Classification

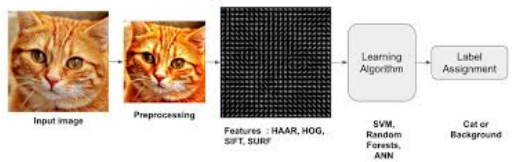
Jonathan Hersh[1]

Chapman University, Argyros School of Business
hersh@chapman.edu

February 27, 2018

How do ML engineers use classification?

- **Image classification**

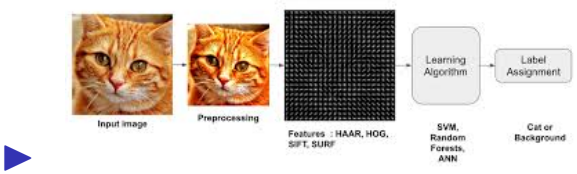


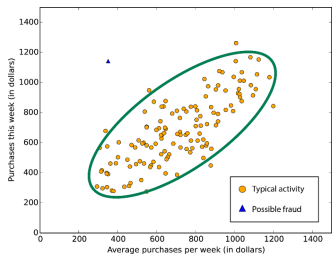
How do ML engineers use classification?

- **Image classification**
- **Speech recognition**

How do ML engineers use classification?

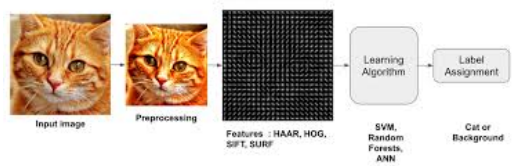
- **Image classification**
- **Speech recognition**
- **Fraud detection**





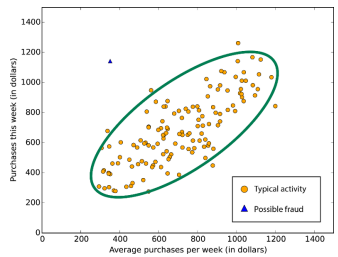
How do ML engineers use classification?

- **Image classification**



- **Speech recognition**

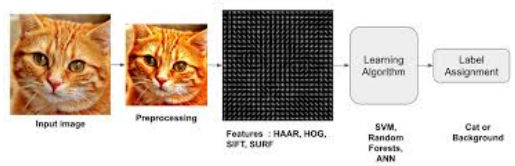
- **Fraud detection**

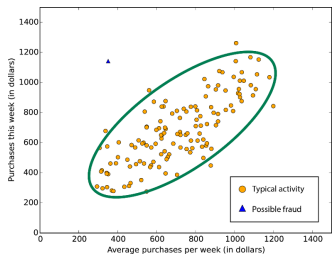


- **Spam detection**

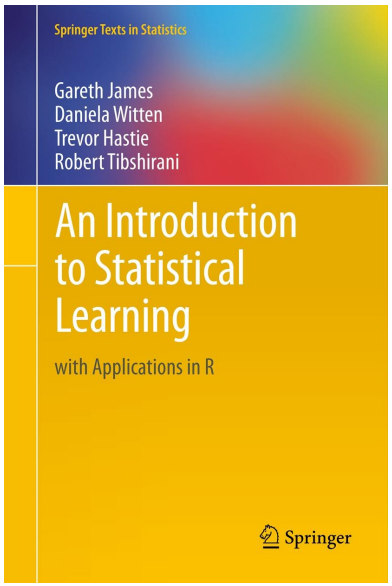
How do ML engineers use classification?

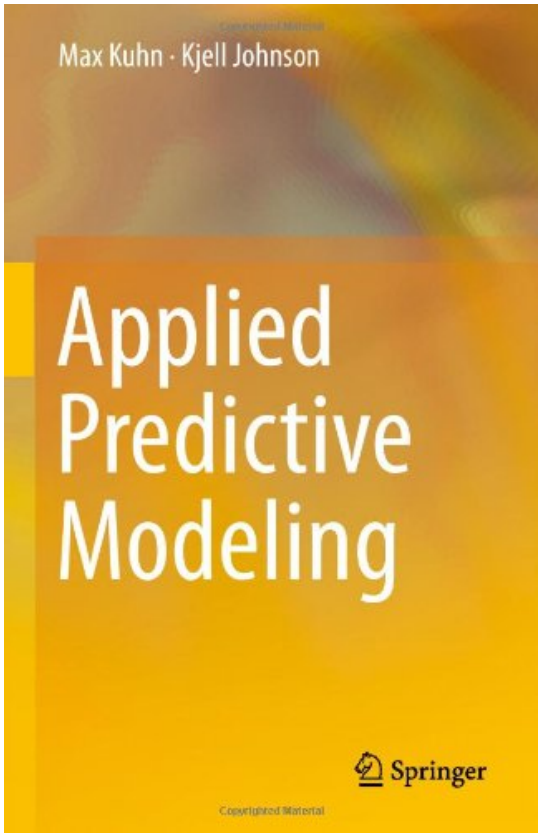
- **Image classification**
- **Speech recognition**
- **Fraud detection**
- **Spam detection**
- **Advertising**





Source material





- **ISLR Chapter 4**; APM Chapters 11, 12 & 16

Plan

Machine Learning Classification Methods

Linear Classification Methods

- Linear Regression
- Probit
- Logit
- Linear Discriminant Analysis
- Regularized Probit/Logit

Machine Learning Classification Methods

Linear Classification Methods

- Linear Regression
- Probit
- Logit
- Linear Discriminant Analysis
- Regularized Probit/Logit

Nonlinear Methods

- Neural Networks
- Support Vector Machines (SVM)
- K-Nearest Neighbors (k-NN)
- Regression Trees
- Random Forests
- Deep learning (autoencoders)

What is classification?

- Modeling of dependent variable in a discrete class
- Include binary dependent variable models:

$$y_i \in \{\text{spam, not spam}\}$$

$$y_i \in \{\text{poor, not poor}\}$$

What is classification?

- Modeling of dependent variable in a discrete class
- Include binary dependent variable models:

$$y_i \in \{\text{spam, not spam}\}$$

$$y_i \in \{\text{poor, not poor}\}$$

- As well as multinomial dependent variable models

$$y_i \in \{\text{brown, black, blonde, red}\}$$

What is classification?

- Modeling of dependent variable in a discrete class
- Include binary dependent variable models:

$$y_i \in \{\text{spam, not spam}\}$$

$$y_i \in \{\text{poor, not poor}\}$$

- As well as multinomial dependent variable models

$$y_i \in \{\text{brown, black, blonde, red}\}$$

- Often we are more interested in **class probabilities**, rather than classifying objects themselves

Example: credit card default

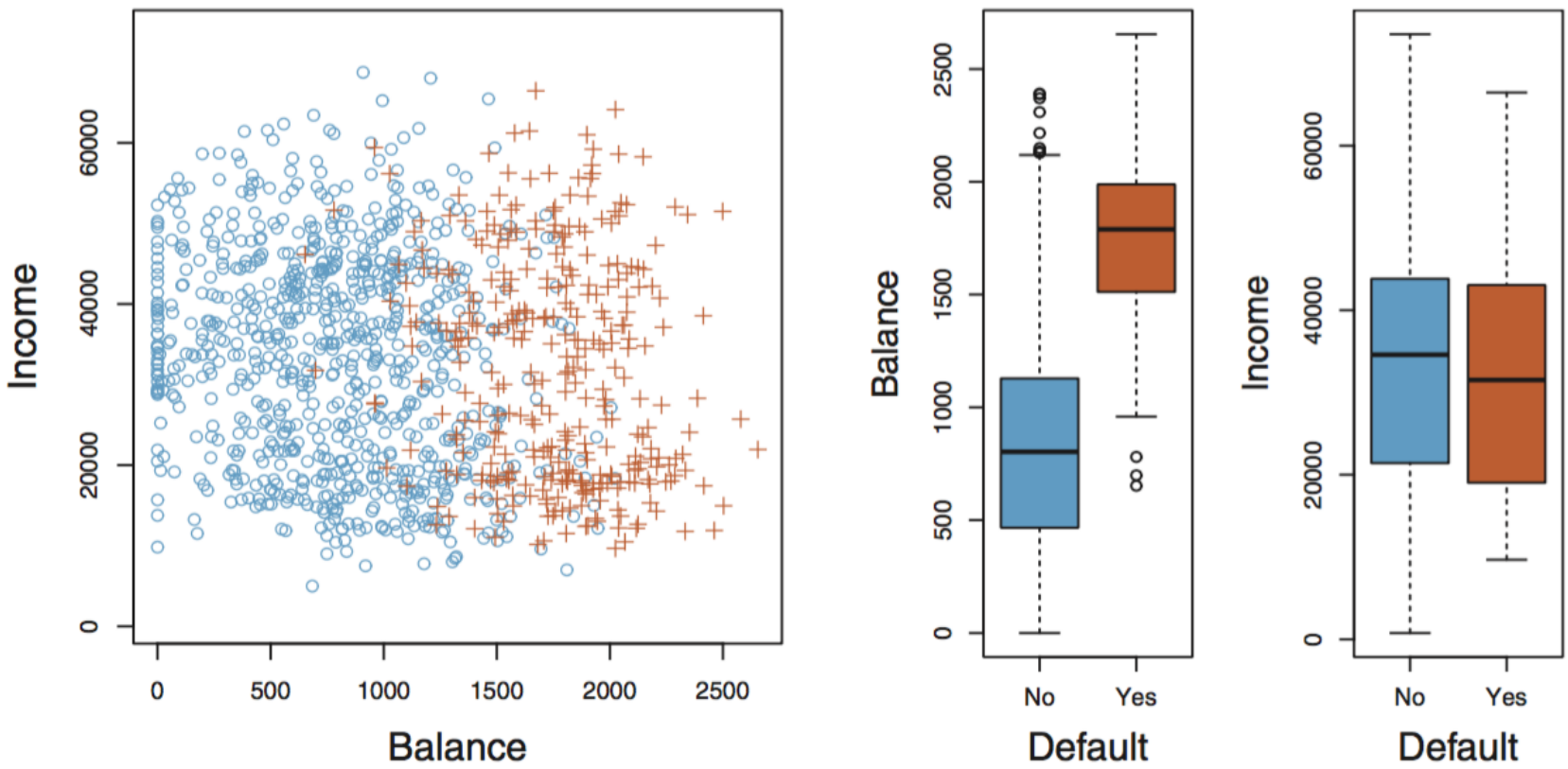


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of* `default` *status.* Right: *Boxplots of* `income` *as a function of* `default` *status.*

Can we use linear regression?

- For the default classification task

$$Y = \begin{array}{ll} 0 & \text{if } \textbf{No} \text{ default} \\ 1 & \text{if } \textbf{Yes} \text{ default} \end{array}$$

- Can we just linearly regress X on Y? $\Rightarrow$ classify as **Yes** if $\hat{y} > 0.5$?

Can we use linear regression?

- For the default classification task

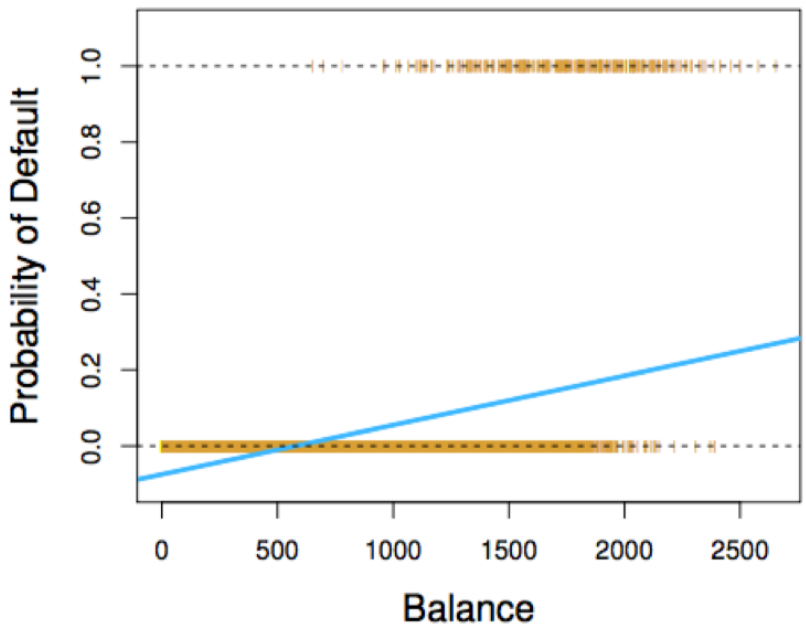
$$Y = \begin{array}{ll} 0 & \text{if } \textbf{No} \text{ default} \\ 1 & \text{if } \textbf{Yes} \text{ default} \end{array}$$

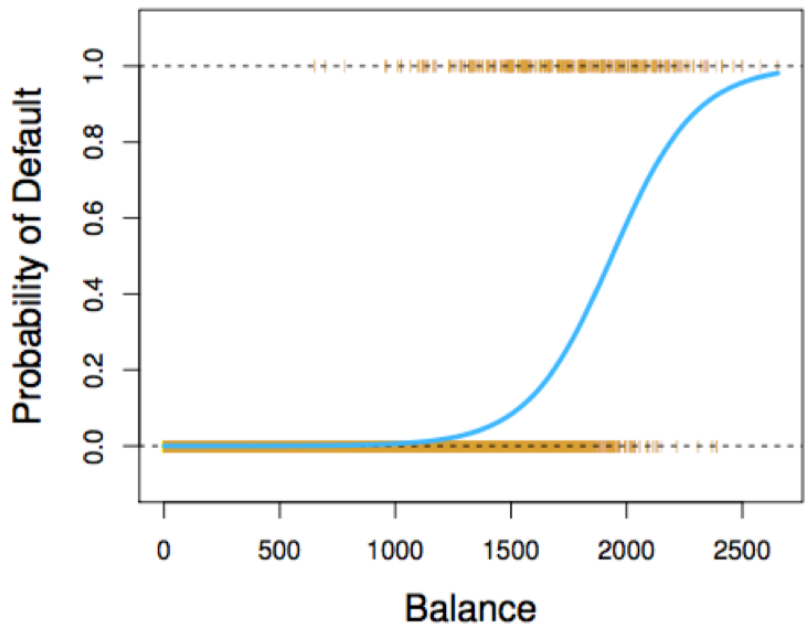
- Can we just linearly regress X on Y? $\Rightarrow$ classify as **Yes** if $\hat{y} > 0.5$?
- In many cases, yes, as $\mathbb{E}[Y|X = x] = Pr(Y = 1|X = x)$
- However, this might produce $\hat{y} \notin [0, 1]$, which may be a problem for prediction $\Rightarrow$ **Logistic regression**

Logistic regression

- Logistic regression uses a logit transform to ensure predicted values are always between 0 and 1.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_1+\beta_1 X}}$$





Making predictions

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

$$p(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055*1000}}{1+e^{-10.6513+0.0055*1000}} = 0.006$$

Making predictions

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

$$p(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055*1000}}{1+e^{-10.6513+0.0055*1000}} = 0.006$$

- with a balance of \$2000?

$$p(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055*2000}}{1+e^{-10.6513+0.0055*2000}} = 0.586$$

Where this can go wrong

- **It turns out, for many variables, estimation via maximum likelihood breaks down**

Where this can go wrong

- **It turns out, for many variables, estimation via maximum likelihood breaks down**

- To see this, note that we estimate (that is, choose βs) via maximum likelihood

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p((x_i))$$

Where this can go wrong

- **It turns out, for many variables, estimation via maximum likelihood breaks down**

- To see this, note that we estimate (that is, choose βs) via maximum likelihood

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p((x_i))$$

- Our likelihood often becomes non-concave, and can't estimate coefficients with precision

Regularized logistic

$$\beta_{\text{LogitLasso}} = \underset{\beta}{\text{argmin}} \sum_{j=1}^{N} \left\{ \underbrace{y_j \left(X_j^{\mathsf{T}} \beta \right) - \ln \left(1 + \exp \left(X_j^{\mathsf{T}} \beta \right) \right)}_{\text{Logistic LLH}} + \underbrace{\lambda \sum_{j=1}^{K} |\beta_j|}_{\text{regularization}} \right\}$$

- Performs very well given large number of variables
- **Cross-validation ensures model doesn't overfit**

Plan

Confusion Matrix

		Observed	
		False (Y = 0)	**True (Y = 1)**
Predicted	**False (Y=0)**	True Negative (TN)	False Negative (FN)
	True (Y=1)	False Positive (FN)	True Positive (TP)

Confusion Matrix

		Observed	
		False (Y = 0)	**True (Y = 1)**
Predicted	**False (Y=0)**	True Negative (TN)	False Negative (FN)
	True (Y=1)	False Positive (FN)	True Positive (TP)

Diagonals (good job)

- TN: Predicted false, true false
- TP: Predicted true, observed true

Confusion Matrix

		Observed	
		False (Y = 0)	**True (Y = 1)**
Predicted	**False (Y=0)**	True Negative (TN)	False Negative (FN)
	True (Y=1)	False Positive (FN)	True Positive (TP)

Diagonals (good job)

- TN: Predicted false, true false
- TP: Predicted true, observed true

Off-diagonals (bad job)

- FP: Predicted true, observed false (Type I Error)
- FN: Predicted false, observed true (Type II Error)

Example confusion matrix with default data

		True default status		
		No	**Yes**	
Predicted default status	**No**	9,644	252	9,896
	Yes	23	81	104
		9,667	333	10,000

- **Accuracy:** "How often is the classifier correct?"
  $(TP + TN)/Total = (9,644 + 81)/10,000 = 97.25$

Example confusion matrix with default data

		True default status		
		No	**Yes**	
Predicted default status	**No**	9,644	252	9,896
	Yes	23	81	104
		9,667	333	10,000

- **Accuracy:** "How often is the classifier correct?"
  $(TP + TN)/Total = (9,644 + 81)/10,000 = 97.25$
- **Mis-classification rate:** "How often is the classifier wrong?"
  $(FP + FN)/Total = (23 + 252)/10,000 = 2.75$

Example confusion matrix with default data

		True default status		
		No	**Yes**	
Predicted default status	**No**	9,644	252	9,896
	Yes	23	81	104
		9,667	333	10,000

- **Accuracy:** "How often is the classifier correct?"
  $(TP + TN)/Total = (9,644 + 81)/10,000 = 97.25$
- **Mis-classification rate:** "How often is the classifier wrong?"
  $(FP + FN)/Total = (23 + 252)/10,000 = 2.75$
- Note if we classified everything to **No, we would make 333/1000 errors, only 3.33% error rate !**

Example confusion matrix with default data

		True default status		
		No	**Yes**	
Predicted default status	**No**	9,644	252	9,896
	Yes	23	81	104
		9,667	333	10,000

- **Accuracy:** "How often is the classifier correct?"
  $(TP + TN)/Total = (9,644 + 81)/10,000 = 97.25$
- **Mis-classification rate:** "How often is the classifier wrong?"
  $(FP + FN)/Total = (23 + 252)/10,000 = 2.75$
- Note if we classified everything to **No, we would make 333/1000 errors, only 3.33% error rate !**
- Our classifier seems unbalanced:
  - Of the true **No**'s: $23/9667 = 0.2\%$ errors!
  - Of the true **Yes**'s: $252/333 = 75.7\%$ errors!

Tradeoff between FP and FN

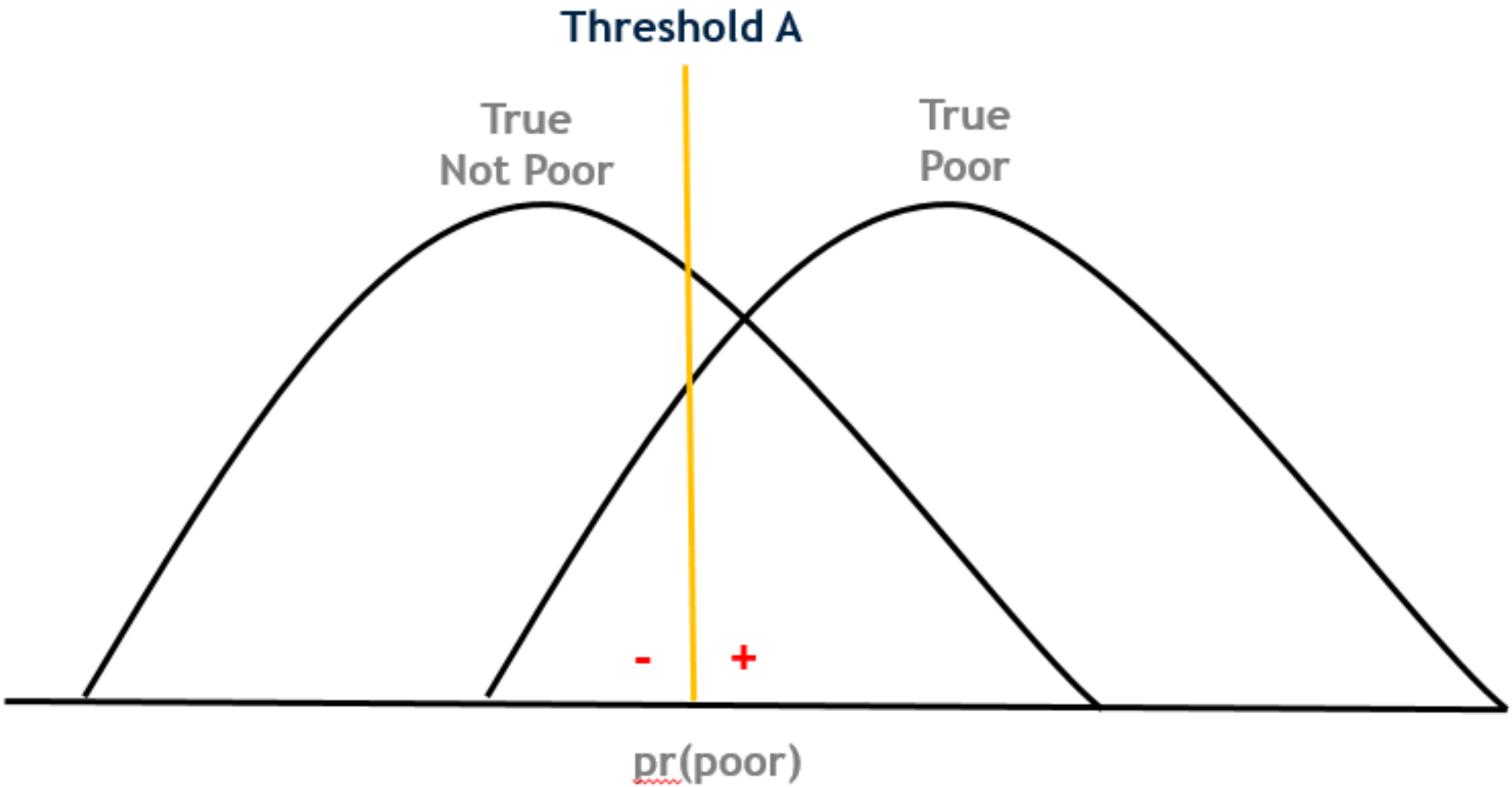
- Think of two medical tests:
  1. **One that often flags a disease** (at the expense of flagging many healthy patients)
  2. **One that seldom flags a disease** (at the expense of not flagging many sick patients)

Tradeoff between FP and FN

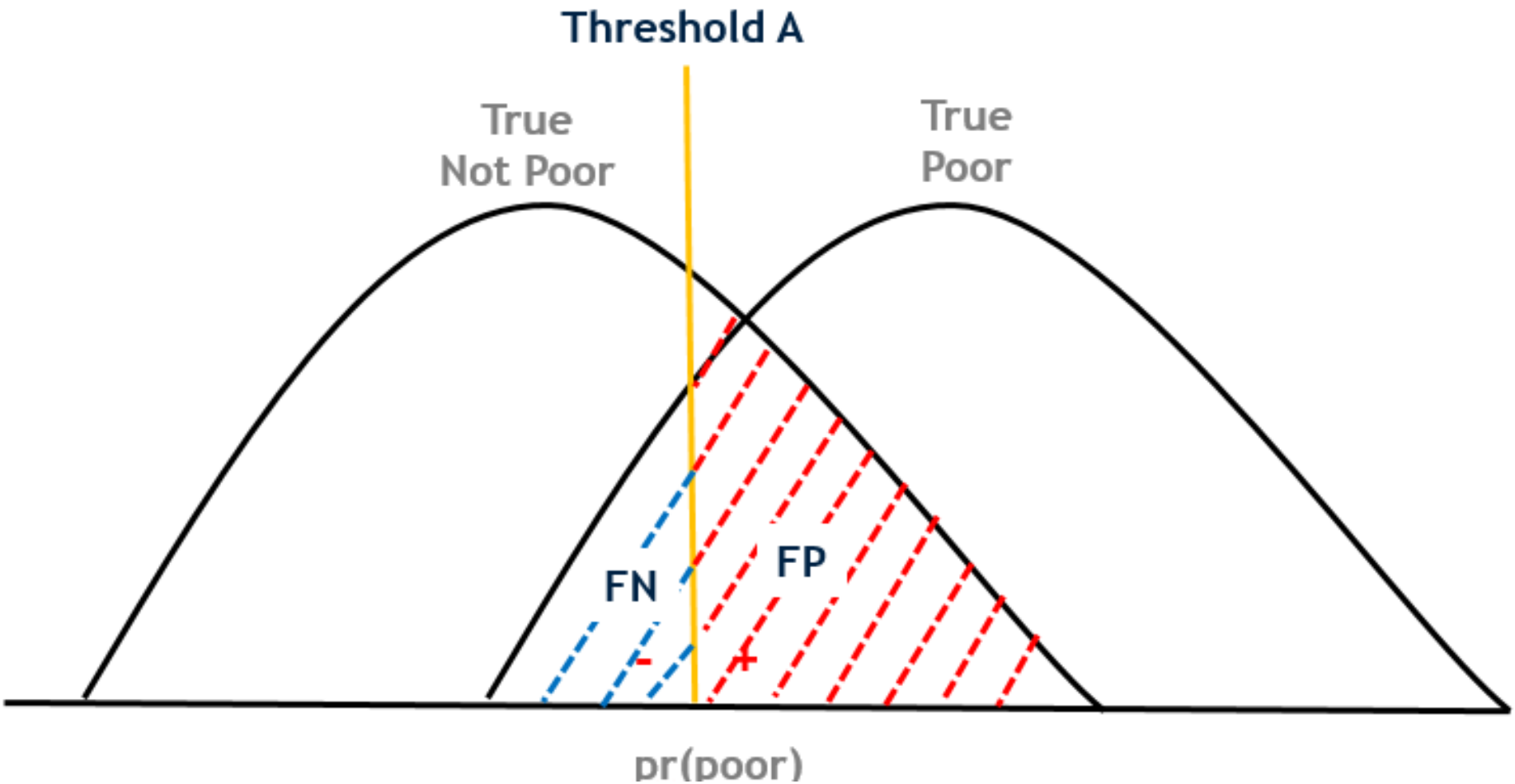
- Think of two medical tests:
  1. **One that often flags a disease** (at the expense of flagging many healthy patients)
  2. **One that seldom flags a disease** (at the expense of not flagging many sick patients)

- **In ML, we stay that test 1 has a high sensitivity, low specificity**
- **and test 2 has a low sensitivity, high specificity**
  - Specificity "Proportion of negatives correctly identified"
  - Sensitivity: "Proportion of positives correctly identified"

Specificity and Sensitivity Tradeoff

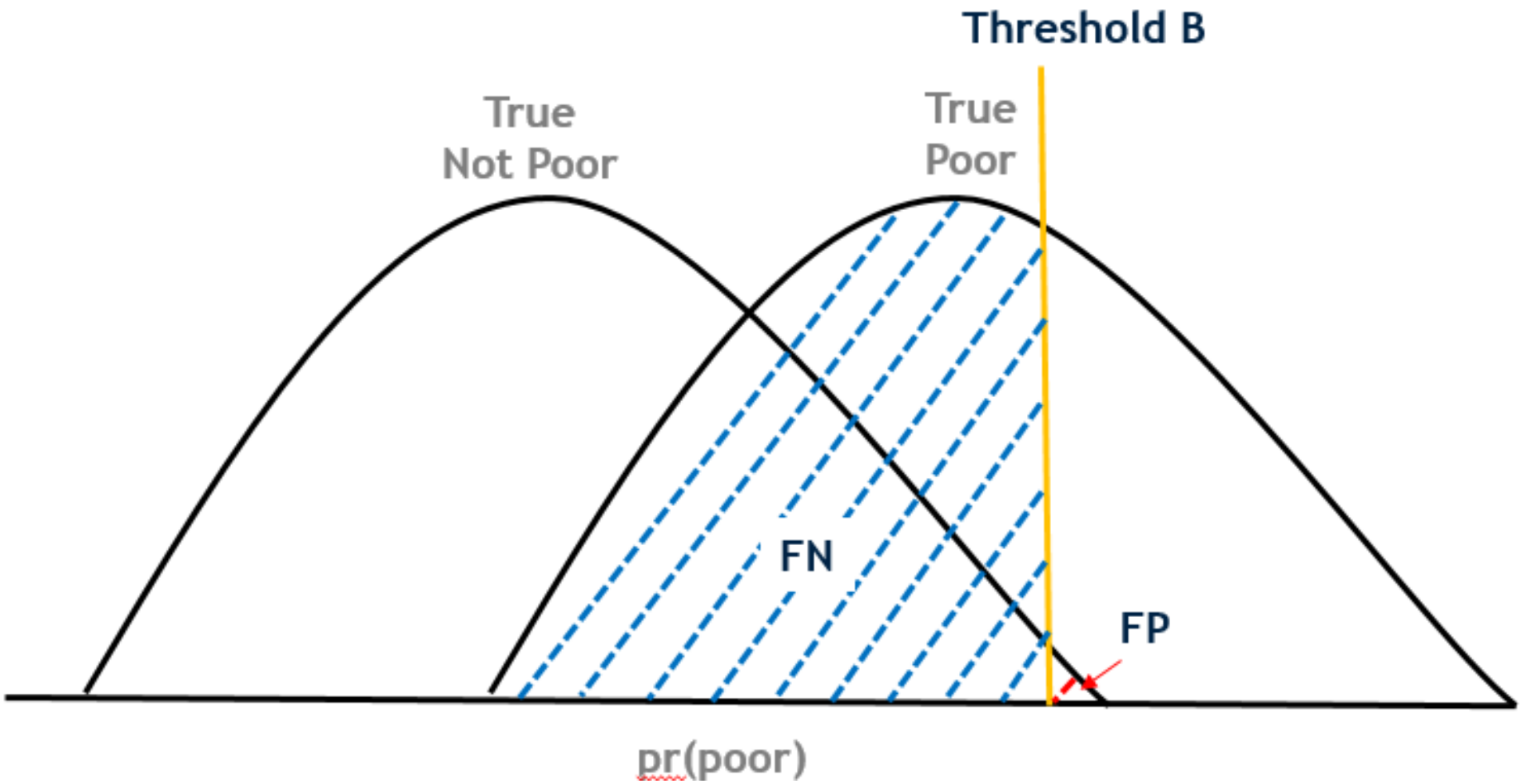


Specificity and Sensitivity Tradeoff



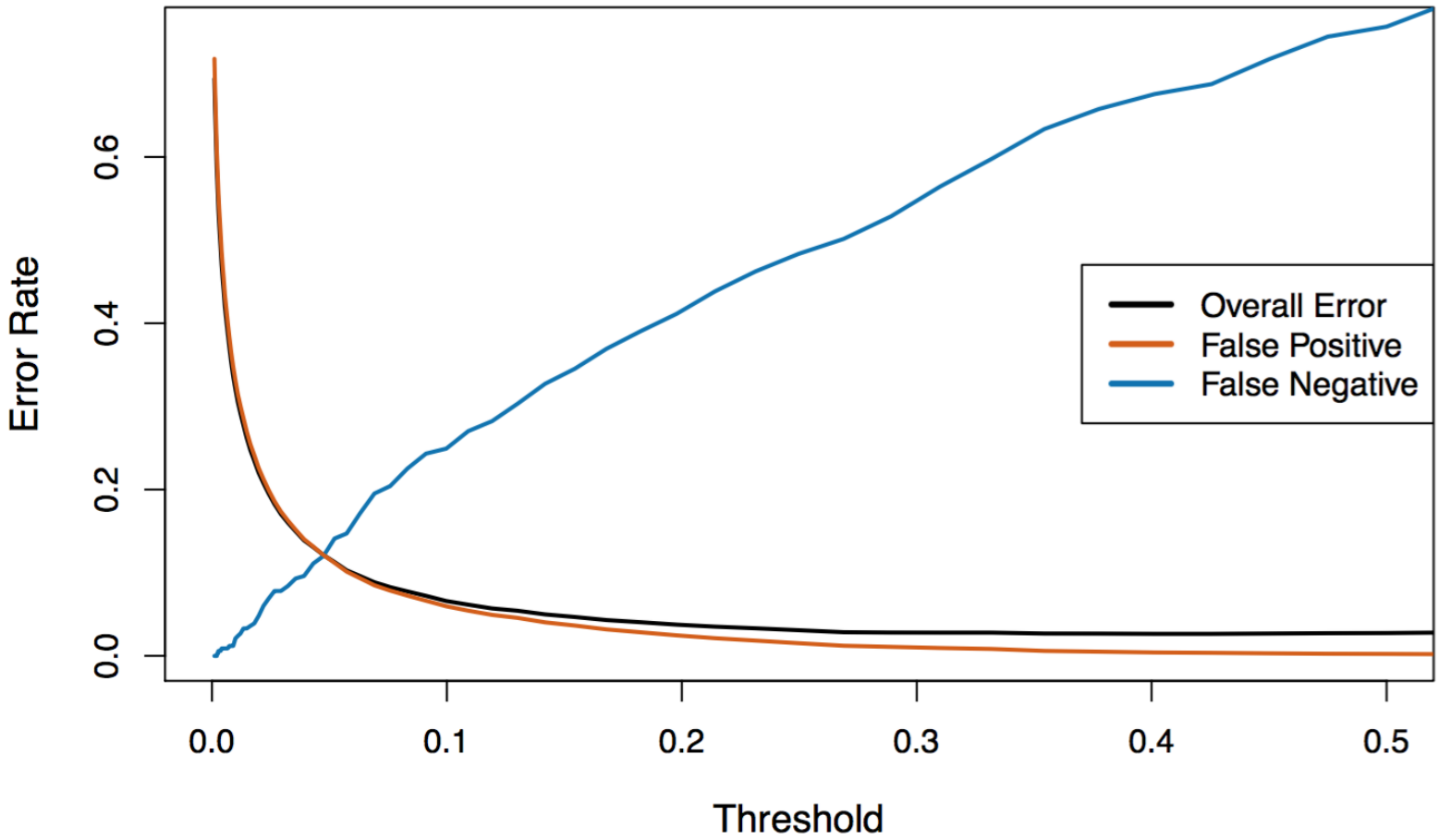
- Threshold A is highly sensitive – high TPR

Specificity and Sensitivity Tradeoff



- Threshold B has high specificity – high TNR

Varying the threshold



ROC Curve

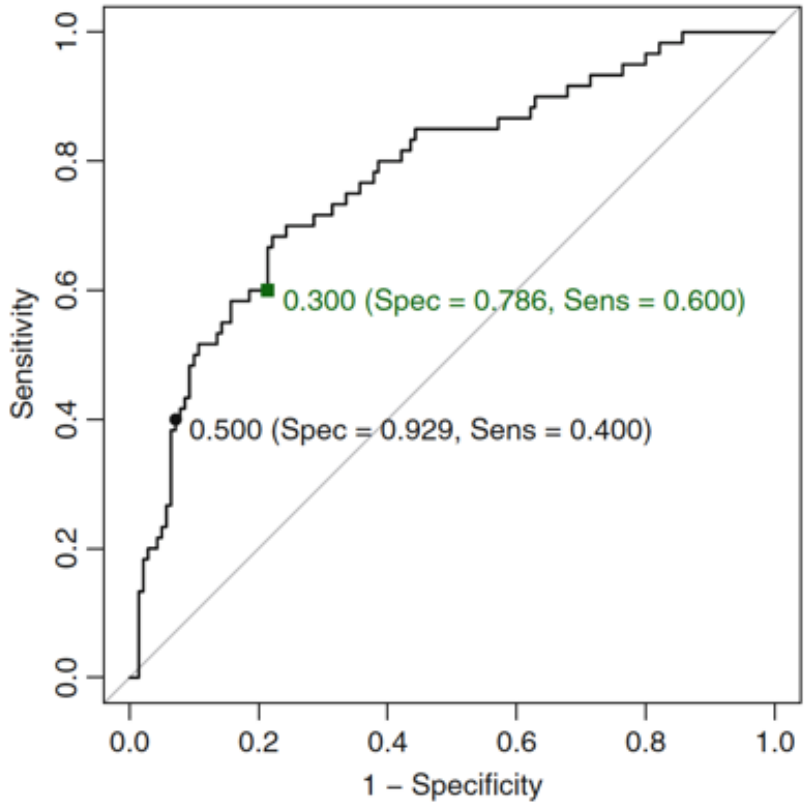


Fig. 11.6: A receiver operator characteristic (ROC) curve for the logistic regression model results for the credit model. The dot indicates the value corresponding to a cutoff of 50 % while the green square corresponds to a cutoff of 30 % (i.e., probabilities greater than 0.30 are called events)

- **AUC**: "Area under the curve"
- **AUC:** 1 = perfect accuracy**; Diagonal line:** no better than chance

Lift Chart

- **Lift charts are a visualization tool for assessing accuracy in binary models**
- It shows **best and worst models** (perfect accuracy and random chance), showing how a given model performs relative to these two

Lift Chart

- **Lift charts are a visualization tool for assessing accuracy in binary models**
- It shows **best and worst models** (perfect accuracy and random chance), showing how a given model performs relative to these two
- To construct a lift charge, use any method to get predicted probabilities $\hat{p}_i$, then order observations by these $\hat{p}_i$.

Lift Chart

- **Lift charts are a visualization tool for assessing accuracy in binary models**
- It shows **best and worst models** (perfect accuracy and random chance), showing how a given model performs relative to these two
- To construct a lift charge, use any method to get predicted probabilities $\hat{p}_i$, then order observations by these $\hat{p}_i$.
- For each $\hat{p}_i$ count whether the observation event occurred
- Calculate counterfactual perfect and random model accuracy

Lift Chart

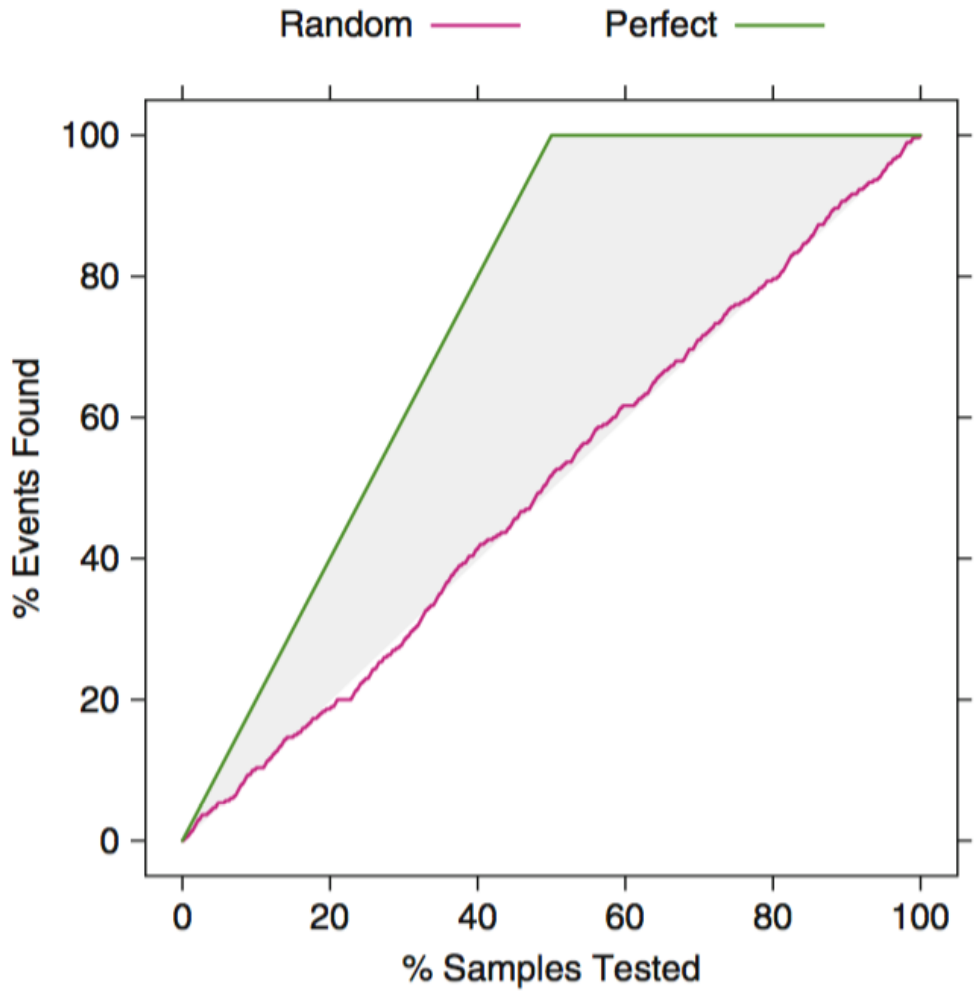


Fig. 11.7: An example lift plot with two models: one that perfectly separates two classes and another that is completely non-informative

Lift Chart example

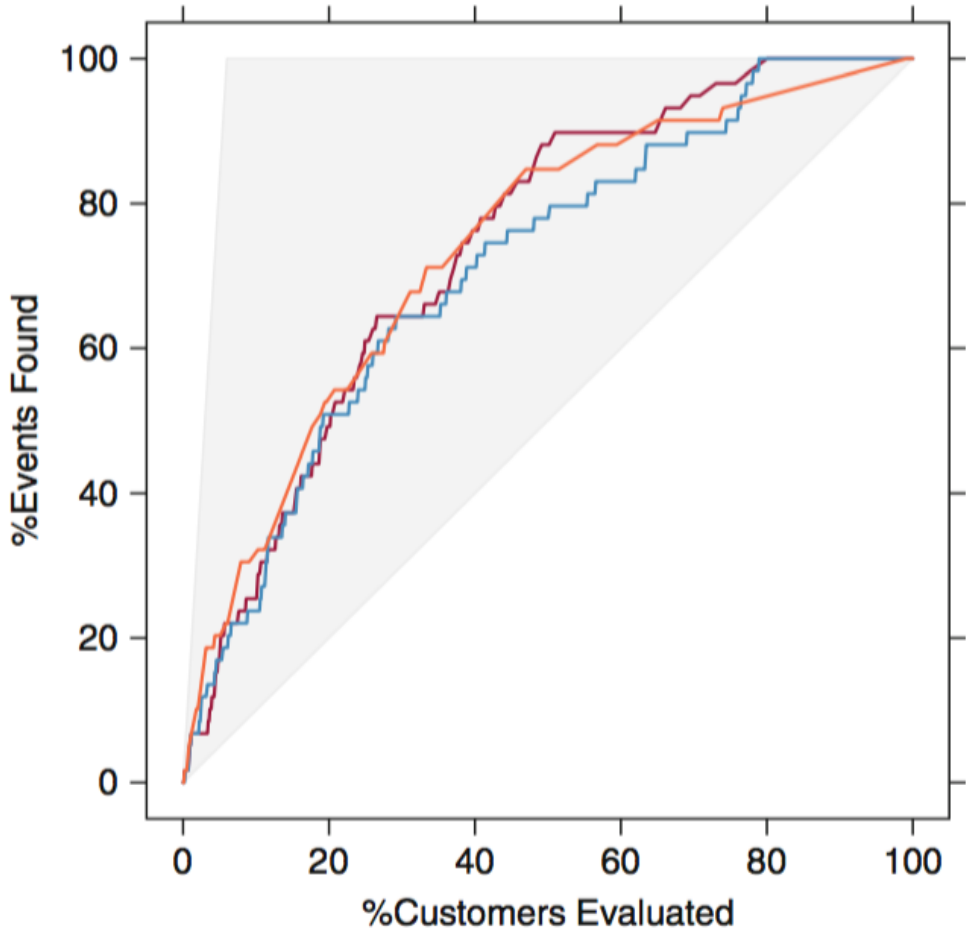


Fig. 16.1: *Top*: Evaluation set ROC curves for each of the three baseline models. *Bottom*: The corresponding lift plots

Severe Class Imbalance

- **Severe class imbalance** occurs when one class is vastly overrepresented
- **The log-likelihood is maximized by settings coefficients to predict well the majority class, and poorly predict the minority class.**

Severe Class Imbalance

- **Severe class imbalance** occurs when one class is vastly overrepresented
- **The log-likelihood is maximized by settings coefficients to predict well the majority class, and poorly predict the minority class.**

Table 16.1: Results for three predictive models using the evaluation set

Model	Accuracy	Kappa	Sensitivity	Specificity	ROC AUC
Random forest	93.5	0.091	6.78	99.0	0.757
FDA (MARS)	93.8	0.024	1.69	99.7	0.754
Logistic regression	93.9	0.027	1.69	99.8	0.727

Severe Class Imbalance

- **Severe class imbalance** occurs when one class is vastly overrepresented
- **The log-likelihood is maximized by settings coefficients to predict well the majority class, and poorly predict the minority class.**

Table 16.1: Results for three predictive models using the evaluation set

Model	Accuracy	Kappa	Sensitivity	Specificity	ROC AUC
Random forest	93.5	0.091	6.78	99.0	0.757
FDA (MARS)	93.8	0.024	1.69	99.7	0.754
Logistic regression	93.9	0.027	1.69	99.8	0.727

- Fancier methods: random forest, neural networks, even deep learning will not solve this

Remedy 1: alternative $\hat{p}$ cutoff

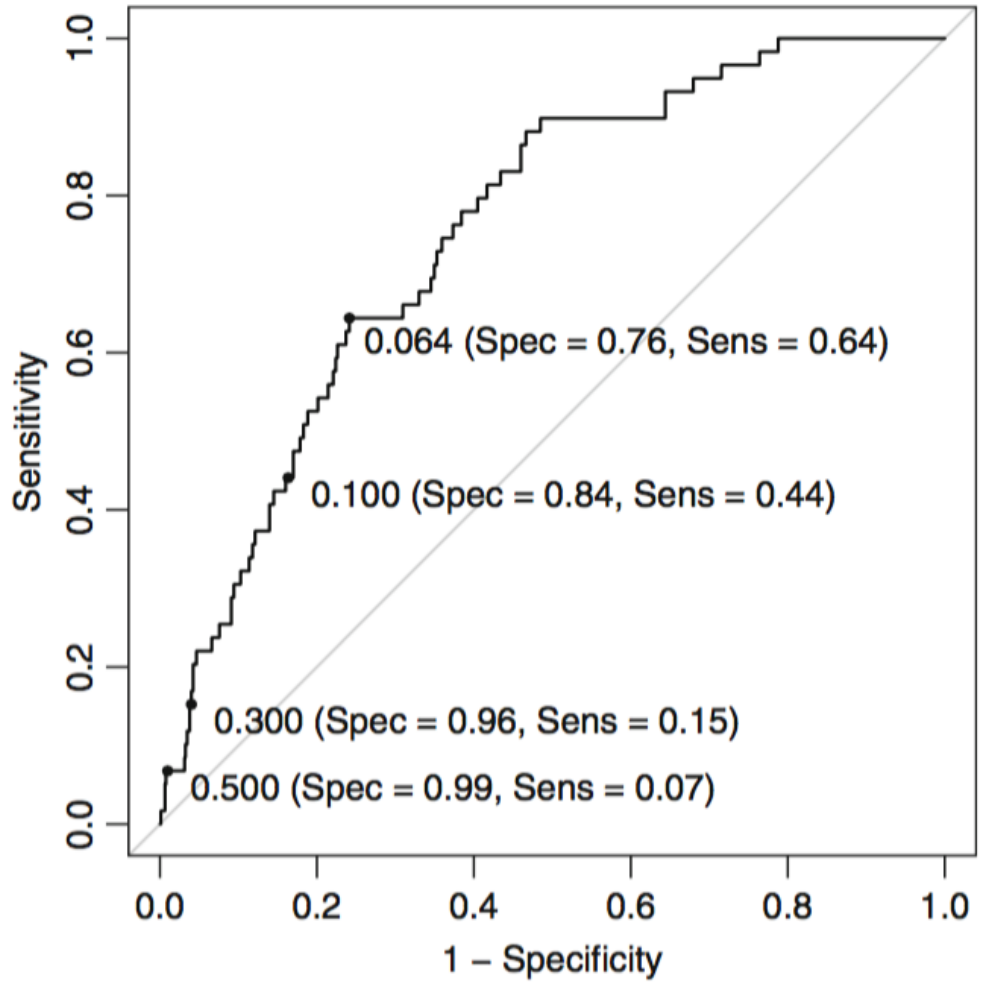


Fig. 16.2: The random forest ROC curve for predicting the classes using the evaluation set. The number on the left represents the probability cutoff, and the numbers in the parentheses are the specificity and sensitivity, respectively. Several possible probability cutoffs are used, including the threshold geometrically closest to the perfect model (0.064)

Remedies 2 & 3: undersampling majority class, SMOTE

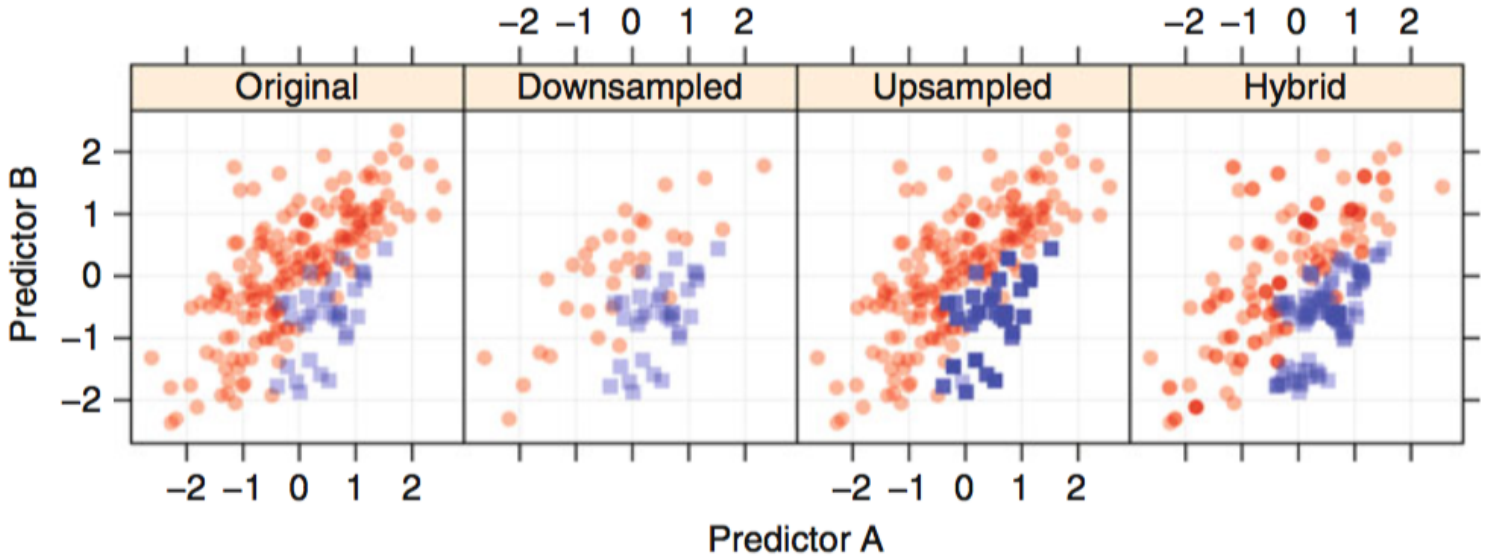
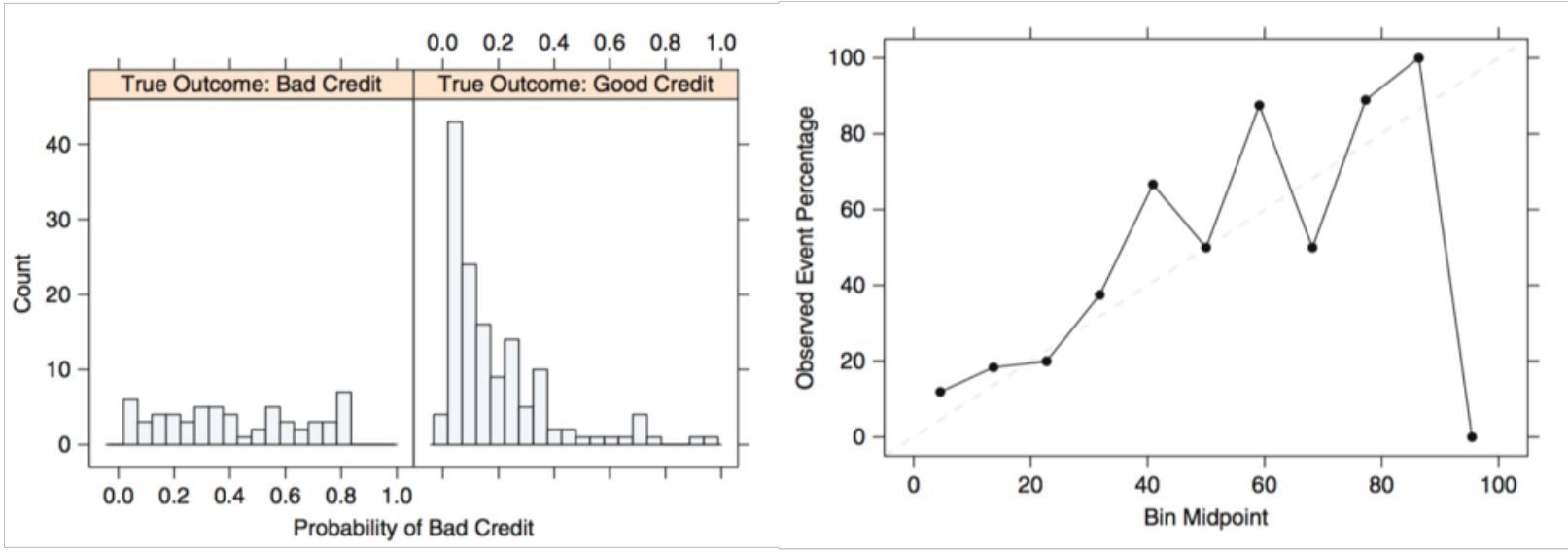


Fig. 16.3: *From left to right*: The original simulated data set and realizations of a down-sampled version, an up-sampled version, and sampling using SMOTE where the cases are sampled and/or imputed

- **SMOTE**: uses interpolation to create new minority classes

Calibration plot to check predictions



- **Calibration plot:** bin $\hat{p}$ by deciles, and plot against observed event frequencies.

Calibration plot to check predictions

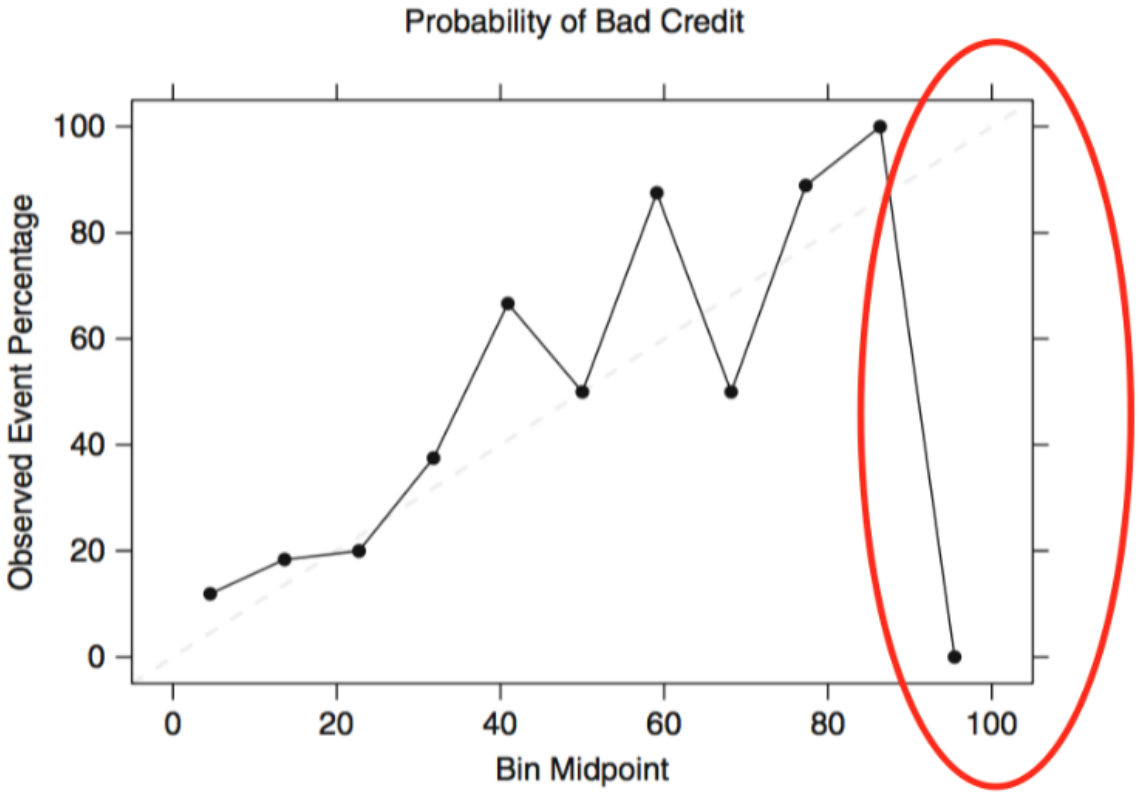


Fig. 11.3: *Top*: Histograms for a set of probabilities associated with bad credit. The two panels split the customers by their true class. *Bottom*: A calibration plot for these probabilities

Plan

Conclusion

- Use confusion matrices to compare predictive performance
- Lift charts present model performance against useful bar of random or perfect assignment

Conclusion

- Use confusion matrices to compare predictive performance
- Lift charts present model performance against useful bar of random or perfect assignment
- Severe class imbalance cannot be solved through fancier methods $\rightarrow$ must use brain
- Calibration plots help model diagnostic