ISEN 613

Assignment 7

The goal of the project is to predict what type of persons were more likely to survive? The features available are Name, Age, Gender, Fare Class, etc. Data dictionary is provided in the appendix. Data is partitioned into (1) ProjectTrain.csv, and (2) ProjectTest.csv. Use Train data to develop the model and report performance results on the Test dataset.

1) Develop Logistic Repression, LDA, QDA and KNN based survival prediction models using Pclass, Sex, Age, SibSp, Parch, and Embarked as predictor variables. Note that some of these variables may need to be case of categorical (factors in R). Also, Age has lot of missing values. The missing values may need to be imputed (e.g., mean) for using this variable. Try few values of k in KNN to determine suitable value for K. Compare and interpret True Positive (TP) and False Positive (FP) of the different models using test data.

Code:

```r
setwd("C:/Users/RPJ/Desktop/Notes/Data Analysis/R")

#-------------Loading data-------------------

Train.data=read.csv("ProjectTrain.csv",header=T)
Test.data=read.csv("ProjectTest.csv",header=T)
library(plyr)

library(MASS)
library(class)
library(caret)

#------------------Data Preparation-------------------

attach(Train.data)

#Replacing missing age values with mean age for training data. This is done s
ince there are not that many missing values (125 out of 624)
mean.age.train=mean(Age,na.rm=TRUE)
Train.data$Age[is.na(Train.data$Age)]=mean.age.train
Train.data$Age=lapply(Train.data$Age,round,2)

#Replacing missing age values with mean age for testing data (52 out of 267)
mean.age.test=mean(Test.data$Age,na.rm=TRUE)
Test.data$Age[is.na(Test.data$Age)]=mean.age.test
Test.data$Age=lapply(Test.data$Age,round,2)

#Filling missing Embarked values with mode for training data. This is a reaso
nable choice since there are only 2 missing values
```

```r
unique_embarked<-unique(Embarked)
mode_embarked=unique_embarked[which.max(tabulate(match(Embarked, unique_embar
ked)))]
mode_embarked
```

```
## [1] S
## Levels:  C Q S
```

```r
mode_embarked="S" #Since the mode obtained is S
Train.data$Embarked[Train.data$Embarked==""]="S"

#Changing data type to factor for training data
Train.data$Survived=factor(Train.data$Survived)
Train.data$Pclass=factor(Train.data$Pclass)
Train.data$Sex=factor(Train.data$Sex)
Train.data$Embarked=factor(Train.data$Embarked)
Train.data$Age=unlist(Train.data$Age)
str(Train.data)
```

```
## 'data.frame':    624 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 6 8 9 10 11 14 ...
##  $ Survived   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 2 2 2 1 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 3 2 3 3 ...
##  $ Name       : Factor w/ 624 levels "Abbing, Mr. Anthony",..: 81 141 255
197 395 441 299 412 513 21 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 1 1 1 2 ..
.
##  $ Age        : num  22 38 26 35 29.6 ...
##  $ SibSp      : int  1 1 0 1 0 3 0 1 1 1 ...
##  $ Parch      : int  0 0 0 0 0 1 2 0 1 5 ...
##  $ Ticket     : Factor w/ 516 levels "110152","110413",..: 391 447 507 34
211 294 259 105 466 252 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.46 ...
##  $ Cabin      : Factor w/ 109 levels "","A10","A14",..: 1 59 1 41 1 1 1 1
108 1 ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 1 3 3 2 3 3 1 3 3 ...
```

```r
attach(Train.data)

#Changing data type to factor for testing data
Test.data$Survived=factor(Test.data$Survived)
Test.data$Pclass=factor(Test.data$Pclass)
Test.data$Sex=factor(Test.data$Sex)
Test.data$Embarked=factor(Test.data$Embarked)
Test.data$Age=unlist(Test.data$Age)
str(Test.data)
```

```
## 'data.frame':    267 obs. of  12 variables:
##  $ PassengerId: int  450 212 363 498 543 230 232 475 825 753 ...
##  $ Survived   : Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 1 1 1 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 1 2 3 3 3 3 3 3 3 3 ...
##  $ Name       : Factor w/ 267 levels "Abelson, Mrs. Samuel (Hannah Wizosky
)",..: 200 34 18 223 8 136 133 232 190 248 ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 2 1 1 2 1 2 2 ..
.
##  $ Age        : num  52 35 45 29.9 11 ...
##  $ SibSp      : int  0 0 0 0 4 3 0 0 4 0 ...
##  $ Parch      : int  0 0 1 0 2 1 0 0 1 0 ...
##  $ Ticket     : Factor w/ 232 levels "110152","110413",..: 16 194 71 191 1
14 170 109 175 86 106 ...
##  $ Fare       : num  30.5 21 14.5 15.1 31.3 ...
##  $ Cabin      : Factor w/ 59 levels "","A24","A32",..: 18 1 1 1 1 1 1 1 1
1 ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 3 1 3 3 3 3 3 3 3 ...

#------------------------Logistic regression--------------------------------
logistic.fit=glm(Survived~Pclass+Age+Sex+SibSp+Parch+Embarked,data=Train.data
,family='binomial')
summary(logistic.fit)

##
## Call:
## glm(formula = Survived ~ Pclass + Age + Sex + SibSp + Parch +
##     Embarked, family = "binomial", data = Train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5753  -0.5879  -0.4036   0.6158   2.4139
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.066936   0.497422   8.176 2.93e-16 ***
## Pclass2     -1.094381   0.322242  -3.396 0.000683 ***
## Pclass3     -2.433505   0.307497  -7.914 2.49e-15 ***
## Age         -0.032762   0.009291  -3.526 0.000421 ***
## Sexmale     -2.717339   0.239800 -11.332  < 2e-16 ***
## SibSp       -0.343770   0.132037  -2.604 0.009225 **
## Parch       -0.039605   0.152235  -0.260 0.794743
## EmbarkedQ    0.097034   0.444679   0.218 0.827264
## EmbarkedS   -0.299407   0.278653  -1.074 0.282608
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 835.17  on 623  degrees of freedom
## Residual deviance: 550.88  on 615  degrees of freedom
## AIC: 568.88
##
## Number of Fisher Scoring iterations: 5
```

```
logistic.probability=predict(logistic.fit,Test.data,type="response")

logistic.predict=rep("0",267)
logistic.predict[logistic.probability>0.5]="1"
attach(Test.data)

confusion.logistic=table(logistic.predict,Survived) #confusion matrix
confusion.logistic

##                  Survived
## logistic.predict   0   1
##                0 144  29
##                1  25  69

Accuracy.logistic=mean(logistic.predict==Survived)
Accuracy.logistic

## [1] 0.7977528

tp.logistic=confusion.logistic[2,2]/(confusion.logistic[1,2]+confusion.logist
ic[2,2])
tp.logistic #true positive rate

## [1] 0.7040816

fp.logistic=confusion.logistic[2,1]/(confusion.logistic[1,1]+confusion.logist
ic[2,1])
fp.logistic #false positive rate

## [1] 0.147929

#---------------------Linear Discriminant Analysis-------------------------

lda.fit=lda(Survived~Pclass+Age+Sex+SibSp+Parch+Embarked,data=Train.data)
lda.predict=predict(lda.fit,Test.data)

lda.class=lda.predict$class
attach(Test.data)

confusion.lda=table(lda.class,Survived)

confusion.lda

##          Survived
## lda.class   0   1
##         0 146  30
##         1  23  68

Accuracy.lda=mean(lda.class==Survived)
Accuracy.lda

## [1] 0.8014981
```

```r
tp.lda=confusion.lda[2,2]/(confusion.lda[1,2]+confusion.lda[2,2])
tp.lda

## [1] 0.6938776

fp.lda=confusion.lda[2,1]/(confusion.lda[1,1]+confusion.lda[2,1])
fp.lda

## [1] 0.1360947

# -----------------------Quadratic Discriminant Analysis---------------------

qda.fit=qda(Survived~Pclass+Age+Sex+SibSp+Parch+Embarked,data=Train.data)
qda.predict=predict(qda.fit,Test.data)

qda.class=qda.predict$class
attach(Test.data)

confusion.qda=table(qda.class,Survived)

confusion.qda

##          Survived
## qda.class   0   1
##         0 130  20
##         1  39  78

Accuracy.qda=mean(qda.class==Survived)
Accuracy.qda

## [1] 0.7790262

tp.qda=confusion.qda[2,2]/(confusion.qda[1,2]+confusion.qda[2,2])
tp.qda

## [1] 0.7959184

fp.qda=confusion.qda[2,1]/(confusion.qda[1,1]+confusion.qda[2,1])
fp.qda

## [1] 0.2307692

# ---------------------------------KNN--------------------------------

library(psych)

attach(Train.data)

train.X=data.frame(Pclass,Age,Sex,SibSp,Parch,Embarked)
train.Y=Train.data$Survived
attach(Test.data)

test.X=data.frame(Pclass,Age,Sex,SibSp,Parch,Embarked)
test.Y=Test.data$Survived
```

```
attach(train.X)

# Dummy variable encoding for training data
train.X$Sex=dummy.code(train.X$Sex)
train.X$Pclass=as.data.frame(dummy.code(train.X$Pclass))
train.X$Embarked=as.data.frame(dummy.code(train.X$Embarked))
str(train.X)

attach(test.X)

# Dummy variable encoding for testing data
test.X$Sex=dummy.code(test.X$Sex)
test.X$Pclass=as.data.frame(dummy.code(test.X$Pclass))
test.X$Embarked=as.data.frame(dummy.code(test.X$Embarked))


# 5 different values of K are used to compare model accuracy
#K=1
set.seed(5)
knn.predict=knn(train.X,test.X,train.Y,k=1)

confusion.knn=table(knn.predict,test.Y)
confusion.knn

##            test.Y
## knn.predict   0   1
##           0 140  33
##           1  29  65

mean.knn=mean(knn.predict==Test.data$Survived)
Accuracy.knn=mean(knn.predict==Survived)
Accuracy.knn

## [1] 0.7677903

tp.knn=confusion.knn[2,2]/(confusion.knn[1,2]+confusion.knn[2,2])
tp.knn

## [1] 0.6632653

fp.knn=confusion.knn[2,1]/(confusion.knn[1,1]+confusion.knn[2,1])
fp.knn

## [1] 0.1715976

#K=5
set.seed(5)
knn.predict=knn(train.X,test.X,train.Y,k=5)

confusion.knn=table(knn.predict,test.Y)
confusion.knn
```

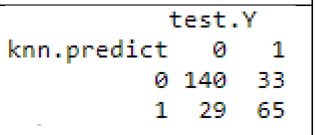
```
##             test.Y
## knn.predict   0   1
##           0 150  32
##           1  19  66

mean.knn=mean(knn.predict==Test.data$Survived)
Accuracy.knn=mean(knn.predict==Survived)
Accuracy.knn

## [1] 0.8089888

tp.knn=confusion.knn[2,2]/(confusion.knn[1,2]+confusion.knn[2,2])
tp.knn

## [1] 0.6734694

fp.knn=confusion.knn[2,1]/(confusion.knn[1,1]+confusion.knn[2,1])
fp.knn

## [1] 0.112426

#K=10
set.seed(5)
knn.predict=knn(train.X,test.X,train.Y,k=10)

confusion.knn=table(knn.predict,test.Y)
confusion.knn

##             test.Y
## knn.predict   0   1
##           0 152  41
##           1  17  57

mean.knn=mean(knn.predict==Test.data$Survived)
Accuracy.knn=mean(knn.predict==Survived)
Accuracy.knn

## [1] 0.7827715

tp.knn=confusion.knn[2,2]/(confusion.knn[1,2]+confusion.knn[2,2])
tp.knn

## [1] 0.5816327

fp.knn=confusion.knn[2,1]/(confusion.knn[1,1]+confusion.knn[2,1])
fp.knn

## [1] 0.1005917

#K=15
set.seed(5)
knn.predict=knn(train.X,test.X,train.Y,k=15)
```

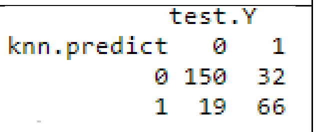
```
confusion.knn=table(knn.predict,test.Y)
confusion.knn

##             test.Y
## knn.predict   0   1
##           0 151  47
##           1  18  51

mean.knn=mean(knn.predict==Test.data$Survived)
Accuracy.knn=mean(knn.predict==Survived)
Accuracy.knn

## [1] 0.7565543

tp.knn=confusion.knn[2,2]/(confusion.knn[1,2]+confusion.knn[2,2])
tp.knn

## [1] 0.5204082

fp.knn=confusion.knn[2,1]/(confusion.knn[1,1]+confusion.knn[2,1])
fp.knn

## [1] 0.1065089
```

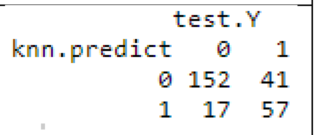
Explanation:

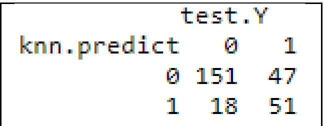
For KNN, 4 different values of k were selected (1,5,10 and 15) and parameters were calculated for it, which are shown in the table below:

K value	Accuracy	True Positive rate	False Positive rate	Confusion matrix

1	76.8%	66.3%	17.2%	 `            test.Y`
				`knn.predict   0   1`
				`          0 140  33`
				`          1  29  65`

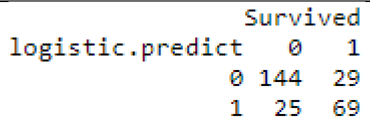
5	80.9%	67.3%	11.2%	 `            test.Y`
				`knn.predict   0   1`
				`          0 150  32`
				`          1  19  66`

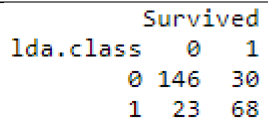
10	78.3%	58.2%	10.1%	 `            test.Y`
				`knn.predict   0   1`
				`          0 152  41`
				`          1  17  57`

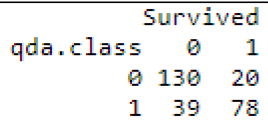
15	75.6%	52%	10.6%	 `          test.Y`
				`knn.predict   0   1`
				`          0 151  47`
				`          1  18  51`

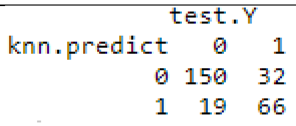
From the table, it can be seen that K=5 gives the highest accuracy, highest True Positive Rate (TPR) and considerably low False Positive Rate (FPR) and so, it is a good selection for the model.

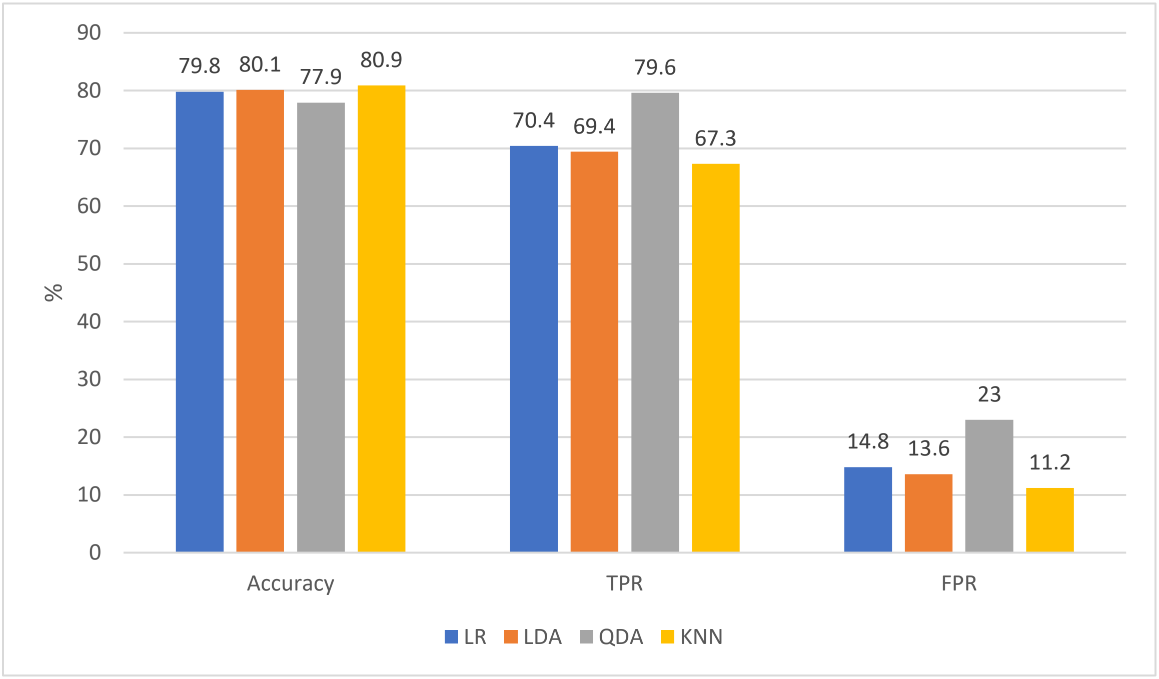
Method	**Accuracy**	**True Positive rate**	**False Positive rate**	**Confusion matrix**

Logistic Regression	79.8%	70.4%	14.8%	 `                Survived`
				`logistic.predict   0   1`
				`               0 144  29`
				`               1  25  69`

Linear Discriminant Analysis	80.1%	69.4%	13.6%	 `          Survived`
				`lda.class   0   1`
				`        0 146  30`
				`        1  23  68`

Quadratic Discriminant Analysis	77.9%	79.6%	23%	 `          Survived`
				`qda.class   0   1`
				`        0 130  20`
				`        1  39  78`

KNN (K=5)	80.9%	67.3%	11.2%	 `          test.Y`
				`knn.predict   0   1`
				`          0 150  32`
				`          1  19  66`



Since the number of datapoints are not equal for survived and not survived, accuracy would not be an appropriate metric to compare different models. Therefore, true positive rate (TPR) and false positive rate (FPR) are used. Ideally, TPR should be maximum and FPR should be minimum. From the above graph, it is seen that the TPR is maximum for QDA but its FPR is also the highest. Therefore, it is not the best choice. For KNN, the FPR is the lowest, but its TPR is also the lowest. For LDA and LR, it is seen that the performance is nearly similar, implying that the data does not follow Gaussian distribution, in which case the LDA would have given better results.

For TPR, QDA rightly predicts 79.6% survival out of the total number of passengers that survived. Interpreting values of FPR, QDA has the highest FPR (23%) which means that out of all the people that died, QDA wrongly predicts that 23% of them survived. FPR should be minimum, as it does not capture the seriousness of the disaster (It says that people survived, when they, in reality did not survive).

2) “Cabin” has sparse data content. One approach to handle the missing data is to have a special value “Not Available” for all the missing values. For the Logistic Regression model, evaluate performance improvement with and without including the cabin feature using test data.

Code:

```
# ---------------With Cabin data - Logistic Regression-----------------------
```

```r
#Cabin data categorization-Here, the grouping is done based on cabin alphabet
. For eg, A1 to A30 are grouped into A and so on

#Training data
Train.data$Cabin=as.character(Train.data$Cabin)
Train.data$Cabin[Train.data$Cabin==""]='X'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='A']='A'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='B']='B'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='C']='C'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='D']='D'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='E']='E'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='F']='F'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='G']='G'
Train.data$Cabin[substr(Train.data$Cabin, 1, 1)=='T']='T'
Train.data$Cabin=as.factor(Train.data$Cabin)

#Test data
Test.data$Cabin=as.character(Test.data$Cabin)
Test.data$Cabin[Test.data$Cabin==""]='X'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='A']='A'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='B']='B'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='C']='C'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='D']='D'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='E']='E'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='F']='F'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='G']='G'
Test.data$Cabin[substr(Test.data$Cabin, 1, 1)=='T']='T'
Test.data$Cabin=as.factor(Test.data$Cabin)

#logistic regression including cabin data
lfit.cabin=glm(Survived~Pclass+Age+Sex+SibSp+Parch+Embarked+Cabin,data=Train.
data,family='binomial')
summary(lfit.cabin)

##
## Call:
## glm(formula = Survived ~ Pclass + Age + Sex + SibSp + Parch +
##     Embarked + Cabin, family = "binomial", data = Train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4989  -0.5766  -0.3877   0.6252   2.4456
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.191832   0.839339   4.994 5.91e-07 ***
## Pclass2     -0.446378   0.472945  -0.944 0.345258
## Pclass3     -1.724529   0.464571  -3.712 0.000206 ***
## Age         -0.035122   0.009756  -3.600 0.000318 ***
## Sexmale     -2.711557   0.245309 -11.054  < 2e-16 ***
## SibSp        -0.346998   0.135816  -2.555 0.010621 *
## Parch        -0.058231   0.156577  -0.372 0.709967
## EmbarkedQ     0.073313   0.449761   0.163 0.870516
## EmbarkedS    -0.345842   0.288199  -1.200 0.230135
## CabinB        0.329149   0.820743   0.401 0.688392
## CabinC       -0.235187   0.781492  -0.301 0.763455
## CabinD        0.740830   0.913747   0.811 0.417503
## CabinE        0.955627   0.887081   1.077 0.281359
## CabinF        0.517899   1.126989   0.460 0.645845
## CabinG       -0.864630   1.509269  -0.573 0.566726
## CabinT      -13.120009 535.411595  -0.025 0.980450
## CabinX       -0.768359   0.779861  -0.985 0.324501
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 835.17  on 623  degrees of freedom
## Residual deviance: 537.22  on 607  degrees of freedom
## AIC: 571.22
##
## Number of Fisher Scoring iterations: 12
```

```
lprobability.cabin=predict(lfit.cabin,Test.data,type="response")

lpredict.cabin=rep("0",267)
lpredict.cabin[lprobability.cabin>0.5]="1"
attach(Test.data)

confusion.cabin=table(lpredict.cabin,Survived)
confusion.cabin
```

```
##                 Survived
## lpredict.cabin   0   1
##              0 147  24
##              1  22  74
```

```
Accuracy.cabin=mean(lpredict.cabin==Survived)

Accuracy.cabin
```

```
## [1] 0.8277154
```

```
tp.cabin=confusion.cabin[2,2]/(confusion.cabin[1,2]+confusion.cabin[2,2])
tp.cabin
```

```
## [1] 0.755102
```

```
fp.cabin=confusion.cabin[2,1]/(confusion.cabin[1,1]+confusion.cabin[2,1])
fp.cabin
```

```
## [1] 0.1301775
```

Explanation:

Since there were a lot of missing values in Cabin data, a new factor X was created to fill these missing values first. Then the cabin numbers were categorized based on their alphabetical index. For e.g., cabin B54 and B67 into B, G 2 and G98 into G and so on. The results of logistic regression after including this data are shown below.

Method	Accuracy	True Positive rate	False Positive rate	Confusion matrix

LR without cabin data	79.8%	70.4%	14.8%	`Survived`
				`logistic.predict   0   1`
				`0 144  29`
				`1  25  69`

LR with cabin data	82.7%	75.5%	13%	`Survived`
				`lpredict.cabin   0   1`
				`0 147  24`
				`1  22  74`

It can be seen that accuracy and TPR of the model with cabin data is higher. Also, the FPR is lower. This implies that cabin is an important predictor in predicting the survival .

3) Like linear regression, Logistic regression (LR) has the advantage of interpretability. Research the concepts of “Unadjusted Odds Ratio” and “Adjusted Odds Ratio”. Determine the adjusted odds ratio for Sex, Pclass, and Embarked using LR. Interpret the results.

Code:

```
#-------------------------Adjusted Odds ratio------------------------------

logistic2.fit=glm(Survived~Pclass+Sex+Embarked,data=Train.data,family='binomi
al')
summary(logistic2.fit)

##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Embarked, family = "binomial",
##     data = Train.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3367  -0.7403  -0.4132   0.6404   2.2375
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.66260    0.31576   8.432  < 2e-16 ***
## Pclass2     -0.78825    0.30438  -2.590  0.00961 **
## Pclass3     -2.05165    0.27413  -7.484  7.2e-14 ***
## Sexmale     -2.63466    0.22388 -11.768  < 2e-16 ***
## EmbarkedQ    0.02365    0.42574   0.056  0.95571
## EmbarkedS   -0.39419    0.27101  -1.455  0.14579
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 835.17  on 623  degrees of freedom
## Residual deviance: 570.75  on 618  degrees of freedom
## AIC: 582.75
##
## Number of Fisher Scoring iterations: 4
```

```
aor=exp(logistic2.fit$coefficients)
round(aor,3)
```

```
## (Intercept)     Pclass2     Pclass3     Sexmale   EmbarkedQ   EmbarkedS
##      14.333       0.455       0.129       0.072       1.024       0.674
```

Explanation:

From the values of adjusted odds ratio, following observations can be made:

- Out of all Pclasses, the odds of survival are highest in Pclass1. People in Pclass3 have the least odd of survival, 0.129 to 1, compared with people with Pclass1 tickets.
- It is also seen that males have lesser chance of survival, as compared to females. Odds of survival of a male onboard are 0.072 to that of a female.
- Interestingly, it is seen that passengers who boarded from Southampton have lesser odds of survival. Odds of survival of passengers from Queenstown are marginally higher (1.024 to 1) when compared to odds of survival from Cherbourg passengers.

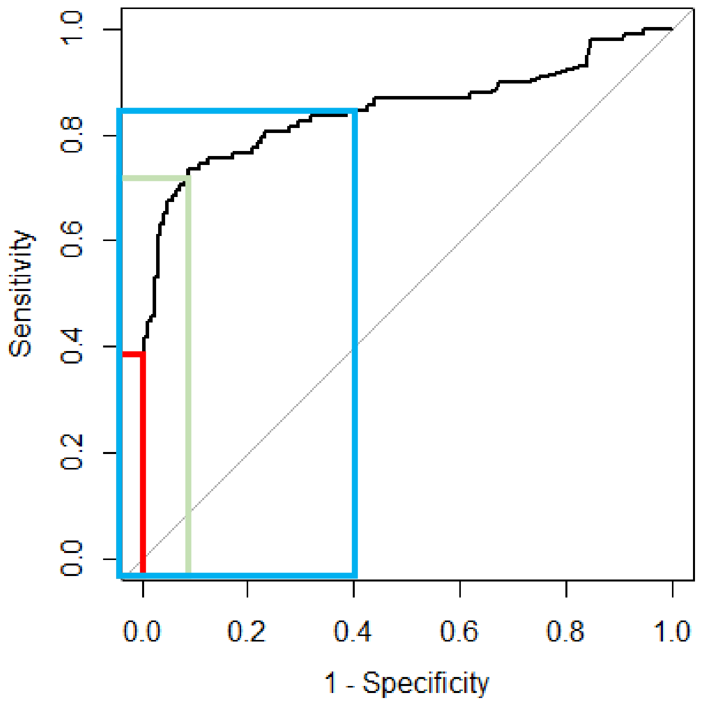
4) The default threshold to classify an entity to a class is 0.5. For the LR models, vary the threshold to 0.8, 0.5, and 0.2. Which threshold value do you think is appropriate for survival prediction? Why? Justify your answer with respect to misclassification rate on test data

Code:

```
#----------------------------Threshold=0.8---------------------------------

lpredict.cabin.h=rep("0",267)
lpredict.cabin.h[lprobability.cabin>0.8]="1"
attach(Test.data)

confusion.cabin.h=table(lpredict.cabin.h,Survived)
confusion.cabin.h
```

```
##                  Survived
## lpredict.cabin.h   0   1
##                0 168  59
##                1   1  39

Accuracy.cabin.h=mean(lpredict.cabin.h==Survived)
Accuracy.cabin.h

## [1] 0.7752809

tp.cabin.h=confusion.cabin.h[2,2]/(confusion.cabin.h[1,2]+confusion.cabin.h[2
,2])
tp.cabin.h

## [1] 0.3979592

fp.cabin.h=confusion.cabin.h[2,1]/(confusion.cabin.h[1,1]+confusion.cabin.h[2
,1])
fp.cabin.h

## [1] 0.00591716

#-----------------------------Threshold=0.2---------------------------------

lpredict.cabin.l=rep("0",267)
lpredict.cabin.l[lprobability.cabin>0.2]="1"
attach(Test.data)

confusion.cabin.l=table(lpredict.cabin.l,Survived)
confusion.cabin.l

##                  Survived
## lpredict.cabin.l   0   1
##                0 100  15
##                1  69  83

Accuracy.cabin.l=mean(lpredict.cabin.l==Survived)
Accuracy.cabin.l

## [1] 0.6853933

tp.cabin.l=confusion.cabin.l[2,2]/(confusion.cabin.l[1,2]+confusion.cabin.l[2
,2])
tp.cabin.l

## [1] 0.8469388

fp.cabin.l=confusion.cabin.l[2,1]/(confusion.cabin.l[1,1]+confusion.cabin.l[2
,1])
fp.cabin.l

## [1] 0.408284

library(pROC)
```

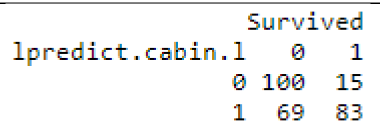
```
par(pty="s")
roc(Test.data$Survived,lprobability.cabin,plot=TRUE,ci=TRUE,legacy.axes=TRUE)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

##
## Call:
## roc.default(response = Test.data$Survived, predictor = lprobability.cabin,
ci = TRUE, plot = TRUE, legacy.axes = TRUE)
##
## Data: lprobability.cabin in 169 controls (Test.data$Survived 0) < 98 cases
(Test.data$Survived 1).
## Area under the curve: 0.8466
## 95% CI: 0.7903-0.9029 (DeLong)
```

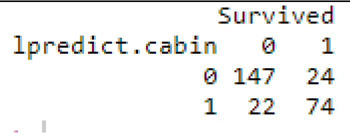


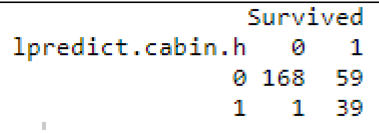
Explanation:

The ROC graph is plotted for the logistic regression model. The three threshold values are shown in the plot as well.

Threshold value	Accuracy	Misclassification rate	TPR	FPR	Confusion matrix

0.2 (blue)	68.5%	31.5%	84.7%	40.8%	 `Survived`
					`lpredict.cabin.l   0   1`
					`0 100  15`
					`1  69  83`

0.5 (green)	82.7%	17.3%	75.5%	13%	 `Survived`
					`lpredict.cabin   0   1`
					`0 147  24`
					`1  22  74`

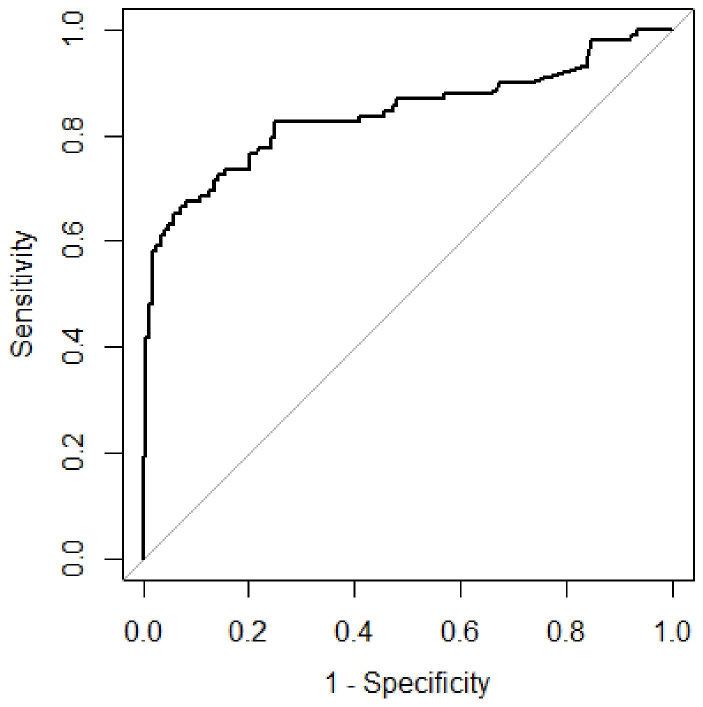
0.8 (red)	77.5%	22.5%	39.8%	0.6%	 `Survived`
					`lpredict.cabin.h   0   1`
					`0 168  59`
					`1   1  39`

From the table, it is found that misclassification rate is the highest for 0.2 threshold. This is because the false positives are too high (FPR is 40.8%). By increasing the threshold value, the FPR decreases, but at the same time, TPR also decreases (as seen from 0.8 threshold value). Therefore, threshold value of 0.5 is the ideal choice in this case, as it has less misclassification rate (and therefore high accuracy), as wells as not too many false positives.

5) Develop ROC plot for the LDA model.

Code:

```
#---------------------------ROC curve LDA------------------------------------

library(pROC)

par(pty="s")
roc(Test.data$Survived,lda.predict$posterior[,2],plot=TRUE,ci=TRUE,legacy.axe
s=TRUE)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



```
##
## Call:
## roc.default(response = Test.data$Survived, predictor = lda.predict$posteri
or[,     2], ci = TRUE, plot = TRUE, legacy.axes = TRUE)
##
## Data: lda.predict$posterior[, 2] in 169 controls (Test.data$Survived 0) <
98 cases (Test.data$Survived 1).
## Area under the curve: 0.8387
## 95% CI: 0.7819-0.8956 (DeLong)
```

Explanation:

Thus, the area under the curve is 83.87%, which is reasonably good value

6) What features do you think are important to make the prediction? Why? Evaluate the KNN model performance by including just the important features

Explanation:

Features important to make the survival prediction are Age, PClass, Sex and SibSp. This is because these factors had a p value <0.05 in the logistic regression, so they are statistically significant.

Code:

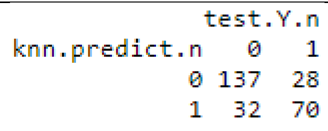
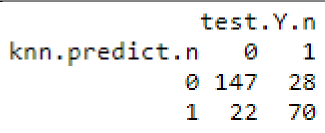
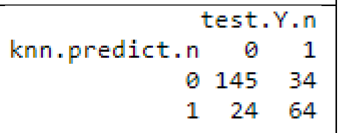
```
#------------------------------- KNN_updated---------------------------------
```

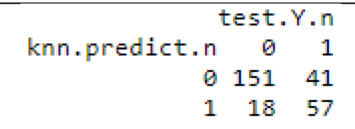
```r
#Here, only those predictors that were found to be statistically significant
in LR have been considered.
attach(Train.data)

train.X.n=data.frame(Pclass,Age,Sex,SibSp)
train.Y.n=Train.data$Survived
attach(Test.data)

test.X.n=data.frame(Pclass,Age,Sex,SibSp)
test.Y.n=Test.data$Survived

attach(train.X.n)

# Dummy variable
train.X.n$Sex=dummy.code(train.X.n$Sex)
train.X.n$Pclass=as.data.frame(dummy.code(train.X.n$Pclass))


attach(test.X.n)

# Dummy variable
test.X.n$Sex=dummy.code(test.X.n$Sex)
test.X.n$Pclass=as.data.frame(dummy.code(test.X.n$Pclass))

#K=1
set.seed(5)
knn.predict.n=knn(train.X.n,test.X.n,train.Y.n,k=1)

confusion.knn.n=table(knn.predict.n,test.Y.n)
confusion.knn.n

##                test.Y.n
## knn.predict.n   0   1
##               0 137  28
##               1  32  70

Accuracy.knn.n=mean(knn.predict.n==Survived)
Accuracy.knn.n

## [1] 0.7752809

tp.knn.n=confusion.knn.n[2,2]/(confusion.knn.n[1,2]+confusion.knn.n[2,2])
tp.knn.n

## [1] 0.7142857

fp.knn.n=confusion.knn.n[2,1]/(confusion.knn.n[1,1]+confusion.knn.n[2,1])
fp.knn.n

## [1] 0.1893491

#K=5
set.seed(5)
```

```r
knn.predict.n=knn(train.X.n,test.X.n,train.Y.n,k=5)

confusion.knn.n=table(knn.predict.n,test.Y.n)
confusion.knn.n

##               test.Y.n
## knn.predict.n   0   1
##             0 147  28
##             1  22  70

Accuracy.knn.n=mean(knn.predict.n==Survived)
Accuracy.knn.n

## [1] 0.8127341

tp.knn.n=confusion.knn.n[2,2]/(confusion.knn.n[1,2]+confusion.knn.n[2,2])
tp.knn.n

## [1] 0.7142857

fp.knn.n=confusion.knn.n[2,1]/(confusion.knn.n[1,1]+confusion.knn.n[2,1])
fp.knn.n

## [1] 0.1301775

#K=10
set.seed(5)
knn.predict.n=knn(train.X.n,test.X.n,train.Y.n,k=10)

confusion.knn.n=table(knn.predict.n,test.Y.n)
confusion.knn.n

##               test.Y.n
## knn.predict.n   0   1
##             0 145  34
##             1  24  64

Accuracy.knn.n=mean(knn.predict.n==Survived)
Accuracy.knn.n

## [1] 0.7827715

tp.knn.n=confusion.knn.n[2,2]/(confusion.knn.n[1,2]+confusion.knn.n[2,2])
tp.knn.n

## [1] 0.6530612

fp.knn.n=confusion.knn.n[2,1]/(confusion.knn.n[1,1]+confusion.knn.n[2,1])
fp.knn.n

## [1] 0.1420118
```

```
#K=15
set.seed(5)
knn.predict.n=knn(train.X.n,test.X.n,train.Y.n,k=15)

confusion.knn.n=table(knn.predict.n,test.Y.n)
confusion.knn.n

##                test.Y.n
## knn.predict.n   0   1
##              0 151  41
##              1  18  57

Accuracy.knn.n=mean(knn.predict.n==Survived)
Accuracy.knn.n

## [1] 0.7790262

tp.knn.n=confusion.knn.n[2,2]/(confusion.knn.n[1,2]+confusion.knn.n[2,2])
tp.knn.n

## [1] 0.5816327

fp.knn.n=confusion.knn.n[2,1]/(confusion.knn.n[1,1]+confusion.knn.n[2,1])
fp.knn.n

## [1] 0.1065089
```

The KNN model with only essential predictors is run with 5 different values of K, as before, and the results are shown in the below table:

K value	**Accuracy**	**True Positive rate**	**False Positive rate**	**Confusion matrix**
1	77.5%	71.4%	18.9%	 `test.Y.n` `knn.predict.n 0 1` `0 137 28` `1 32 70`
5	81.3%	71.4%	13%	 `test.Y.n` `knn.predict.n 0 1` `0 147 28` `1 22 70`
10	78.3%	65.3%	14.2%	 `test.Y.n` `knn.predict.n 0 1` `0 145 34` `1 24 64`

15	77.9%	58.2%	10.6%	

```
                test.Y.n
knn.predict.n     0    1
            0   151   41
            1    18   57
```

The KNN models with multiple predictors and significant predictors only are compared below:

K value	**Accuracy**		**True Positive rate**		**False Positive rate**	
	Multiple predictors	**Significant predictors**	**Multiple predictors**	**Significant predictors**	**Multiple predictors**	**Significant predictors**
1	76.8%	77.5%	66.3%	71.4%	17.2%	18.9%
5	80.9%	81.3%	67.3%	71.4%	11.2%	13%
10	78.3%	78.3%	58.2%	65.3%	10.1%	14.2%
15	75.6%	77.9%	52%	58.2%	10.6%	10.6%

It is observed that by including significant predictors only, the accuracy of model increased. The TPR also increased significantly (4 to 6%). There was also an increase in FPR (0 to 4%). It is to be noted that K=5 is the optimal choice for K among all the above values, since there was significant increase in TPR and accuracy and only minimal increase in FPR.