Introduction to Machine Learning: Lab 3

Hao Chi KIANG

November 16, 2016

Assignment 1: LDA and Logistic Regression

Solution 1.1

Figure 1 shows the carapace length and rear width of the provided Australian Crabs data set. It seems there is possibility to classify the crabs into two groups: the group which tends to have higher position and the one that tends to have lower position. Linear discriminant is a reasonable choice here, as the gap between the cluster seems to be forming a straight line.

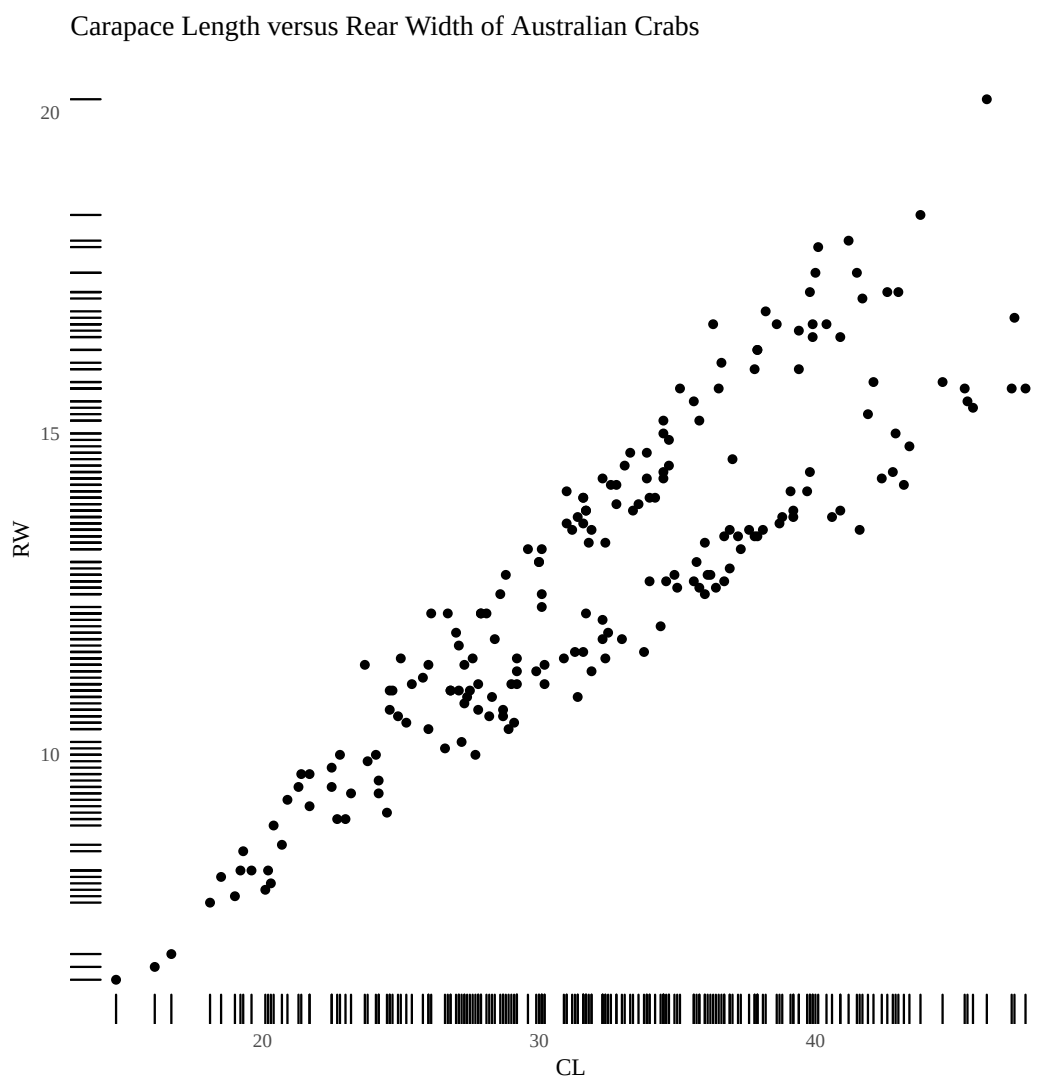

Figure 1: Carapace Length versus Rear Width of Australian Crabs

Solution 1.2

Figure 2 shows a classification result using linear discriminant analysis, and a plot of the decision boundary is included.

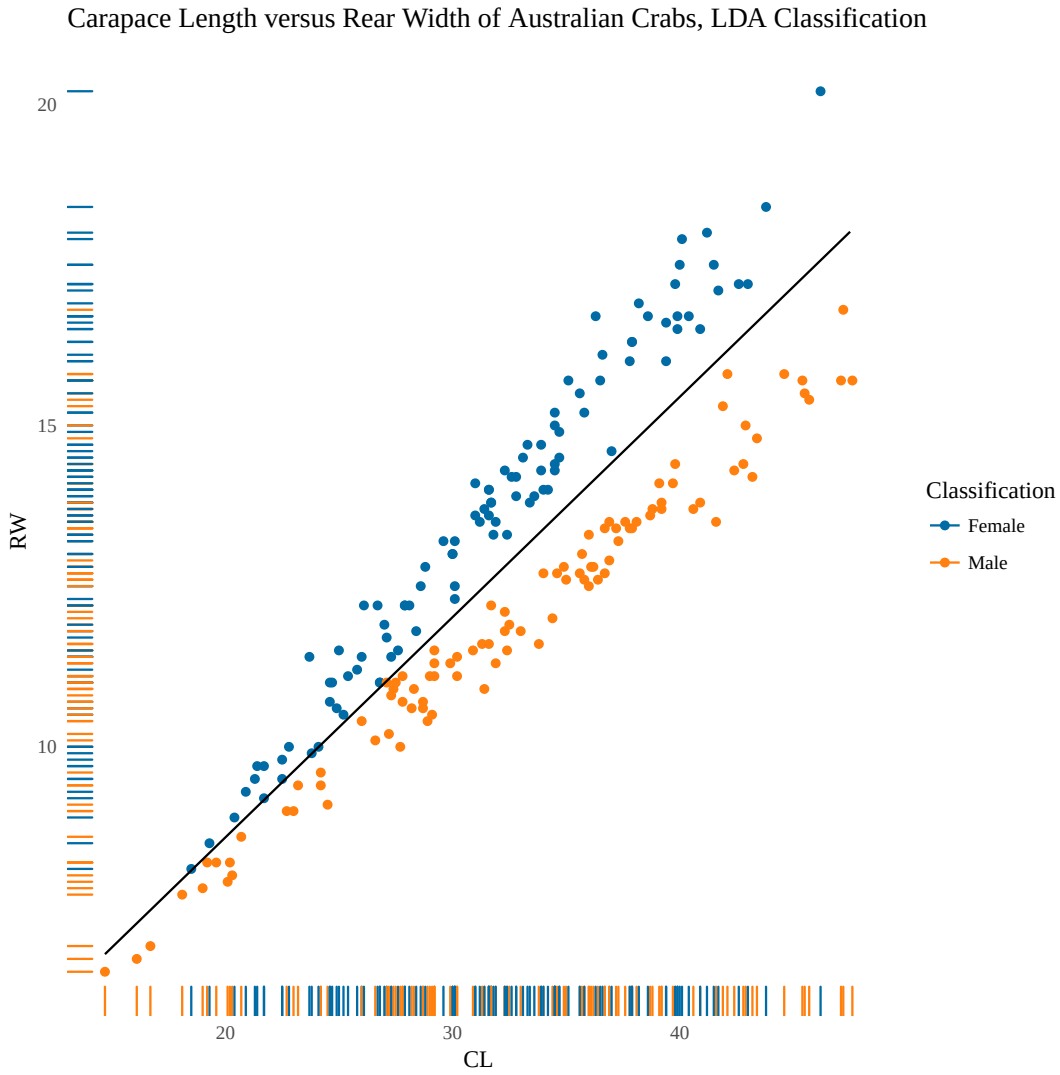

Figure 2: Carapace Length versus Rear Width of Australian Crabs, Classification result

The estimated discriminant function is

$$\delta_i(x_1, x_2) = w_{0i} + w_{1i}x_1 + w_{2i}x_2$$

where i is the identification number of features: CL be 1, and RW be 2, and

w_{ij} have the following estimated values:

$$\begin{aligned} w_{01} &= -12.6833192, & w_{02} &= -22.648321 \\ w_{11} &= -0.2159742, & w_{12} &= -2.183150 \\ w_{21} &= 2.5917691, & w_{22} &= 8.332018 \end{aligned}$$

The equation of decision boundary can be derived easily as follow: The set of points (x_1, x_2) on the boundary are the points which satisfies $\delta_1(x_1, x_2) = \delta_2(x_1, x_2)$. That is,

$$w_{01} + w_{11}x_1 + w_{21}x_2 = w_{02} + w_{12}x_1 + w_{22}x_2$$

Plugging in our value of estimated w_{ij} and normalize on x_1's term, term, the equation is

$$x_1 - 2.918x_2 + 5.066 = 0$$

This function was used to plot the boundary with *ggplot*.

Solution 1.3

Figure 3 displays the decision boundary and the real data on the same plot. We can see that our classification is quite accurate.

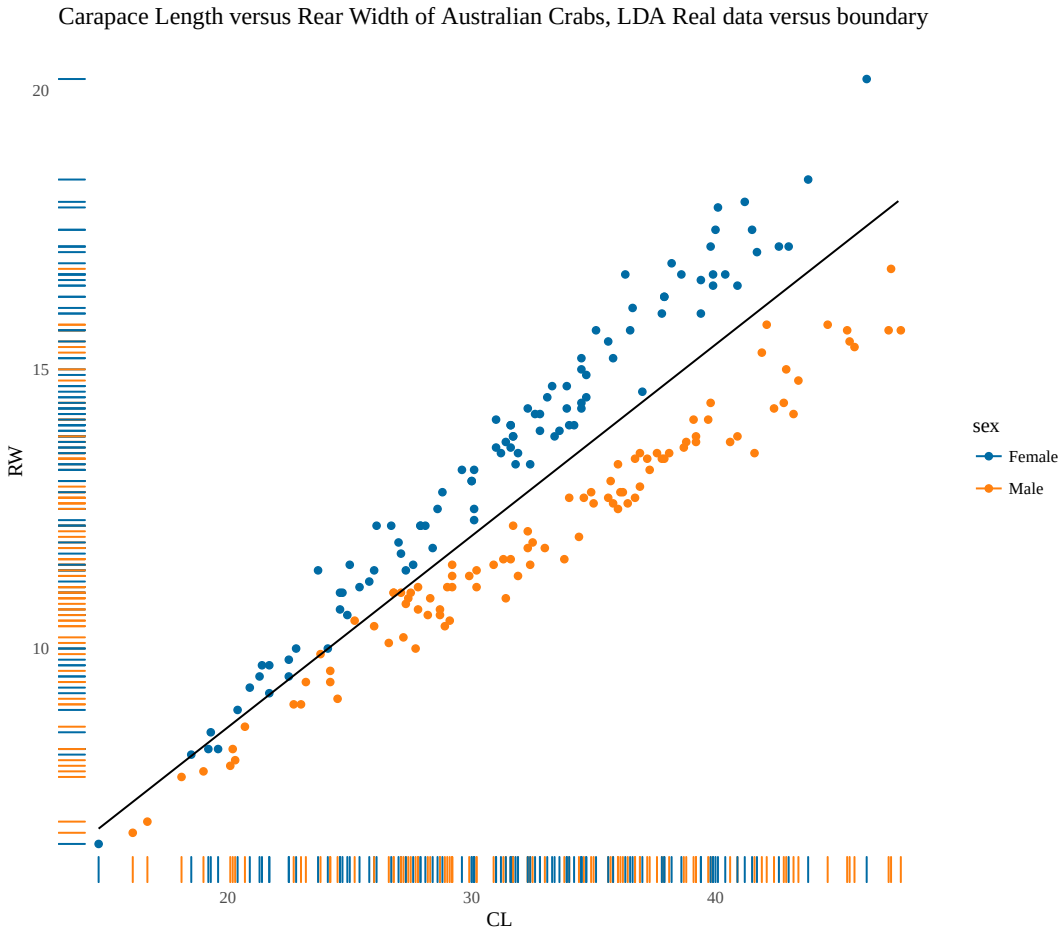

Figure 3: Carapace Length versus Rear Width of Australian Crabs, Real data vs. Decision Boundary

Solution 1.4

Figure 4 shown a logistic regression using the $glm()$ function. The result of logistic regression is almost exactly the same as the

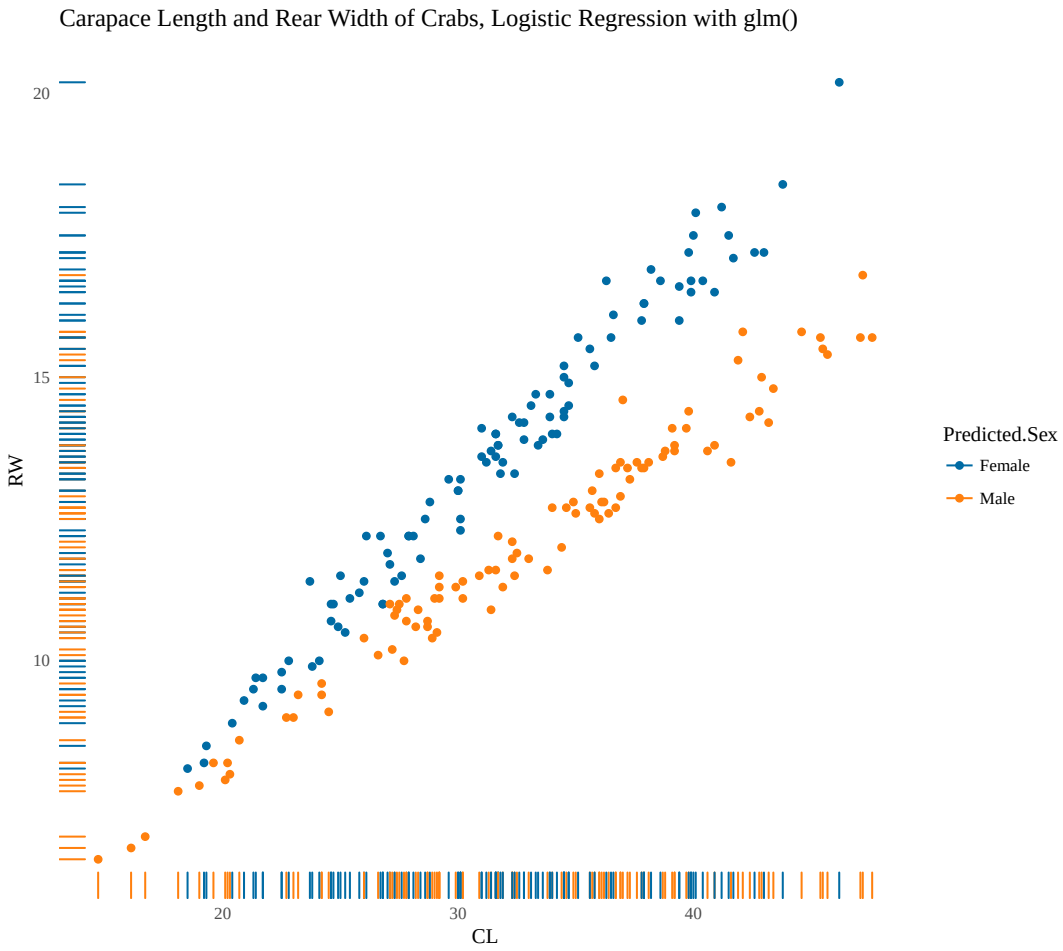

Figure 4: Carapace Length and Rear Width of Crabs, Logistic Regression with glm()

The equation of decision boundary can be derived as follows:
Consider any general logistic function $f(x) = \frac{1}{1+e^{-x}}$. The solution of the

equation $f(x) = 1 - f(x)$ is

$$\begin{aligned}
\frac{1}{1+e^{-x}} &= 1 - \frac{1}{1+e^{-x}} \\
\frac{1}{1+e^{-x}} &= \frac{e^{-x}}{1+e^{-x}} \\
1 &= e^{-x} \\
x &= 0
\end{aligned}$$

Now we want to find the set of points (x_1, x_2) whose probabilty of being female is the same as being male. That is, we want a set of points which satisfy $f(w_0 + w_1x_1 + w_2x_2) = 1 - f(w_0 + w_1x_1 + w_2x_2)$, where w_i are the linear regression coefficients. By , this set of points satisfies

$$w_0 + w_1x_1 + w_2x_2 = 0 \tag{1}$$

Plugging in our estimated w_i and normalize on x_1's term, the equation is

$$x_1 - 2.713x_2 + 2.940 = 0 \tag{2}$$

This is very similar to to the result we got from logistic regression.

Assignment 2: Linear Regression and Regularization

Solution 2.1, 2.2

The misclassification rate is as below:

Gini Train	0.236
Gini Testing	0.36
Deviance Train	0.212
Deviance Testing	0.268

From the above table, deviance seems to be a better measure of impurity in our data set.

Solution 2.3

According to Figure 5, it is reasonable to use tree depth = 4 as the best tree for prediction. The tree uses savings, duration, and history as predictor,

which is very consistent to our common sense. The misclassification rate for testing data of this tree is 0.256.

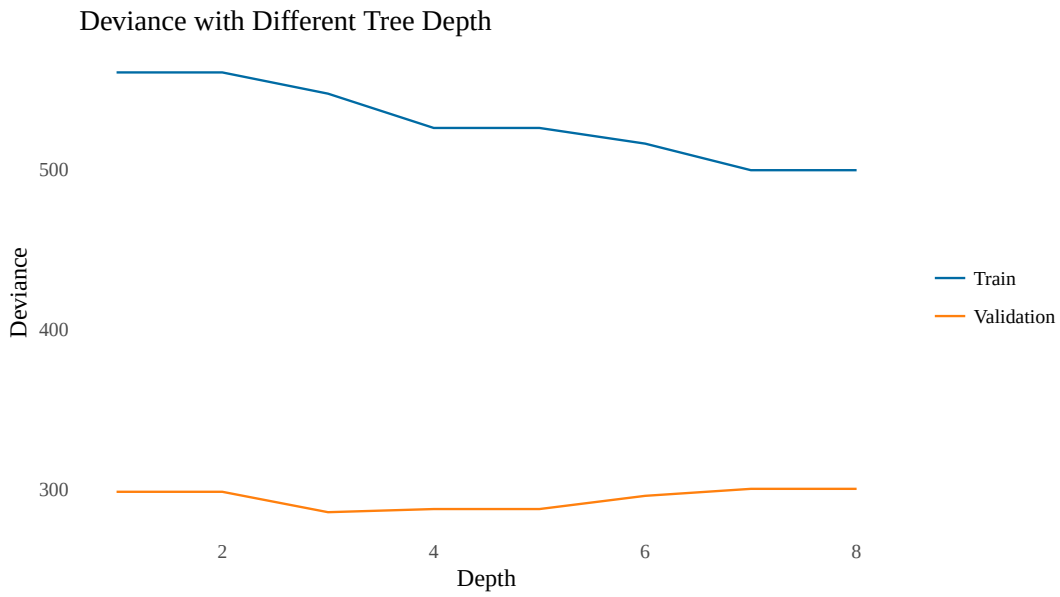

Figure 5: Deviance with Different Tree Depth

Solution 2.4

The confusion matrix of Naive Bayes classification using the training data set is as follows:

	bad	good
bad	95	98
good	52	255

Misclassification rate of training data is 0.3. The confusion matrix using the testing data set is as follows:

	bad	good
bad	46	49
good	30	125

Misclassification rate of testing data is 0.316. The classification is signif-

icantly worse than that of decision tree.

Solution 2.5

The confusion matrix of Naive Bayes classification with the provided custom loss function and the training data set is as follows:

	bad	good
bad	27	17
good	120	336

Misclassification rate of training data is 0.274. The confusion matrix using the testing data set is as follows:

	bad	good
bad	14	10
good	62	164

Misclassification rate of testing data is 0.288. This classification gives better overall misclassification rate, but it is more likely misclassify bad loaners as good: it is very conservative. The fact that such a conservative criterion gives better overall rate than that using normal loss function is due to that the normal loss function is more likely misclassify good loaner as bad: the adjustment of loss function corrected these bias. Also, most of the customers are good anyway, so adjusting the algorithm to guess more 'good' doesn't harm much.

Appendix 1: R Codes

```
library('tree')
library('glmnet')
library('ggthemes')
library('ggplot2')
library('functional')
library('magrittr')
library('reshape2')
library('MASS')
library('e1071')
library('readxl')

crabs <- read.csv('../question/australian-crabs.csv')

assignment1.1 <- function () {
    ggplot( crabs, aes(x = CL, y = RW) ) +
        geom_rug() +
        theme_tufte(ticks = F) +
        geom_point() +
        ggtitle('Carapace Length versus Rear Width of Australian Crabs')
}


class_mean_est <- function (X, Y, classes) {
    sapply( classes,
            . %>% { X[which(Y == .),] } %>% apply(2, mean)
          )
    ## Each column is a class, row is a feature in X
}


class_cov_est <- function (X, Y, classes, means = NULL) {
    if (is.null(means))
        means <- class_mean_est(X, Y, classes)

    lapply( classes,
           function (cls) {
               rowidx <- which(Y == cls)
               mats <-
                   lapply( rowidx,
                          function (rownum) {
                              dev <- as.numeric(
                                  matrix(X[rownum, ] - means[ ,cls]) )
                              dev %*% t(dev)
                          })
               lmatsum(mats) / length(rowidx)
           }) %>%
```

```
    set_names( classes )
}

lmatsum <- function (mats) {            # Sums a list of matrices
    nr <- nrow(mats[[1]])
    nc <- ncol(mats[[1]])
    Reduce(function (s1,s2) {s1 + s2},
           mats,
           matrix(0,
                  nrow = nr,
                  ncol = nc))
}

total_cov_est <- function (X, Y, classes, class_covs = NULL, means = NULL) {
    if (is.null(class_covs))
        class_covs <- class_cov_est(X, Y, classes, means = means)

    lapply(classes,
           function (cls) {
               length(which(Y == cls)) * class_covs[[cls]]
           }) %>% lmatsum %>% divide_by(nrow(X))
}

lda_coef <- function (X, Y, classes) {
    means <- class_mean_est(X, Y, classes)
    cov <- total_cov_est(X, Y, classes, means = means)

    slants <- sapply(
        classes,
        function (cls) {
            solve(cov) %*% means[, cls, drop = F]
        }
    ) %>% set_colnames( classes ) %>% set_rownames( colnames(X) )

    intercepts <- sapply(
        classes,
        function (cls) {
            mu <- means[, cls, drop = F]
            - 0.5 * t(mu) %*% solve(cov) %*% mu + log(length(which(Y == cls))/ length(
        }
    ) %>% t %>% set_colnames( classes )  %>% set_rownames( c('Intercepts') )

    rbind(intercepts, slants)
}

lda_2D_discrim_func <- function (coef) {
    Vectorize(
        function (x1) {
            ## Given a value of feature 1, return the boundary value of feature 2
```

```
            ((coef[2,2] - coef[2,1]) * x1 + coef[1,2] - coef[1,1]) /
                (coef[3,1] - coef[3,2])
        }
    )
}

predict_lda <- function (coef, X, classes) {
    ex <- sapply(
        classes,
        function (cls) {
            exp(coef[1, cls] + t(coef[-1, cls]) %*% X)
        }
    )
    (ex / sum(ex)) %>% which.max %>% {classes[.]}
}

assignment1.23 <- function () {
    X <- crabs[, c('CL', 'RW')]
    Y <- crabs[, 'sex']
    classes <- c('Male', 'Female')

    co <- lda_coef(X, Y, classes)
    print(co)

    X$Classification <- apply( X, 1, . %>% { predict_lda(co, ., classes) } )

    discrim.x <- seq(min(X$CL), max(X$CL), by = 0.2 )
    discrim.y <- lda_2D_discrim_func(co)(discrim.x)
    discrim <- data.frame( CL = discrim.x, RW = discrim.y )

    ggplot( X, aes(x = CL, y = RW, color = Classification) ) +
        geom_rug() +
        theme_tufte(ticks = F) +
        geom_point() +
        geom_line( data = discrim, color = 'black' ) +
        ggtitle(
            paste('Carapace Length versus Rear Width of Australian',
                  'Crabs, LDA Classification')) +
        scale_colour_tableau("colorblind10")
    ## ggsave('lda.pdf')

    ggplot( cbind(X, sex = Y), aes(x = CL, y = RW, color = sex) ) +
        geom_rug() +
        theme_tufte(ticks = F) +
        geom_point() +
        geom_line( data = discrim, color = 'black' ) +
        ggtitle(
            paste('Carapace Length versus Rear Width of Australian',
                  'Crabs, LDA Real data versus boundary')) +
```

```
        scale_colour_tableau("colorblind10")
    ## ggsave('realdat_lda.pdf')
}

assignment1.4 <- function () {
    m <- glm(sex ~ CL + RW, data = crabs, family=binomial())

    crabs.pred <- crabs
    crabs.pred$Predicted.Sex <- predict(m, crabs, type='response')

    crabs.pred$Predicted.Sex <- sapply(
        crabs.pred$Predicted.Sex,
        function (p) {
            if (p > 0.5)
                'Male'
            else
                'Female'

        }
    )

    print(crabs.pred$Predicted.Sex)

    ggplot( crabs.pred, aes(x = CL, y = RW, color = Predicted.Sex) ) +
        geom_rug() +
        theme_tufte(ticks = F) +
        geom_point() +
        ggtitle(
            paste('Carapace Length and Rear Width of',
                  'Crabs, Logistic Regression with glm()')) +
        scale_colour_tableau("colorblind10")
}



## Assignment2.1
credits <- read_excel('../question/creditscoring.xls')
credits$good_bad %<>% as.factor
set.seed(12345)
train.idx <- sample(1:nrow(credits), nrow(credits) / 2)
credits.train <- credits[train.idx, ]
set.seed(12345)
valid.idx <- sample(1:(nrow(credits)/2), nrow(credits) / 4)
credits.valid <- credits[-train.idx,][valid.idx,]
credits.test <- credits[-train.idx,][-valid.idx,]


misclass.rate <- function (est, data, real_class) {
    predict(est, newdata = data, type = 'class') %>% {
```

```
        1 - length( which(. == real_class) ) / nrow(data)
    }
}

assignment2.2 <- function () {
    print.rate <- function (title, est, data) {
        cat(title,
            misclass.rate(est, data, data$good_bad),
            '\n')
    }

    cat('Misclassification rates using different methods:\n')
    set.seed(12345)
    est.gini <- tree(good_bad ~ ., data = credits.train, split = 'gini')
    print.rate('Gini Train:', est.gini, credits.train)
    print.rate('Gini Testing:', est.gini, credits.test)

    set.seed(12345)
    est.dev <- tree(good_bad ~ ., data = credits.train, split = 'deviance')
    print.rate('Deviance Train:', est.dev, credits.train)
    print.rate('Deviance Testing:', est.dev, credits.test)
}

assignment2.34 <- function () {
    depths <- 2:9

    set.seed(12345)
    est.dev <- tree(good_bad ~ ., data = credits.train, split = 'deviance')

    trees <-
        lapply(
            depths,
            . %>% {
                prune.tree(est.dev, best = .)
            }) %>% set_names(depths)

    depths <-
        sapply( trees,
                . %>% {
                    c(deviance( . ),
                      deviance( predict( ., credits.valid, type = 'tree' )))
                }) %>% t %>% set_colnames(c('Train', 'Validation')) %>% melt

    ggplot(data = depths, aes(x = Var1, y = value, group = Var2, color = Var2) ) +
        geom_line() +
        theme_tufte(ticks = F) +
        xlab('Depth') +
        ylab('Deviance') +
        ggtitle( 'Deviance with Different Tree Depth' ) +
```

```
        scale_colour_tableau("colorblind10") +
        guides( color = guide_legend(title=NULL))

    best <- 4
    best.tree <- trees[[as.character(best)]]
    cat( 'Best tree misclassification rate:',
        misclass.rate(best.tree, credits.test, credits.test$good_bad),
        '\n')
    best.tree
}

assignment2.5 <- function () {
    m <- naiveBayes( good_bad ~ ., data = credits.train )

    table.train <-
        table( predict(m, newdata = credits.train), credits.train$good_bad )
    table.test <-
        table( predict(m, newdata = credits.test), credits.test$good_bad )

    cat('Training Naive Bayes:')
    print(table.train)
    print(1 - sum(diag(table.train)) / sum(table.train))
    cat('Testing Naive Bayes:')
    print(table.test)
    print(1 - sum(diag(table.test)) / sum(table.test))
}


assignment2.6 <- function () {
    m <- naiveBayes( good_bad ~ ., data = credits.train )
    confusion <- function (newdata) {
        probs <- predict(m, newdata = newdata, type='raw') *
            matrix(c(rep(1, nrow(newdata)), rep(10, nrow(newdata))) ,ncol = 2)
        which.bads <- which( probs[,'bad'] > probs[,'good'] )
        which.goods <- - which.bads
        pred <- rep(0, nrow(newdata))
        pred[which.bads] <- 'bad'
        pred[which.goods] <- 'good'
        pred %<>% as.factor
        table( pred, newdata$good_bad )
    }
    table.train <- confusion( credits.train )
    table.test <- confusion( credits.test )

    cat('Training Naive Bayes:')
    print(table.train)
    print(1 - sum(diag(table.train)) / sum(table.train))
    cat('Testing Naive Bayes:')
    print(table.test)
```

```
    print(1 - sum(diag(table.test)) / sum(table.test))
}
```