

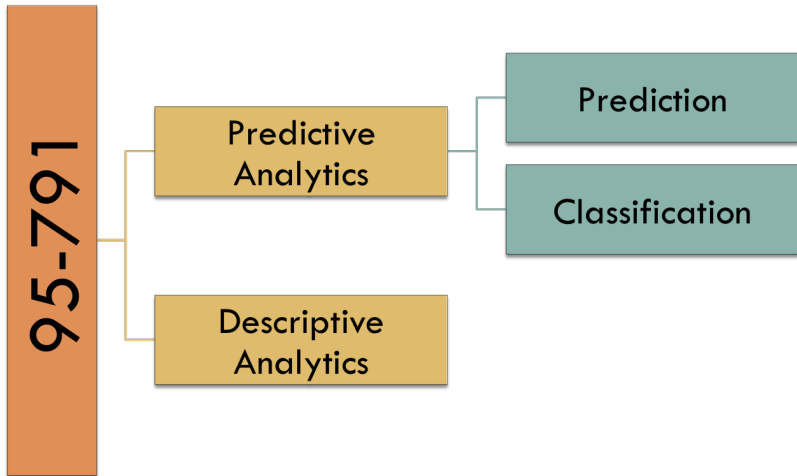
Lecture 4: Classification

Part I: Classification methods

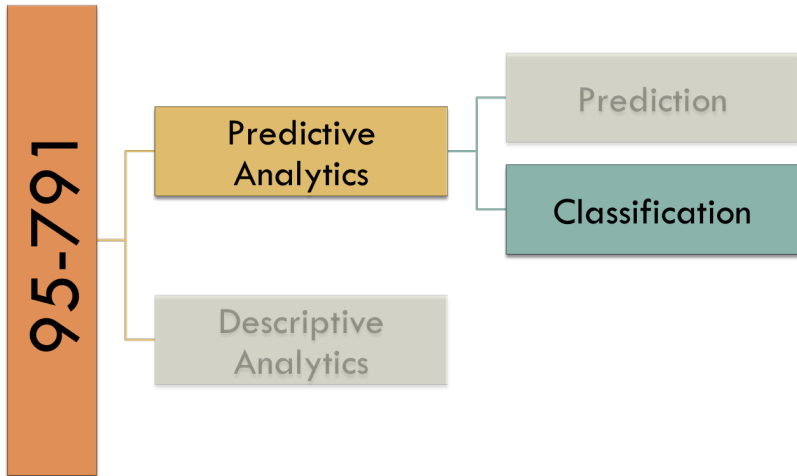
Part II: Assessing classifier performance

Prof. Alexandra Chouldechova
95-791: Data Mining

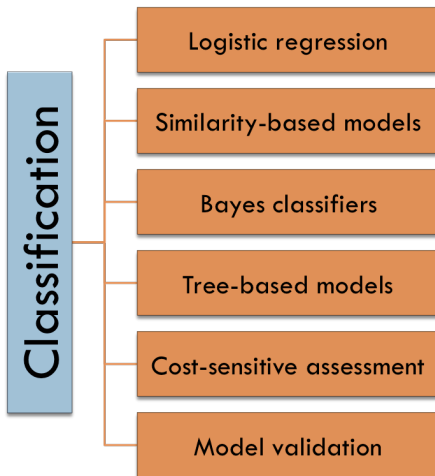
Course Roadmap



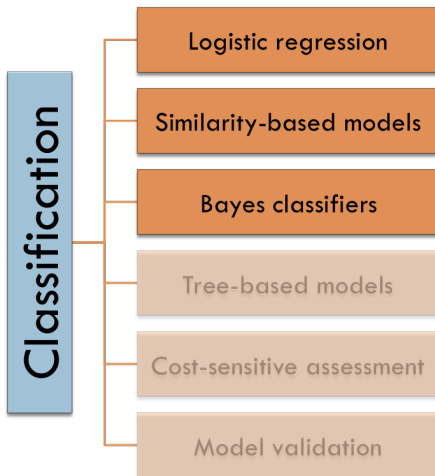
Course Roadmap



Prediction topics



Prediction topics



Agenda for Part I

- **Decision Boundary for Logistic Regression**
- **Nearest-Neighbours methods**
- **Bayes Methods**

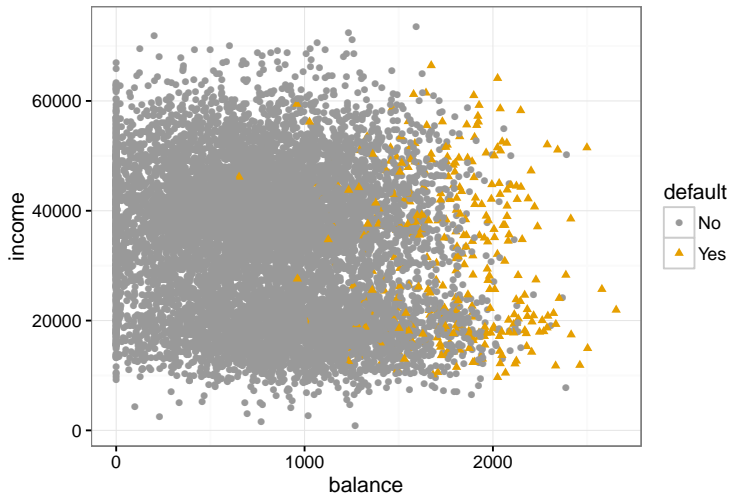
Logistic regression as a classifier

- Last class we saw how to *interpret* logistic regression
- But what does it look like as a **classifier**?
- i.e., What does the **decision rule**

$$\hat{Y} = \begin{cases} 1 & \text{if } \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 \text{balance} + \hat{\beta}_2 \text{income}}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 \text{balance} + \hat{\beta}_2 \text{income}}} > \alpha \\ 0 & \text{otherwise} \end{cases}$$

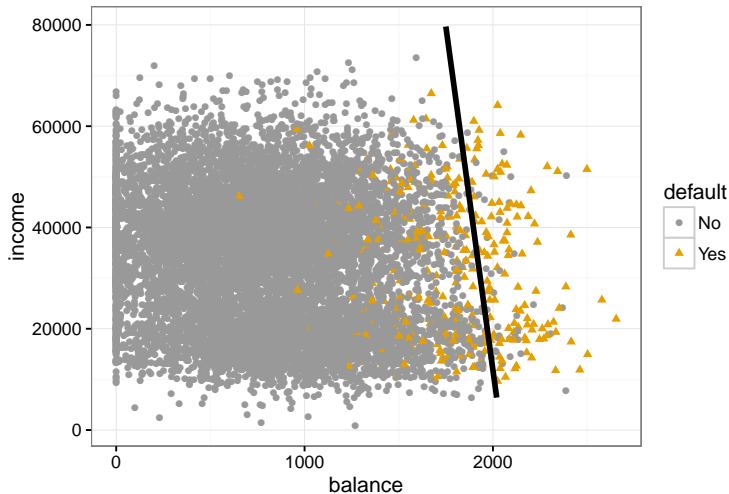
actually look like for various choices of the cutoff α ?

Logistic regression: Decision boundary



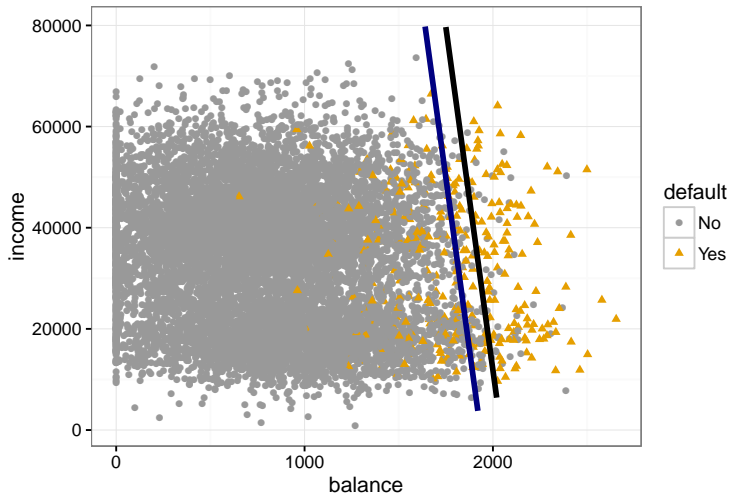
This is what the data actually looks like (previous scatterplot *undersampled* the `default = No` group.)

Logistic regression: Decision boundary



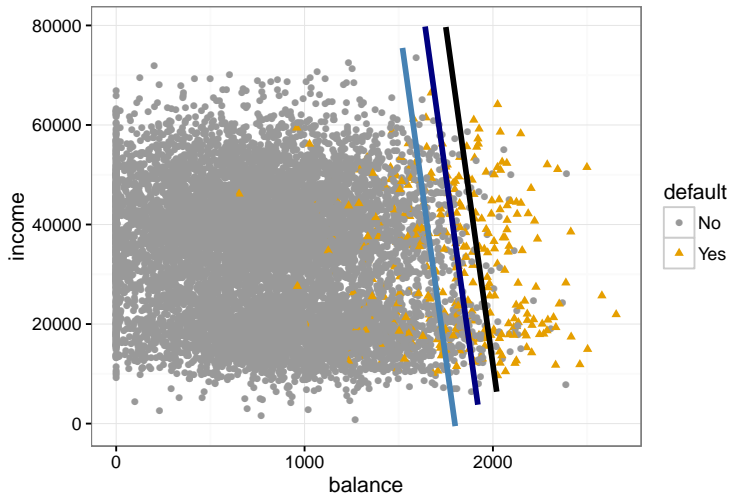
Black line shows **decision boundary** for the rule: $\hat{Y} = 1$ if $\hat{p}(x) > 0.5$
Points to the **right** of the boundary get classified as **default = Yes**

Logistic regression: Decision boundary



Navy line shows **decision boundary** for the rule: $\hat{Y} = 1$ if $\hat{p}(x) > 0.35$
Points to the **right** of the boundary get classified as **default = Yes**

Logistic regression: Decision boundary



Blue line shows decision boundary for the rule: $\hat{Y} = 1$ if $\hat{p}(x) > 0.2$
Points to the right of the boundary get classified as default = Yes

More general decision boundaries

- We see that the **decision boundary** provided by logistic regression turns out to be **linear**:
 - Points on one side of the boundary get classified as $\hat{Y} = 1$
 - Points on the other side get classified as $\hat{Y} = 0$
- When we have $p > 2$ covariates, instead of getting a **line**, we get a **(hyper)-plane**
- Just as we don't think that linear models are accurate representations of numeric outcomes, we may not believe that a **linear decision boundary** is the best way to classify points

k-Nearest Neighbours Classifier

- Setup: Data (x_i, y_i) , $x_i \in \mathbb{R}^p$ vector of inputs, $y_i \in \{0, 1\}$.
- **k-Nearest Neighbours** (k-NN) classification is an example of a **non-parametric** “lazy learning” (**memory-based**) method
- Unlike the methods we've seen before¹, which estimate **parameters** in some model, k-NN looks at the training data each time it is queried to make a classification²
- Let $\mathcal{N}_k(x)$ denote the k training points that are *closest* to x
- If we want to classify a new individual with covariates $X = x$, we simply classify it to the **majority class** of the points in $\mathcal{N}_k(x)$
- As an **estimator** of the conditional probability, we can think of k-NN as

$$\hat{p}_{\text{kNN}}(x) = \frac{1}{k} \sum_{x_i \in \mathcal{N}_k(x)} y_i$$

¹With the exception of **local regression**, which also has these properties.

²k-NN also works for prediction, though we didn't discuss it at the time

k-Nearest Neighbours Classifier: 3-NN example

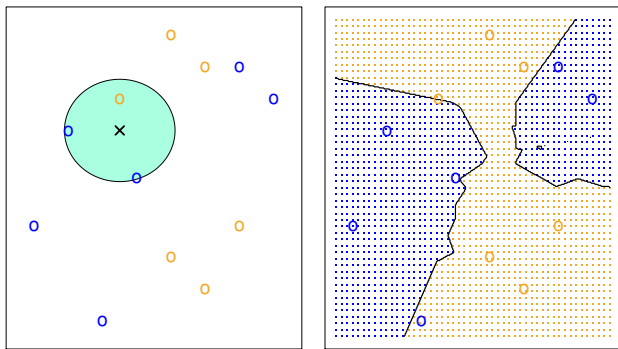


Figure: 2.14 from ISL. Axes represent $p = 2$ predictors.

- *Left:* We have two classes: $Y \in \{\text{orange, blue}\}$. 3-NN is used to classify the \times as a blue point. 2 of \times 's 3 nearest neighbours are blue, one is orange, so blue wins.
- *Right:* The decision boundary. Any point in the orange shaded region gets classified by 3-NN as orange. Any point in the blue shaded region area gets classified by 3-NN as blue.

k-Nearest Neighbours: Simulated data

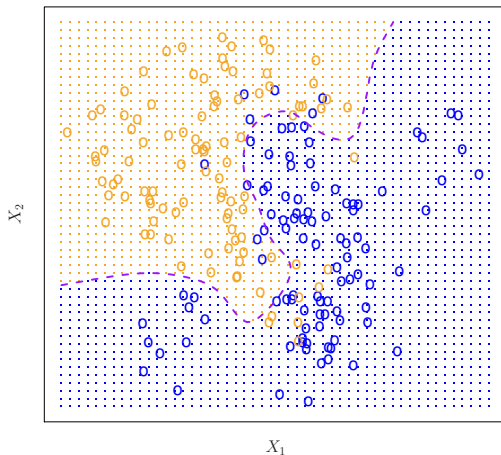


Figure 2.15 from ISL. Dashed purple line is the Bayes classifier “optimal” decision boundary.

k-Nearest Neighbours: Simulated data

KNN: K=1

KNN: K=100

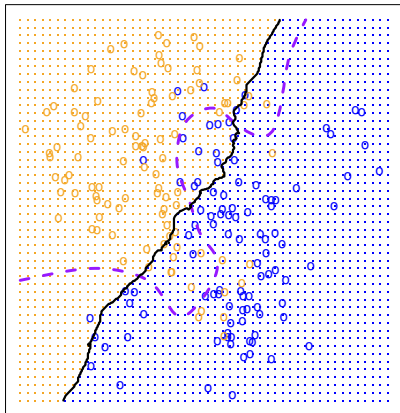
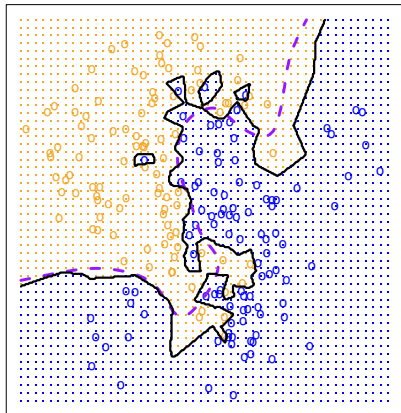


Figure 2.15 from ISL. Dashed purple line is the Bayes classifier “optimal” decision boundary. Solid black lines are 1-NN and 100-NN classifiers.

Can we get non-linear boundaries from Logistic regression?

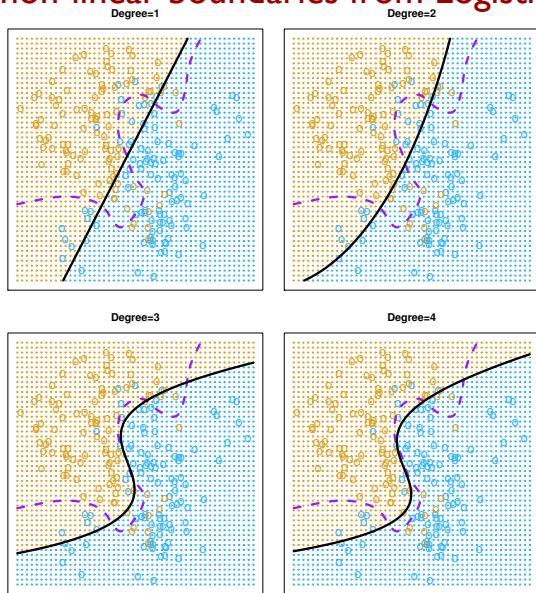
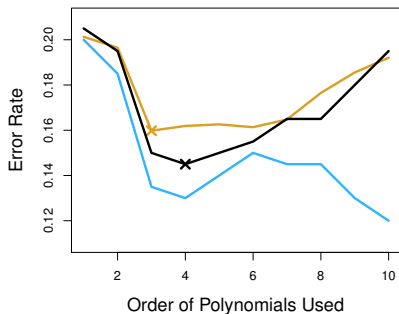


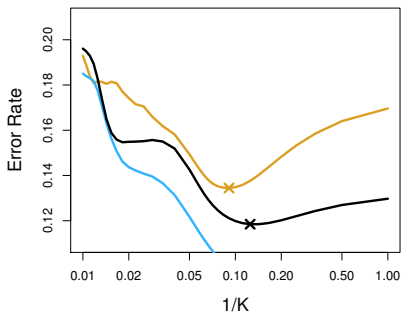
Figure 5.7 from ISL. Decision boundaries are obtained by running logistic polynomial regression for various choices of degree.

Choosing k : Logistic polynomial regression, k -NN

Logistic regression



K -Nearest Neighbours



- Error rate: $\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$
- Test error (computed through simulation)
- Training error
- Cross-validation error estimate

Other extensions of logistic regression

- When we discussed **Linear models**, we saw that we could easily extend them to **Additive models** and **Regularized regression models**
- The same is true of **Logistic regression**
- To fit **Additive Logistic models**:
Specify `family = binomial()` in a `gam()` command
- To fit **ℓ -1 regularized Logistic regression**:
Specify `family = "binomial"` in a `glmnet` command
- Indeed, **Google** famously uses *massive* regularized logistic regressions as a core component of their AdClick prediction system. This system is trained on many *billions* of observations, with potentially *millions* of predictors

Multi-class classification

- The examples we have considered thus far have all been cases of **binary classification**
- Y could take on one of two possible values: $\{0, 1\}$, **{orange, blue}**, etc.
- Going forward, it will be just as straightforward to consider the multi-class case where we have $J \geq 2$ classes
- Our **goal** is to use inputs $X = x$ to classify Y to one of $\{1, 2, \dots, J\}$, seeing to minimize the (test) **error rate** (aka **misclassification rate**)

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$$

- As we said earlier, the best *any classifier whatsoever* could do is to classify to the group j that has the largest value of

$$\mathbb{P}(Y = j \mid X = x)$$

Multi-class classification

- We can now think of our **intermediate goal** as getting good estimates of the conditional probability functions

$$\mathbb{P}(Y = j \mid X = x)$$

- There is a multi-class version of **Logistic regression**, which typically goes by the name **Multinomial regression**
- We will not discuss this method here, because it's not a particularly popular approach
- For k -NN, there's really no generalization that needs to take place to go from $J = 2$ classes to $J \geq 2$ classes
 - We find that k closest points to x , and classify to the **plurality class**: the class that appears most often among x 's k nearest neighbours

Multi-class classification: Generative models

- Logistic regression is an example of what is called a **Discriminative model**
- Such models estimate $\mathbb{P}(Y = j \mid X = x)$, but do not try to model the *joint distribution* between the inputs and the response Y
- Since the **Bayes classifier** itself only uses $\mathbb{P}(Y = j \mid X = x)$, it may seem that to model anything beyond that is *too much work* and seemingly *unnecessary*
- However, there are some nice classification methods that *do* proceed by modeling the joint distribution of (Y, X_1, \dots, X_p) .
- Such methods are called **Generative models**

A closer look at Bayes theorem

- Suppose we have an outcome $Y \in \{1, 2, \dots, K\}$ and an input vector $X = (X_1, \dots, X_p)$

- **Bayes theorem** tells us that

$$\mathbb{P}(Y = k | X = x) = \frac{\mathbb{P}(X = x | Y = k)\mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- $\mathbb{P}(X = x | Y = k)$ is the **density** for X in class k
- $\mathbb{P}(Y = k)$ is the **prior probability** of class k
- **Bayes theorem** gives us a way of combining prior beliefs about the likelihood that $Y = k$ with the **new evidence** that we observed $X = x$.
- $\mathbb{P}(Y = k | X = x)$ is often called the **posterior probability** of class k

Bayes theorem for discriminant analysis

$$\mathbb{P}(Y = k | X = x) = \frac{\mathbb{P}(X = x | Y = k)\mathbb{P}(Y = k)}{\mathbb{P}(X = x)}$$

- Let $f_k(x) = \mathbb{P}(X = x | Y = k)$ denote the **density** of X in class k
- Let $\pi_k = \mathbb{P}(Y = k)$ denote the **prior probability** of class k
- By the *Law of total probability*,

$$\mathbb{P}(X = x) = \sum_{\ell=1}^K \pi_{\ell} f_{\ell}(x)$$

and so we can rewrite Bayes theorem as

$$\mathbb{P}(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^K \pi_{\ell} f_{\ell}(x)}$$

How do we use this as a classifier?

$$\mathbb{P}(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x)}$$

- To minimize the **misclassification rate**, we should classify an observation with inputs x_0 to the class with the **highest posterior probability**

$$\hat{y}_0 = \operatorname{argmax}_{k=1, \dots, K} \frac{\pi_k f_k(x_0)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x_0)}$$

- Note that the denominator term is exactly the same for each term (it depends on x , but not on k).
- Thus the above rule is the same as the simpler rule:

$$\hat{y}_0 = \operatorname{argmax}_{k=1, \dots, K} \pi_k f_k(x_0)$$

Why is this interesting?

$$\hat{y}_0 = \operatorname{argmax}_{k=1,\dots,K} \frac{\pi_k f_k(x_0)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x_0)} = \operatorname{argmax}_{k=1,\dots,K} \pi_k f_k(x_0)$$

- Prior to collecting data, we generally don't know what any of the π_k or $f_k(x)$ should be.
- Thus we must **estimate** the π_k 's and the densities $f_k(x)$ from the data
- The π_k are easy:

$$\hat{\pi}_k = \frac{\#\{i : y_i = k\}}{n}$$

- The $f_k(x)$ are *harder* and *more interesting* to estimate
- Different ways of **estimating** $f_k(x)$ give us **different classifiers!**

The Canadian vs American example

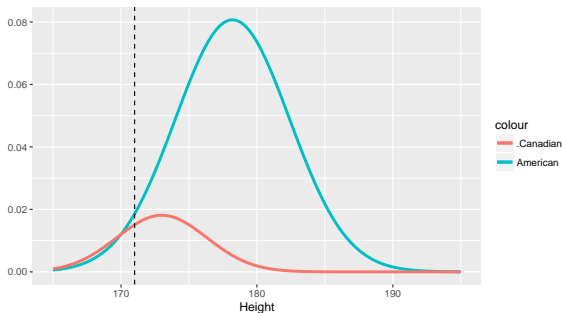
- In the Toy example we looked at earlier, we knew in advance that

$$H \mid \text{American} \sim \text{Normal}(178.2, \sigma = 4.7) = f_A(x)$$

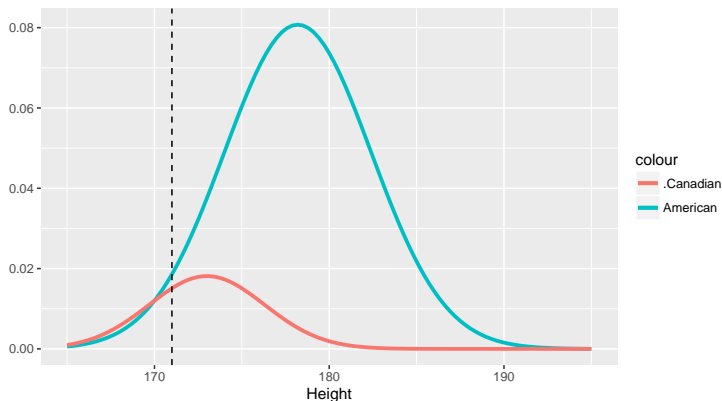
$$H \mid \text{Canadian} \sim \text{Normal}(173, \sigma = 3.3) = f_C(x)$$

and $\pi_A = 0.85$, $\pi_C = 0.15$

- Plot below shows value of $\pi_k f_k(x)$ for $k = \text{Canadian}$ and American



Plot below shows value of $\pi_k f_k(x)$ for $k = \text{Canadian}$ and American



- If $\pi_A f_A(x) > \pi_C f_C(x)$: Classify as **American**
- If $\pi_A f_A(x) < \pi_C f_C(x)$: Classify as **Canadian**
- $\pi_A = 0.85$, $\pi_C = 0.15$ so we have a very high prior that any “American sounding” male we encounter is actually American
- So there's a very small height range where we would classify someone as Canadian

Linear discriminant analysis

$$\hat{y}_0 = \operatorname{argmax}_{k=1,\dots,K} \frac{\pi_k f_k(x_0)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x_0)} = \operatorname{argmax}_{k=1,\dots,K} \pi_k f_k(x_0)$$

- **Linear discriminant analysis (LDA)**³ assumes all the $f_k(x)$ are **Multivariate Normal**(μ_k, Σ)
 - Different means, *same* covariance matrix

³Not to be confused with Latent Dirichlet Allocation... also abbreviated LDA

What is a Multivariate Normal?

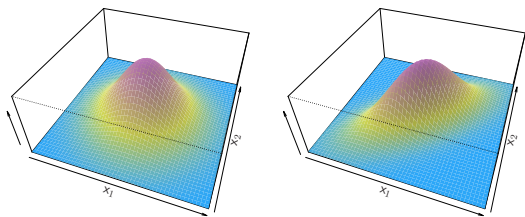
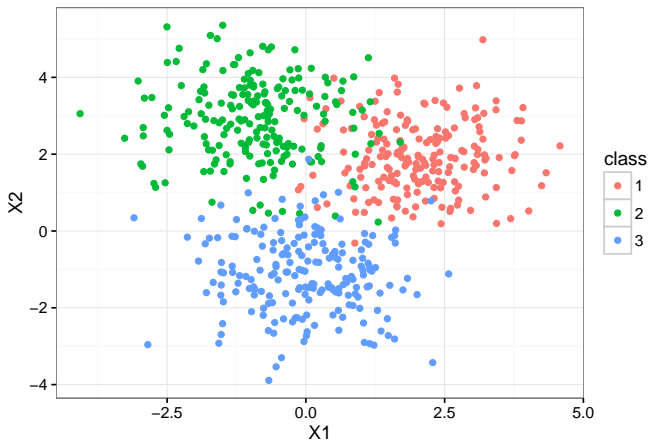
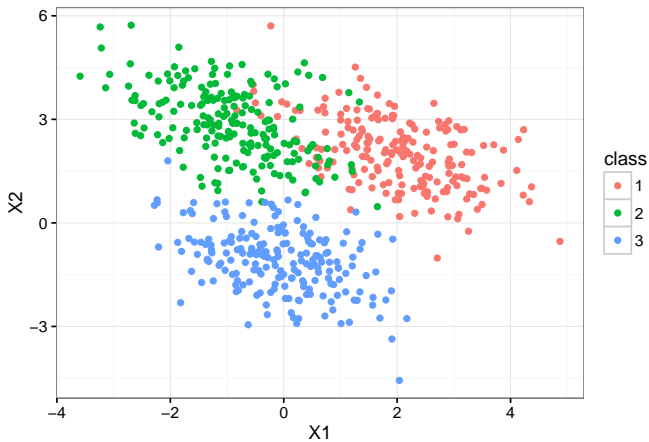


Figure 4.5 from ISL. Bivariate normal density for two choices of Σ .



Covariance matrix $\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

| $\mu_k =$ | Class 1 | Class 2 | Class 3 |
|-----------|---------|---------|---------|
| | (2, 2) | (-1, 3) | (0, -1) |



Covariance matrix $\Sigma = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$

| | Class 1 | Class 2 | Class 3 |
|-----------|---------|---------|---------|
| $\mu_k =$ | (2, 2) | (-1, 3) | (0, -1) |

Linear discriminant analysis

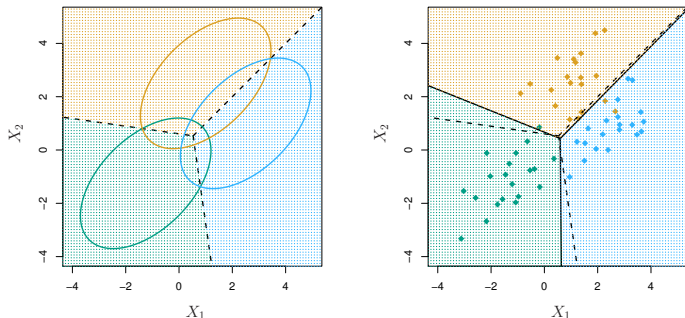


Figure 4.6 from ISL

- **Left:** $K = 3$ classes. True f_k are Bivariate Normal. Ovals represent 95% density contours for each class (95% of points generated from a class will fall into the ellipse). Dashed lines are the LDA boundary from the true f_k
- **Right:** 20 points are observed from each class. Solid lines are the estimated LDA decision boundaries. We had to estimate μ_k and Σ from the data.

Linear discriminant analysis

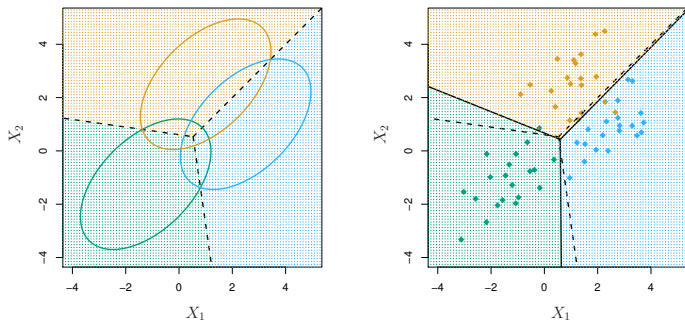


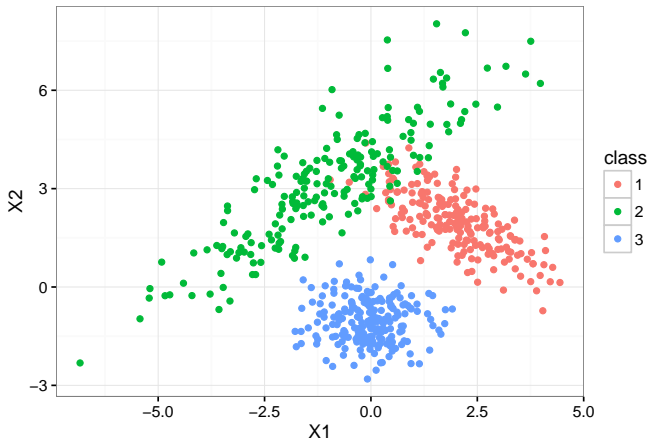
Figure 4.6 from ISL

- It's called **Linear** discriminant analysis because the choice $f_k = \text{MVN}(\mu_k, \Sigma)$ always results in *linear decision boundaries*
- The LDA rule amounts to classifying x_0 to the class with the highest value of the **discriminant function** δ_k :

$$\delta_k(x_0) = x_0^T \hat{\Sigma}^{-1} \hat{\mu}_k - \frac{1}{2} \hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k + \log(\hat{\pi}_k)$$

Quadratic discriminant analysis

- Quadratic discriminant analysis (QDA) models $f_k(x)$ as $MVN(\mu_k, \Sigma_k)$: it relaxes LDA's assumption that all the covariance matrices are the same
- This produces *quadratic* decision boundaries



$$\Sigma_1 = \begin{pmatrix} 1 & -0.7 \\ -0.7 & 1 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 3 & 2.5 \\ 2.5 & 3 \end{pmatrix} \quad \Sigma_3 = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

| | Class 1 | Class 2 | Class 3 |
|-----------|---------|---------|---------|
| $\mu_k =$ | (2, 2) | (-1, 3) | (0, -1) |

Quadratic discriminant analysis

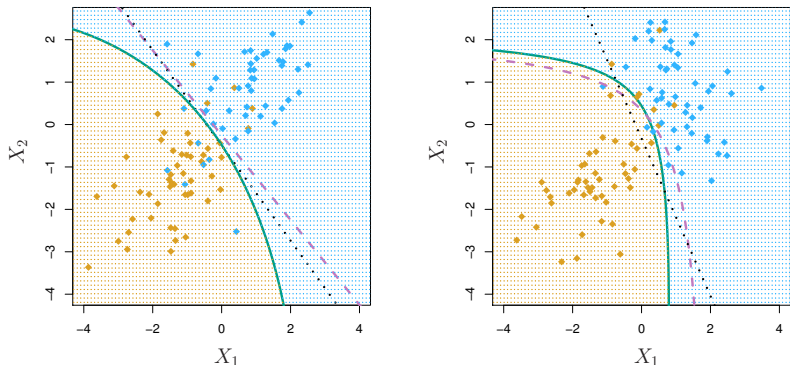


Figure 4.9 from ISL. Dashed purple curve is the Bayes classifier decision boundary. Solid green curve is QDA, dotted black line is LDA. Left: True boundary is linear. Right: True boundary is quadratic.

Logistic regression vs. Linear discriminant analysis

- When the classes are *well separated*, Logistic regression coefficient estimates are **really unstable**. LDA does not have this problem.
- LDA **makes assumptions** on the distribution of $X | Y = k$
 - When the assumptions hold (or approximately hold), LDA can produce better, *more stable* decision boundaries, even when n is small
- When we have $K = 2$ classes, Logistic regression can be extended in all kinds of ways (additive models, regularized models, etc.), and is highly interpretable
- Logistic regression gets hard to interpret for $K > 2$ classes. LDA is essentially the same regardless of how many classes we have.

Naive Bayes

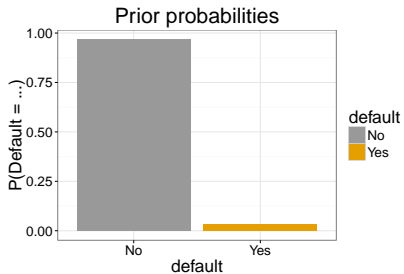
- **Naive Bayes** is a popular simple classifier in cases where we have a lot of predictors p
- Imagine $p = 1000$ and $n = 2000$. It's going to be *extremely difficult* to accurately estimate $f_k(x)$ without *really strong assumptions* on f_k
- **Naive Bayes** says: Let's *assume* that all components of $X = (X_1, X_2, \dots, X_p)$ are **independent**⁴
- Under the **independence** assumption, $f_k(x)$ simplifies to:

$$\begin{aligned}f_k(x) &= P(X_1 = x_1, \dots, X_p = x_p \mid Y = k) \\&= P(X_1 = x_1 \mid Y = k) \times \dots \times P(X_p = x_p \mid Y = k) \\&= \prod_{j=1}^p P(X_j = x_j \mid Y = k)\end{aligned}$$

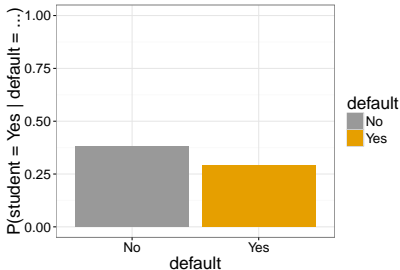
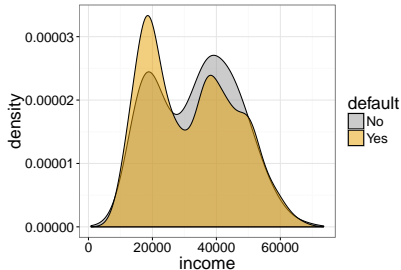
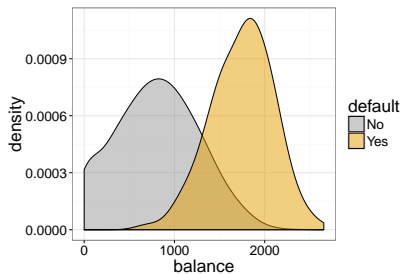
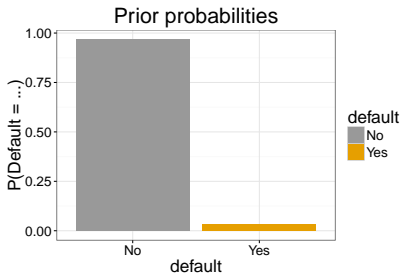
- So now to estimate $f_k(x)$ we just need to estimate p **univariate densities**: $f_k(x_j) = P(X_j = x_j \mid Y = k)$

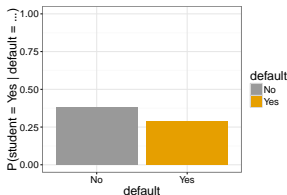
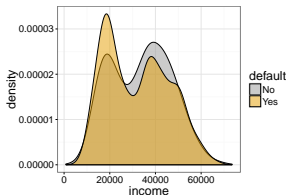
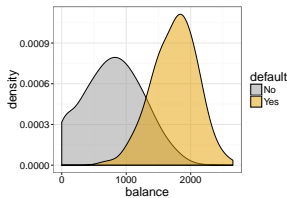
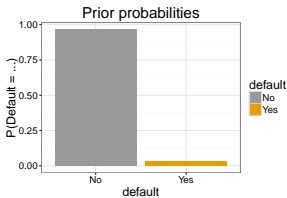
⁴We're actually assuming *conditional independence*: The inputs are independent *given* the class labels

Naive Bayes with the Default Data



Naive Bayes with the Default Data





Calculate:

$$\hat{f}_{Yes}(\mathbf{i}, \mathbf{b}, \mathbf{s}) = \hat{f}_{Yes}(\text{income}) \times \hat{f}_{Yes}(\text{balance}) \times \hat{f}_{Yes}(\text{student})$$

$$\hat{f}_{No}(\mathbf{i}, \mathbf{b}, \mathbf{s}) = \hat{\pi}_{No} \hat{f}_{No}(\text{income}) \times \hat{f}_{No}(\text{balance}) \times \hat{f}_{No}(\text{student})$$

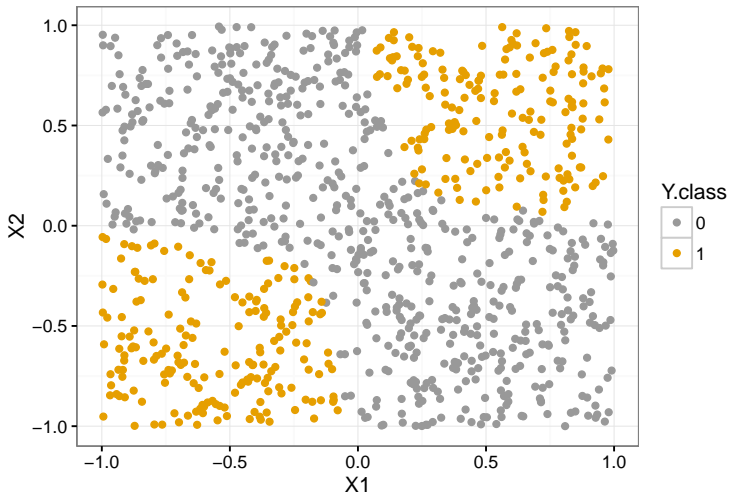
Naive Bayes posterior probability estimate of **default = Yes**:

$$\hat{\mathbb{P}}(\text{default} = \text{Yes} \mid \mathbf{i}, \mathbf{b}, \mathbf{s}) = \frac{\hat{\pi}_{Yes} \hat{f}_{Yes}(\mathbf{i}, \mathbf{b}, \mathbf{s})}{\hat{\pi}_{Yes} \hat{f}_{Yes}(\mathbf{i}, \mathbf{b}, \mathbf{s}) + \hat{\pi}_{No} \hat{f}_{No}(\mathbf{i}, \mathbf{b}, \mathbf{s})}$$

Naive Bayes vs. LDA vs. QDA

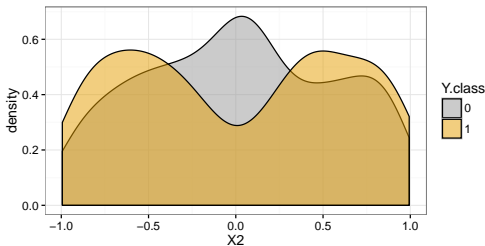
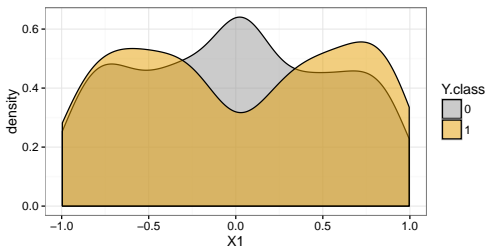
- Naive Bayes scales well to problems where p is large
 - If you have enough data to estimate the univariate density of each predictor (i.e., enough to form a *nice* histogram), you can apply Naive Bayes
- In LDA, we have to estimate $K \times p$ parameters to get $\hat{\mu}_k$'s and another $\frac{1}{2}p(p+1)$ parameters to estimate the $p \times p$ covariance matrix $\hat{\Sigma}$.
- In QDA, we have to estimate the means and K $p \times p$ covariance matrices. That's $\frac{1}{2}Kp(p+1)$ parameters!
- So why do we even bother with methods like LDA or QDA?
 - They can capture meaningful **interactions**. Naive Bayes **cannot**.

An example where Naive Bayes fails



This is **not** a difficult classification problem. Logistic regression with formula $y \sim X1 * X2$ gives a **perfect** classifier.

What does Naive Bayes see on this problem?



Naive Bayes views X_1 and X_2 independently, and *cannot* produce the rule: “If X_1 and X_2 are both small or both large, classify as $\hat{Y} = 1$ ”

End of Part I

10 minute break

Assessing the performance of Classifiers

Agenda for Part II

Assessing performance of classification models

- Calibration plots
- Confusion matrices
- Sensitivity, Specificity, Accuracy, Precision, Recall
- Cost-based criteria
- ROC curves

Assessing Classifier Performance

- We now have a bunch of different ways of estimating the **conditional probability** (aka **posterior probability**, if we take a Bayes approach)

$$p_k(x) = \mathbb{P}(Y = k \mid X = x)$$

- Now we can start asking questions about whether the estimate $\hat{p}_k(x)$ is a good one, and whether it results in a good **decision rule**
- Let's start with the question:

Is $\hat{p}_k(x)$ a good estimate of $p_k(x)$?

Are we doing a good job of estimating $p_k(x)$?

Is $\hat{p}_k(x)$ a good estimate of $p_k(x)$?

- Why do we care?
 - An email whose spam probability is $\hat{p}_k(x) = 0.52$ will get classified to spam the same as an email whose spam probability is $\hat{p}_k(x) = 0.999$. But one email is borderline, while the other is extremely likely to be spam. This is important information.
 - Whether to pursue suspicious insurance claim may depend on the estimated **probability** that it's fraudulent and various cost-related factors
 - Customer Lifetime Value (CLV) calculations require an estimate of the **probability** that a customer will make a purchase

Calibration plots

- ① Bin the data according to $\hat{p}_k(x)$: E.g., bins could be $[0, 0.1], (0.1, 0.2], \dots, (0.9, 1]$.
 - ② For each bin, calculate the proportion of observations in bin b that had class $Y = k$
 - ③ Plot the midpoints of the bins on the x -axis and the proportions from Step 2. on the y -axis
- **Note:** Certain methods are obviously poorly calibrated. E.g., we saw that linear regression can return *negative values*
 - In such cases, a popular approach is to use the **softmax transformation**

$$\hat{p}_k^* = \frac{e^{\hat{y}_k}}{\sum_{\ell=1}^K e^{\hat{y}_\ell}}$$

These values at least all lie in $[0, 1]$ and sum to 1

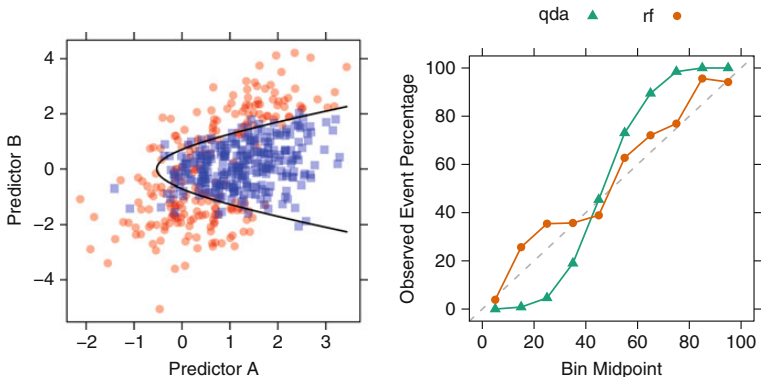


Fig. 11.1: *Left*: A simulated two-class data set with two predictors. The *solid black line* denotes the 50% probability contour. *Right*: A calibration plot of the test set probabilities for random forest and quadratic discriminant analysis models

The RF is well-calibrated. The QDA model **overestimates** the true probability when it's low, and **underestimates** it when it's high.

[source: Applied Predictive Modeling]

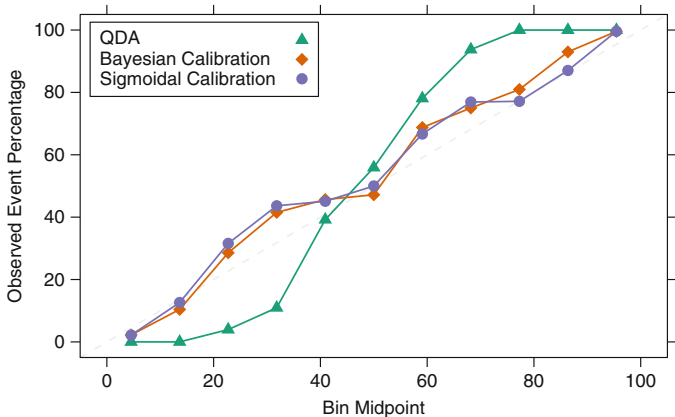


Fig. 11.2: The original QDA class probabilities and recalibrated versions using two different methodologies

The QDA calibration curve looks S-shaped. The technical term to describe this shape is: *sigmoidal*. We can try **recalibrating** by feeding the $\hat{p}(x)$ into a logistic regression to predict y_i , and then use the resulting probabilities. In some cases this actually works.

Evaluating Classifications

- Let's focus again on the **binary classification** setting:
 - $Y = 1$: if the event happened
 - $Y = 0$: if the event did not happen
- The primary building block of essentially all approaches to evaluating a Classifier is the **confusion matrix**

Table 11.1: The confusion matrix for the two-class problem (“events” and “nonevents.”) The table cells indicate number of the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)

| Predicted | Observed | |
|-----------|----------|----------|
| | Event | Nonevent |
| Event | TP | FP |
| Nonevent | FN | TN |

[source: Applied Predictive Modeling]

Evaluating Classifications

Table 11.1: The confusion matrix for the two-class problem (“events” and “nonevents.”) The table cells indicate number of the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)

| Predicted | Observed | |
|-----------|----------|----------|
| | Event | Nonevent |
| Event | TP | FP |
| Nonevent | FN | TN |

- In **R** (and often in practice), it's more natural to form confusion matrices with the Non-event and Event labels swapped.

| Predicted | Observed | |
|-----------|----------|-----|
| | No | Yes |
| No | TN | FN |
| Yes | FP | TP |

[source: Applied Predictive Modeling]

The Confusion Matrix

Table 11.1: The confusion matrix for the two-class problem (“events” and “nonevents.”) The table cells indicate number of the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)

| Predicted | Observed | |
|-----------|----------|----------|
| | Event | Nonevent |
| Event | TP | FP |
| Nonevent | FN | TN |

- Each cell is a count. The cells in total add up to n
- The **diagonal** entries are **correct classifications**
- The **off-diagonal** entries are classification **errors**

[source: Applied Predictive Modeling]

What does it mean to have a good classifier?

| Predicted | Observed | |
|-----------|----------|----------|
| | Event | Nonevent |
| Event | TP | FP |
| Nonevent | FN | TN |

- **Accuracy** = $(TP + TN)/n$
- **Misclassification rate** = $(FN + FP)/n = 1 - \text{Accuracy}$
- If FP and FN have the same **cost**, then our goal is to minimize the **Misclassification rate**
- E.g., if $Y \in \{\text{cat photo, dog photo}\}$, we don't care if our error was mistaking a cat for a dog or vice versa
- If $Y \in \{\text{fraud, not fraud}\}$, the cost of a FN (failing to catch fraud) is typically *much higher* than the cost of a FP (further investigating a false lead)

What does it mean to have a good classifier?

- Suppose I tell you my classifier has **97% accuracy**. Are you impressed?
- Well, what if $Y \in \{\text{wins lottery, does not win}\}$?
 - The chances that a ticket wins the Powerball lottery are way smaller than 1% (they're 1 in 175,000,000)
 - By classifying all tickets as “does not win”, we'd have an accuracy of $1 - 1/175,000,000 = 0.99999\dots$
 - So a **97% accuracy** in this case is actually **really bad**
- We must take the **baseline probabilities** (i.e., prior probabilities) of each class into account when assessing performance

Sensitivity, Specificity

- We're now going to look at ways of quantifying the performance of a classifier that go beyond the simple notion of **Accuracy**
- **Sensitivity**: aka Recall

$$\begin{aligned} &= \frac{\text{\#observations correctly classified to have the event}}{\text{\#observations that had the event}} \\ &= \frac{TP}{TP + FN} \end{aligned}$$

- **Specificity**: aka True Negative Rate (TNR)

$$\begin{aligned} &= \frac{\text{\#observations correctly classified as non-events}}{\text{\#observations that did not have the event}} \\ &= \frac{TN}{TN + FP} \end{aligned}$$

Sensitivity - Specificity trade-off

- A classifier with high **Sensitivity** is desirable when **False Negatives** (e.g., failing to detect fraud) are more costly than **False Positives** (flagging a case that turns out to be non-fraudulent)
- A classifier with high **Specificity** is desirable when **False Positives** (convicting an innocent person of a crime) are more costly than **False Negatives** (failing to convict a guilty person of a crime)
- **Sensitivity** and **Specificity** tend to move in opposite directions
 - To increase **Sensitivity**, we can always flag more cases as potentially fraudulent. But this would decrease **Specificity** because more non-fraud cases would now be misclassified as fraud.

An example: Marketing data

- The Marketing data set contains information on bank customers who were contacted by marketers wanting them to open an account
- $Y = \text{“Yes”}$ if the customer opened an account when contacted, “No” otherwise
- Here's a confusion matrix obtained from fitting a logistic regression model, and classifying $Y = 1$ if $\hat{p}(x) \geq 0.25$

| | | Observed | |
|-----------|-----|----------|------|
| | | Yes | No |
| Predicted | Yes | 121 | 234 |
| | No | 916 | 7771 |

- $n = 121 + 234 + 916 + 7771 = 9042$
- **Accuracy** = $(121 + 7771)/n = 87\%$
- **Misclassification rate** = $(234 + 916)/n = 13\%$
- **Prevalence** = $(121 + 916)/n = 11.5\%$

Marketing data

| | | Observed | |
|-----------|-----|----------|------|
| | | Yes | No |
| Predicted | Yes | 121 | 234 |
| | No | 916 | 7771 |

- $n = 121 + 234 + 916 + 7771 = 9042$
- **Accuracy** = $(121 + 7771)/n = 87$
- **Misclassification rate** = $(234 + 916)/n = 13\%$
- **Prevalence** = $(121 + 916)/n = 11.5\%$
- **Sensitivity (Recall)** = $121/(121 + 916) = 11.6\%$
- **Specificity** = $7771/(7771 + 234) = 97.1\%$
- So at the cutoff, $\hat{p}(x) \geq 0.25$, our classifier has *low Sensitivity* and *high Specificity*
- This is not a good setting for marketing.

Some other measures of performance

- Here are some other quantities that people have names for
- **Positive Predictive Value (PPV):**

$$\begin{aligned} &= \frac{\text{\#observations correctly classified to have the event}}{\text{\#observations classified to have the event}} \\ &= \frac{TP}{TP + FP} \end{aligned}$$

- Suppose you take a diagnostic test for a disease. The **PPV** of the diagnostic is the probability that you actually have disease when you test positive.
- **Negative Predictive Value: (NPV)**

$$\begin{aligned} &= \frac{\text{\#observations correctly classified as non-events}}{\text{\#observations classified as non-events}} \\ &= \frac{TN}{TN + FN} \end{aligned}$$

- Suppose you take a pregnancy test. The **NPV** is the probability that you actually aren't pregnant given that the test comes up negative.

Cost-Based Criteria

- All of the criteria we've discussed so far take the form of counts and proportions
- But many real world problems have real **costs** and **benefits**
- We may be interested in:
 - Predicting which investments to make to maximize return
 - Improving customer satisfaction through market segmentation
 - Minimize costs associated with fraudulent transactions
- Suppose you work for a clothing retailer and are tasked with mailing out promotional offers
 - It costs you **\$2.00** to mail a promotion
 - A customer who *Responds* to the promotion yields you an average gain of **\$28.40**
- We can use the confusion matrix now to calculate **profit**

Simple profit calculation

Table 11.4: Left: A hypothetical test confusion matrix for a predictive model with a sensitivity of 75% and a specificity of 94.4%. Right: The confusion matrix when a mass mailing is used for all customers

| Predicted | Observed | | Observed | |
|-------------|----------|-------------|----------|-------------|
| | Response | Nonresponse | Response | Nonresponse |
| Response | 1,500 | 1,000 | 2,000 | 18,000 |
| Nonresponse | 500 | 17,000 | 0 | 0 |

- It costs you **\$2.00** to mail a promotion
- A customer who *Responds* to the promotion yields you an average gain of **\$28.40**. This is a *net gain* of **\$26.40** (benefit of a TP).
- A customer who *would have Responded* but who you did not reach out to thus loses you **\$28.40** (cost of a FN)
- Thus the strategy for the Left confusion matrix gives us a profit of:

$$\text{profit} = \$26.40TP - \$2.00FP - \$28.40FN = \$23,400$$

- If we mailed *everyone* (strategy on right), the profit would be \$16,800

[source: Applied Predictive Modeling]

Probabilities as ranking functions

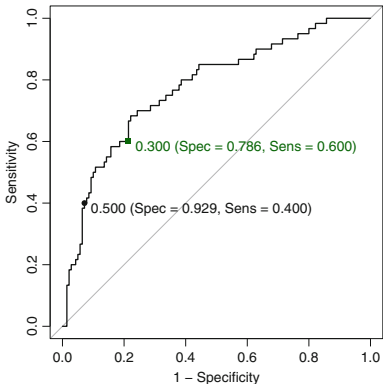
- Suppose we have a probability estimate $\hat{p}(x)$
- We can use $\hat{p}(x)$ to order the observations from *most likely to have the Event* to *least likely*

| i | $\hat{p}(x_i)$ | y_i |
|----------|----------------|----------|
| 45 | 0.975 | 1 |
| 12 | 0.824 | 0 |
| 191 | 0.762 | 1 |
| 77 | 0.754 | 1 |
| \vdots | \vdots | \vdots |

- We can think about how well $\hat{p}(x)$ performs by asking: When we order the y_i according to $\hat{p}(x)$, do most of the observations with $y_i = 1$ appear at the top of the list?
- A **perfect ranking function** will score all of the observations where $y_i = 1$ higher than those where $y_i = 0$
- We're now going to discuss various approaches for visualizing how well $\hat{p}(x)$ does at ranking observations

ROC Curves

- As we vary our probability cutoff α , we get different classification rules and hence different values of all of our performance metrics
- You can think of getting a different confusion matrix at each α
- It's useful to plot the values of various performance metrics as you vary the cutoff α
- Perhaps the most widely used plot is the **ROC Curve**



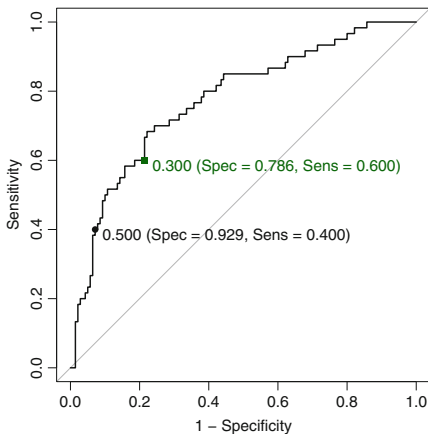
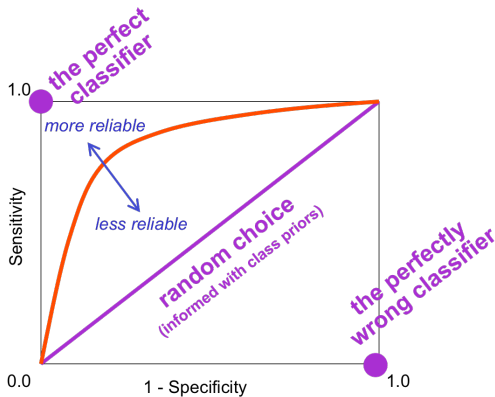


Fig. 11.6: A receiver operator characteristic (ROC) curve for the logistic regression model results for the credit model. The dot indicates the value corresponding to a cutoff of 50% while the green square corresponds to a cutoff of 30% (i.e., probabilities greater than 0.30 are called events)

Each point on the curve corresponds to the value of (1-Specificity, Sensitivity) calculated at a particular choice of cutoff α

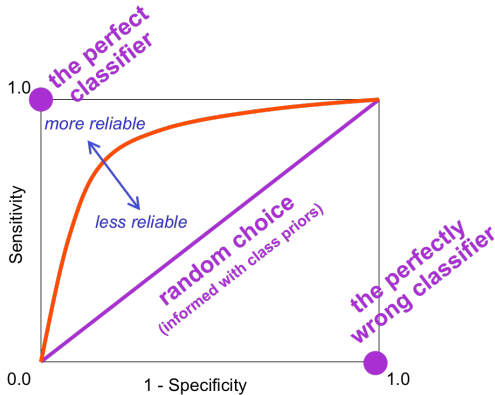
[source: Applied Predictive Modeling]

ROC



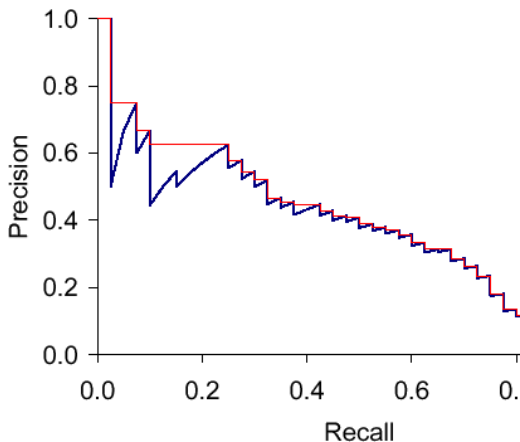
- The diagonal is the ROC you would get from randomly picking proportion π_k of the observations to classify to class k
- Higher ROC is better
- The **perfect classifier** has $(1 - \text{Specificity}, \text{Sensitivity}) = (0, 1)$

Area under the curve



- The **AUC** is the *area under the ROC curve*
- AUC has a nice **interpretation**: The AUC is the probability that the classifier will rank a *randomly selected* observation where $y_i = 1$ higher than a *randomly selected* observation where $y_i = 0$

Precision-Recall curves



- **Precision:** $TP / (TP + FP)$ (aka, **PPV**)
- **Recall:** $TP / (TP + FN)$ (aka, **Sensitivity**)
- Precision @50% Recall is a common performance metric

[source: Introduction to Information Retrieval, Manning et al.]

Lift charts

- **Lift charts** are kind of like ROC curves, but may be more useful depending on the application

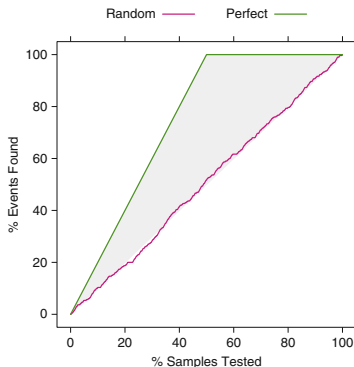
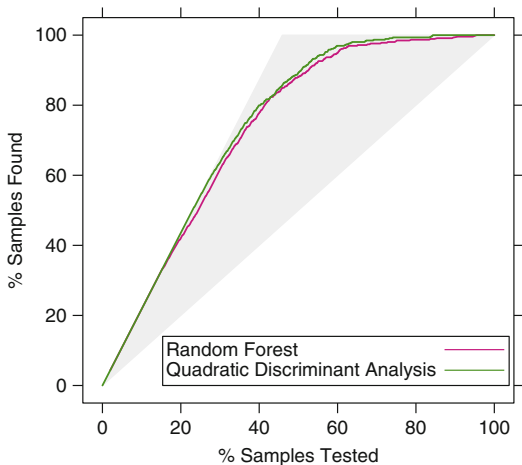


Fig. 11.7: An example lift plot with two models: one that perfectly separates two classes and another that is completely non-informative

- **y-axis: Recall (Sensitivity)**
- **x-axis: $\#\{i : \hat{p}(x_i) > \alpha\} / n = (FP + TP) / n$**

Lift charts

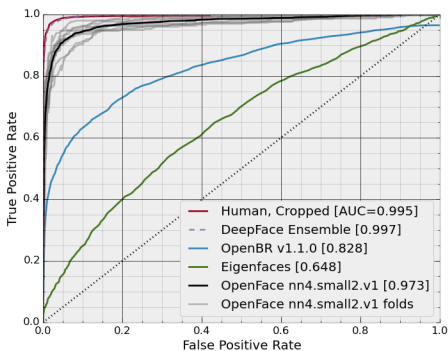


- In this example, of the top 40% of observations ordered according to $\hat{p}(x)$, essentially all of them have $y_i = 1$. This is great!

How do we pick the best classifier?

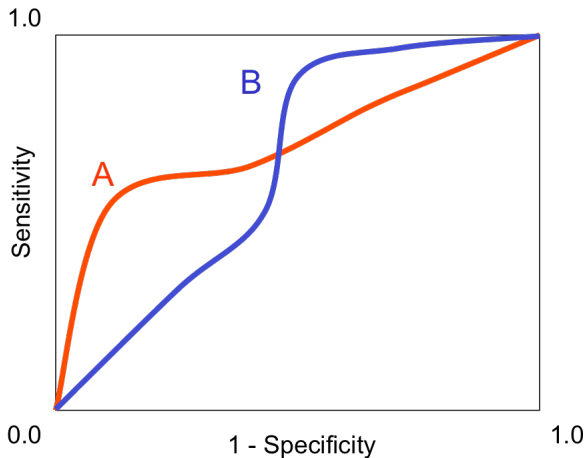
You can overlay the ROC curves from a bunch of different methods.

Here's a facial recognition example. [source: [cmusatyalab/openface GitHub](#)]



- The DeepFace Ensemble method is amazing
- The proposed method, OpenFace nn4.small2.v1 does really well. Its ROC curve is the solid black line.
 - 10 grey curves are Test Fold ROC curves from 10-Fold CV

Now which one is better?



It depends on what region of the curve you care most about.

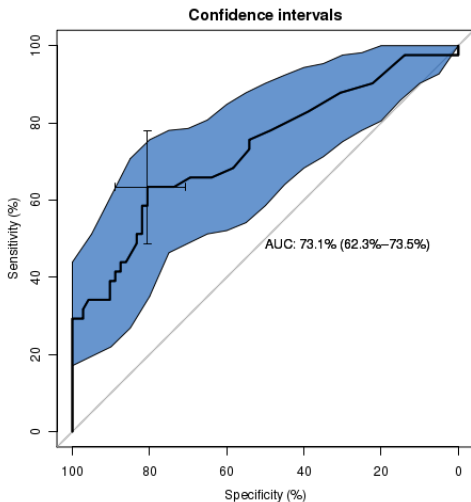
Example: Cancer diagnosis

We can think of two settings: **Screening**, and **Referred examinations**

- **Screening:** E.g., we want annual screening of all people above age 40
 - Most people we test will *not* have cancer (high ratio of **non-Events** to **Events**)
 - Cost(False positives)/Cost(False negatives) moderate or high
 - Focus on: performance at High Specificity (small x -axis values)
- **Referred examination:** E.g., you're exhibiting symptoms and your doctor feels a bump under your skin, and refers you for a biopsy to get it tested for cancer
 - Many referred individuals *will* have cancer (*low* or *equal* ratio of **non-Events** to **Events**)
 - False negatives are really costly
 - Focus on: performance at High Sensitivity (high y -axis values)

Putting confidence bands on ROC curves

- We like putting standard error bars on our curves so that we can visually discern which trends/differences are *statistically significant*



Using Cross-validation

- In the Prediction setting, we focussed entirely on MSE as our performance metric
- To **validate** our model, we would use K -Fold CV to estimate the **Test MSE**
- In the Classification setting, there are *many* metrics out there. The set I presented is by no means exhaustive.
- To estimate **Test performance**:
 - ➊ Pick a metric (E.g., Accuracy, profit, AUC, Sensitivity @ $x\%$ Specificity, Precision @ $x\%$ Recall, etc.)
 - ➋ Calculate the metric on each fold of K -fold CV
 - ➌ Average over all of the folds
- For ROC and Precision-Recall curves, you may want to show the curve you get from each Test fold. This gives a visual representation of the variability of the curve estimates.

Takeaways

- In the Prediction setting, we focussed entirely on MSE as our performance metric
- There are other options out there, but MSE is by the far the most widely accepted
- In the Classification setting, there are *many* metrics out there. The set I presented is by no means exhaustive.
- Whatever criterion it is we wish to maximize (e.g., Accuracy, profit, Sensitivity @ x % Specificity, Precision @ x % Recall, etc.), we can use Cross-validation to **estimate the Test set performance** according to this metric

Main takeaway:

Classification is way **more interesting** than Prediction.

Acknowledgements

All of the lectures notes for this class feature content borrowed with or without modification from the following sources:

- 36-462/36-662 Lecture notes (Prof. Tibshirani, Prof. G'Sell, Prof. Shalizi)
- 95-791 Lecture notes (Prof. Dubrawski)
- *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
- *Applied Predictive Modeling*, (Springer, 2013), Max Kuhn and Kjell Johnson