CLASSIFICATION

Esam Mahdi

Data Analytics (ECMP 5005B)
School of Mathematics and Statistics
Master of Engineering - Engineering Practice
Carleton University

October 13, 2023

LEARNING OBJECTIVES

By the end of this topic, you should be able to do the following:

1. Compare and contrast classification with linear regression.
2. Perform classification using logistic regression.
3. Interpret the odds and odds ratios in logistic regression.
4. Calculate Bayes decision boundaries.
5. Make predictions using linear discriminant analysis (LDA).
6. Make predictions using quadratic discriminant analysis (QDA).
7. Make predictions using Naive Bayes.
8. Make predictions using K-nearest neighbors (KNN).
9. Perform a Poisson regression (Note: This is a generalized linear model (GLM) and not considered as a classification methods).
10. Use **R** statistical software with real-life applications using different classification methods.

Definition

Classification: A method is used to make inference and/or predict *qualitative (categorical) response variable*.

Common classification techniques (classifiers):

- Logistic regression.
- Linear discriminant analysis (LDA).
- Quadratic discriminant analysis (QDA).
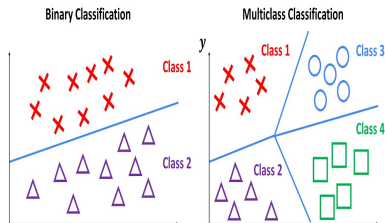- Naive Bayes.
- K-nearest neighbors.

FIGURE: Source: `https://datahacker.rs`.

EXAMPLES OF CLASSIFICATION PROBLEMS

1. A patient arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
   - Predictors: Symptoms.
   - Response: Type of medical conditions.

2. An online banking service needs to determine whether or not a transaction being performed on the site is fraudulent, given the user's IP address, past transaction history, and other factors.
   - Predictors: User's IP address, past transaction history, etc.
   - Response: Fraudulent activity (Yes/No).

CREDIT CARD DEFAULT: DEFAULT DATASET

The annual incomes (X_1 = income) and monthly credit card balances (X_2 = balance) are used to predict whether an individual will default on his or her credit card payment Y.

Suppose for the *Default* classification task that we code

$$Y = \begin{cases} 0, & \text{if No} \\ 1, & \text{if Yes} \end{cases}$$
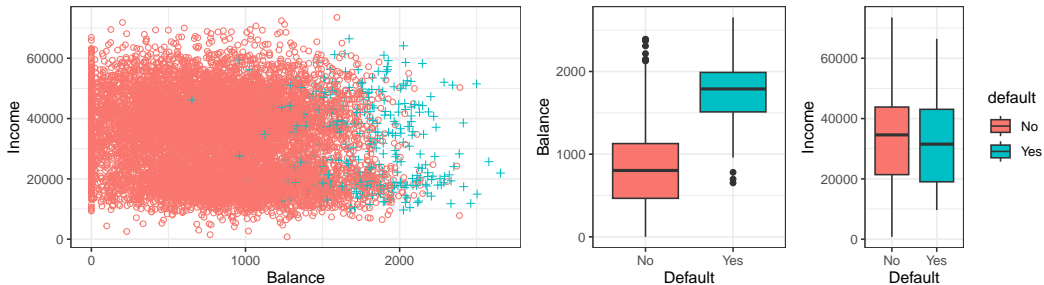

FIGURE: The distribution of balance and income split by the binary default variable respectively. (turquoise) represents defaulters and (pink) represents non-defaulters.
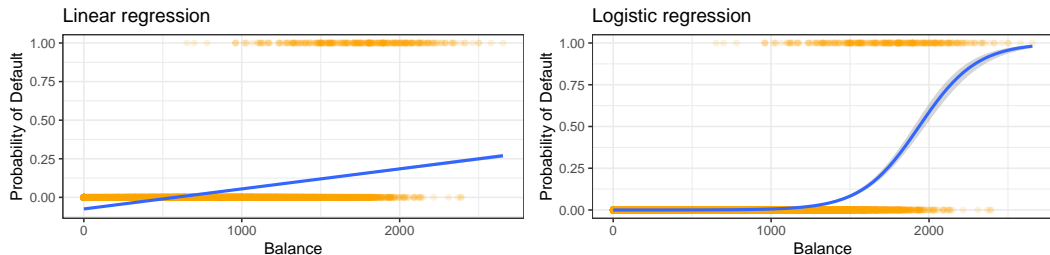
FIGURE: Classification using the *Default* data set. **Left:** Estimated probability of default using linear regression. **Right:** Predicted probabilities of default using logistic regression.

- The orange marks indicate the response Y, either 0 or 1.
- Can we simply perform a linear regression of Y on X and classify as Yes if $\hat{Y} > 0.5$?
- Instead of estimating Y by $\hat{Y}$, it is better to estimate the probability of Y given some values of X.
- Linear regression might produce probabilities less than zero or bigger than one. This is bad!
- Using logistic regression, all probabilities lie between 0 and 1. More appropriate!

Recall that the linear model (Ordinary lease squares (OLS) model) has the form

$$Y_i = \mu_i + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}\left(0, \sigma^2\right),$$

or

$$E\left(Y_i\right) = \mu_i = \boldsymbol{X}_i^T \boldsymbol{\beta}; \quad Y_i \sim \mathcal{N}\left(\mu_i, \sigma^2\right), \tag{1}$$

where

- $\boldsymbol{X}_i$ are $p+1$ covariates for observation i $(1, X_{i1}, X_{i2}, \cdots, X_{ip})^T$.
- $\boldsymbol{\beta}$ is the $(p+1) \times 1$ vector of regression coefficients $(\beta_0, \beta_1, \cdots, \beta_p)^T$.

- In many circumstances, the model (1) is not a valid model because:
  - The relationship between the response and predictors are not seen in the linear form that is given in the form (1).
  - Response variables are not normally distributed. They may even be categorical rather than continuous.
    - For example, there are many applications in which the response is a binary variable, taking on only two possible states. In this situation, a transformation to normality (e.g., using Box-Cox method) is out of the question.
- Generalized linear models (GLMs) are a flexible class of models that let us generalize from the linear model to include more types of response variables, such as count, binary, and proportion data.

GENERALIZED LINEAR MODELS (GLMs)

The *generalized linear models (GLMs)* have the form

$$g(\mu_i) = \eta_i = \boldsymbol{X}_i^T \boldsymbol{\beta}; \quad Y_i \sim \mathcal{G}\left(\mu_i, \theta\right), \tag{2}$$

where

- $\mathcal{G}$ is the *distribution* of the response variable.
- μ_i is a *location parameter* for observation i.
- θ are *additional parameters (dispersion)* for the density of $\mathcal{G}$.
- $g(\cdot)$ is a *link function*.

  > - The link function is a monotone, differentiable function maps a non-linear function relating $E\left(Y_i\right) = \mu_i$ to the linear component $\boldsymbol{X}_i^T \boldsymbol{\beta}$ using the following transformation
  >
  > $$g(\mu_i) = \eta_i = \boldsymbol{X}_i^T \boldsymbol{\beta}.$$
  >
  > - It is the link function that allows us to generalize the linear models for count, binomial and percent data.
  > - It ensures the linearity and constrains the predictions to be within a range of possible values.

- $\boldsymbol{X}_i$ are *covariates* for observation i.
- $\boldsymbol{\beta}$ is a vector of *regression coefficients*.

SIMPLE LOGISTIC REGRESSION MODEL

- Let Y be a *binary categorical response variable*; i.e., a *dichotomous* variable whose values are *success* and *failure*, which we convert to "1/0" for analysis, and let X be a single *predictor variable* which can be *categorical* or *numerical*,

- so that $Y|x \sim \text{Bernoulli}(\pi)$, where $\pi = \mathcal{P}(\text{success})$ is the probability of the success and $1 - \pi = \mathcal{P}(\text{failure})$ is the probability of the failure, where π and $(1 - \pi) \in (0, 1)$.

- Note that the mean of $Y|x$ is $E(Y) = \mu = 1 \times \mathcal{P}(Y = 1) + 0 \times \mathcal{P}(Y = 0) = \pi$.

- The *simple logistic regression model* can be used to predict the probability of $Y = 1|X = x$ and $Y = 0|X = x$.
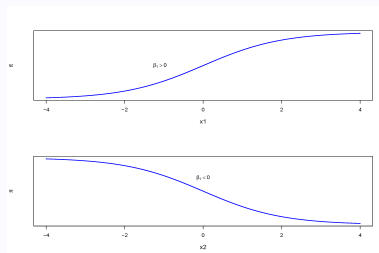

FIGURE: Logistic regression functions

SIMPLE LOGISTIC REGRESSION MODEL

DEFINITION

The probability of a response variable, with dichotomous values, estimated by the *logistic regression model* is represented by *S-shaped curve*. This curve is called *logistic function* given by

$$\pi = \mathcal{P}(X) = \mathcal{P}(Y = 1|X) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}, \qquad (3)$$

where $\exp(\cdot) = e \approx 2.71828$ is a mathematical constant (Euler's number.)

Note that π takes values between 0 and 1.

LOGIT MODELS

The *logistic regression* uses the *logit function* as a *link function* between the probability of Y and the linear regression expression. A bit of rearrangement to Eq. (3) gives

$$g(\mu) = \text{logit}(\pi) = \log \frac{\pi}{1-\pi} = \beta_0 + \beta_1 x, \quad \text{where} \quad \mu = \pi. \tag{4}$$

- Logistic regression model is often called *log odds* or *logit model* (not to confused with logistic function). Simple logistic regression is a regression line with a logit function.

- $\text{logit}(\pi) \in (-\infty, \infty)$.

The parameters β_0 and β_1 can be estimated by the *maximum likelihood estimation (MLE)* using iterative numerical methods such as Newton-Raphson (not the *sum squares residuals*).
Now, we can use estimate model of (4) to:

- Predict the probability of the success for a particular value of $X = x$ (using Eq. (3)).

- Estimate the odds in favor of "success" for a particular value of $X = x$.

- Estimate the odds ratio when X is increased by ℓ-units, where $\ell \geq 1$.

ODDS AND ODDS RATIO

- The *odds* in favor of "success" at a given value $X = x$ is

$$\text{odds} = \frac{\mathcal{P}(\text{success})}{\mathcal{P}(\text{failure})} = \frac{\pi}{1 - \pi} = e^{\beta_0 + \beta_1 x}. \tag{5}$$

  A one-unit increase in x is associated with multiplying the odds of that $Y = 1$ by a factor of e^{β_1}.

- The *odds ratio* is the ratio of two odds (at $X = x_1$ and $X = x_2$) given by

$$\text{odds ratio} = \frac{\left[\frac{\mathcal{P}(x_1)}{1 - \mathcal{P}(x_1)} \right]}{\left[\frac{\mathcal{P}(x_2)}{1 - \mathcal{P}(x_2)} \right]} = e^{\beta_1 (x_1 - x_2)}, \tag{6}$$

  where $\mathcal{P}(x_i) = \mathcal{P}(Y = 1 | X = x_i) = \pi_i$ for $i = 1, 2$.

- The *logit (also called the log-odds)* function is the natural logarithm of the odds.

$$\text{logit}(\pi) = \log(\text{odds})$$

  A one-unit increase in x is associated with β_1 increase/decrease in the log-odds that $Y = 1$.

EXAMPLE

- Let $\pi = 0.8$ denotes the probability that a person will get infected by COVID-19 ($Y = 1$, where 1 denotes success).

    The odds in favor of getting infected by COVID-19 is $0.8/0.2 = 4$;   i.e., $4 : 1$. This means that 4 out of 5 will get infected by COVID-19.

- Now consider the fitted logit model

$$\text{logit}(\hat{\pi}) = \log(\widehat{\text{odds}}) = \hat{\beta}_0 + \hat{\beta}_1 x = -40.2 + 1.2x,$$

where X is an explanatory variable representing the age of a person. The results from the logit model can be interpreted in two ways:

- *(In the log odds scale):* For a one year increase in the age of a person, the log of the odds of getting infected by COVID-19 increases by 1.2.
- *(In the odds scale):* A one year increase in the age multiplies the odds of getting infected by COVID-19 by $\exp(\hat{\beta}_1) = e^{1.2} \approx 3.32$. Alternatively, a one year increase in the age increases the odds of getting infected by COVID-19 by $100 \times (3.32 - 1) \approx 232\%$.

- We can also use the estimated logistic function given in (3) to predict the probability at any age. For example, the estimated probability of getting a person of age 30 infected by COVID-19 is

$$\hat{\pi} = \hat{\mathcal{P}}(Y|X = 30) = \frac{\exp(-40.2 + 1.2(30))}{1 + \exp(-40.2 + 1.2(30))} \approx 0.015.$$

ASSUMPTIONS FOR BINARY LOGISTIC REGRESSION

The assumptions used for binary logistic regression:

- The response variable is a binary (the outcome is either success or failure).
- The success should be represented by the factor level 1 of the response variable, the failure is 0.
- The sample size is large enough.
- No multicollinearity in the independent variables.
- Log odds and independent variables have to be linearly related.

MULTIPLE LOGISTIC REGRESSION

- *Multiple Logistic Regression* is used to relate a categorical (binary) response variable Y to $p \geq 1$ predictor variables, $X_1, \cdots, X_p$.

- The predictors may be true quantitative predictors or indicator variables coding categorical predictors.

- The unknown β_j's parameters can be estimated by the point estimators $\hat{\beta}_j$'s, where $j = 0, 1, \cdots, p$, using Newton-Raphson based MLE method. The resulting estimated *logit function (probability of success)* is

$$\hat{\pi} = \hat{\mathcal{P}}(Y|x_1, \cdots, x_p) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p}} = \frac{1}{1 + e^{-(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p)}}. \quad (7)$$

- The estimated *logit model (multiple logistic regression function)* is

$$\text{logit}(\hat{\pi}) = \log(\widehat{\text{odds}}) = \log \frac{\hat{\pi}}{1 - \hat{\pi}} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p. \quad (8)$$

ODDS AND ODDS RATIO IN MULTIPLE LOGISTIC REGRESSION

- The *odds* is given by

$$\text{odds} = \frac{\pi}{1-\pi} = \frac{\mathcal{P}(Y=1|x_1,\cdots,x_p)}{1-\mathcal{P}(Y=1|x_1,\cdots,x_p)} = e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}. \tag{9}$$

- The estimated *odds* is given by

$$\widehat{\text{odds}} = \frac{\hat{\pi}}{1-\hat{\pi}} = \frac{\hat{\mathcal{P}}(Y=1|x_1,\cdots,x_p)}{1-\hat{\mathcal{P}}(Y=1|x_1,\cdots,x_p)} = e^{\hat{\beta}_0+\hat{\beta}_1 x_1+\cdots+\hat{\beta}_p x_p}. \tag{10}$$

- A one-unit increase in x_j when the other x's are hold constant is associated with multiplying the odds of $Y=1$ by a factor of $e^{\hat{\beta}_j}$.

- An h-unit increase in x_j when the other x's are hold constant is associated with multiplying the odds of $Y=1$ by a factor of $e^{h\beta_j}$.

- The odds ratio associated with h unit increase in x_i and s unit increase in x_j where the other predictors do not change is

$$\widehat{\text{odds ratio}}_{x_i,x_j} = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x_1+\cdots+\hat{\beta}_i(x_i+h)+\cdots+\hat{\beta}_p x_p}}{e^{\hat{\beta}_0+\hat{\beta}_1 x_1+\cdots+\hat{\beta}_j(x_j+s)+\cdots+\hat{\beta}_p x_p}} = e^{h\hat{\beta}_i - s\hat{\beta}_j}. \tag{11}$$

LOGISTIC REGRESSION IN R

There are four types of logistic regression models:

Binary (dichotomous) logistic regression: The `glm()` function in the **stats** package can be used to fit a binary logistic when the response variable has two unordered categories (e.g., "success/failure", "yes/no", which we convert to "1/0" for analysis). *Note "glm" stands for generalized linear models*.

Multinomial logistic regression (Multiclass logistic regression): The `mlogit()` function in the **mlogit** package can be used to fit a multinomial logistic when the response variable has more than two unordered categories (e.g., "black/white/blue/others", "married/divorce/widow"). Alternatively, you can use the `multinom()` function in the **nnet** package.

Ordinal logistic regression: The `polyr()` function in the **MASS** package can be used to fit an ordinal logistic when the response variable has ordinal categories (e.g., "low/middle/high income", "high school/BS/MS/PhD").

Robust logistic regression: The `glmRob()` function in the **robustbase** package can be used to fit a robust generalized linear model, including robust logistic, when data contains outliers and influential observations.

THE GLM() FUNCTION

The form of the function `glm()` is similar to `lm()` with some additional parameters. The basic format of the function `glm()` is

```
glm(formula, family = family(link = link function), data)
```

TABLE: Probability distribution (family) and link function parameter in `glm()` function

Family	Default link function
binomial	link = "logit"
gaussian	link = "identity"
poisson	link = "log"
gamma	link = "inverse"
quasi	link = "identity", variance = "constant"
quasibinomial	link = "logit"
quasipoisson	link = "log"
inverse.gaussian	link = "1/mu^2"

EXAMPLE

Suppose we have a data frame (mydata) with a single dichotomous (0 or 1) response variable Y and two predictor variables $X1, X2$.

- The logistic regression model can be fit using

  ```
  glm(Y ~ X1+X2, family = binomial(link = "logit"), data = mydata)
  ```

  Alternatively

  ```
  glm(Y ~ X1+X2, family = binomial(), data = mydata)
  ```

- The Poisson regression model can be fit using

  ```
  glm(Y ~ X1+X2, family = poisson(link = "log"), data = mydata)
  ```

  Alternatively

  ```
  glm(Y ~ X1+X2, family = poisson(), data = mydata)
  ```

- Note that that the standard linear model is a special case of the generalized linear model. Thus

  ```
  glm(Y ~ X1+X2, family = gaussian(link = "identity"), data = mydata)
  ```

  would produce the same results as

  ```
  lm(Y ~ X1+X2, data = mydata)
  ```

ASSESSMENT OF MODEL ADEQUACY

Once you fit the logistic regression model using the function `glm()`, you can diagnose this model. One way to assess the model adequacy is to plot the predicted probabilities against the residuals of the deviance by using the command

```
pred <- predict(your.fitted.model, type = "response")
res <- residuals(your.fitted.model, type = "deviance")
plot(pred, res)
```

The model is good if the plot shows no pattern (i.e., the plot should be a random scatter). You could also check the plots of hat values, studentized residuals, and Cook's D statistics using the function `influencePlot()` in the package **car**

```
car::influencePlot(your.fitted.model)
```

ASSESS THE PERFORMANCE OF THE MODEL USING CONFUSION MATRIX

You can use the function `ifelse()` to turn your predicted probabilities into *classification* by using the threshold 0.5, so that the predicted values with probability more than 0.5 will be considered as *"successes" = "yes" = 1* whereas the values with probability less than or equal to 0.5 will be considered as *"failures" = "no" = 0*. After that, you can assess the performance of your model by comparing the "yes/no" in the predicted data with those observed in the response variable using the function `table()` (*confusion matrix*).

```
pred.class=ifelse(pred > 0.5, 1, 0)
pred.class=factor(pred.class,levels=c(0,1),labels=c("No","Yes"))
table(pred.class, mydata$response)
```

Another elegant way to calculate the confusion matrix is to use the function `confusionMatrix()`, which is a part of the **caret** library

```
require(caret)
confusionMatrix(pred.class, mydata$response, positive = "Yes")
```

BINARY LOGISTIC REGRESSION WITH CATEGORICAL EXPLANATORY VARIABLES

Consider the logistic model

$$\text{logit}[P(Y = 1)] = \beta_0 + \beta_1 D_1 + \beta_2 D_2,$$

where D_1 and D_2 are two *dummy variables* for two *categorical explanatory variables* X_1 and X_2 coded by the values 0 and 1. Then D_1 and D_2 are *called indicator variables*, because each indicates the category of an explanatory variable. In this case, we can display the data in a $2 \times 2 \times 2$ contingency table. The following table shows the logit values at the four combinations of values 0 and 1 of two categorical predictors X_1 and X_2.

D_1	D_2	Logit
0	0	β_0
1	0	$\beta_0 + \beta_1$
0	1	$\beta_0 + \beta_2$
1	1	$\beta_0 + \beta_1 + \beta_2$

This model assumes no interaction between the two categorical predictors.

- The effect of one factor is the same at each category of the other factor. For instance, at a fixed category D_2, the effect on the log odds (logit) of changing from $D_1 = 0$ to $D_1 = 1$ is

$$(\beta_0 + \beta_1 \times 1 + \beta_2 D_2) - (\beta_0 + \beta_1 \times 0 + \beta_2 D_2) = \beta_1.$$

- Thus, when the category D_2 is hold constant, the odds of success at $D_1 = 1$ will be equal to $\exp(\beta_1)$ times the odds of success at $D_1 = 0$.

EXAMPLE: SIMPLE LOGISTIC REGRESSION FOR CREDIT CARD DEFAULT

```r
# Load the package "xtable" to print an output of R as a LaTeX or HTML
library(xtable)
fit1 <- glm(default~balance, family = binomial(), data = Default)
xtable(fit1)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-10.6513	0.3612	-29.49	0.0000
balance	0.0055	0.0002	24.95	0.0000

- What is our estimated probability of default for someone with a balance of \$1000?

$$\hat{\mathcal{P}}(Y=1|X=1000) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 x}} = \frac{e^{-10.6513+0.0055\times1000}}{1+e^{-10.6513+0.0055\times1000}} = 0.006$$

- With a balance of \$2000?

$$\hat{\mathcal{P}}(Y=1|X=2000) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 x}} = \frac{e^{-10.6513+0.0055\times2000}}{1+e^{-10.6513+0.0055\times2000}} = 0.586$$

LOG ODDS AND ODDS FOR THE SIMPLE LOGISTIC REGRESSION FOR CREDIT CARD DEFAULT USING BALANCE AS A PREDICTOR

From the summary of the GLM model, we can write the fitted logit model as follows

$$\text{logit}(\hat{\pi}) = \log(\widehat{\text{odds}}) = \log \frac{\hat{\pi}}{1 - \hat{\pi}} = -10.6513 + 0.0055 \text{ balance}.$$

Thus, the estimated odds for being default are given by

$$\widehat{\text{odds}} = \exp\{-10.6513 + 0.0055 \text{ balance}\}.$$

For example,

- For a one unit increase in the balance (one dollar), the odds of being default increases by $\exp(0.0055) = 1.006$; (i.e., the odds for being in default class are about 0.6% higher than being in not default class).

- One thousand dollars increase in the balance implies that the odds for being included in default class increases by $\exp(1000 \times 0.0055) = 244.69$; (i.e., $(244.69 - 100) \times 100\% = 144.69\%$). *Is this make sense?*

CONFUSION MATRIX

Let's assess the performance of our fitted model by comparing the predicted values with the true values.

> A *confusion matrix* represents the prediction summary in matrix form. It shows how many prediction are correct and incorrect per class.

```r
pred <- predict(fit1, type = "response")
pred.class <- ifelse(pred > 0.5, 1, 0)
pred.class <- factor(pred.class, levels = c(0, 1),
                                 labels = c("No", "Yes"))
table(pred.class, Default$default) # confusion matrix

##
## pred.class   No   Yes
##         No  9625  233
##         Yes   42  100

#try: caret::confusionMatrix(pred.class,Default$default,positive="Yes")
```

Rate misclassification is $(42 + 233)/10000 = 2.75\%$. Of the true No's, we make $42/9667 = 0.43\%$ errors; of the true Yes's, we make $233/333 = 69.97\%$ errors!. Note that these errors are called *training errors*.

```r
library(caret)
confusionMatrix(pred.class, Default$default, positive = "Yes")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9625  233
##        Yes   42  100
##
##                Accuracy : 0.9725
##                  95% CI : (0.9691, 0.9756)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 0.0004973
##
##                   Kappa : 0.4093
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3003
##             Specificity : 0.9957
##          Pos Pred Value : 0.7042
##          Neg Pred Value : 0.9764
```

EXAMPLE: LETS FIT A LOGISTIC REGRESSION FOR CREDIT CARD DEFAULT AGAIN, USING STUDENT AS A SINGLE PREDICTOR

```
fit2 <- glm(default ~ student, family = binomial(), data = Default)
summary(fit2)
```

```
##
## Call:
## glm(formula = default ~ student, family = binomial(), data = Default)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
## studentYes   0.40489    0.11502    3.52 0.000431 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 2908.7  on 9998  degrees of freedom
## AIC: 2912.7
##
## Number of Fisher Scoring iterations: 6
```

EXAMPLE: LOGISTIC REGRESSION FOR CREDIT CARD DEFAULT AGAIN, USING STUDENT AS A SINGLE PREDICTOR (CONT.)

	Estimate	Std. Error	z value	Pr($>$|z|)
(Intercept)	-3.5041	0.0707	-49.55	0.0000
studentYes	0.4049	0.1150	3.52	0.0004

- What is our estimated probability of default for a student?

$$\hat{\mathcal{P}}(\text{default = Yes}|\text{student = Yes}) = \frac{e^{-3.5041+0.4049\times1}}{1+e^{-3.5041+0.4049\times1}} = 0.0431.$$

- What is our estimated probability of default for a non student?

$$\hat{\mathcal{P}}(\text{default = Yes}|\text{student = No}) = \frac{e^{-3.5041+0.4049\times0}}{1+e^{-3.5041+0.4049\times0}} = 0.0292.$$

FIT A LOGISTIC REGRESSION FOR CREDIT CARD DEFAULT WITH SEVERAL PREDICTORS (MULTIPLE LOGISTIC MODEL)
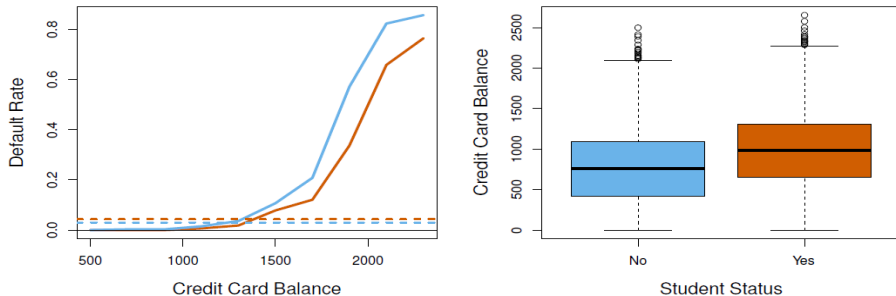
FIGURE: Source: An introduction to statistical learning with applications in R.

- Students tend to have higher balances than non-students, so their marginal default rate is higher than for non-students.

- But for each level of balance, students default less than non-students.

- Multiple logistic regression can tease this out.

FIT A LOGISTIC REGRESSION FOR CREDIT CARD DEFAULT WITH SEVERAL PREDICTORS (MULTIPLE LOGISTIC MODEL)

```
fit3 = glm(default ~ student + balance + income,
                 family = binomial(), data = Default)
# same as: glm(default ~., family = binomial(), data = Default)
summary(fit3)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-10.8690	0.4923	-22.08	0.0000
studentYes	-0.6468	0.2363	-2.74	0.0062
balance	0.0057	0.0002	24.74	0.0000
income	0.00003	0.00008	0.37	0.7115

Why is coefficient for student negative, while it was positive before?

Answer: Maybe *confounding available*! Confounding variable is a variable that influences both the dependent and independent variable, causing a *spurious* association.

LOG ODDS AND ODDS FOR THE MULTIPLE LOGISTIC REGRESSION - CREDIT CARD DEFAULT

From the summary of the multiple logistic model, we can write the fitted logit model as follows

$$\text{logit}(\hat{\pi}) = \log(\widehat{\text{odds}}) = -10.8690 - 0.6468 \text{ Student} + 0.0057 \text{ Balance} + 3.033 \times 10^{-6} \text{ Income}.$$

Thus, the estimated odds for being default are given by

$$\widehat{\text{odds}} = \exp\{-10.8690 - 0.6468 \text{ Student} + 0.0057 \text{ Balance} + 3.033 \times 10^{-6} \text{ Income}\}.$$

For example,

- The odds of being default for a student when the other variables hold constant decreases by $\exp(-0.6468) = 0.52$; (i.e., about less than 48% for non students.)

- One thousand dollars increase in the balance implies that the odds for being included in default class increases by $\exp(1000 \times 0.0057) = 298.87$.

CONFUSION MATRIX

Let's now assess the performance of the multiple logistic model by calculating the confusion matrix.

```
pred3 <- predict(fit3, type="response")
pred3.class <- ifelse(pred3 > 0.5, 1, 0)
pred3.class <- factor(pred3.class, levels = c(0, 1),
                                   labels=c("No", "Yes"))
table(pred3.class, Default$default) # confusion matrix

##
## pred3.class   No   Yes
##         No  9627   228
##         Yes   40   105

mean(pred3.class == Default$default) # prediction accuracy

## [1] 0.9732
```

The logistic regression predictions are accurate almost 97.3% of the time.

Split the Data into Training and Testing (Hold-out Set Cross Validation Method): Credit Card Default Data Set

- The previous results show that the logistic regression does a good job in predicting who will default.

- However, this result is misleading because we trained and tested the model on the same set.

- The errors that we calculated are known as *training error*, which has a rate that is often underestimate the *testing error* rate.

- A better assess for the accuracy of the fitted logistic regression model is to split the data, randomly, into two datasets: *Training data* and *testing data*.

- After that, we can fit the logistic regression model using the training data, and then examine how well it predicts testing *the held out data*.

REASSESS THE LOGISTIC MODEL USING THE TESTING DATA: THE CODE ...

```r
set.seed(1) # to replicate the random selections
# select 50% (5000 observations) of the data randomly
Sample <- sample(1:10000, 5000) #default argument: sample(replace = FALSE)
train <- Default[Sample,]
test <- Default[-Sample,]
default.test <- test$default
# In glm() function, you may use "data = train" as follows
fit3=glm(default~student+balance+income, data = train, family = binomial)
# Alternatively, use "data = Default, subset = Sample"
fit3=glm(default~student+balance+income,data=Default,family=binomial,subset=Sample)
# Get the prediction probabilties using the test data set
pred3 <- predict(fit3, test, type = "response")
# Use threshold 0.5, so that P(X)> 0.5 is classified as "yes"
pred3.class <- ifelse(pred3 > 0.5, "Yes", "No")
# Confusion matrix using the response variable in the test set
table(pred3.class, default.test)
# Accuarcy rate using the response variable in the test set (output is 0.974)
mean(pred3.class == default.test)
# Calculate the test set error rate (output is 0.026)
mean(pred3.class != default.test) # != notation means "not equal to".
```

```
##            default.test
## pred3.class  No  Yes
##         No  4825  112
##        Yes   18   45
## [1] 0.974
```

- Logistic regression is popular for classification problems with two classes ($K = 2$), but not ideal with more than two. *Linear discriminant analysis* is more robust, especially when we have more than two response classes.

- When there is significant separation between the classes, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis estimates are more stable.

- If the distribution of the predictors X is approximately normal in each of the classes and the sample size is small, then the discriminant analysis may be more accurate than logistic regression.

Here the approach is to model the distribution of the predictors X in each of the classes separately, and then use *Bayes theorem* to flip things around and obtain $\Pr(Y|X)$.

A popular types for *Discriminant analysis* are:

- *Linear discriminant analysis (LDA)*
- *Quadratic discriminant analysis (QDA)*
- *Naive Bayes*

When we use normal (Gaussian) distributions for each class, this leads to linear or quadratic discriminant analysis.

BAYES THEOREM

$$\Pr(Y = k \mid X = x) = \frac{\Pr(X = x \mid Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)} \quad (12)$$

For discriminant analysis, we represent Eq.(12) in another form as follows

$$\Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} \pi_i f_i(x)}, \quad (13)$$

where

- $\Pr(Y = k \mid X = x)$ is the *posterior* probability distribution.
- $f_k(x) = \Pr(X = x \mid Y = k)$ is the *likelihood (density)* for X in class k. Here we will use normal densities for these, separately in each class.
- $\pi_k = \Pr(Y = k)$ is the marginal or *prior* probability for class k.
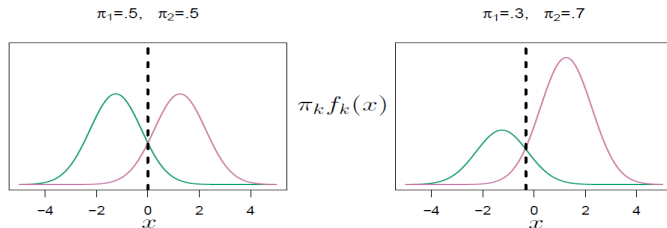
$\pi_1=.5, \quad \pi_2=.5$ $\qquad$ $\pi_1=.3, \quad \pi_2=.7$

$\pi_k f_k(x)$

FIGURE: Source: An introduction to statistical learning with applications in R (slides of chapter 4.)

- We classify a new point according to which class has the highest value of $\pi_k f_k(x)$. (Note that $\sum_{l=i}^{K} \pi_i f_i(x)$ is constant.)
- Any thing to the left of the *decision boundary (a vertical dashed line)* is classified as *green*, and any thing to the right is *pink*.
- **Left plot:** When priors are the same, we classify a new point (determine the decision boundary) according to which density ($f_k(x)$) is highest. Here, the decision boundary is located in the middle distance between the means of the densities of the two classes, where both densities are the same.
- **Right plot:** When the priors are different, we take them into account as well, and compare $\pi_k f_k(x)$. Here, the *pink* class has a higher density than the *green*. Thus, the decision boundary is shifted to the left to include more values from the *pink* class.

The Gaussian density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k}e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

Here μ_k is the mean and σ_k^2 the variance (in class k). We will assume that all the $\sigma_k = \sigma$ are the same.

Plugging this into Bayes formula, we get

$$\mathcal{P}_k(x) = \Pr(Y = k \mid X = x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{i=1}^{K} \pi_i \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma}\right)^2}}$$

To classify at the value $X = x$, we need to see which of the $p_k(x)$ is largest.

Taking logs, and discarding terms that do not depend on k, we see that this is equivalent to assigning x to the class with the largest discriminant score:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$
$$= c_0 + c_1 x \tag{14}$$

where $c_0 = -\frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$ and $c_1 = \frac{\mu_k}{\sigma^2}$.

Note that $\delta_k(x)$ is a *linear* function of x with intercept c_0 and slope c_1.

If there are $K = 2$ classes and $\pi_1 = \pi_2 = 0.5$, then one can see that the decision boundary is point for which $\delta_1(x) = \delta_2(x)$

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}.$$
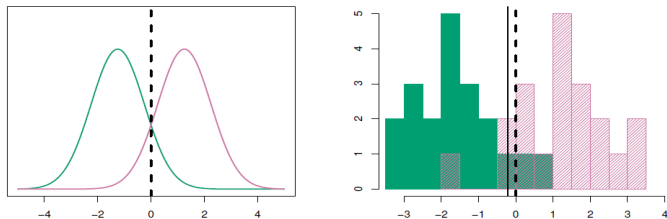
FIGURE: Source: An introduction to statistical learning with applications in R (slides of chapter 4.)

Example with $\mu_1 = -1.5$, $\mu_2 = 1.5$, $\pi_1 = \pi_2 = 0.5$, and $\sigma^2 = 1$.

Assuming we know these parameters, we can plot the Gaussian densities of the two classes as shown left. The dashed vertical line represents the Bayes decision boundary. Typically we don't know these parameters; we just have the training data. Thus, we use the histograms to estimate these densities as shown right. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

$$= \sum_{k=1}^{K} \frac{n_k - 1}{n - K} \cdot \hat{\sigma}_k^2$$

where $\hat{\sigma}_k^2 = \frac{1}{n_k-1} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$ is the usual formula for the estimated variance in the k th class.
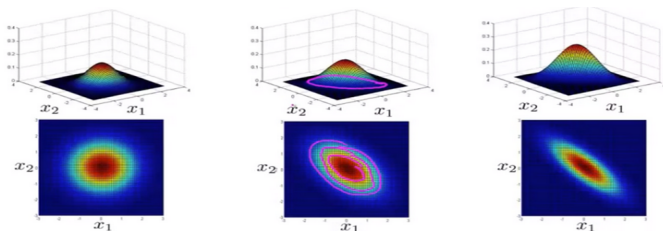
FIGURE: Bivariate normal distribution: Left: X_1 and X_2 are independent. Middle: X_1 and X_2 are weekly dependent. Right: X_1 and X_2 are strongly dependent.

The multivariate Gaussian density is defined as

$$f(x) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)}$$

The discriminant function is

$$\delta_k(x) = x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2}\mu_k^T \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k$$
$$= c_{k0} + c_{k1}x_1 + c_{k2}x_2 + \ldots + c_{kp}x_p, \quad \text{which is a linear function of predictors.} \quad (15)$$
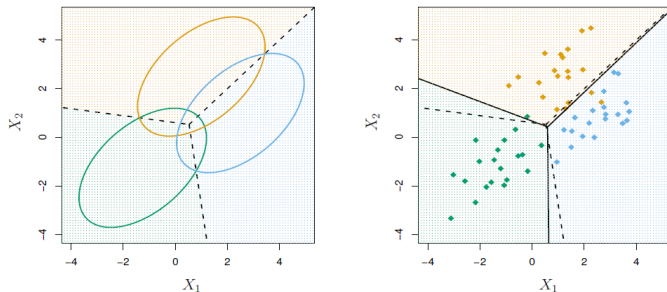
FIGURE: Source: `https://r4ds.github.io/bookclub-islr/a-comparison-of-classification-methods.html`. An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95% of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines. Overall, the LDA decision boundaries are pretty close to the Bayes decision boundaries, shown again as dashed lines.

FROM $\delta_k(x)$ TO PROBABILITIES

Once we have estimates $\hat{\delta}_k(x)$, we can turn these into estimates for class probabilities:

$$\widehat{\Pr}(Y = k \mid X = x) = \frac{e^{\hat{\delta}_k(x)}}{\sum_{i=1}^{K} e^{\hat{\delta}_i(x)}}.$$

So classifying to the largest $\hat{\delta}_k(x)$ amounts to classifying to the class for which $\widehat{\Pr}(Y = k \mid X = x)$ is largest.
When $K = 2$, we classify to class 2 if $\widehat{\Pr}(Y = 2 \mid X = x) \geq 0.5$, otherwise to class 1.

```
pairs(iris[,1:4],col=c("#1AA7EC","#DAB135","#358856")[iris[,5]],lwd =1.5)
```
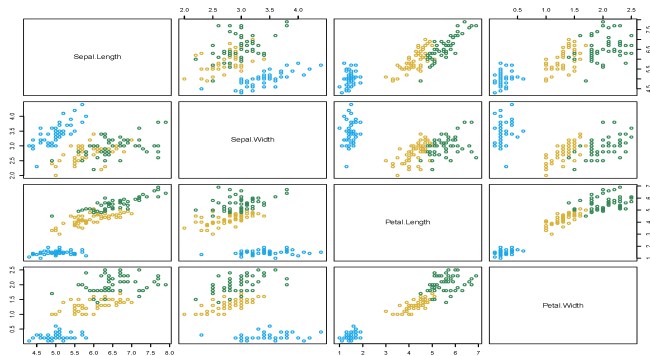

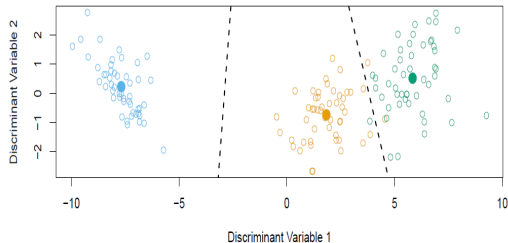FIGURE: Fisher's Iris Data: 4 variables, 3 species (Setosa, Versicolor, Virginica), 50 samples/class.

FIGURE: Source: An introduction to statistical learning with applications in R (slides of chapter 4.)

- When there are K classes, linear discriminant analysis can be viewed exactly in a $K - 1$ dimensional plot. This is because it essentially classifies to the closest centroid, and they span a $K - 1$ dimensional plane.
- Even when $K > 3$, we can find the "best" 2-dimensional plane for visualizing the discriminant rule.

LDA CONFUSION MATRIX FOR CREDIT CARD DATA

Let's assess the performance of LDA on the credit card data set. In R, we fit an LDA model using the `lda()` function, which is part of the **MASS** library.

```
library(MASS)
lda.fit <- lda(default~balance+student, data = Default)
lda.pred <- predict(lda.fit, Default)
table(lda.pred$class, Default$default)

##
##          No   Yes
##   No   9644   252
##   Yes    23    81

mean(lda.pred$class == Default$default) #prediction accuracy

## [1] 0.9725
```

$(23 + 252)/10000$ errors — a 2.75% misclassification rate!

CAVEATS OF ERROR RATE

- This is *training error*, and we may be *overfitting*. Not a big concern here since $n = 10000$ (large) and $p = 2$ (small)!
- If we classified to the prior — always to class No in this case — we would make $333/10000 = 3.33\%$ errors!
- Of the true No's, we make $23/9667 = 0.2\%$ errors; of the true Yes's, we make $252/333 = 75.7\%$ errors!

LDA CONFUSION MATRIX FOR CREDIT CARD DATA

Let's reassess the performance of LDA on the credit card data set using the training-testing sets.

```
lda.fit <- lda(default~balance+student, data = train)
lda.pred <- predict(lda.fit, test)
table(lda.pred$class, default.test)

##      default.test
##        No  Yes
##   No  4832  118
##   Yes   11   39

mean(lda.pred$class == default.test) #prediction accuracy

## [1] 0.9742
```

$(11 + 118)/5000$ errors — a 2.58% misclassification rate. Note too much different from what we got before when we fit the model using the full data.

		Reference	
		Yes (+)	No (−)
Prediction	Yes (+)	TP (True positive)	FP (False positive)
	No (−)	FN (False negative)	TN (True negative)

- Let's choose a threshold of 0.5, so that all observations satisfy

$$\widehat{\Pr}(\text{Default} = \text{Yes}|\text{Balance, Student}) \geq 0.5$$

will be classified as 1 (yes). Otherwise they will be classified as 0 (no).

- Then we can calculate the *sensitivity* $= \frac{TP}{TP+FN}$ and *specificity* $= \frac{TN}{FB+TN}$.

- But we can select any threshold:

$$\widehat{\Pr}(\text{Default} = \text{Yes}|\text{Balance, Student}) \geq threshold$$

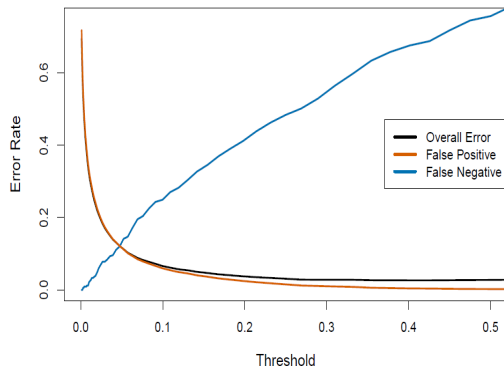and for each threshold we will have different sensitivity and specificity.

FIGURE: Source: An introduction to statistical learning with applications in R.

FIGURE: Source: https://en.wikipedia.org/wiki/Sensitivity_and_specificity.

In order to reduce the false negative rate, we may want to reduce the threshold to 0.1 or less.

- Another way to check the model performance is through the *Receiver Operating Characteristic (ROC)* curve.
- The *ROC* curve is a classification error metric obtained by plotting *sensitivity (true positive rate)* against the *one minus specificity (false positive rate)*.
- If the model is perfect at detecting true positives then the *false positive rate (FPR)* will always be 0 and *true positive rate (TPR)* will always be 1.
- This will provide a curve which is the upper triangular. The area under that curve would be the area of the square that is 1.
- If the model is randomly deciding the positives then the *TPR* will always be equal to *FPR*.
- This will provide the 45 degree line. The area under the curve is the area under the right triangle which 0.5
- Most of the times we will get a curve which can have an area between 0.5 and 1.
- Thus we need to calculate the *Area Under the Curve (AUC)* to check whether it is closer to 1 or 0.5
- The *AUC* is often referred to as *c-index*.
- If the *AUC* is closer to 1 then we can interpret that the model has a very good discrimination ability.
- We select the threshold value that leads to increase the *TPR* and reduce the *FPR* as much as possible.

THE ROC CURVE FOR CREDIT CARD DATA

```r
library(ROCR)
fit<- glm(default~balance+student, family = binomial(),data=Default)
pred <- predict(fit, type = "response")
roc_score <- prediction(pred,Default$default)
roc_perf <- performance(roc_score,'tpr','fpr')
plot(roc_perf,colorize=TRUE,print.cutoffs.at=seq(0.1,by=0.1),lwd=3)
```
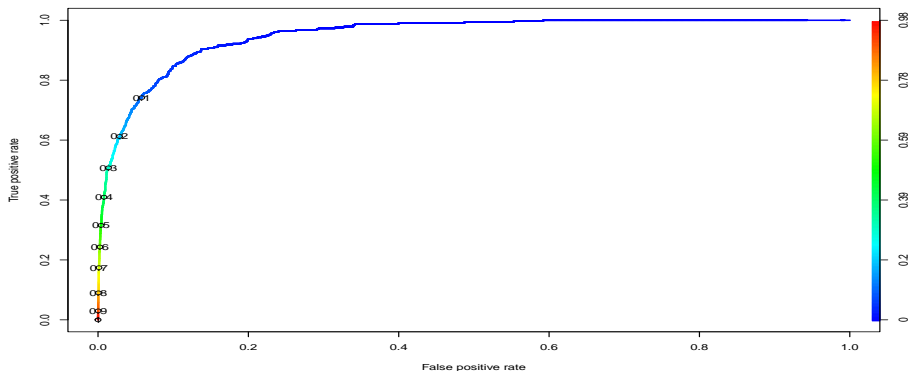
Recall, the posterior probability is given by

$$\Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_i f_i(x)}.$$

When $f_k(x)$ are Gaussian densities, with the same covariance matrix $\boldsymbol{\Sigma}$ in each class, this leads to *linear discriminant analysis (LDA)*. By altering the forms for $f_k(x)$, we get different classifiers.

- With Gaussians but different $\boldsymbol{\Sigma}_k$ in each class, we get *quadratic discriminant analysis (QDA)*.
- With $f_k(x) = \prod_{j=1}^{p} f_{jk}(x_j)$ (conditional independence model) in each class we get *naive Bayes*. For Gaussian this means the $\boldsymbol{\Sigma}_k$ are diagonal.
- Many other forms, by proposing specific density models for $f_k(x)$, including *nonparametric* approaches.

QUADRATIC DISCRIMINANT ANALYSIS (QDA)

Quadratic discriminant analysis (QDA) is similar to LDA, but it assumes that each class has its own covariance matrix. i.e., $X \sim \mathcal{N}(\mu_k, \Sigma_k)$, where the values of Σ_k are different.
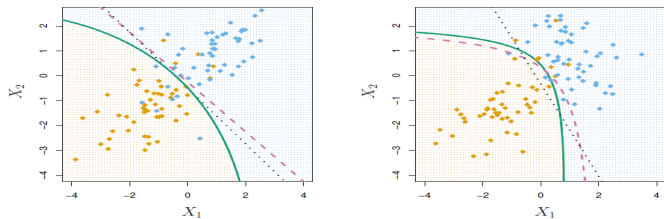

FIGURE: Performances of LDA and QDA in two scenarios. Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\Sigma_1 = \Sigma_2 = 0.7$. Right: Details are as given in the left-hand panel, except that $0.7 = \Sigma_1 \neq \Sigma_2 = -0.7$.

Bayes classifier

$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k - \frac{1}{2}\log|\Sigma_k|$$

With some algebraic, the Bayes classifier can be written as a quadratic form of X

$$\delta_k(x) = -\frac{1}{2}x^T \Sigma_k^{-1} x - x^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k + \log \pi_k - \frac{1}{2}\log|\Sigma_k|$$

QDA FOR CREDIT CARD DATA

QDA is implemented in R using the qda() function, which is also part of the **MASS** library. The qda() syntax is identical to that of lda(). We fit the model on the training data that we obtained before and then we assess the performance of this model on the test set.

```
library(MASS)
qda.fit <- qda(default~balance+student,data=Default,subset=Sample)
qda.pred <- predict(qda.fit, test)
table(qda.pred$class, default.test)

##      default.test
##         No  Yes
##   No  4829  114
##   Yes   14   43

mean(qda.pred$class == default.test)

## [1] 0.9744
```

The QDA predictions are accurate almost 97.4% of the time. Comparing this value with what we achieved by using the LDA method, we conclude that the quadratic form assumed by QDA might not needed as it doesn't improved our prediction.

NAIVE BAYES

- When p is large, multivariate methods like LDA and even QDA break down.

- *Naive Bayes* is useful in this case. It assumes that the features are independent in each class (strong assumption).

- Gaussian naive Bayes assumes each $\boldsymbol{\Sigma}_k$ is diagonal:

$$\delta_k(x) \propto \log \left[\pi_k \prod_{j=1}^{p} f_{kj}(x_j) \right]$$
$$= -\frac{1}{2} \sum_{j=1}^{p} \left[\frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2} + \log \sigma_{kj}^2 \right] + \log \pi_k$$

- can use for mixed feature vectors (qualitative and quantitative). If X_j is qualitative, replace $f_{kj}(x_j)$ with probability mass function (histogram) over discrete categories.

- Despite strong assumptions, *naive Bayes* often produces good classification results.

SIMPLE EMPIRICAL EXAMPLE TO ILLUSTRATE NAIVE BAYES

The following is a sample of size 10 that we observed from two features X_1, X_2 and categorical response variable Y takes two binary values $\{0, 1\}$.

X_1	1	0	0	1	2	0	1	2	2	0
X_2	0	0	1	2	1	2	1	0	2	0
Y	0	0	1	1	0	1	0	0	0	1

Suppose you need to predict the class of a given features, say $(x_1, x_2) = (0, 2)$. That is, we need to find

$$\max\left[\mathcal{P}(Y = 0 | (X_1 = 0, X_2 = 2)), \mathcal{P}(Y = 1 | (X_1 = 0, X_2 = 2))\right]$$

Solution: Use Bayes theorem and assume that X_1 and X_2 are independent in each class, so that $\mathcal{P}(X_1, X_2) = \mathcal{P}(X_1)\mathcal{P}(X_2)$ is satisfied. Thus,

$$\mathcal{P}(Y = k | (X_1 = 0, X_2 = 2)) = \frac{\mathcal{P}((X_1 = 0, X_2 = 2) | Y = k)\mathcal{P}(Y = k)}{\sum_{k=0}^{1} \mathcal{P}((X_1 = 0, X_2 = 2) | Y = k)\mathcal{P}(Y = k)}, \quad k = 0, 1$$

$$\propto \mathcal{P}(X_1 = 0 | Y = k) \times \mathcal{P}(X_2 = 2 | Y = k) \times \mathcal{P}(Y = k) \tag{16}$$

Note that you need to calculate the probability in the numerator and no need to calculate the denominator! Why?

Continue to next slide!

SIMPLE EMPIRICAL EXAMPLE TO ILLUSTRATE NAIVE BAYES (CONT.)

$$\mathcal{P}(Y = 0) = \frac{\#\{Y = 0\}}{10} = \frac{6}{10}$$

$$\mathcal{P}(Y = 1) = \frac{\#\{Y = 1\}}{10} = \frac{4}{10} \tag{17}$$

$$\mathcal{P}(X_1 = 0 | Y = 0) \times \mathcal{P}(X_1 = 2 | Y = 0) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

$$\mathcal{P}(X_1 = 0 | Y = 1) \times \mathcal{P}(X_1 = 2 | Y = 1) = \frac{3}{4} \times \frac{2}{4} = \frac{3}{8} \tag{18}$$

Multiply what you got in (18) by (17) to get (16):

$$\mathcal{P}(Y = k | (X_1 = 0, X_2 = 2)) = \begin{cases} \dfrac{1}{36} \times \dfrac{6}{10} = \dfrac{1}{60} = 0.0167, & \text{if } k = 0 \\[2mm] \dfrac{3}{8} \times \dfrac{4}{10} = \dfrac{3}{20} = 0.15, & \text{if } k = 1 \end{cases}$$

Thus, the *point* $(x_1, x_2) = (0, 2)$ *will be classified in class 1* because
$\max \left[\mathcal{P}(Y = 0 | (X_1 = 0, X_2 = 2)), \mathcal{P}(Y = 1 | (X_1 = 0, X_2 = 2)) \right] = 0.15$ which achieved when $Y = 1$.

NAIVE BAYES FOR CREDIT CARD DATA

Naive Bayes is implemented in R using the `naiveBayes()` function, which is part of the **e1071** library. The `naiveBayes()` syntax is identical to that of `lda()` and `qda()`. By default, naive Bayes classifier models each quantitative feature using a Gaussian distribution. However, a kernel density method can also be used to estimate the distributions.

```
library(e1071)
nb.fit <- naiveBayes(default~balance+student,data=train)
nb.pred <- predict(nb.fit, test)
table(nb.pred, default.test)

##        default.test
## nb.pred   No  Yes
##     No  4823  108
##     Yes   20   49

mean(nb.pred == default.test)

## [1] 0.9744
```

The naive Bayes predictions are accurate almost 97.4% of the time, which is almost the same value that we got by using the discriminant methods.

K-NEAREST NEIGHBORS (KNN)

K-NEAREST NEIGHBORS (KNN) ALGORITHM STEPS

Given n data points, the steps involved in performing KNN algorithm as follows:

- Calculate the distance (*Euclidean, Manhattan, or Hamming*) from the point you need to classify or predict to all other data points in the dataset.

- Get the closet k points. The choice of k is crucial! Small k usually leads to low bias and high variance (*overfitting*) . Large k usually leads to high bias and low variance (*underfitting*) . The best value can be found with cross validation and grid search, that is control the balance between overfitting and underfitting , known as the *bias-variance tradeoff*.

- You may start with $k = \sqrt{n}$, where n denotes the sample size. Typically, k is set somewhere between 3 and 10.
  - For *Classification* case: Get the class label of data point with the majority vote.
  - For *Regression* case: Get the average of the values of the k nearest neighbors. In this case, features usually transformed to a standard scale (e.g. *min-max normalization* or *z-score standardization*).

Min-max normalization

$$X_{\text{new}} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Z-score standardization

$$X_{\text{new}} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$
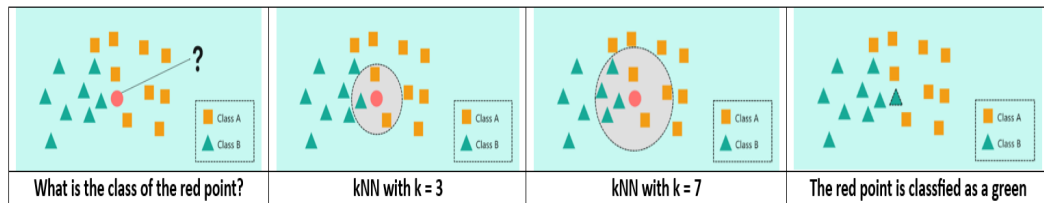

FIGURE: Source: Edureka. `https://www.edureka.co`.

We can perform the KNN using the function `knn()` in the library **class**. A better way is to use the library **caret**! Here, we give the KNN classifier implementation in the package **class**.

This function works differently from the other previous functions that we discussed thus far!

Rather than a two-step approach in which we first fit the model and then we use the model to make predictions, `knn()` forms predictions using a single command.

> The function requires four inputs.
> 1. A matrix containing the predictors associated with the training data, labeled `train.X`.
> 2. matrix containing the predictors associated with the data for which we wish to make predictions, labeled `test.X`.
> 3. A vector containing the class labels for the training observations, labeled `train.Direction`.
> 4. A value for K, the number of nearest neighbors to be used by the classifier.

KNN FOR FOR CREDIT CARD DEFAULT DATA USING $K = 3$

```r
library(class)
attach(Default)
train.X <- cbind(balance,student)[Sample, ]
test.X <- cbind(balance,student)[-Sample, ]
train.default <- default[Sample]
knn.pred <- class::knn(train.X, test.X, train.default, k = 3)
table(knn.pred, train.default)

##          train.default
## knn.pred   No  Yes
##      No  4710  175
##      Yes  114    1

mean(knn.pred==train.default)

## [1] 0.9422

detach(Default)
```

The results using $K = 3$ shows that about 94.22% of the observations are correctly predicted. Not too much improved compared with logistic model, LDA, QDA, and naive Bayes.

THE FUNCTION TRAIN() IN THE LIBRARY **caret**

The **caret** package (https://topepo.github.io/caret/) is a new package which supports more than 230 machine learning models including the KNN algorithm. The syntax of this function is

```
# Default S3 method
train(x, y, method = "rf", preProcess = NULL, ..., weights = NULL,
metric = ifelse(is.factor(y), "Accuracy", "RMSE"),
maximize = ifelse(metric%in%c("RMSE", "logLoss", "MAE", "logLoss"), FALSE, TRUE),
trControl = trainControl(), tuneGrid = NULL,
tuneLength = ifelse(trControl$method == "none", 1, 3)
)

## S3 method for class 'formula'
train(form, data, ..., weights, subset, na.action = na.fail, contrasts = NULL)

## S3 method for class 'recipe'
train(x, data, method = "rf", ...,
metric = ifelse(is.factor(y_dat), "Accuracy", "RMSE"),
maximize = ifelse(metric %in% c("RMSE", "logLoss", "MAE"), FALSE, TRUE),
trControl = trainControl(), tuneGrid = NULL,
tuneLength = ifelse(trControl$method == "none", 1, 3)
)
```

You can learn more about this function by typing the help ?train.

Note: The same authors of the **caret** package introduced a more recent flexible framework **tidymodels**

(https://www.tidymodels.org/) that might supersede the **caret** package.

REGRESSION FOR COUNTING DATA

- **Linear regression:** Response variable is *continuous*, $y \in \mathbb{R} \sim \mathcal{N}(\mu, \sigma^2)$.

- **Logistic regression:** Response variable is *binary*, $y \in \{0, 1\} \sim \text{Bern}(p)$.

- **Poisson regression:** Response variable is *count/rate*, $y \in \{0, 1, 2, \cdots\} \sim \text{Po}(\mu)$.

WHY LINEAR REGRESSION IS NOT RECOMMENDED TO MODEL COUNT DATA?

1. Count variable is a discrete random variable, usually, follows a Poisson distribution

$$\mathcal{P}(Y = y) = \frac{\mu^y e^{-\mu}}{y!}, \quad y = 0, 1, 2, \cdots$$

where $\text{E}(Y) = \text{Var}(Y) = \mu$. Thus, as the mean rate for a Poisson variable increases, the variance also increases which violates the linear regression assumption that the variance is homogeneous.

2. The linear regression might estimates negative values for μ which contradicts with the fact that this average can only take on values from 0 to ∞.

Examples of counts are the number of accidents, number of deaths due to COVID-19 in 2021, and number of rats. Counts also occur in summarizing categorical variables with *contingency tables*.

For the case of Poisson regression, the three Components of GLM are

- *Random component:* Poisson distribution and model the expected value of the response variable Y as $E(Y) = \mu$.
- *Systematic component:* The predictor variables are $x_1, \cdots, x_p$.
- *Link function:* There are two links that can be used:
  - *Identity link:* $\mu = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$. This approach is not commonly used as the linear model can yield $\mu < 0$ while μ takes only positive values.
  - *Log link:* In this case, the log link is the "canonical link" for GLMs with Poisson distribution where $\log(\mu)$ is the "natural parameter" that is defined as

$$\log(\mu) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

The *Poisson regression (Poisson log-linear) model* for counts (with a log link) is given by

$$\log(\mu) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{19}$$

INTERPRETATION OF THE REGRESSION COEFFICIENTS

For a single explanatory variable x, the Poisson log-linear model has the form

$$\log(\mu) = \beta_0 + \beta_1 x$$

The mean satisfies the exponential relationship

$$\mu = \exp(\beta_0 + \beta_1 x) = e^{\beta_0} e^{\beta_1 x} \tag{20}$$

- A one-unit increase in x has a multiplicative impact of e^{β_1} on μ. That is, the mean of Y at $x+1$ equals the mean of Y at x multiplied by factor e^{β_1}.
- An s-units increase in x has a multiplicative impact of $e^{\beta_1 s}$ on the mean of Y.
- If $\beta_1 = 0$, then $e^{\beta_1 s} = e^0 = 1$. In this case, the mean of Y does not change as x changes as the the multiplicative factor is 1.
- If $\beta_1 > 0$, then $e^{\beta_1 s} > 1$. In this case, the mean of Y increases as x increases.
- If $\beta_1 < 0$, then $e^{\beta_1 s} < 1$. In this case, the mean of Y decreases as x increases.

INTERPRETATION OF THE REGRESSION COEFFICIENTS (CONT.)

For a multiple explanatory variables $x_1, \cdots, x_p$, the Poisson log-linear model has the form

$$\log(\mu) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

The mean satisfies the exponential relationship

$$\mu = \exp\left(\beta_0 + \sum_{i=1}^{p} \beta_i x_i\right) = \left[e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_{j-1} x_{j-1} + \beta_{j+1} x_{j+1} + \cdots + \beta_p x_p}\right] e^{\beta_j x_j} \qquad (21)$$

- A one-unit increase in x_j provided all the other explanatory variables remain the same has a multiplicative impact of e^{β_j} on μ. That is, the mean of Y at $x_j + 1$ equals the mean of Y at x_j (when the other explanatory variables are hold constant) multiplied by factor e^{β_j}.
- An s-units increase in x_j has a multiplicative impact of $e^{\beta_j s}$ on the mean of Y (when the other explanatory variables remain the same).
- If $\beta_j = 0$, then $e^{\beta_j s} = e^0 = 1$. In this case, the mean of Y does not change as x_j changes (when the other explanatory variables remain the same).
- If $\beta_j > 0$, then $e^{\beta_j s} > 1$. In this case, the mean of Y increases as x_j increases (when the other explanatory variables remain the same).
- If $\beta_j < 0$, then $e^{\beta_j s} < 1$. In this case, the mean of Y decreases as x_j increases (when the other explanatory variables remain the same).

Note that this interpretation can be applied for more general case. For instant, an s-units increase in x_j and r-units increase in x_ℓ while the other x' remain the same has a multiplicative impact of $\exp\{s\beta_j + r\beta_\ell\}$ on the mean of Y.

EXAMPLE: HORSESHOE CRABS WITH COLOR AND WIDTH PREDICTORS

Consider the female horseshoe crab data set available with the name `Crabs` from the package **touchard** Each female horseshoe crab in the data set has a male crab attached to her in her nest. The data has the following variables:

- Y : the number of satellites.

- $X_1 =$ width : the female crab's shell width, in centimeters (cm), between 21 and 33.5 cm.

- $X_2 =$ color : ordinal variable (1 = medium light, 2 = medium, 3 = medium dark, 4 = dark).

- $X_3 =$ weight : weight in kg.

- $X_4 =$ spine : spine condition (1 = both good; 2 = one broken; 3 = both broken).

We want to investigate the factors that affect whether the female crab had any other males, called *satellites*, residing near her. The response variable is

$Sat = \begin{cases} 1, & \text{if the female crab has at least one satellite (other males who could mate with her).} \\ 0, & \text{if the female crab has no satellite.} \end{cases}$

In this example, we want to model the count response variable, sat using Poisson regression with a predictor width.

EXAMPLE: FEMALE HORSESHOE CRABS AND THEIR SATELLITES

```r
library("touchard") # load the package
data("Crabs") # load the data
# create the binary response variable:
# 1 = "yes" if the number of satellites is greater than one.
# 0 = "no" if the number of satellites is less than or equal to one.
Crabs <- Crabs %>% mutate(sat = ifelse(y > 0, 1, 0))
        ggplot(Crabs, aes(x = width, y = y)) + geom_point() + theme_bw() +
        ylab("Number of satellites") + xlab("Width") + stat_smooth(method = "gam",
        formula = y ~ s(x), method.args = list(family = "poisson"), se = FALSE)
```
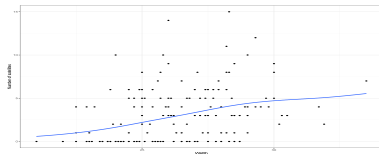

FIGURE: Number of satellites by female crab shell width in cm and generalized additive model smoothing fit.

Generalized additive models (GAM) provide more general structural form than GLMs. They find possibly complex functions of the explanatory variables that serve as the best predictors of an additive type.

EXAMPLE: GLM WITH A POISSON DISTRIBUTION AND IDENTITY LINK

If you try to GLM with a Poisson distribution and identity link, *you will get an error message*.

```
fit1 <- glm(y ~ width, family = poisson(link = identity), data = Crabs)

## Error:  no valid set of coefficients has been found:  please supply
starting values
```

EXAMPLE: LM WITH A NORMAL DISTRIBUTION AND IDENTITY LINK

If you try to fit a linear model with a normal distribution and identity link, you might get *negative* estimated mean number of satellites (for instance, when $x = 20$, $\hat{\mu} = -10.4244 + 0.5074(20) = -0.2764$), which does not make sense!

```
fit2 <- lm(y ~ width, data = Crabs)
fit2

##
## Call:
## lm(formula = y ~ width, data = Crabs)
##
## Coefficients:
## (Intercept)        width
##    -10.4244       0.5074
```

EXAMPLE: FIT POISSON DISTRIBUTION WITH LOG LINK

```
fit3 <- glm(y ~ width, family = poisson(link = log), data = Crabs)
# Default canonical link for Poisson is log, so "(link=log)" is not necessary
summary(fit3)

##
## Call:
## glm(formula = y ~ width, family = poisson(link = log), data = Crabs)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.30476    0.54224  -6.095  1.1e-09 ***
## width        0.16405    0.01997   8.216  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 632.79  on 172  degrees of freedom
## Residual deviance: 567.88  on 171  degrees of freedom
## AIC: 927.18
##
## Number of Fisher Scoring iterations: 6
```

The estimated log-linear model is

$$\log(\hat{\mu}_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i = -3.305 + 0.164 x_i$$

and the estimated mean number of satellites is

$$\hat{\mu}_i = \exp(\hat{\beta}_0 + \hat{\beta}_1 x_i) = \exp(-3.305 + 0.164 x_i)$$

For this model:

- The estimated mean number of satellites at the mean width $\bar{x} = 26.3$ cm is
  $\hat{\mu}_{|x=\bar{x}} = \exp[-3.305 + 0.164(26.3)] = 2.74$.

- A 1-cm increase in the shell width of a female crab has a multiplicative impact of
  $\exp(\hat{\beta}_1) = e^{0.164} = 1.178$. That is, a 1-cm increase in the shell width has an 17.8% increase
  in the estimated mean number of satellites.

  - For instance, the estimated mean number of satellites at shell width equals
    $x = 27.3 = \bar{x} + 1 = 26.3 + 1$ is
    $\hat{\mu} = \exp[-3.305 + 0.164(27.3)] = 3.23 = 1.178\hat{\mu}_{|x=\bar{x}} = 1.178(2.74)$.

EXAMPLE: CONFIDENCE INTERVAL

Recall that $\hat{\beta}_1 = 0.164$ and its asymptotic standard errors is $SE(\hat{\beta}_1) = 0.01997$. Thus,

- The approximate $(1 - \alpha)100\%$ confidence interval for β_1 is given by

$$\hat{\beta}_1 \pm z_{\alpha/2}SE(\hat{\beta}_1) \approx 0.164 \pm 1.96(0.01997) \longmapsto [0.125, 0.203].$$

```
confint(fit3)
##                    2.5 %      97.5 %
## (Intercept) -4.3662326 -2.2406858
## width        0.1247244  0.2029871
```

- The approximate $(1 - \alpha)100\%$ confidence interval for $\exp(\beta_1)$ is given by

$$\exp[\hat{\beta}_1 \pm z_{\alpha/2}SE(\hat{\beta}_1)] \approx \exp[0.164 \pm 1.96(0.01997)] \longmapsto [1.133, 1.225].$$

```
exp(confint(fit3))
##                    2.5 %      97.5 %
## (Intercept) 0.01269899 0.1063855
## width       1.13283621 1.2250566
```

Note that the limits of the second confidence interval are greater than 1. This means that Y is expected to increase as x increases.

For a two-class problem, one can show that for LDA

$$\log \left[\frac{\mathcal{P}_1(x)}{1 - \mathcal{P}_1(x)} \right] = \log \left[\frac{\mathcal{P}_1(x)}{\mathcal{P}_2(x)} \right] = \text{ log odds} = c_0 + c_1 x_1 + \ldots + c_p x_p$$

So it has the same form as logistic regression, where both assume that the *log odds* of the posterior probabilities is *linear* in X.

The difference is in how the parameters are estimated.

- Logistic regression uses the conditional likelihood based on $\Pr(Y \mid X)$ (known as *discriminative learning*).

- LDA uses the full likelihood based on $\Pr(X, Y)$ (known as *generative learning*).

- Despite these differences, in practice the results are often very similar.

Note: logistic regression can also fit quadratic boundaries like QDA, by explicitly including quadratic terms in the model.

Summary of the Classification Methods

In general, the choice of classification method depends on

1. The true distribution of the predictors in each of the K classes.
2. The bias-variance trade-off, sample size (n), and number of features (p).

- As seen in the previous slide, **LDA** and **logistic regression** assume that the *log odds* of the posterior probabilities is *linear* in X.
- **QDA** assumes that the *log odds* of the posterior probabilities is *quadratic* in X.
- **LDA** is simply a restricted version of **QDA** with $\Sigma_1 = \cdots = \Sigma_K = \Sigma$.
- **LDA** is a special case of **naive Bayes** and vice-versa!
- **Naive Bayes** can produce a more flexible fit, especially when p is very large.
- **LDA** assumes that the features are normally distributed with a common within-class covariance matrix, and **naive Bayes** instead assumes *independence* of the features.
- **QDA** might be more accurate in settings where *interactions* among the predictors are important in discriminating between classes.
- **LDA** is more efficient than **logistic regression** when the observations at each Kth class are normally distributed.
- **K-nearest neighbors (KNN)** will be better classifiers when decision boundary is *non-linear*, n is large, and p is small. On the other hand, if the decision boundary is *non-linear* but n and p are small, then **QDA** may be preferred to **KNN**.
- **KNN** has *low bias* but *large variance*; as such, it requires a lot of observations relative to the number of predictors.
- **KNN** does not tell us which predictors are important!

Scenario 1: Binary class response, equal observations in each class, uncorrelated predictors.

Scenario 2: Similar to Scenario 1, but the predictors had a correlation of -0.5.

Scenario 3: Predictors had a negative correlation, t-distribution (more extreme points at the tails)

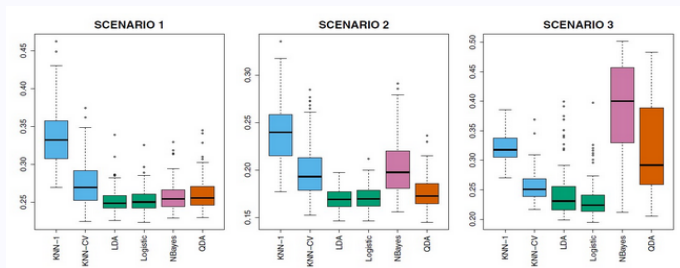Boxplots of the test error rates for each of the *linear scenarios*.

FIGURE: Source:https://r4ds.github.io/bookclub-islr/
summary-of-the-classification-methods.html.

EMPIRICAL COMPARISON: WHEN BAYES DECISION BOUNDARY IS NON-LINEAR

Scenario 4: normal distiibution, correlation of 0.5 between the predictors in the first class, and correlation of -0.5 between the predictors in the second class.

Scenario 5: Normal distribution, uncorrelated predictors.

Scenario 6: Normal distribution, different diagonal covariance matrix for each class, small n.

Boxplots of the test error rates for each of the *non-linear scenarios*.
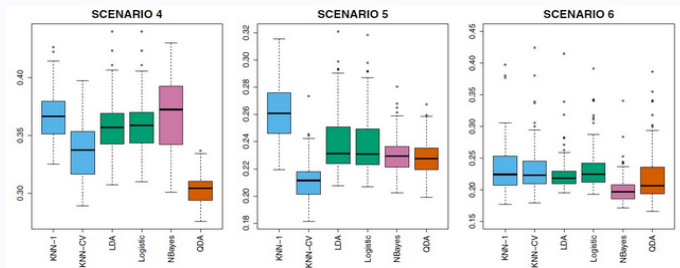

FIGURE: Source:`https://r4ds.github.io/bookclub-islr/summary-of-the-classification-methods.html`.

WHAT NEXT?

You need to

- Read and understand the examples and code in Section 4.7 Lab: Classification Methods from the textbook *"An Introduction to Statistical Learning: With Applications in R"*
- Solve question 4 in Exercise 4.8.
- Solve question 8 in Exercise 4.8.
- Solve question 12 in Exercise 4.8.
- Solve questions 13 to 16 in Exercise 4.8.