the above result error in test and valiodation set is higher than training set / # Problem 2

```
# Problem 2 (a)
library(readxl)
library(rsample)
```

```
## Loading required package: tidyr
```

```
housing <- read_excel("BostonHousing.xlsx",sheet="Data")
# Split into training and validation data
indices <- sample(nrow(housing), 0.6*nrow(housing))
train <- housing[indices,-14] #60% of the data
```

```
test <- housing[-indices,-14] #40% of the data
# Normlaize Data Function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
train_df <- as.data.frame(lapply(train, normalize))
val_df <- as.data.frame(lapply(test, normalize))
# Split Data
train_l <- train_df[,13]
val_l <- val_df[,13]
housing_predict1 <- class::knn(train = train_df[,-13], test = val_df[,-13],cl = train_l, k=1)
housing_predict2 <- class::knn(train = train_df[,-13], test = val_df[,-13],cl = train_l, k=2)
housing_predict3 <- class::knn(train = train_df[,-13], test = val_df[,-13],cl = train_l, k=3)
housing_predict4 <- class::knn(train = train_df[,-13], test = val_df[,-13],cl = train_l, k=4)
housing_predict5 <- class::knn(train = train_df[,-13], test = val_df[,-13],cl = train_l, k=5)

library(caret)
knn1 <-  100 * sum(val_l == housing_predict1)/NROW(val_l)
knn2 <-  100 * sum(val_l == housing_predict2)/NROW(val_l)
knn3 <-  100 * sum(val_l == housing_predict3)/NROW(val_l)
knn4 <-  100 * sum(val_l == housing_predict4)/NROW(val_l)
knn5 <-  100 * sum(val_l == housing_predict5)/NROW(val_l)
acc <- data.frame("k"=c(1,2,3,4,5),"Accuracy"=c(knn1,knn2,knn3,knn4,knn5))
acc
```

```
##   k  Accuracy
## 1 1 1.9704433
## 2 2 0.9852217
## 3 3 0.0000000
## 4 4 0.9852217
## 5 5 1.4778325
```

For the given model the best value for k is 1 which means the new data will be classified based on the majority vote of it's nearest neighbour.

Problem (b)

```
# Problem 2(b)
predict_df<-data.frame(0.2,0,7,0,0.538,6,62,4.7,4,307,21,10)
names(predict_df) <- names(train_df)[-13]
pred_values <- preProcess(predict_df, method=c("center", "scale"))
pred_norm <- predict(pred_values, newdata = predict_df)
knn_pred <- class::knn(train = train_df[,-13],
                       test = pred_norm,
                       cl = train_df$MEDV, k = 1)
knn_pred
```

```
## [1] 0.664414414414414
## 180 Levels: 0 0.0157657657657658 0.0315315315315315 ... 1
```

Problem 2 (C)

‘ The error in training data is 0 at k=1 because we have performed knn prediction based on training set

Problem 2(D)

Thevalidation data error overly optimistic compared to the error rate when applying this k-NN predictor to new data because we select value of k that performs best on validation set

Problem 2(E)

The difficulty usinh KNN for large training set is : Firsly, It taKES time to find nearest neighbour . Operations that algorithm : find the distnace of each new record from each record in training set. Find the minimum from all the distance obtained.

Problem 3

Problem 3(a)

```
library(readxl)
acc <- read_excel("Accidents.xlsx", sheet = "Data")
acc$injury <- "yes"
acc$injury[acc$MAX_SEV_IR == 0] <- "no"
yes_count <-length(acc$injury[acc$injury =="yes"])
total_injury <- length(acc$injury)
p_yes <- yes_count/total_injury
p_yes
```

```
## [1] 0.5087832
```

Using the information in this dataset, if an accident has just been reported and no further information is available, the prediction for injury would be “yes” because the probability of an having an injury irrespective of the conditions is greater than not having an injury.

Problem 3 (b)(ii)

```
# Select first 12 records
acc_df1 <- dplyr::select(acc,TRAF_CON_R,WEATHER_R,injury)
acc_df1 <- acc_df1[1:12,]
acc_df <- acc_df1[1:12,]

# P(injury=yes|TRAF_CON_R = 0,WEATHER_R = 1)
n <- nrow(acc_df[which(acc_df$injury=="yes" & acc_df$TRAF_CON_R ==0 & acc_df$WEATHER_R==1),])
d <- nrow(acc_df[which(acc_df$injury=="yes"),])
p1 <- n/d
p1
```

```
## [1] 0.6666667
```

```
# P(injury=yes|TRAF_CON_R = 0,WEATHER_R = 2)
n <- nrow(acc_df[which(acc_df$injury=="yes" & acc_df$TRAF_CON_R ==0 & acc_df$WEATHER_R==2),])
p2 <- n/d
p2
```

```
## [1] 0.3333333
```

```
# P(injury=yes|TRAF_CON_R = 1,WEATHER_R = 1)
n <- nrow(acc_df[which(acc_df$injury=="yes" & acc_df$TRAF_CON_R ==1 & acc_df$WEATHER_R==1),])
p3 <- n/d
p3
```

```
## [1] 0
```

```
# P(injury=yes|TRAF_CON_R = 1,WEATHER_R = 2)
n <- nrow(acc_df[which(acc_df$injury=="yes" & acc_df$TRAF_CON_R ==1 & acc_df$WEATHER_R==2),])
p4 <- n/d
p4
```

```
## [1] 0
```

```
# P(injury=yes|TRAF_CON_R = 2,WEATHER_R = 1)
n <- nrow(acc_df[which(acc_df$injury=="yes" & acc_df$TRAF_CON_R ==2 & acc_df$WEATHER_R==1),])
p5 <- n/d
p5
```

```
## [1] 0
```

```
# P(injury=yes|TRAF_CON_R = 2,WEATHER_R = 2)
n <- nrow(acc_df[which(acc_df$injury=="yes" & acc_df$TRAF_CON_R ==2 & acc_df$WEATHER_R==2),])
p6 <- n/d
p6
```

```
## [1] 0
```

Problem 3 (b)(iii)

```
# Problem 3 (b)(iii)
acc_df$Probability <- c(0.667,0.167,0,0,0.67,0.167,0.167,0.67,0.167,0.167,0.167,0)
acc_df$Predicted_Injury <- 0
acc_df$Predicted_Injury <- ifelse(acc_df$Probability >0.5,"yes","no")
```

Problem 3 (b)(iv)

P(INJURY = YES|WEATHER_R=1,TRAF_CON_R=1)= P(WEATHER_R=1|INJURY = YES)*P(WEATHER_R=1|INJURY = YES)*P(INJURY = YES)/ (P(WEATHER_R=1|INJURY = YES)*P(WEATHER_R=1|INJURY = YES)*P(INJURY = YES)+ P(WEATHER_R=1|INJURY = NO)*P(WEATHER_R=1|INJURY = NO)*P(INJURY = NO)) = 0

```
# Problem 3 (b)(v)
library(e1071)
library(klaR)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     housing
```

```
acc_df2 <- acc_df1
nb<-naiveBayes(injury ~ ., data = acc_df2)
predict(nb, newdata = acc_df2,type = "raw")
```

```
##                no          yes
##  [1,] 0.001916916 0.9980830837
##  [2,] 0.006129754 0.9938702459
##  [3,] 0.999548668 0.0004513316
##  [4,] 0.998552097 0.0014479028
##  [5,] 0.001916916 0.9980830837
##  [6,] 0.006129754 0.9938702459
##  [7,] 0.006129754 0.9938702459
##  [8,] 0.001916916 0.9980830837
##  [9,] 0.006129754 0.9938702459
## [10,] 0.006129754 0.9938702459
## [11,] 0.006129754 0.9938702459
## [12,] 0.989399428 0.0106005719
```

```
model <- train(acc_df2[,-3],acc_df2$injury,'nb', trControl = trainControl(method = 'cv',number=10))
ModelPred<-predict(model$finalModel,acc_df2[,-3])
table(ModelPred$class,acc_df2$injury)
```

```
##
##       no yes
##   no   9   3
##   yes  0   0
```

```
acc_df$Prob_NB<-ModelPred$class
acc_df
```

```
## # A tibble: 12 x 6
##    TRAF_CON_R WEATHER_R injury Probability Predicted_Injury Prob_NB
##         <dbl>     <dbl> <chr>        <dbl> <chr>            <fct>
##  1          0         1 yes          0.667 yes              no
##  2          0         2 no           0.167 no               no
##  3          1         2 no           0     no               no
##  4          1         1 no           0     no               no
##  5          0         1 no           0.67  yes              no
##  6          0         2 yes          0.167 no               no
##  7          0         2 no           0.167 no               no
##  8          0         1 yes          0.67  yes              no
##  9          0         2 no           0.167 no               no
## 10          0         2 no           0.167 no               no
## 11          0         2 no           0.167 no               no
## 12          2         1 no           0     no               no
```

The resulting classifications in this case are not equivalent.

Problem 3 (c)(i) Below are the predictors that will be included for analysis: HOUR_I_R,ALCOHOL_I,ALIGN_I,WRK_ZON

```
#Problem 3 (c)(ii)
acc_df3 <- as.data.frame(acc)
for (i in c(1:dim(acc_df3)[2])){
  acc_df3[,i] <- as.factor(acc_df3[,i])
}

split <- sample(c(1:dim(acc_df3)[1]), dim(acc_df3)[1]*0.6)
Train_df <- acc_df3[split,]
Valid_df <- acc_df3[-split,]
variables <- c("injury","HOUR_I_R","ALCHL_I","ALIGN_I","WRK_ZONE","WKDY_I_R","INT_HWY","LGTCON_I_R","SPI

nb_train <- naiveBayes(injury ~ ., data = Train_df[,variables])
confusionMatrix(Train_df$injury, predict(nb_train, Train_df[, variables]), positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5065 7340
##        yes 4278 8626
##
##                Accuracy : 0.541
##                  95% CI : (0.5348, 0.5471)
##     No Information Rate : 0.6308
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0771
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5403
##             Specificity : 0.5421
##          Pos Pred Value : 0.6685
##          Neg Pred Value : 0.4083
```

```
##              Prevalence : 0.6308
##          Detection Rate : 0.3408
##    Detection Prevalence : 0.5099
##       Balanced Accuracy : 0.5412
##
##        'Positive' Class : yes
##
```

```
# Overall error for training data
er_train <- 1-0.542
ep_train <- scales::percent(er_train,0.01)
paste("Overall Percentage Error for training set:",ep_train)
```

```
## [1] "Overall Percentage Error for training set: 45.80%"
```

```
# Problem 3 (c)(iii)
nb_valid <- naiveBayes(injury ~ ., data = Valid_df[,variables])
confusionMatrix(Valid_df$injury, predict(nb_valid, Valid_df[, variables]), positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  3200 5116
##        yes 2655 5903
##
##                Accuracy : 0.5395
##                  95% CI : (0.5319, 0.547)
##     No Information Rate : 0.653
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0749
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5357
##             Specificity : 0.5465
##          Pos Pred Value : 0.6898
##          Neg Pred Value : 0.3848
##              Prevalence : 0.6530
##          Detection Rate : 0.3498
##    Detection Prevalence : 0.5072
##       Balanced Accuracy : 0.5411
##
##        'Positive' Class : yes
##
```

```
# Error Rate for validation set
er_valid <- 1-0.541
er_valid
```

```
## [1] 0.459
```

```r
ep_valid=scales::percent(er_valid,0.01)
paste("Overall Percentage Error for validation set:",ep_valid)
```

```
## [1] "Overall Percentage Error for validation set: 45.90%"
```

```r
# Problem 3 (c)(iv)
i <- er_valid - er_train
paste("The percent improvement:",scales::percent(i,0.01))
```

```
## [1] "The percent improvement: 0.10%"
```

```r
# Problem 3 (c)(v)

# P(injury=no|SPD_LIM=50)
n <- nrow(Train_df[which(Train_df$SPD_LIM==50 & Train_df$injury=="no"),])
d <- nrow(Train_df[which(Train_df$injury=="no"),])
p4 <- n/d
p4
```

```
## [1] 0.03998388
```

The conditional probability of an accident leading to injury at a low speed is as low and not zero beacuse,in reality the chance of injury at such low speed is quite low.