

Lecture 17

Logistic regression and decisions

COURSE MATH 637: Mathematical Techniques in Data Science

WEBSITE <https://jacobcalvert.com/teaching/math637fall2026/>

INSTRUCTOR Jacob Calvert (calvert@udel.edu)

OFFICE HOURS MW 2:45pm–3:45pm, Ewing Hall 509

Plan for today

- | | |
|---|--------|
| • Naive Bayes | 8 min |
| • Two repairs so far | 2 min |
| • The logistic model | 13 min |
| • From probabilities to decisions | 15 min |
| • Coefficients; generative vs. discriminative | 7 min |

Definition: Naive Bayes

Recall: To estimate $\eta_k(x)$, a generative classifier first models π_k and f_k , then uses Bayes' rule: $\eta_k(x) = \pi_k f_k(x) / \sum_j \pi_j f_j(x)$.

Definition (Naive Bayes)

Model: priors π_k , and features independent within each class: $f_k(x) = f_{k1}(x_1) \cdots f_{kp}(x_p)$.

Fit: $\hat{\pi}_k = n_k/n$; each $\hat{f}_{kj}$ from the values of feature j in class k alone.

Predict: $\hat{\eta}_k(x) \propto \hat{\pi}_k \hat{f}_{k1}(x_1) \cdots \hat{f}_{kp}(x_p)$.

- each f_{kj} : Gaussian for a measurement; frequencies for a category or yes/no

Definition: Naive Bayes

Recall: To estimate $\eta_k(x)$, a generative classifier first models π_k and f_k , then uses Bayes' rule: $\eta_k(x) = \pi_k f_k(x) / \sum_j \pi_j f_j(x)$.

Definition (Naive Bayes)

Model: priors π_k , and features independent within each class: $f_k(x) = f_{k1}(x_1) \cdots f_{kp}(x_p)$.

Fit: $\hat{\pi}_k = n_k/n$; each $\hat{f}_{kj}$ from the values of feature j in class k alone.

Predict: $\hat{\eta}_k(x) \propto \hat{\pi}_k \hat{f}_{k1}(x_1) \cdots \hat{f}_{kp}(x_p)$.

- each f_{kj} : Gaussian for a measurement; frequencies for a category or yes/no
- with Gaussian f_{kj} : QDA with every Σ_k diagonal

Naive Bayes versus QDA

QDA

- models the correlations within each class
- $Kp + K\frac{p(p+1)}{2} + K - 1$ parameters: 314 for wine
- needs at least $p + 1$ examples in every class
- Gaussian features

naive Bayes

- ignores the correlations within each class
- $2Kp + K - 1$ parameters: 80 for wine
- needs 2 examples in every class
- any one-dimensional model per feature

Naive Bayes versus QDA

QDA

- models the correlations within each class
- $Kp + K\frac{p(p+1)}{2} + K - 1$ parameters: 314 for wine
- needs at least $p + 1$ examples in every class
- Gaussian features

naive Bayes

- ignores the correlations within each class
- $2Kp + K - 1$ parameters: 80 for wine
- needs 2 examples in every class
- any one-dimensional model per feature

Idea: LDA (shared Σ) and naive Bayes (diagonal Σ_k) restrict QDA in different ways; neither contains the other. Which restriction helps depends on the data.

Python: Naive Bayes and QDA

```
from sklearn.discriminant_analysis import (
    QuadraticDiscriminantAnalysis)
from sklearn.naive_bayes import GaussianNB
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
gnb = GaussianNB().fit(X_train, y_train)
p_hat = gnb.predict_proba(X_test)    #  $\eta_k(x)$ 
gnb.theta_, gnb.var_                 #  $K \times p$  means, variances
```

- the same calls for both: `fit`, `predict`, `predict_proba`, `score`

Python: Naive Bayes and QDA

```
from sklearn.discriminant_analysis import (
    QuadraticDiscriminantAnalysis)
from sklearn.naive_bayes import GaussianNB
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
gnb = GaussianNB().fit(X_train, y_train)
p_hat = gnb.predict_proba(X_test)    #  $\eta_k(x)$ 
gnb.theta_, gnb.var_                 #  $K \times p$  means, variances
```

- the same calls for both: `fit`, `predict`, `predict_proba`, `score`
- with 27 training wines, `qda.fit` raises an error (a singular $\hat{\Sigma}_k$);
`gnb.fit` does not

Example: Fewer training wines

All 13 measurements; test accuracy averaged over 20 random splits:

training wines (per cultivar)	QDA	LDA	naive Bayes
90 (30, 36, 24)	96.6%	97.7%	97.2%
45 (15, 18, 12)	cannot be fit	95.8%	95.5%
27 (9, 11, 7)	cannot be fit	91.1%	92.9%

- 12 wines of cultivar 3, 13 measurements: $\hat{\Sigma}_3$ has rank at most 11

Example: Fewer training wines

All 13 measurements; test accuracy averaged over 20 random splits:

training wines (per cultivar)	QDA	LDA	naive Bayes
90 (30, 36, 24)	96.6%	97.7%	97.2%
45 (15, 18, 12)	cannot be fit	95.8%	95.5%
27 (9, 11, 7)	cannot be fit	91.1%	92.9%

- 12 wines of cultivar 3, 13 measurements: $\hat{\Sigma}_3$ has rank at most 11
- here, with few examples per class, the restricted models do better

Plan for today

- Naive Bayes 8 min
- Two repairs so far 2 min
- The logistic model 13 min
- From probabilities to decisions 15 min
- Coefficients; generative vs. discriminative 7 min

Recall: Three ways to estimate η_k

Least squares on indicators estimates $\eta_k(x) = \mathbb{P}(Y = k \mid X = x)$, but the line leaves $[0, 1]$ and can mask a class.

method	main idea
k -NN	average the class indicators of nearby points
generative classifiers	model each class's x , then use Bayes' rule
logistic regression (today)	linear model for log-odds

Plan for today

- Naive Bayes 8 min
- Two repairs so far 2 min
- The logistic model 13 min
- From probabilities to decisions 15 min
- Coefficients; generative vs. discriminative 7 min

Recall: LDA's log-odds are linear

Recall: For two classes, LDA's discriminants satisfy

$$\delta_1(x) - \delta_0(x) = w^T x + b,$$

where $w = \Sigma^{-1}(\mu_1 - \mu_0)$ is Fisher's direction and b is a constant.

Recall: LDA's log-odds are linear

Recall: For two classes, LDA's discriminants satisfy

$$\delta_1(x) - \delta_0(x) = \textcolor{brown}{w}^\top x + \textcolor{brown}{b},$$

where $w = \Sigma^{-1}(\mu_1 - \mu_0)$ is Fisher's direction and b is a constant.

Because $\eta_k(x) \propto e^{\delta_k(x)}$, the log-odds are

$$\log \frac{\mathbb{P}(Y = 1 \mid X = x)}{\mathbb{P}(Y = 0 \mid X = x)} = \log \frac{\eta_1(x)}{\eta_0(x)} = \delta_1(x) - \delta_0(x) = \textcolor{brown}{w}^\top x + \textcolor{brown}{b}.$$

Recall: LDA's log-odds are linear

Recall: For two classes, LDA's discriminants satisfy

$$\delta_1(x) - \delta_0(x) = \mathbf{w}^\top x + b,$$

where $\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_0)$ is Fisher's direction and b is a constant.

Because $\eta_k(x) \propto e^{\delta_k(x)}$, the log-odds are

$$\log \frac{\mathbb{P}(Y = 1 \mid X = x)}{\mathbb{P}(Y = 0 \mid X = x)} = \log \frac{\eta_1(x)}{\eta_0(x)} = \delta_1(x) - \delta_0(x) = \mathbf{w}^\top x + b.$$

Solve for the probability:

$$\mathbb{P}(Y = 1 \mid X = x) = \sigma(\mathbf{w}^\top x + b), \quad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Definition: Logistic regression

Idea: Keep $\mathbb{P}(Y = 1 \mid X = x) = \sigma(\theta^\top x)$ as LDA has for binary Y , but no longer assume that θ is a function of π_k, μ_k, Σ !

Definition: Logistic regression

Idea: Keep $\mathbb{P}(Y = 1 \mid X = x) = \sigma(\theta^\top x)$ as LDA has for binary Y , but no longer assume that θ is a function of π_k, μ_k, Σ !

Definition (Logistic regression)

Data (x_i, y_i) with $y_i \in \{0, 1\}$ for $i = 1, \dots, n$. Assume each x_i has first entry = 1 so that $\theta^\top x_i$ includes intercept.

- (i) **Model:** $\mathbb{P}(Y = 1 \mid X = x) = \sigma(\theta^\top x)$.
- (ii) **Fit:** choose θ to maximize the **conditional likelihood**

$$L(\theta) = \prod_{i=1}^n \mathbb{P}(Y = y_i \mid X = x_i) = \prod_{i=1}^n \sigma(\theta^\top x_i)^{y_i} (1 - \sigma(\theta^\top x_i))^{1-y_i}.$$

Aside: Several classes

Definition (Multinomial logistic regression)

$$\mathbb{P}(Y = k \mid X = x) = \frac{e^{\theta_k^\top x}}{\sum_{j=1}^K e^{\theta_j^\top x}},$$

the form LDA's probabilities take for K classes (L16), with every θ_k free.

- adding the same vector to every θ_k changes nothing: fix one, e.g. $\theta_K = 0$, to make the coefficients unique

Aside: Several classes

Definition (Multinomial logistic regression)

$$\mathbb{P}(Y = k \mid X = x) = \frac{e^{\theta_k^\top x}}{\sum_{j=1}^K e^{\theta_j^\top x}},$$

the form LDA's probabilities take for K classes (L16), with every θ_k free.

- adding the same vector to every θ_k changes nothing: fix one, e.g. $\theta_K = 0$, to make the coefficients unique
- two classes, labels 0 and 1, with $\theta_0 = 0$: $e^{\theta_1^\top x} / (1 + e^{\theta_1^\top x}) = \sigma(\theta_1^\top x)$, the definition above

Aside: Several classes

Definition (Multinomial logistic regression)

$$\mathbb{P}(Y = k \mid X = x) = \frac{e^{\theta_k^\top x}}{\sum_{j=1}^K e^{\theta_j^\top x}},$$

the form LDA's probabilities take for K classes (L16), with every θ_k free.

- adding the same vector to every θ_k changes nothing: fix one, e.g. $\theta_K = 0$, to make the coefficients unique
- two classes, labels 0 and 1, with $\theta_0 = 0$: $e^{\theta_1^\top x} / (1 + e^{\theta_1^\top x}) = \sigma(\theta_1^\top x)$, the definition above
- scikit-learn fits this by default when $K > 2$

One form, two likelihoods

LDA

maximize the joint likelihood

$$\prod_i \pi_{y_i} \mathcal{N}(x_i \mid \mu_{y_i}, \Sigma)$$

- estimates π_k, μ_k, Σ , which determine $\theta = (b, w)$
- also models x : a mixture of Gaussians with a shared Σ

Logistic regression

maximize the conditional likelihood

$$\prod_i \mathbb{P}(Y = y_i \mid X = x_i)$$

- estimates θ directly
- says nothing about how x is distributed!

The Bernoulli likelihood gives **log loss**

With $p_i = \sigma(\theta^\top x_i)$, maximizing

$$L(\theta) = \prod_{i=1}^n \mathbb{P}(Y = y_i \mid X = x_i) = \prod_{i=1}^n p_i^{y_i} (1 - p_i)^{1-y_i}$$

is the same as minimizing the average negative log-likelihood

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^n \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right].$$

model for the response	maximum likelihood minimizes
Gaussian noise	squared loss
Bernoulli label	log loss

The gradient **looks familiar**

Problem

Using $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, compute the gradient of

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^n \left[y_i \log \sigma(\theta^\top x_i) + (1 - y_i) \log(1 - \sigma(\theta^\top x_i)) \right].$$

The gradient **looks familiar**

Problem

Using $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, compute the gradient of

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^n \left[y_i \log \sigma(\theta^\top x_i) + (1 - y_i) \log(1 - \sigma(\theta^\top x_i)) \right].$$

least squares

$$\nabla \text{MSE}(\theta) = \frac{2}{n} X^\top (X\theta - y)$$

logistic regression

$$\nabla \ell(\theta) = \frac{1}{n} X^\top (p - y), \quad p_i = \sigma(\theta^\top x_i)$$

The gradient **looks familiar**

Problem

Using $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, compute the gradient of

$$\ell(\theta) = -\frac{1}{n} \sum_{i=1}^n \left[y_i \log \sigma(\theta^\top x_i) + (1 - y_i) \log(1 - \sigma(\theta^\top x_i)) \right].$$

least squares

$$\nabla \text{MSE}(\theta) = \frac{2}{n} X^\top (X\theta - y)$$

logistic regression

$$\nabla \ell(\theta) = \frac{1}{n} X^\top (p - y), \quad p_i = \sigma(\theta^\top x_i)$$

Idea: $\nabla \ell(\theta) = 0$ is nonlinear in θ , which enters through σ , so there is no closed form. The standard is to solve it **iteratively**.

Python: Logistic regression

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=np.inf) # the MLE
logreg.fit(X_train, y_train)
logreg.intercept_, logreg.coef_ # b and w
p_hat = logreg.predict_proba(X_test)[: , 1]
y_hat = logreg.predict(X_test) # p_hat >= 0.5
```

- column 1 of `predict_proba` estimates $\mathbb{P}(Y = 1 \mid X = x)$
- the default `C=1.0` adds a ridge penalty on w (a Gaussian prior);
`C=np.inf` gives the maximum likelihood estimate
- if a hyperplane separates the two classes, the estimate does not exist:
keep `C` finite

Plan for today

- Naive Bayes 8 min
- Two repairs so far 2 min
- The logistic model 13 min
- From probabilities to decisions 15 min
- Coefficients; generative vs. discriminative 7 min

Definition: Confusion matrix

Positive ($y = 1$) and **negative** ($y = 0$) classes. Classify x positive when $\hat{\eta}_1(x) \geq t$ for a **threshold** t .

	predicted positive	predicted negative
actually positive	true positive (TP)	false negative (FN)
actually negative	false positive (FP)	true negative (TN)

Note: Each threshold t gives a different confusion matrix.

Example: Credit default data with threshold $t = 0.5$

Logistic regression on balance, income, and student indicator; fitted on 5,000 customers, tested on the other 5,000:

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

- $\text{error} = (FP + FN)/n = 122/5,000 = 2.44\%$

Example: Credit default data with threshold $t = 0.5$

Logistic regression on balance, income, and student indicator; fitted on 5,000 customers, tested on the other 5,000:

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

- $\text{error} = (FP + FN)/n = 122/5,000 = 2.44\%$
- the trivial rule “flag no one” has error $167/5,000 = 3.34\%$

Rates from the **rows** and the **columns**

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

Rates from the **rows** and the **columns**

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

- **sensitivity (= recall)** = $\frac{TP}{TP+FN} = \frac{60}{167} = 35.9\%$ defaulters flagged

Rates from the **rows** and the **columns**

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

- **sensitivity (= recall)** = $\frac{TP}{TP+FN} = \frac{60}{167} = 35.9\%$ defaulters flagged
- **specificity** = $\frac{TN}{TN+FP} = \frac{4,818}{4,833} = 99.7\%$ non-defaulters not flagged

Rates from the **rows** and the **columns**

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

- **sensitivity (= recall)** = $\frac{TP}{TP+FN} = \frac{60}{167} = 35.9\%$ defaulters flagged
- **specificity** = $\frac{TN}{TN+FP} = \frac{4,818}{4,833} = 99.7\%$ non-defaulters not flagged
- **precision** = $\frac{TP}{TP+FP} = \frac{60}{75} = 80.0\%$ flags that are defaulters

Rates from the **rows** and the **columns**

	flagged	not flagged	total
defaulted	TP = 60	FN = 107	167
did not default	FP = 15	TN = 4,818	4,833
total	75	4,925	5,000

- **sensitivity** (= **recall**) = $\frac{TP}{TP+FN} = \frac{60}{167} = 35.9\%$ defaulters flagged
- **specificity** = $\frac{TN}{TN+FP} = \frac{4,818}{4,833} = 99.7\%$ non-defaulters not flagged
- **precision** = $\frac{TP}{TP+FP} = \frac{60}{75} = 80.0\%$ flags that are defaulters

Note: Sensitivity and specificity divide by a row total (a true class); precision divides by a column total (the flags).

Example: Which threshold?

Problem

A false alarm costs 1 and a missed default costs 9.

rule	TP	FP	FN	TN	error	sensitivity	specificity	cost
flag no one	0	0	167	4,833	3.34%	0%	100%	1,503
$\hat{\eta}_1(x) \geq 0.5$	60	15	107	4,818	2.44%	35.9%	99.7%	978
$\hat{\eta}_1(x) \geq 0.1$	128	273	39	4,560	6.24%	76.6%	94.4%	624

Example: Which threshold?

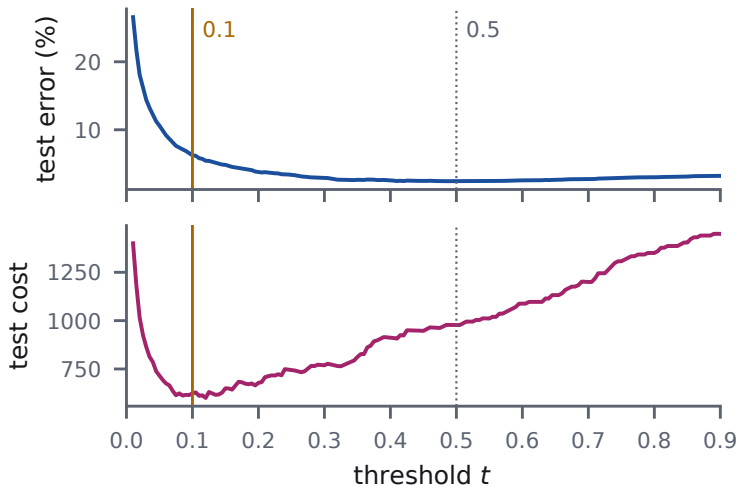
Problem

A false alarm costs 1 and a missed default costs 9.

rule	TP	FP	FN	TN	error	sensitivity	specificity	cost
flag no one	0	0	167	4,833	3.34%	0%	100%	1,503
$\hat{\eta}_1(x) \geq 0.5$	60	15	107	4,818	2.44%	35.9%	99.7%	978
$\hat{\eta}_1(x) \geq 0.1$	128	273	39	4,560	6.24%	76.6%	94.4%	624

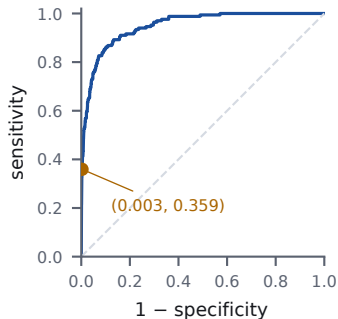
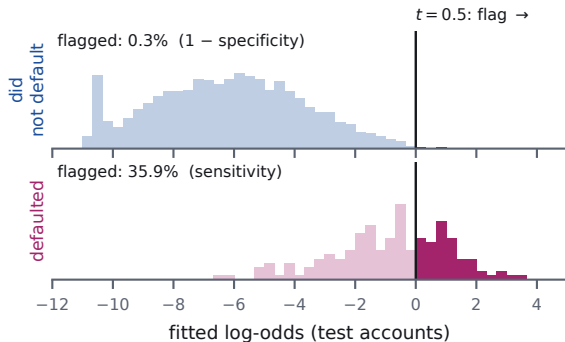
Idea: The threshold $t = \frac{1}{1+9} = 0.1$ has the **highest** error and the **lowest** cost. It trades specificity for sensitivity.

Error and cost pick **different** thresholds



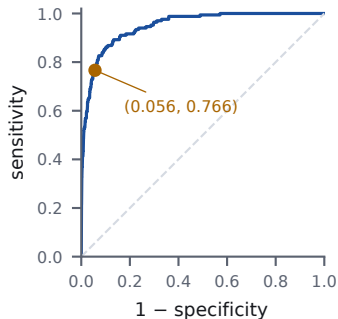
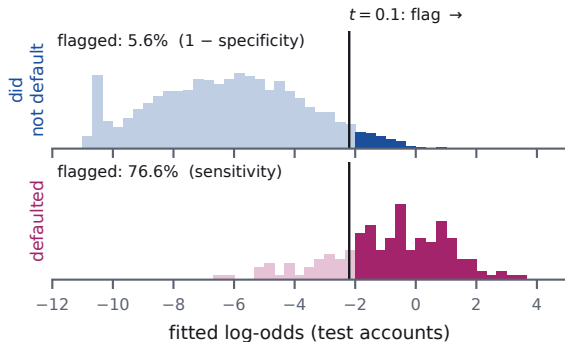
From thresholds to the ROC curve

Each threshold gives one point, $(1 - \text{specificity}, \text{sensitivity})$. Lowering t flags more accounts of both kinds.



From thresholds to the ROC curve

Each threshold gives one point, $(1 - \text{specificity}, \text{sensitivity})$. Lowering t flags more accounts of both kinds.

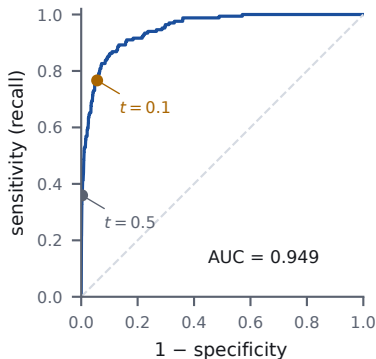


Definition: The ROC curve

Definition (ROC curve)

Sensitivity against $1 - \text{specificity} = \text{FP}/(\text{FP} + \text{TN})$, one point per threshold.

- from $(0, 0)$, flag no one, to $(1, 1)$, flag everyone

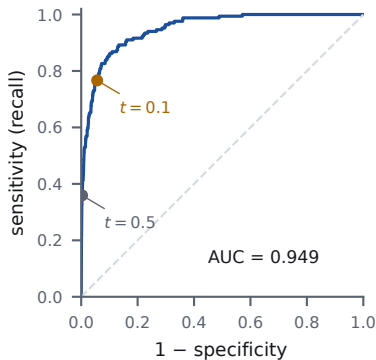


Definition: The ROC curve

Definition (ROC curve)

Sensitivity against $1 - \text{specificity} = \text{FP}/(\text{FP} + \text{TN})$, one point per threshold.

- from $(0, 0)$, flag no one, to $(1, 1)$, flag everyone
- flagging at random with probability q gives (q, q) : the diagonal

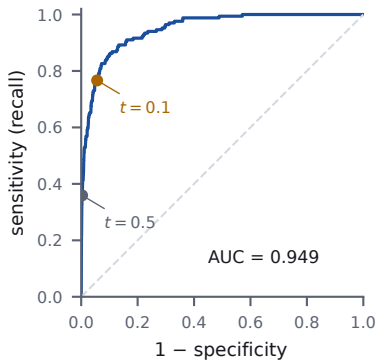


Definition: The ROC curve

Definition (ROC curve)

Sensitivity against $1 - \text{specificity} = \text{FP}/(\text{FP} + \text{TN})$, one point per threshold.

- from $(0, 0)$, flag no one, to $(1, 1)$, flag everyone
- flagging at random with probability q gives (q, q) : the diagonal
- a perfect rule passes through $(0, 1)$



What the **AUC** measures

Definition (AUC)

The area under the ROC curve. It equals the chance that a randomly chosen defaulter gets a higher $\hat{\eta}_1$ than a randomly chosen non-defaulter.

- 0.5 for random flagging, 1 for a perfect ranking; 0.949 here

What the **AUC** measures

Definition (AUC)

The area under the ROC curve. It equals the chance that a randomly chosen defaulter gets a higher $\hat{\eta}_1$ than a randomly chosen non-defaulter.

- 0.5 for random flagging, 1 for a perfect ranking; 0.949 here
- if only the share of defaulters changes, the curve does not: sensitivity and specificity condition on the true class (precision changes)

What the **AUC** measures

Definition (AUC)

The area under the ROC curve. It equals the chance that a randomly chosen defaulter gets a higher $\hat{\eta}_1$ than a randomly chosen non-defaulter.

- 0.5 for random flagging, 1 for a perfect ranking; 0.949 here
- if only the share of defaulters changes, the curve does not: sensitivity and specificity condition on the true class (precision changes)
- AUC measures ranking, not a decision: choosing t still takes the costs

Python: Confusion matrix and ROC

```
from sklearn.metrics import (confusion_matrix,
                              recall_score, roc_auc_score, roc_curve)
p_hat = logreg.predict_proba(X_test)[: , 1]
flag = p_hat >= 0.5
confusion_matrix(y_test, flag)  # [[TN, FP], [FN, TP]]
recall_score(y_test, flag)      # sensitivity
recall_score(y_test, flag, pos_label=0)  # specificity
fpr, tpr, thresholds = roc_curve(y_test, p_hat)
roc_auc_score(y_test, p_hat)    # AUC
```

- rows and columns are sorted by label, 0 first: the negative class comes first, the reverse of our table

Python: Tune, then threshold

```
pipe = make_pipeline(StandardScaler(),  
                      LogisticRegression())  
grid = {"logisticregression__C": np.logspace(-3, 3, 7)}  
search = GridSearchCV(pipe, grid, cv=5,  
                      scoring="neg_log_loss")  
search.fit(X_train, y_train) # C = tau^2  
p_hat = search.predict_proba(X_test)[: , 1]  
flag = p_hat >= 1 / (1 + 9) # costs 1 and 9
```

- `search.predict` uses 0.5, i.e., equal costs

Plan for today

- Naive Bayes 8 min
- Two repairs so far 2 min
- The logistic model 13 min
- From probabilities to decisions 15 min
- Coefficients; generative vs. discriminative 7 min

Example: Are students riskier?

Logistic regression on credit default data, coefficient of the indicator “is a student”:

student indicator alone	+0.40
student indicator, balance, income	−0.65

Example: Are students riskier?

Logistic regression on credit default data, coefficient of the indicator “is a student”:

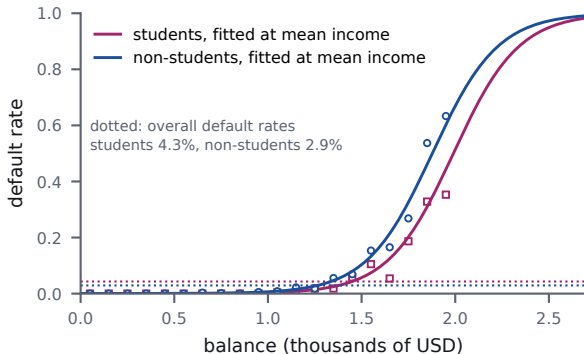
student indicator alone	+0.40
student indicator, balance, income	−0.65

Question

The first says students are more likely to default; the second says less. How can both be right?

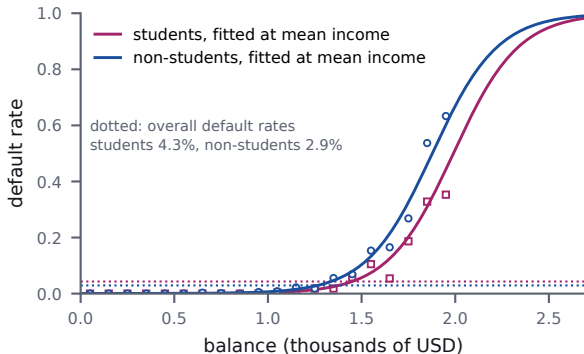
Example: Are students riskier?

- students default more:
4.3% vs. 2.9%



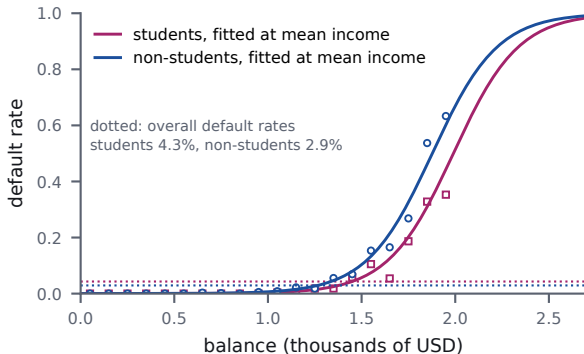
Example: Are students riskier?

- students default more:
4.3% vs. 2.9%
- and carry larger balances:
\$988 vs. \$772



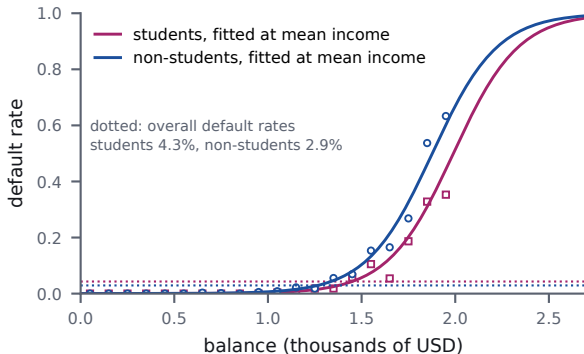
Example: Are students riskier?

- students default more:
4.3% vs. 2.9%
- and carry larger balances:
\$988 vs. \$772
- fixed balance and income:
 $\text{odds} \times e^{-0.65} \approx 0.52$



Example: Are students riskier?

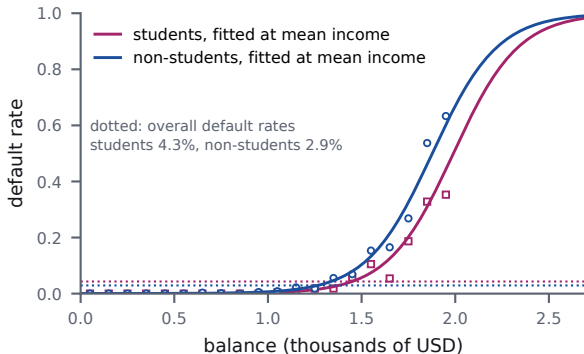
- students default more:
4.3% vs. 2.9%
- and carry larger balances:
\$988 vs. \$772
- fixed balance and income:
 $\text{odds} \times e^{-0.65} \approx 0.52$



Example: Are students riskier?

- students default more:
4.3% vs. 2.9%
- and carry larger balances:
\$988 vs. \$772
- fixed balance and income:
 $\text{odds} \times e^{-0.65} \approx 0.52$

Note: e^{β_j} : the odds multiplier per unit of feature j , others fixed. An association, not an effect.



Generative, discriminative, **discriminant**

generative

model each class, π_k and $f_k(x)$; get $\eta_k(x)$ from Bayes' rule

QDA, LDA (L15–L16), naive Bayes; fit by the joint likelihood

discriminative

model $\eta_k(x)$, or the boundary, directly; no model for x

k -NN (L14); logistic regression, fit by the conditional likelihood

Generative, discriminative, **discriminant**

generative

model each class, π_k and $f_k(x)$; get $\eta_k(x)$ from Bayes' rule

QDA, LDA (L15–L16), naïve Bayes; fit by the joint likelihood

discriminative

model $\eta_k(x)$, or the boundary, directly; no model for x

k -NN (L14); logistic regression, fit by the conditional likelihood

Watch out (Names): Linear and quadratic *discriminant* analysis are generative. “Discriminant” refers to the functions $\delta_k(x)$ that are compared to classify, a term from Fisher’s 1936 linear discriminant.

Logistic regression or LDA?

- if the classes really are Gaussian with a shared covariance, logistic regression is asymptotically about 30% less efficient (Efron, 1975)

Logistic regression or LDA?

- if the classes really are Gaussian with a shared covariance, logistic regression is asymptotically about 30% less efficient (Efron, 1975)
- if not, LDA's extra assumption can bias it

Logistic regression or LDA?

- if the classes really are Gaussian with a shared covariance, logistic regression is asymptotically about 30% less efficient (Efron, 1975)
- if not, LDA's extra assumption can bias it
- Default: test AUC 0.949 for both

Logistic regression or LDA?

- if the classes really are Gaussian with a shared covariance, logistic regression is asymptotically about 30% less efficient (Efron, 1975)
- if not, LDA's extra assumption can bias it
- Default: test AUC 0.949 for both
- wine, 20 splits, correct of 88: LDA 73.5, logistic 76.9, QDA 78.9

Logistic regression or LDA?

- if the classes really are Gaussian with a shared covariance, logistic regression is asymptotically about 30% less efficient (Efron, 1975)
- if not, LDA's extra assumption can bias it
- Default: test AUC 0.949 for both
- wine, 20 splits, correct of 88: LDA 73.5, logistic 76.9, QDA 78.9

Logistic regression or LDA?

- if the classes really are Gaussian with a shared covariance, logistic regression is asymptotically about 30% less efficient (Efron, 1975)
- if not, LDA's extra assumption can bias it
- Default: test AUC 0.949 for both
- wine, 20 splits, correct of 88: LDA 73.5, logistic 76.9, QDA 78.9

Key idea: The unequal class covariances make LDA's assumption doubtful. Logistic regression does better here despite linear log-odds; QDA, which models the covariances, does best.

Summary

- naive Bayes: features independent within each class; with Gaussian features, QDA with diagonal covariances, and far fewer parameters
- logistic regression models the log-odds as linear, the form LDA implies, and maximizes the conditional likelihood
- a confusion matrix counts correct and incorrect predictions; sensitivity and specificity are rates within the true classes, precision within the flags
- ROC curve shows sensitivity against 1 – specificity for every threshold; AUC measures ranking
- a decision needs costs: flag when $\hat{\eta}_1 \geq c_{FP} / (c_{FP} + c_{FN})$