1(a)

When $0.5 - 0.3x_{i1} - 2.7x_{i2} + 3.9x_{i3} \geq log\frac{0.7}{1-0.7}$, that is to say when $0.5 - 0.3x_{i1} - 2.7x_{i2} + 3.9x_{i3} \geq 0.8472979$, the model will predict $Y_i = 1$.

If $X_i = (3,1,1)$, $0.5 - 0.3x_{i1} - 2.7x_{i2} + 3.9x_{i3} = 0.5 - 0.3 * 3 - 2.7 * 1 + 3.9 * 1 = 0.8$, which is smaller than 0.8472979, so the model will predict $Y_i = 0$.

If t is smaller, false-positive rate will increase and false-negative rate will decrease.

1(b)

For $Prob\left[Y = k \middle| \hat{Y} = k\right] = p_k$, the 0-1 loss for $\hat{Y} = k$ (l_k), is $1 - p_k$.

k	0	1	2	3	4
l_k	0.8	0.65	0.85	0.8	0.9

- **#2(a)**

```
> library(readr)
> wine <- read_csv("wine.csv")
Parsed with column specification:
cols(
  fixed.acidity = col_double(),
  volatile.acidity = col_double(),
  citric.acid = col_double(),
  residual.sugar = col_double(),
  chlorides = col_double(),
  free.sulfur.dioxide = col_double(),
  total.sulfur.dioxide = col_double(),
  density = col_double(),
  pH = col_double(),
  sulphates = col_double(),
  alcohol = col_double(),
  quality = col_double()
)
> View(wine)

> # Data pre-processing

> #set a high_quality column as Y
> wine$high_quality = 0
> wine$high_quality[wine$quality > 5] = 1
> wine_high_qulaity = subset(wine, select =
c("fixed.acidity","volatile.acidity","citric.acid","residu
al.sugar",
+
"chlorides","free.sulfur.dioxide","total.sulfur.dioxide",
+
"density","pH","sulphates","alcohol","high_quality"))

> # Random sampling into training, validation, and testing
> set.seed(15)
> training.rows <- sample(1:nrow(wine_high_qulaity),
nrow(wine_high_qulaity)*0.6)
> wine_quality_training =
wine_high_qulaity[training.rows,]
> wine_quality_vt = wine_high_qulaity[-training.rows,]

> validation.rows <-
sample(1:nrow(wine_quality_vt),nrow(wine_quality_vt)*0.5)
> wine_quality_validation =
wine_quality_vt[validation.rows,]
```

```
> wine_quality_testing = wine_quality_vt[-
validation.rows,]

> # Check whether the rows of training+validation+testing
equals to the total number
> nrow(wine_high_qulaity) ==
nrow(wine_quality_training)+nrow(wine_quality_validation)+
+   nrow(wine_quality_testing)
[1] TRUE
```

Fist, split the training set(60%) from the whole data frame, and then equally split the rest of the data into validation(20%) and testing set(20%).

- **(i) Logistic Regression Model**

```
> # Basic logistic regression model trained on training
set
> # glm:generalize linear model; family=binomial:use
logistic regression
> model_lrg = glm(high_quality ~
fixed.acidity+volatile.acidity+citric.acid+residual.sugar+
+
chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
+                density+pH+sulphates+alcohol,
+              data=wine_quality_training,family=binomial)
```

```
> summary(model_lrg)
Call:
glm(formula = high_quality ~ fixed.acidity + volatile.acidity +
    citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + density + pH + sulphates + alcohol,
    family = binomial, data = wine_quality_training)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6061  -0.8978   0.4431   0.8053   2.5536

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          3.809e+02  9.481e+01   4.017 5.88e-05 ***
fixed.acidity        1.044e-01  9.589e-02   1.089 0.276360
volatile.acidity    -6.457e+00  5.311e-01 -12.157  < 2e-16 ***
citric.acid          2.417e-01  3.937e-01   0.614 0.539257
residual.sugar       2.187e-01  3.579e-02   6.112 9.83e-10 ***
chlorides            2.110e+00  2.089e+00   1.010 0.312403
free.sulfur.dioxide  6.839e-03  3.538e-03   1.933 0.053222 .
total.sulfur.dioxide 3.348e-05  1.574e-03   0.021 0.983030
density             -3.948e+02  9.613e+01  -4.107 4.01e-05 ***
pH                   1.471e+00  4.769e-01   3.084 0.002040 **
sulphates            1.518e+00  4.512e-01   3.365 0.000767 ***
alcohol              5.973e-01  1.245e-01   4.799 1.60e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3738.3  on 2937  degrees of freedom
Residual deviance: 2961.2  on 2926  degrees of freedom
AIC: 2985.2

Number of Fisher Scoring iterations: 5


> # Make predictions on the validation set
> # The type="response" option tells R to output probabilities of the form P(Y = 1|X), as opposed to other information such as the logit.
> predictVal_lrg = predict(model_lrg, type="response", newdata = wine_quality_validation)

> # Analyze predictions
> summary(predictVal_lrg)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.004204 0.477606 0.707180 0.662999 0.869147 0.991805

> # Confusion matrix for threshold of 0.5
> table(wine_quality_validation$high_quality, predictVal_lrg > 0.5)
```

```
      FALSE TRUE
  0     159  176
  1      64  581
> # Calculate the overall accuracy
> accuracy_lrg = (159+581)/(159+176+64+581)
> accuracy_lrg
[1] 0.755102
```

The overall accuracy of logistic regression model is ***75.51%***.

- **(ii)k-NN model**

```
> # Create the training data set
> # Scaling:(x-mean(x))/sd(x)
> f_a = (wine_quality_training$fixed.acidity-mean(wine_quality_training$fixed.acidity))/sd(wine_quality_training$fixed.acidity)
> v_a = (wine_quality_training$volatile.acidity-mean(wine_quality_training$volatile.acidity))/sd(wine_quality_training$volatile.acidity)
> c_a = (wine_quality_training$citric.acid-mean(wine_quality_training$citric.acid))/sd(wine_quality_training$citric.acid)
> r_s = (wine_quality_training$residual.sugar-mean(wine_quality_training$residual.sugar))/sd(wine_quality_training$residual.sugar)
> chl = (wine_quality_training$chlorides-mean(wine_quality_training$chlorides))/sd(wine_quality_training$chlorides)
> f_s_d = (wine_quality_training$free.sulfur.dioxide-mean(wine_quality_training$free.sulfur.dioxide))/sd(wine_quality_training$free.sulfur.dioxide)
> t_s_d = (wine_quality_training$total.sulfur.dioxide-mean(wine_quality_training$total.sulfur.dioxide))/sd(wine_quality_training$total.sulfur.dioxide)
> den = (wine_quality_training$density-mean(wine_quality_training$density))/sd(wine_quality_training$density)
> ph = (wine_quality_training$pH-mean(wine_quality_training$pH))/sd(wine_quality_training$pH)
> sul = (wine_quality_training$sulphates-mean(wine_quality_training$sulphates))/sd(wine_quality_training$sulphates)
```

```
> alc = (wine_quality_training$alcohol-
mean(wine_quality_training$alcohol))/sd(wine_quality_train
ing$alcohol)
> h_q = as.factor(wine_quality_training$high_quality)

knn_train=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den,p
h,sul,alc)
knn_training=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,de
n,ph,sul,alc,h_q)

> # Create the validation data set
> # Scaling:(x-mean(x))/sd(x)
> f_a = (wine_quality_validation$fixed.acidity-
mean(wine_quality_validation$fixed.acidity))/sd(wine_quali
ty_validation$fixed.acidity)
> v_a = (wine_quality_validation$volatile.acidity-
mean(wine_quality_validation$volatile.acidity))/sd(wine_qu
ality_validation$volatile.acidity)
> c_a = (wine_quality_validation$citric.acid-
mean(wine_quality_validation$citric.acid))/sd(wine_quality
_validation$citric.acid)
> r_s = (wine_quality_validation$residual.sugar-
mean(wine_quality_validation$residual.sugar))/sd(wine_qual
ity_validation$residual.sugar)
> chl = (wine_quality_validation$chlorides-
mean(wine_quality_validation$chlorides))/sd(wine_quality_v
alidation$chlorides)
> f_s_d = (wine_quality_validation$free.sulfur.dioxide-
mean(wine_quality_validation$free.sulfur.dioxide))/sd(wine
_quality_validation$free.sulfur.dioxide)
> t_s_d = (wine_quality_validation$total.sulfur.dioxide-
mean(wine_quality_validation$total.sulfur.dioxide))/sd(win
e_quality_validation$total.sulfur.dioxide)
> den = (wine_quality_validation$density-
mean(wine_quality_validation$density))/sd(wine_quality_val
idation$density)
> ph = (wine_quality_validation$pH-
mean(wine_quality_validation$pH))/sd(wine_quality_validati
on$pH)
> sul = (wine_quality_validation$sulphates-
mean(wine_quality_validation$sulphates))/sd(wine_quality_v
alidation$sulphates)
> alc = (wine_quality_validation$alcohol-
mean(wine_quality_validation$alcohol))/sd(wine_quality_val
idation$alcohol)
> h_q = as.factor(wine_quality_validation$high_quality)
```

```
knn_valid=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den,p
h,sul,alc)
knn_validation=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,
den,ph,sul,alc,h_q)
>
> # Test the overall accuracy on the testing set.
> accuracy_knn = function(actual, predicted) {
+   return(mean(actual == predicted))
+ }

> # load the package 'class'
> library(class)

> #Make predictions on validation set using kNN
> predicted_validation = knn(train = knn_train, test =
knn_valid, cl = knn_training$h_q, k = 5)
> high_quality_validation = knn_validation$h_q
>
> #Confusion Matrix
> table(high_quality_validation, predicted_validation)
                       predicted_validation
high_quality_validation   0   1
                      0 205 130
                      1  99 546
> #Calculate the accuracy of kNN
> accuracy_knn(actual = high_quality_validation,predicted
= predicted_validation)
[1] 0. 7663265
```

The overall accuracy of logistic regression model is ***76.63%***. The performance of Logistic Regression Model is better than that of KNN Model. So, I choose ***Logistic Regression Model***.

```
> # Choose KNN model
> # Create the testing data set
> # Scaling:(x-mean(x))/sd(x)
> f_a = (wine_quality_testing$fixed.acidity-
mean(wine_quality_testing$fixed.acidity))/sd(wine_quality_
testing$fixed.acidity)
> v_a = (wine_quality_testing$volatile.acidity-
mean(wine_quality_testing$volatile.acidity))/sd(wine_quali
ty_testing$volatile.acidity)
> c_a = (wine_quality_testing$citric.acid-
mean(wine_quality_testing$citric.acid))/sd(wine_quality_te
sting$citric.acid)
```

```
> r_s = (wine_quality_testing$residual.sugar-
mean(wine_quality_testing$residual.sugar))/sd(wine_quality
_testing$residual.sugar)
> chl = (wine_quality_testing$chlorides-
mean(wine_quality_testing$chlorides))/sd(wine_quality_test
ing$chlorides)
> f_s_d = (wine_quality_testing$free.sulfur.dioxide-
mean(wine_quality_testing$free.sulfur.dioxide))/sd(wine_qu
ality_testing$free.sulfur.dioxide)
> t_s_d = (wine_quality_testing$total.sulfur.dioxide-
mean(wine_quality_testing$total.sulfur.dioxide))/sd(wine_q
uality_testing$total.sulfur.dioxide)
> den = (wine_quality_testing$density-
mean(wine_quality_testing$density))/sd(wine_quality_testin
g$density)
> ph = (wine_quality_testing$pH-
mean(wine_quality_testing$pH))/sd(wine_quality_testing$pH)
> sul = (wine_quality_testing$sulphates-
mean(wine_quality_testing$sulphates))/sd(wine_quality_test
ing$sulphates)
> alc = (wine_quality_testing$alcohol-
mean(wine_quality_testing$alcohol))/sd(wine_quality_testin
g$alcohol)
> h_q = as.factor(wine_quality_testing$high_quality)
>
knn_test=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den,ph
,sul,alc)
>
knn_testing=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den
,ph,sul,alc,h_q)
> #Make predictions on testing set using kNN
> predicted_testing = knn(train = knn_valid, test =
knn_test, cl = knn_validation$h_q, k = 5)
> high_quality_testing = knn_testing$h_q
> #Calculate the accuracy of kNN
> accuracy_knn(actual = high_quality_testing,predicted =
predicted_testing)
[1] 0.7408163
```

So, the testing error of the KNN Model is ***0. 2591837*** (1-0.7408163).

- **2(b)**

```
> # Re-split the testing set
> set.seed(8)
> training.rows <- sample(1:nrow(wine_high_qulaity),
nrow(wine_high_qulaity)*0.7)
```

```
> wine_quality_training =
wine_high_qulaity[training.rows,]
> wine_quality_testing = wine_high_qulaity[-
training.rows,]


> #Cross Validation for K-nn Model

> #Load necessary packages for Cross Validation
> library(caret)
> library(e1071)

> # Create the training data set
> # Scaling:(x-mean(x))/sd(x)
> f_a = (wine_quality_training$fixed.acidity-
mean(wine_quality_training$fixed.acidity))/sd(wine_quality
_training$fixed.acidity)
> v_a = (wine_quality_training$volatile.acidity-
mean(wine_quality_training$volatile.acidity))/sd(wine_qual
ity_training$volatile.acidity)
> c_a = (wine_quality_training$citric.acid-
mean(wine_quality_training$citric.acid))/sd(wine_quality_t
raining$citric.acid)
> r_s = (wine_quality_training$residual.sugar-
mean(wine_quality_training$residual.sugar))/sd(wine_qualit
y_training$residual.sugar)
> chl = (wine_quality_training$chlorides-
mean(wine_quality_training$chlorides))/sd(wine_quality_tra
ining$chlorides)
> f_s_d = (wine_quality_training$free.sulfur.dioxide-
mean(wine_quality_training$free.sulfur.dioxide))/sd(wine_q
uality_training$free.sulfur.dioxide)
> t_s_d = (wine_quality_training$total.sulfur.dioxide-
mean(wine_quality_training$total.sulfur.dioxide))/sd(wine_
quality_training$total.sulfur.dioxide)
> den = (wine_quality_training$density-
mean(wine_quality_training$density))/sd(wine_quality_train
ing$density)
> ph = (wine_quality_training$pH-
mean(wine_quality_training$pH))/sd(wine_quality_training$p
H)
> sul = (wine_quality_training$sulphates-
mean(wine_quality_training$sulphates))/sd(wine_quality_tra
ining$sulphates)
> alc = (wine_quality_training$alcohol-
mean(wine_quality_training$alcohol))/sd(wine_quality_train
ing$alcohol)
```

```
> h_q = as.factor(wine_quality_training$high_quality)

knn_train=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den,p
h,sul,alc)
knn_training=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,de
n,ph,sul,alc,h_q)

> # Create the testing data set
> # Scaling:(x-mean(x))/sd(x)
> f_a = (wine_quality_testing$fixed.acidity-
mean(wine_quality_testing$fixed.acidity))/sd(wine_quality_
testing$fixed.acidity)
> v_a = (wine_quality_testing$volatile.acidity-
mean(wine_quality_testing$volatile.acidity))/sd(wine_quali
ty_testing$volatile.acidity)
> c_a = (wine_quality_testing$citric.acid-
mean(wine_quality_testing$citric.acid))/sd(wine_quality_te
sting$citric.acid)
> r_s = (wine_quality_testing$residual.sugar-
mean(wine_quality_testing$residual.sugar))/sd(wine_quality
_testing$residual.sugar)
> chl = (wine_quality_testing$chlorides-
mean(wine_quality_testing$chlorides))/sd(wine_quality_test
ing$chlorides)
> f_s_d = (wine_quality_testing$free.sulfur.dioxide-
mean(wine_quality_testing$free.sulfur.dioxide))/sd(wine_qu
ality_testing$free.sulfur.dioxide)
> t_s_d = (wine_quality_testing$total.sulfur.dioxide-
mean(wine_quality_testing$total.sulfur.dioxide))/sd(wine_q
uality_testing$total.sulfur.dioxide)
> den = (wine_quality_testing$density-
mean(wine_quality_testing$density))/sd(wine_quality_testin
g$density)
> ph = (wine_quality_testing$pH-
mean(wine_quality_testing$pH))/sd(wine_quality_testing$pH)
> sul = (wine_quality_testing$sulphates-
mean(wine_quality_testing$sulphates))/sd(wine_quality_test
ing$sulphates)
> alc = (wine_quality_testing$alcohol-
mean(wine_quality_testing$alcohol))/sd(wine_quality_testin
g$alcohol)
> h_q = as.factor(wine_quality_testing$high_quality)

knn_test=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den,ph
,sul,alc)
knn_testing=data.frame(f_a,v_a,c_a,r_s,chl,f_s_d,t_s_d,den
,ph,sul,alc,h_q)
```

```
> # 6-fold cross validation
> trControl <- trainControl(method  = "cv", number = 6)

> fit_cv <- train(h_q ~ ., method = "knn", tuneGrid =
expand.grid(k = 1:10),metric="Accuracy",
+                  trControl = trControl,data=knn_training)

> fit_cv

k-Nearest Neighbors

3428 samples
  11 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (6 fold)
Summary of sample sizes: 2857, 2858, 2856, 2856, 2857, 2856, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.7783040  0.5023447
   2  0.7354177  0.4042998
   3  0.7514684  0.4318507
   4  0.7415386  0.4068175
   5  0.7494145  0.4220047
   6  0.7494145  0.4206332
   7  0.7555517  0.4302209
   8  0.7520409  0.4228866
   9  0.7605051  0.4403283
  10  0.7575903  0.4339997

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.

> # The best model is k=1.
> #Make predictions on testing set using kNN
> predicted_testing = knn(train = knn_train, test =
knn_test, cl = knn_training$h_q, k = 1)
> high_quality_testing = knn_testing$h_q
>
> #Calculate the accuracy of kNN
> accuracy_knn(actual = high_quality_testing,predicted =
predicted_testing)
[1] 0.7959184
```

Due to its highest overall accuracy, ***0.7959184***, I choose k=1 as the best model, and its testing error is ***0. 2040816*** (1- 0.7959184).