# 09 - Classification

Logistic Regression, Discriminant Analysis, and Naive Bayes

*SYS 6018 | Fall 2019*

*09-classification.pdf*

## Contents



---


Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

# 1 Classification Intro

## 1.1 Credit Card Default data (`Default`)

The textbook *An Introduction to Statistical Learning (ISL)* has a description of a simulated credit card default dataset. The interest is on predicting whether an individual will default on their credit card payment.

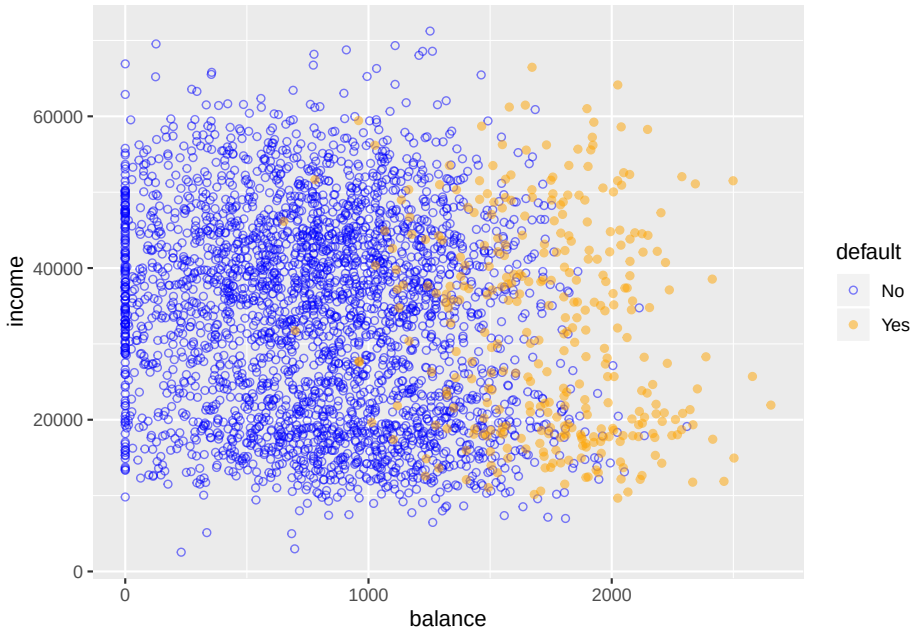
```
data(Default, package="ISLR")
```

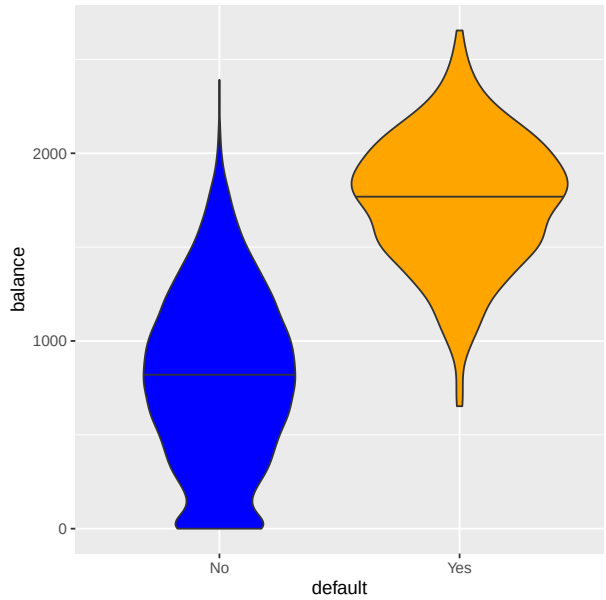
The variables are:

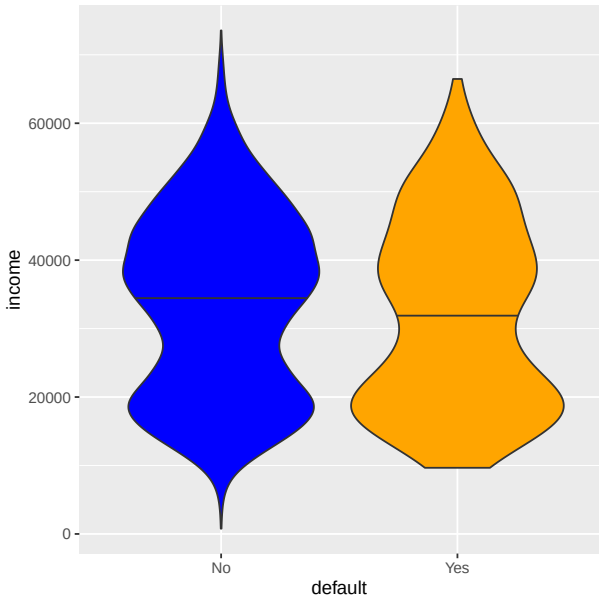
- *response variable* is categorical (factor) `Yes` and `No`, (`default`)
- the categorical (factor) variable (`student`) is either `Yes` or `No`
- the average balance a customer has after making their monthly payment (`balance`)
- the customer's income (`income`)

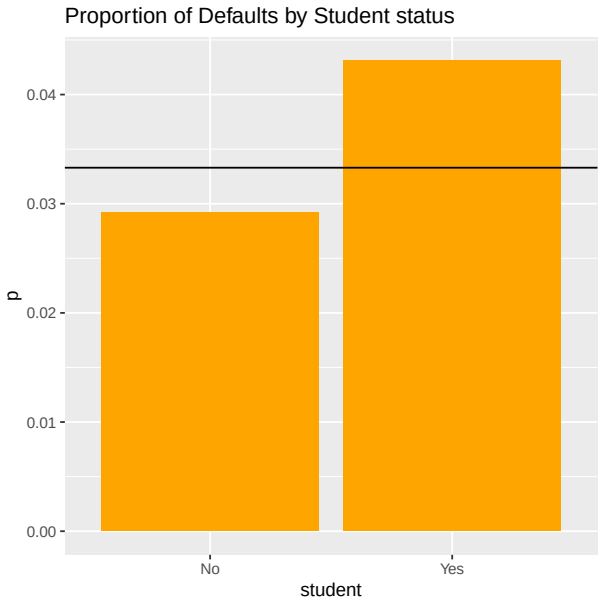
| default | student | balance | income |
|---|---|---|---|
| No | No | 729.5 | 44362 |
| No | Yes | 817.2 | 12106 |
| No | No | 1073.5 | 31767 |
| No | No | 529.3 | 35704 |
| No | No | 785.7 | 38463 |
| No | Yes | 919.6 | 7492 |

```
summary(Default)
#>  default    student       balance           income
#>  No :9667   No :7056   Min.   :   0   Min.   :  772
#>  Yes: 333   Yes:2944   1st Qu.: 482   1st Qu.:21340
#>                        Median : 824   Median :34553
#>                        Mean   : 835   Mean   :33517
#>                        3rd Qu.:1166   3rd Qu.:43808
#>                        Max.   :2654   Max.   :73554
```




balance
2000
1000
0
No
Yes
default


income
60000
40000
20000
0
No
Yes
default


Proportion of Defaults by Student status
p
0.04
0.03
0.02
0.01
0.00
No
Yes
student

**Your Turn #1 : Credit Card Default Modeling**

How would you construct a model to predict defaults?

# 2 Classification and Pattern Recognition

- The response variable is categorical and denoted $G \in \mathcal{G}$
  - Default Credit Card Example: $\mathcal{G} = \{\text{"Yes", "No"}\}$
  - Medical Diagnosis Example: $\mathcal{G} = \{\text{"stroke", "heart attack", "drug overdose", "vertigo"}\}$
- The training data is $D = \{(X_1, G_1), (X_2, G_2), \ldots, (X_n, G_n)\}$
- The optimal decision/classification is often based on the posterior probability $\Pr(G = g \mid \mathbf{X} = \mathbf{x})$

## 2.1 Binary Classification

- Classification is simplified when there are only 2 classes.
  - Many multi-class problems can be addressed by solving a set of binary classification problems (e.g., one-vs-rest).
- It is often convenient to *code* the response variable to a binary $\{0, 1\}$ variable:

$$Y_i = \begin{cases} 1 & G_i = \mathcal{G}_1 \quad \text{(outcome of interest)} \\ 0 & G_i = \mathcal{G}_2 \end{cases}$$

- In the `Default` data, it would be natural to set `default=Yes` to 1 and `default=No` to 0.

### 2.1.1 Linear Regression

- In this set-up we can run linear regression

$$\hat{y}(\mathbf{x}) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$$

```
#-- Create binary column (y)
Default = Default %>% mutate(y = ifelse(default == "Yes", 1L, 0L))

#-- Fit Linear Rergression Model
fit.lm = lm(y~student + balance + income, data=Default)
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | -0.0812 | 0.0084 | -9.685 | 0.0000 |
| studentYes | -0.0103 | 0.0057 | -1.824 | 0.0682 |
| balance | 0.0001 | 0.0000 | 37.412 | 0.0000 |
| income | 0.0000 | 0.0000 | 1.039 | 0.2990 |

**Your Turn #2 : OLS for Binary Responses**

1. For the binary $Y$, what is linear regression estimating?

2. What is the *loss function* that linear regression is using?

3. How could you create a *hard classification* from the linear model?

4. Does is make sense to use linear regression for binary classification?

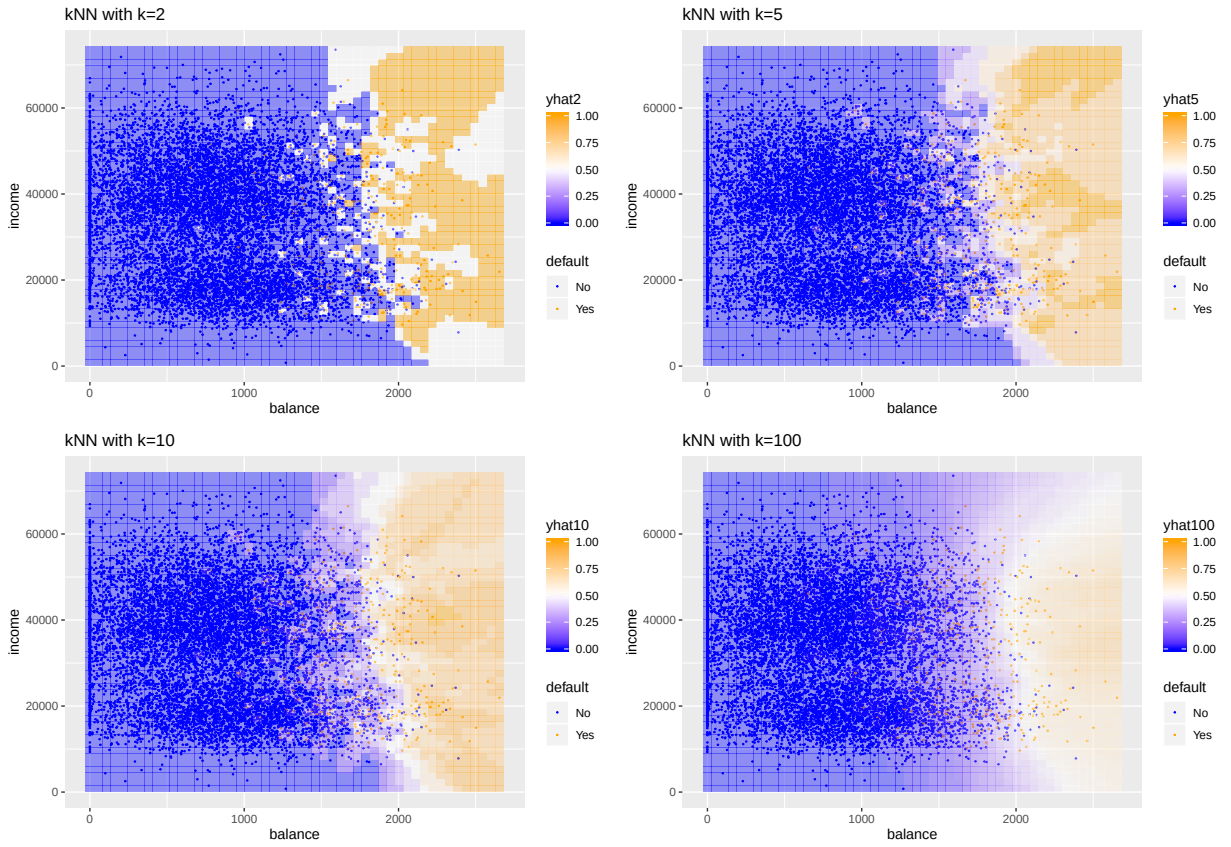
### 2.1.2 $k$-nearest neighbor (kNN)

- The $k$-NN method is a non-parametric *local* method, meaning that to make a prediction $\hat{y}|x$, it only uses the training data in the *vicinity* of $x$.
  - contrast with OLS linear regression, which uses all $X$'s to get prediction.
- The model (for regression and binary classification) is simple to describe

$$\begin{aligned} f_{\text{knn}}(x;k) &= \frac{1}{k} \sum_{i:x_i \in N_k(x)} y_i \\ &= \text{Avg}(y_i \mid x_i \in N_k(x)) \end{aligned}$$

  - $N_k(x)$ are the set of $k$ nearest neighbors
  - only the $k$ closest $y$'s are used to generate a prediction
  - it is a *simple mean* of the $k$ nearest observations
- When $y$ is binary (i.e., $y \in \{0, 1\}$), the kNN model estimates

$$f_{\text{knn}}(x;k) \approx p(x) = \Pr(Y = 1|X = x)$$




## Your Turn #3 : Thoughts about kNN

The above plots show a kNN model using the *continuous* predictors of `balance` and `income`.

- How could you use kNN with the categorical `student` predictor?

- The $k$-NN model also has a more general description when the response variables is categorical $G_i \in \mathcal{G}$

$$f_g^{\text{knn}}(x;k) = \frac{1}{k} \sum_{i:x_i \in N_k(x)} \mathbb{1}(g_i = g)$$
$$= \widehat{\Pr}(G_i = g \mid x_i \in N_k(x))$$

  - $N_k(x)$ are the set of $k$ nearest neighbors
  - only the $k$ closest $y$'s are used to generate a prediction
  - it is a *simple proportion* of the $k$ nearest observations that are of class $g$

# 3 Logistic Regression

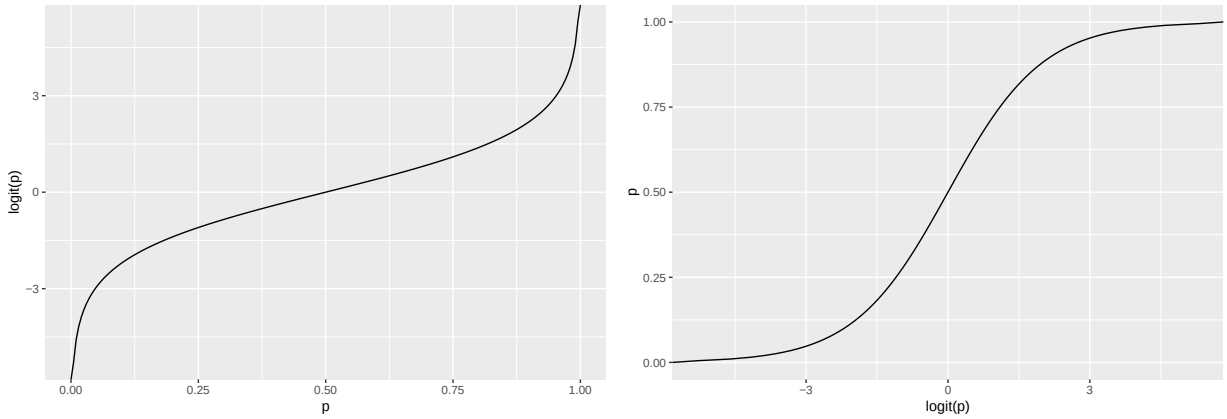
## 3.1 Basics

- Let $0 \leq p \leq 1$ be a probability.
- The log-odds of $p$ is called the *logit*

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

- The inverse logit is the *logistic function*. Let $f = \text{logit(p)}$, then

$$\begin{aligned} p &= \frac{e^f}{1+e^f} \\ &= \frac{1}{1+e^{-f}} \end{aligned}$$




- For binary response variables $Y \in \{0, 1\}$, linear regression models estimate

$$\text{E}[Y \mid X = x] = \Pr(Y = 1 \mid X = x) = \beta^\mathsf{T} x$$

- Logistic Regression models alternatively estimate

$$\log\left(\frac{\Pr(Y = 1 \mid X = x)}{1 - \Pr(Y = 1 \mid X = x)}\right) = \beta^\mathsf{T} x$$

and thus,

$$\begin{aligned} \Pr(Y = 1 \mid X = x) &= \frac{e^{\beta^\mathsf{T} x}}{1 + e^{\beta^\mathsf{T} x}} \\ &= \left(1 + e^{-\beta^\mathsf{T} x}\right)^{-1} \end{aligned}$$

## 3.2 Estimation

- The data for logistic regression is: $(\mathbf{x}_i, y_i)_{i=1}^n$ where $y_i \in \{0, 1\}$, $\mathbf{x}_i = (x_{i0}, x_{i1}, \ldots, x_{ip})^\mathsf{T}$.
- $y_i \mid \mathbf{x}_i \sim \text{Bern}(p_i(\beta))$
  - $p_i(\beta) = \Pr(Y = 1 \mid \mathbf{X} = \mathbf{x}_i; \beta) = \left(1 + e^{-\beta^\mathsf{T} \mathbf{x}_i}\right)^{-1}$

  - $\beta^\mathsf{T}\mathbf{x}_i = \mathbf{x}_i^\mathsf{T}\beta = \beta_0 + \sum_{j=1}^p x_{ij}\beta_j$

- Bernoulli Likelihood Function

$$L(\beta) = \prod_{i=1}^n p_i^{y_1}(1-p_i)^{1-y_i}$$

$$\log L(\beta) = \sum_{i=1}^n \{y_i \ln p_i + (1-y_i)\ln(1-p_i)\}$$

- The usual approach to estimating the Logistic Regression coefficients is *maximum likelihood*

$$\begin{aligned}\hat{\beta} &= \arg\max_\beta L(\beta) \\ &= \arg\max_\beta \log L(\beta)\end{aligned}$$

- We can also view this as the coefficients that minimize the *loss function*, where the loss function is the negative log-likelihood

$$\begin{aligned}\hat{\beta} &= \arg\min_\beta \ell(\beta) \\ &= -C\sum_{i=1}^n \{y_i \ln p_i + (1-y_i)\ln(1-p_i)\}\end{aligned}$$

  - where $C$ is some constant, e.g., $C = 1/n$

- This view facilitates *penalized logistic regression*

$$\hat{\beta} = \arg\min_\beta \ell(\beta) + \lambda P(\beta)$$

  - Ridge Penalty

$$P(\beta) = \sum_{j=1}^p |\beta_j|^2 = \beta^\mathsf{T}\beta$$

  - Lasso Penalty

$$P(\beta) = \sum_{j=1}^p |\beta_j|$$

  - Best Subsets

$$P(\beta) = \sum_{j=1}^p |\beta_j|^0 = \sum_{j=1}^p 1_{(\beta_j \neq 0)}$$

### 3.3 Logistic Regression in Action

- In **R**, logistic regression can be implemented with the `glm()` function since it is a type of *Generalized Linear Model*.

- Because logistic regression is a special case of *Binomial* regression, use the `family=binomial()` argument

```
#-- Fit logistic regression model
fit.lr = glm(y~student + balance + income, data=Default,
             family="binomial")
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | -10.8690 | 0.4923 | -22.0801 | 0.0000 |
| studentYes | -0.6468 | 0.2363 | -2.7376 | 0.0062 |
| balance | 0.0057 | 0.0002 | 24.7376 | 0.0000 |
| income | 0.0000 | 0.0000 | 0.3698 | 0.7115 |

**Your Turn #4 : Interpreting Logistic Regression**

1. What is the estimated probability of default for a Student with a balance of $1000?

2. What is the estimated probability of default for a *Non-Student* with a balance of $1000?

3. Why does `student=Yes` appear to lower risk of default, when the plot of student status vs. default appears to increase risk?

# 4 Linear/Quadratic Discriminant Analysis (LDA/QDA)

- Discriminant analysis seeks to find a function that will *discriminate* between class boundaries.
  - *Linear Discriminant Analysis (LDA)* finds *linear* boundaries between classes
  - *Quadratic Discriminant Analysis (QDA)* finds *quadratic* boundaries between classes

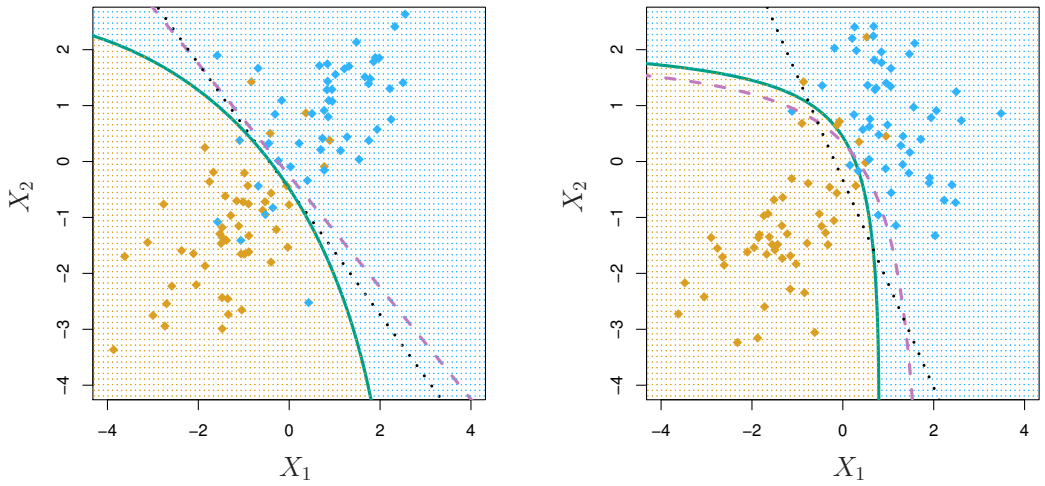



FIGURE 4.9 (from ISLR). Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\Sigma_1 = \Sigma_2$. The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA. Right: Details are as given in the left-hand panel, except that $\Sigma_1 \neq \Sigma_2$. Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.

- Suppose there are $K = |\mathcal{G}|$ classes in the training data, $D = \{(\mathbf{X}_i, G_i)\}_{i=1}^n$
  - where $\mathbf{X}_i \in \mathbf{R}^p$, $G_i \in \mathcal{G}$
- Consider the posterior probability of class $g$, given $X = x$,

$$\begin{aligned}\Pr(G = g \mid \mathbf{X} = \mathbf{x}) &= \frac{f(x \mid G = g)\Pr(G = g)}{f(x)} \\ &= \frac{f_g(x)\pi_g}{\sum_{k=1}^K f_k(x)\pi_k}\end{aligned}$$

  - $f_k(x)$ is the *class conditional density*
  - $0 \leq \pi_k \leq 1$ are the *prior class probabilities*
  - $\sum_{k=1}^K \pi_k = 1$
- The challenge is to estimate the densities $\{f_k(\cdot)\}$
  - Note: $\hat{\pi}_k = n_k/n$ is a natural estimate for the class priors if we think the testing data will have the same proportions as the training data

## 4.1 Estimation

- LDA and QDA have strong connections to model based clustering.
  - But easier, since we have the true class labels in the supervised setting
- Both LDA and QDA model the class conditional densities $f_k(x)$ with *Gaussians*
  - Thus, they model the observations as coming from a *Gaussian mixture model*
  - Each class has its own mean vector $\mu_k$
  - The difference between LDA and QDA is what they use for their covariance matrix

- **LDA**

$$f_k(x) = (2\pi)^{-p/2}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\mu_k)\right\}$$

  - $\Sigma_k = \Sigma \quad \forall k$ (*uses the same variance-covariance for all classes*)
- **QDA**

$$f_k(x) = (2\pi)^{-p/2}|\Sigma_k|^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^\mathsf{T}\Sigma_k{}^{-1}(\mathbf{x}-\mu_k)\right\}$$

  - $\Sigma_k$ is *different* for each classes

**Your Turn #5 : Model Complexity**

The LDA model uses a common covariance matrix while QDA allows each class to have a different covariance (which permits quadratic boundaries). But this flexibility comes at a cost.

1. How many parameters have to be estimated in an LDA model with $K$ classes and $p$ dimensions?

2. How many parameters have to be estimated in an QDA model with $K$ classes and $p$ dimensions?

- There are a few methods to maintain some flexibility, yet protect the model from high variance
- One is to use a *regularlized covariance matrix* (see ESL 4.3.1)

$$\hat{\Sigma}_k(\alpha,\gamma) = \alpha\hat{\Sigma}_k + (1-\alpha)\{\gamma\hat{\Sigma} + (1-\gamma)\hat{\sigma}^2 I_p\}$$

- Another is to fit an LDA model in an *enlarged feature space*
  - E.g., for $p = 2$ dimensions, use $X_1, X_2, X_1 \cdot X_2, X_1^2, X_2^2$ instead of QDA in $X_1, X_2$.
  - Think basis expansion like what we say with polynomial regression or B-splines

### 4.2 LDA/QDA in Action

- In **R**, LDA and QDA can be implemented with the `lda()` and `qda()` functions from the `MASS` package.
- See ISLR 4.6 for details

## 5 Evaluating Classification Models

- Training Data: $\{X_i, G_i\}$
  - $G_i \in \{1, \ldots, K\}$ (i.e., there are $K$ classes)
- Predictor: $\hat{G}(X)$
- Loss function: $L(G, \hat{G}(X))$ is the loss incurred by estimating $G$ with $\hat{G}$

- Risk is the expected loss (or expected prediction error EPE)
  - Expectation is taken wrt future values of $(X, G)$

$$\begin{aligned}\text{Risk}(\hat{G}) &= \text{EPE} \\ &= \text{E}_{XG}\left[L(G, \hat{G}(X))\right] \\ &= \text{E}_X\left[\text{E}_{G|X}\left[L(G, \hat{G}(X)) \mid X\right]\right] \\ &= \text{E}_X\left[R_X(\hat{G})\right]\end{aligned}$$

- The Risk at input $X = x$ is

$$\begin{aligned}R_x(\hat{G}) &= \text{E}_{G|X=X}\left[L(G, \hat{G}(x)) \mid X = x\right] \\ &= \sum_{k=1}^{K} L(G = k, \hat{G}(x)) \Pr(G = k \mid X = x)\end{aligned}$$

- Thus the optimal class label, given $X = x$, is

$$\hat{G}(x) = \arg\min_g R_x(g)$$

## 5.1 Evaluation of Binary Classification Models

- We are considering *binary* outcomes, so use the notation $Y \in \{0, 1\}$
- Let $p(x) = \Pr(Y = 1 \mid X = x)$
- The Risk (for a binary outcome) is:

$$\begin{aligned}R_x(g) &= L(1, g) \Pr(Y = 1 \mid X = x) + L(0, g)(1 - \Pr(Y = 1 \mid X = x)) \\ &= L(1, g)p(x) + L(0, g)(1 - p(x))\end{aligned}$$

- Decision: choose $\hat{G}(x) = 1$ if

$$\begin{aligned}R_x(1) &< R_x(0) \\ L(1,1)p(x) + L(0,1)(1-p(x)) &< L(1,0)p(x) + L(0,0)(1-p(x)) \\ p(x)\left(L(1,1) - L(1,0)\right) &< (1-p(x))\left(L(0,0) - L(0,1)\right) \\ p(x)\left(L(1,0) - L(1,1)\right) &\geq (1-p(x))\left(L(0,1) - L(0,0)\right) \qquad (\textit{multiply both sides by } -1) \\ \frac{p(x)}{1-p(x)} &\geq \frac{L(0,1) - L(0,0)}{L(1,0) - L(1,1)}\end{aligned}$$

### 5.1.1 Example

- Say we have a goal of estimating if a patient has cancer using medical imaging
  - Let $G = 1$ for cancer and $G = 0$ for no cancer
- Suppose we have solicited a loss function with the following values
  - $L(G = 0, \hat{G} = 0) = 0$: There is no loss for correctly diagnosis a patient without cancer
  - $L(G = 1, \hat{G} = 1) = 0$: There is no loss (for our model) for correctly diagnosis a patient with cancer
  - $L(G = 0, \hat{G} = 1) = FP$: There is a cost of $FP$ units if the model issues a *false positive*, estimating the patient has cancer when they don't
  - $L(G = 1, \hat{G} = 0) = FN$: There is a cost of $FN$ units if the model issues a *false negative*, estimating the patient does not have cancer when they really do

  - In these scenarios $FN$ is often much larer than $FP$ ($FN >> FP$) because the the effects of not promptly treating (or further testing, etc) a patient is more severe than starting a treatment path for patients that don't actually have cancer

- Our model will decide to issue a positive indication for cancer if $R_x(1) < R_x(0)$ which occurs when

$$\frac{p(x)}{1-p(x)} \geq \frac{FP}{FN}$$
$$p(x) \geq \frac{FP}{FP+FN}$$

- The ratio of FP to FN is all that matters for the decision. Let's say that FP=1 and FN=10. Then if $p(x) \geq 1/11$, our model will diagnose cancer.
  - Note: $p(x) = \Pr(Y = 1|X = x)$ is affected by the class prior $Pr(Y = 1)$ (e.g., the portion of the population tested with cancer), which is usually going to be small.

## 5.2 Common Binary Loss Functions

- Suppose we are going to estimate a binary reponse $Y \in \{0, 1\}$ with a (possibly continuous) predictor $\hat{y}(x)$
- **0-1 Loss** or **Misclassification Error**

$$L(y, \hat{y}(x)) = \mathbb{1}(y \neq \hat{y}(x)) = \begin{cases} 0 & y = \hat{y}(x) \\ 1 & y \neq \hat{y}(x) \end{cases}$$

  - This assumes $L(0, 1) = L(1, 0)$ (i.e., false positive costs the same as a false negative)
  - Requires that a *hard classification* is made
  - The optimal prediction is $y^*(x) = \mathbb{1}(p(x) > 1 - p(x))$ which is $\mathbb{1}(p(x) > 0.50)$

- **Squared Error**

$$L(y, \hat{y}(x)) = (y - \hat{y}(x))^2$$

  - The optimal prediction is $y^*(x) = \mathrm{E}[Y \mid X = x] = \Pr(Y = 1 \mid X)$

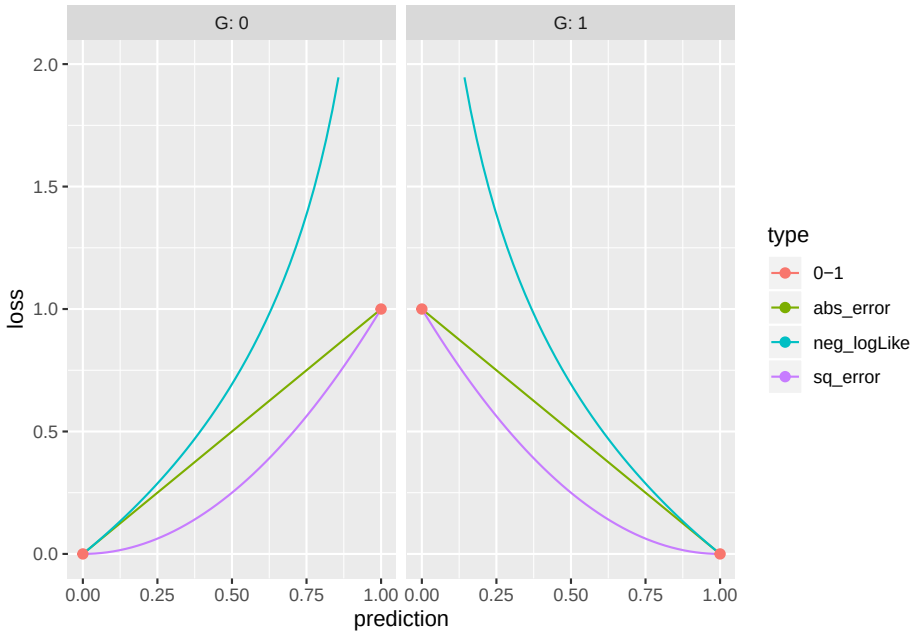
- **Absolute Error**

$$L(y, \hat{y}(x)) = |y - \hat{y}(x)|$$

- **Bernoulli negative log-likelihood**

$$\begin{aligned} L(y, \hat{y}(x)) &= -\{y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\} \\ &= \begin{cases} -\log \hat{y} & y = 1 \\ -\log(1 - \hat{y}) & y = 0 \end{cases} \end{aligned}$$

  - Requires $\hat{y}(x) \in [0, 1]$




## 5.3 Evaluating Binary Classification Models

- Recall, the optimal (*hard classification*) decision is to choose $\hat{G} = 1$ if:

$$\frac{p(x)}{1-p(x)} \geq \frac{L(0,1) - L(0,0)}{L(1,0) - L(1,1)}$$

- Denote $\gamma(x)$ as the *logit* of $p(x)$:

$$\gamma(x) = \log \frac{p(x)}{1-p(x)} = \log \frac{\Pr(G=1 \mid X=x)}{\Pr(G=0 \mid X=x)}$$

- Then we get

$$\begin{aligned} p(x) &= \Pr(G=1 \mid X=x) \\ &= \frac{e^{\gamma(x)}}{1+e^{\gamma(x)}} \end{aligned}$$

- And the optimal (*hard classification*) decision can be described in terms of $\gamma(x)$:

> Choose $\hat{G}(x) = 1$ if $\hat{\gamma}(x) > t$, where $t$ is a threshold

- If the loss/cost is known, then

$$t^* = \log\left(\frac{L(0,1) - L(0,0)}{L(1,0) - L(1,1)}\right)$$

- For a given threshold $t$ and input $x$, the hard classification is $\hat{G}_t(x) = \mathbb{1}(\hat{\gamma}(x) \geq t)$

## 5.4 Performance Metrics

### 5.4.1 Metrics

| Metric | Definition | Estimate |
|---|---|---|
| Mis-classification Rate | $P_{XG}(\hat{G}_t(X) \neq G(X)) =$ $P_X(\hat{G}_t(X) = 0, G(X) = 1) +$ $P_X(\hat{G}_t(X) = 1, G(X) = 0)$ | $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{G}_t(x_i) \neq G_i)$ |
| False Positive Rate (FPR) {1-*Specificity*} | $P_X(\hat{G}_t(X) = 1 \mid G(X) = 0)$ | $\frac{1}{N_0}\sum_{i:G_i=0} \mathbb{1}(\hat{G}_t(x_i) = 1)$ |
| True Positive Rate (TPR) {*Hit Rate, Recall, Sensitivity*} | $P_X(\hat{G}_t(X) = 1 \mid G(X) = 1)$ | $\frac{1}{N_1}\sum_{i:G_i=1} \mathbb{1}(\hat{G}_t(x_i) = 1)$ |
| Precision | $P_X(G(X) = 1 \mid \hat{G}_t(X) = 1)$ | $\frac{1}{\hat{N}_1(t)}\sum_{i:\hat{G}(x_i)=1} \mathbb{1}(G_i = 1)$ |

- Note: Performance estimates are best carried out on *hold-out* data!

### 5.4.2 Confusion Matrix

- Given a threshold $t$, we can make a *confusion matrix* to help analyze our model's performance on data
  - Data = $\{(X_i, G_i)\}_{i=1}^N$ (ideally this is hold-out/test data)
  - $N_k$ is number of observations from class $k$ ($N_0 + N_1 = N$)

| | | **Model Outcome** $\hat{G}_t = 1$ | $\hat{G}_t = 0$ | **total** |
|---|---|---|---|---|
| **True Outcome** | $G = 1$ | True Positive (TP) | False Negative (FN) | $N_1$ |
| | $G = 0$ | False Positive (FP) | True Negative (TN) | $N_0$ |
| | **total** | $\hat{N}_1(t)$ | $\hat{N}_0(t)$ | $N$ |

Table from: https://tex.stackexchange.com/questions/20267/how-to-construct-a-confusion-matrix-in-latex

- See Wikipedia Page: Confusion Matrix for more metrics

## 5.5 Performance over a range of thresholds

To illustrate how we can calculate and visualize different performance aspects in binary classification let's go back to the `Default` data and see how well the basic logistic regression models works.

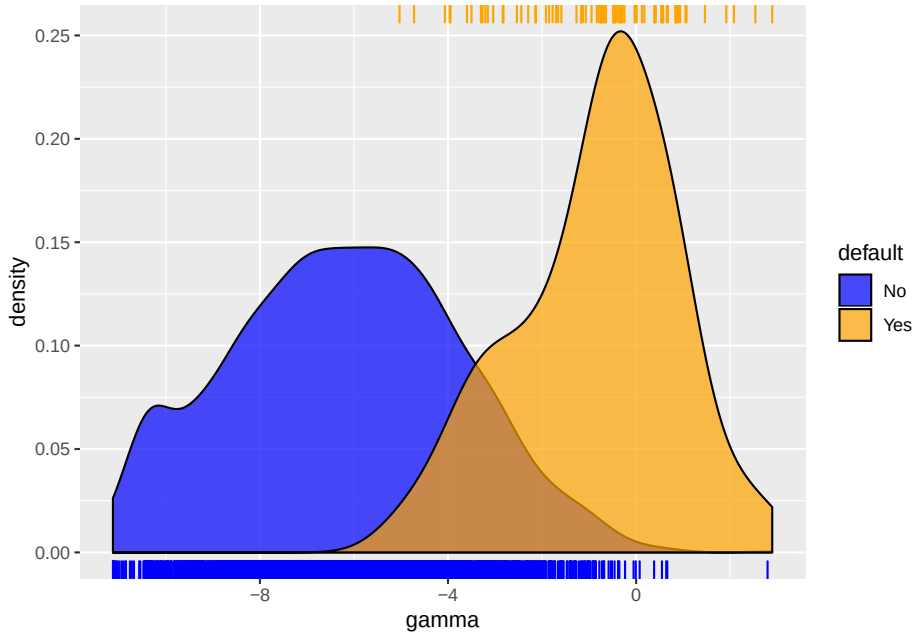
- In order to evaluation on hold-out data, split the data into train/test (used about 75% training, 25% testing), fit a logistic regression model on training data, and make predictions on the test data

```
#-- train/test split
set.seed(2019)
Default = Default %>%
  mutate(group = sample(c('train', 'test'), size=nrow(.),
                        replace=TRUE, prob=c(.75, .25) )) # ~75% train

#-- fit model on training data
fit.lm = glm(y~student + balance + income,
             family='binomial',
             data=filter(Default, group=='train'))

#-- Get predictions (of gamma(x)) on test data
gamma = predict(fit.lm,
                newdata=filter(Default, group=='test'),
                type='link')
Gtest = filter(Default, group=='test') %>% pull(y) # true values
```

- The model is unable to perfectly discriminate between groups, but the *defaults* do get scored higher in general:
  - As a reference point, note that $\hat{\gamma}(x) = 0 \rightarrow \Pr(Y = 1 \mid X = x) = 1/2$

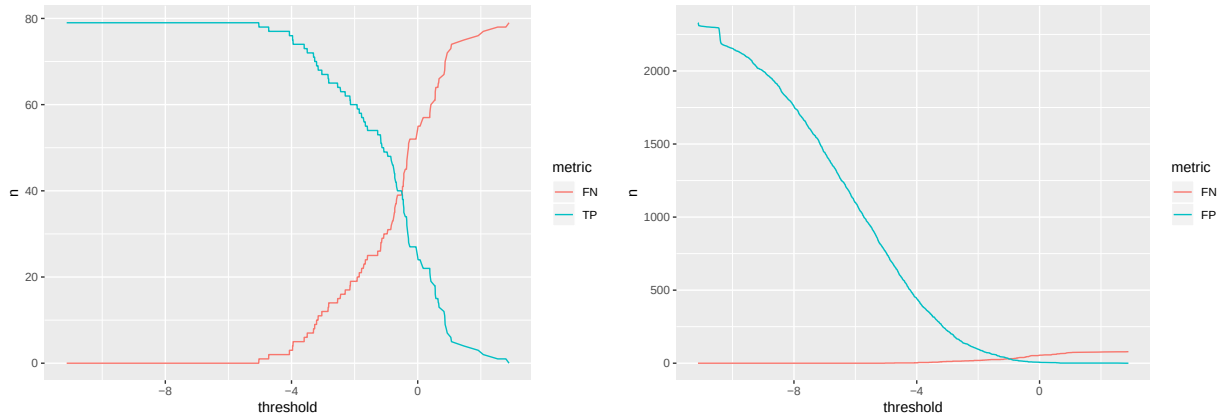



- We can calculate performance over a range of thresholds.
  - Unless the test data is too large, use all unique values of the training data as the tresholds

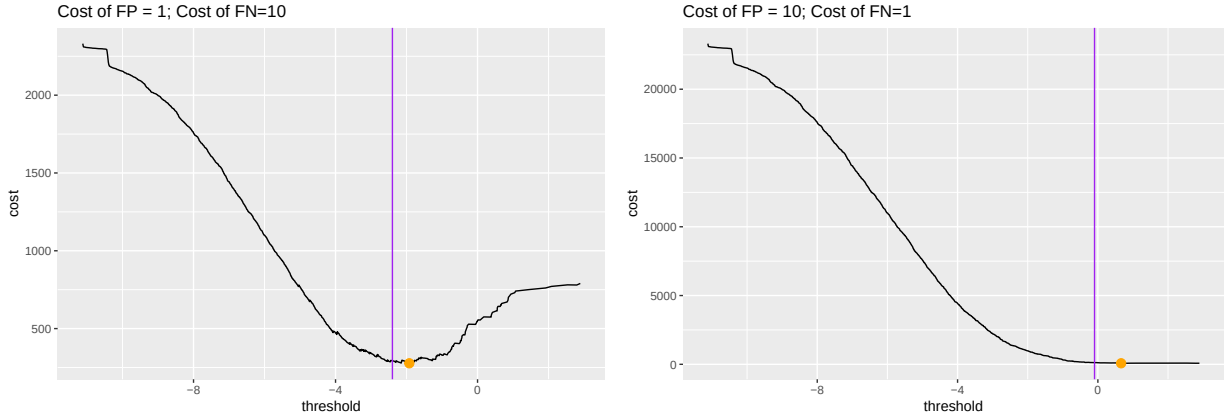
```
#-- Get performance data (by threshold)
perf = tibble(truth = Gtest, prediction = gamma) %>%
  #- group_by() + summarize() in case of ties
  group_by(prediction) %>%
  summarize(n=n(), n.1=sum(truth), n.0=n-sum(truth)) %>%
  #- calculate metrics
  arrange(prediction) %>%
  mutate(FN = cumsum(n.1),     # false negatives
```

```
          TN = cumsum(n.0),     # true positives
          TP = sum(n.1) - FN,   # true positives
          FP = sum(n.0) - TN,   # false positives
          N = cumsum(n),        # number of cases predicted to be 1
          TPR = TP/sum(n.1), FPR = FP/sum(n.0)) %>%
  #- only keep relevant metrics
  select(-n, -n.1, -n.0, threshold=prediction)
```

- **General Performance** (select metrics)



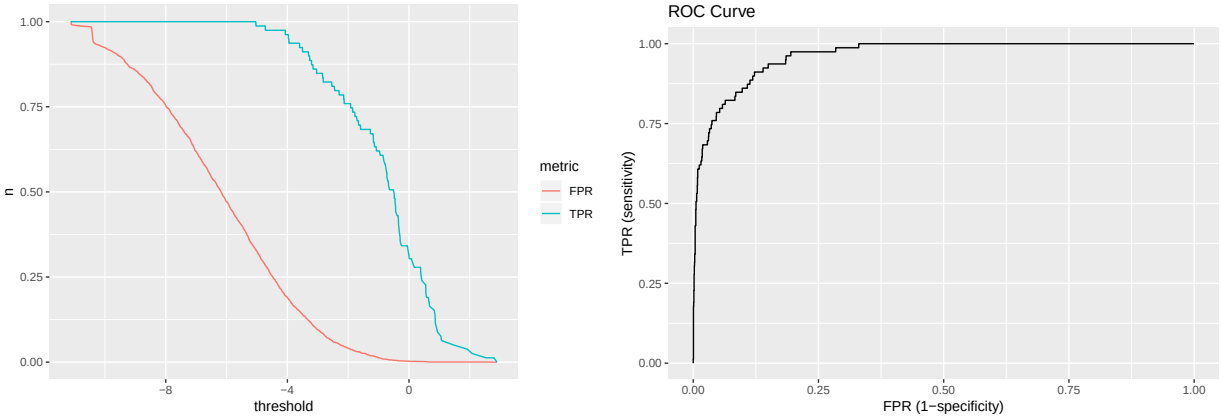

- **Cost Curves**
  - note: the purple is the *theoretical* optimal threshold (using $t^* = \log FP/FN$) and the orange point is at the optimal value using the model




## Optimal Threshold

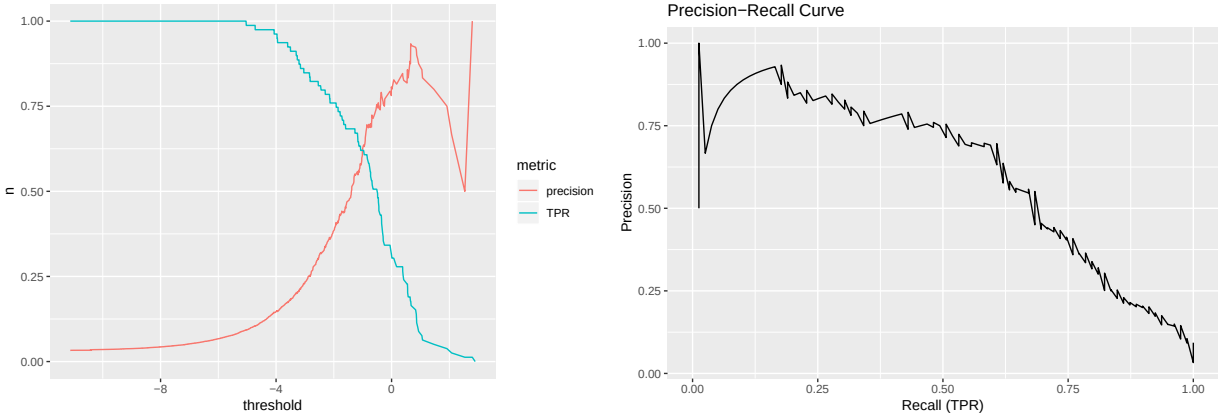
- The *theoretically* optimal threshold is based on the *true* $\gamma(x) = \log \frac{p(x)}{1-p(x)}$ (for a given cost ratio of FP to FN)
- The observed optimal threshold will differ when the model's estimate $\hat{\gamma}(x) \neq \gamma(x)$
  - Hopefully, they are close and it won't make much difference which one you use. But I'd take the estimated threshold if I had sufficent data.
- Note that the estimated values depend on the prior class probabilities. If you suspect these may differ in the future, then you should adjust the threshold.

- **ROC Curves** (Receiver Operating Characteristic)




- The *area under the ROC curve* (AUC) is a popular performance metric
- I don't think it is a great way to compare classifiers for several reasons
  - The main reason is that in a real application you can almost always come up with an estimated cost/loss for the different decisions
  - To say it another way, comparisons should be made at a single point on the curve; the entire FPR region should not factor into the comparison.

- **Precision Recall Curves**
  - Popular for information retrieval/ranking




# 6 Naive Bayes and Generalized Additive Models (GAM)

## 6.1 The Bayes Breakdown

Recall our notation for binary classification problems

- $p(x) = \Pr(Y = 1 \mid X = x) = \frac{f_1(x)\pi}{f_1(x)\pi + f_0(x)(1-\pi)}$
- $\gamma(x) = \frac{p(x)}{1-p(x)}$
- $f_k(x)$ is the *class conditional density*
- $0 \leq \pi_k \leq 1$ are the *prior class probabilities*

- $\pi_0 + \pi_1 = 1$

The log-odds reduces to a combination of prior odds and density (likelihood) ratios

$$\begin{aligned}\gamma(x) &= \log\left(\frac{p(x)}{1-p(x)}\right) \\ &= \underbrace{\log\left(\frac{\pi}{1-\pi}\right)}_{\text{log prior odds}} + \underbrace{\log\left(\frac{f_1(x)}{f_0(x)}\right)}_{\text{log density ratio}}\end{aligned}$$

- Note: $\frac{f_1(x)}{f_0(x)}$ is usually called a *likelihood ratio* when estimated via MLE and *Bayes Factor* when integrating over the model parameters
- We can see that the optimal decision can be based on the density ratios

$$\begin{aligned}&\text{Choose } \hat{G}(x) = 1 \text{ if:} \\ &\hat{\gamma}(x) > \log\left(\frac{L(0,1) - L(0,0)}{L(1,0) - L(1,1)}\right) \\ &= \log\left(\widehat{\frac{f_1(x)}{f_0(x)}}\right) > \log\left(\frac{1-\hat{\pi}}{\hat{\pi}}\right) + \log\left(\frac{L(0,1) - L(0,0)}{L(1,0) - L(1,1)}\right)\end{aligned}$$

## 6.2 Naive Bayes

- Recall in LDA/QDA, the class conditional densities were estimated as Gaussians:

$$\hat{f}_k(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \hat{\mu}_k, \hat{\Sigma}_k)$$

  - But when the dimensionality of $\mathbf{x}$ gets large or there is high correlation, estimation of $\hat{\Sigma}_k$ can be poor
  - Also if the multivariate densities are not unimodal these models may not have good predictive ability
- If we force $\hat{\Sigma}_k$ to be *diagonal* then the densities are product of univariate Gaussians

$$\hat{f}_k = \prod_{j=1}^{p} \mathcal{N}(x_j; \mu_{kj}, \sigma_{kj})$$

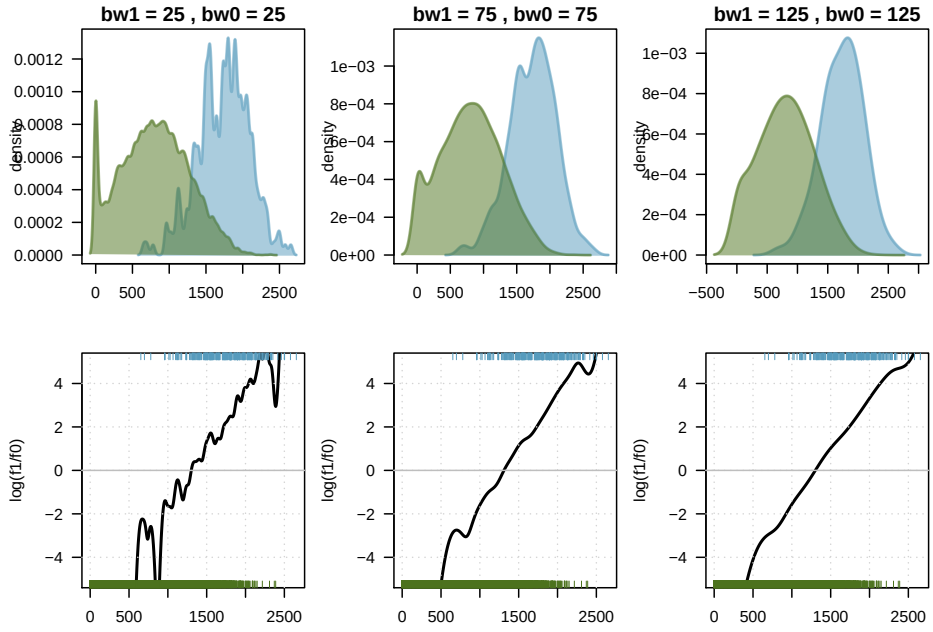
  - Even if the data are not independent, this may give better estimates by reducing the variance (at the expense of a bit of bias)
  - This is an exmaple of *Naive Bayes*
- The so called **Naive Bayes** approach just models the joint multivariate densities as the product of marginal densities

$$\hat{f}_k = \prod_{j=1}^{p} \hat{f}_{kj}(x_j)$$

  - The densities do *not* have to be Gaussian
  - Categorical densities (i.e., pmfs) can be thrown in the mix without a problem
  - Because of the independence, this is easy to implement in parallel (and thus can be fast)

### 6.2.1 KDE based Naive Bayes

- Kernel density estimation (KDE) can be a great way to estimate the component densities
- The bandwidth tuning parameter allows a flexible shape
  - Using the `balance` predictor in the `Default` data




## 6.3 Logistic Regression vs. Linear Discriminant Analysis (LDA) vs. Naive Bayes

It turns out that there is a close connection between Logistic Regression, Naive Bayes, and LDA. To help see this, notice that all three methods can be written:

$$\begin{aligned}\gamma(x) &= \log\left(\frac{\pi}{1-\pi}\right) + \log\left(\frac{f_1(x)}{f_0(x)}\right) \\ &= \alpha_0 + \sum_{j=1}^{p} \alpha_j S_j\end{aligned}$$

- **Logistic Regression**

$$\begin{aligned}\hat{\alpha}_0 &= \hat{\beta}_0 \\ \hat{\alpha}_j &= \hat{\beta}_j \\ \hat{S}_j &= x_j\end{aligned}$$

- **LDA**

$$\begin{aligned}\hat{\alpha}_0 &= \log\frac{\hat{\pi}}{1-\hat{\pi}} - \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_0)^{\mathsf{T}}\hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0) \\ \hat{\alpha}_j &= \hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_0) \\ \hat{S}_j &= x_j\end{aligned}$$

- **Naive Bayes**

$$\hat{\alpha}_0 = \log \frac{\hat{\pi}}{1-\hat{\pi}}$$
$$\hat{\alpha}_j = 1$$
$$\hat{S}_j = \log \frac{\hat{f}_{1j}(x_j)}{\hat{f}_{0j}(x_j)}$$

- **Generalized Additive Models (GAM)**
  - GAM models are made to directly estimate models of this form.

$$\hat{\gamma}(x) = \hat{\alpha} + \sum_{j=1}^{p} \hat{g}_j(x_j)$$

  - In **R**, the `mgcv` package is worth becoming familiar with to implement GAM.
  - See ESL 9.1 for more details