# INFSCI 2915: Machine Learning

## Logistic Regression - Classification

Mai Abdelhakim

School of Computing and Information

610 IS Building

Spring 2018

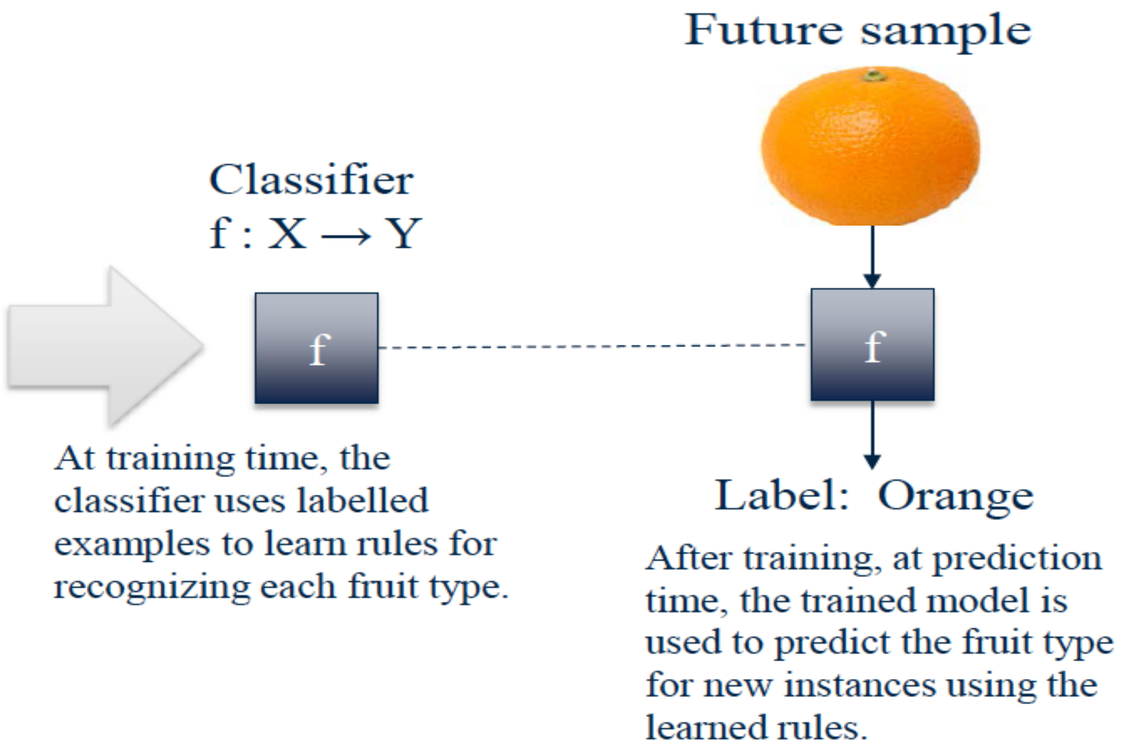
# Classification

- Response is qualitative – Very common problem
- Examples:
  - Email spam detection system: spam or not spam
  - Medical system: Set of symptoms attributed to one of three possible medical conditions. Which of these conditions does the individual have
  - Handwritten digit recognition system: Is the digit 0, 1, 2,..or 9?
- Reponses can be represented by unordered set $C$

# Classification

- Classifier: function that takes feature vector X as an input, and gives corresponding class label Y ∈ C
- Training observations train/build a classifier
- Test data examines accuracy on previously unseen observations by the model
- Example:

| X Sample | Y Target Value (Label) |
|---|---|
|  $x_1$ | Apple $y_1$ |
|  $x_2$ | Lemon $y_2$ |
|  $x_3$ | Apple $y_3$ |
|  $x_4$ | Orange $y_4$ |



# Can We Use Linear Regression for Binary Classification?

- Assume two level quantitative response example:
  - We want to classify whether patient has stoke or drug overdose based on symptoms
  - We can potentially use the dummy variable approach by defining

$$Y = \begin{cases} 0 & \text{if } \texttt{stroke}; \\ 1 & \text{if } \texttt{drug overdose}. \end{cases}$$

  - If the **predicted** Y > 0.5 (**not meaningful**), then classify as drug overdose, otherwise classify as stoke
    - Linear regression may work well for two-level response
    - As if Y estimates the conditional probability: Pr(drug overdose|X)
    - **May not be in correct range**

# Can We Still Use Linear Regression?

- Assume three-level response
  - Three possible diagnosis: stroke, drug overdose and epileptic seizure
  - Then one may use:

$$Y = \begin{cases} 1 & \text{if } \texttt{stroke;} \\ 2 & \text{if } \texttt{drug overdose;} \\ 3 & \text{if } \texttt{epileptic seizure.} \end{cases} \quad \text{OR} \quad Y = \begin{cases} 1 & \text{if } \texttt{epileptic seizure;} \\ 2 & \text{if } \texttt{stroke;} \\ 3 & \text{if } \texttt{drug overdose.} \end{cases}$$

  - Different coding represent different relationship among classes & different linear model
  - Linear regression is not appropriate here
  - There is **no general way to convert qualitative response into quantitative response**

# Logistic Regression

- **A classifier estimates the conditional probability** that X belongs to each possible class label (elements in C)

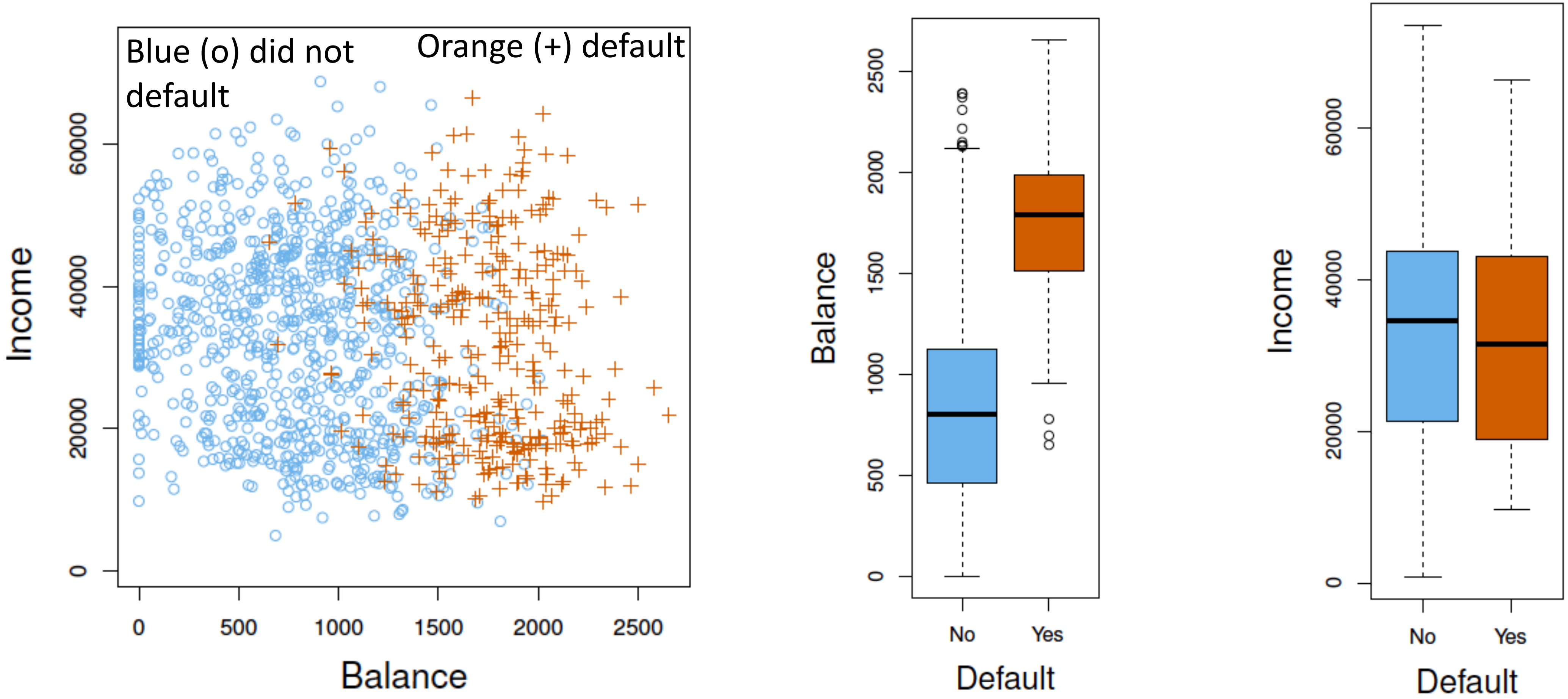
    - What is the probability that the email is spam, and what is the probability that email in not spam

- **The predicted class label is the one with the highest probability**

- **Logistic regression: models the probability** that the **response** Y **belongs to a particular category**

# Example: Default Data Set

- Predict whether an individual will default on his/her credit card payment
  - Two features: income and balance on the credit card
- Dataset contains information of n=10,000 individuals



- Consider that we take a single feature: balance

- Logistic regression models the probability of default given the balance:

$$\Pr(\texttt{default} = \texttt{Yes}|\texttt{balance})$$

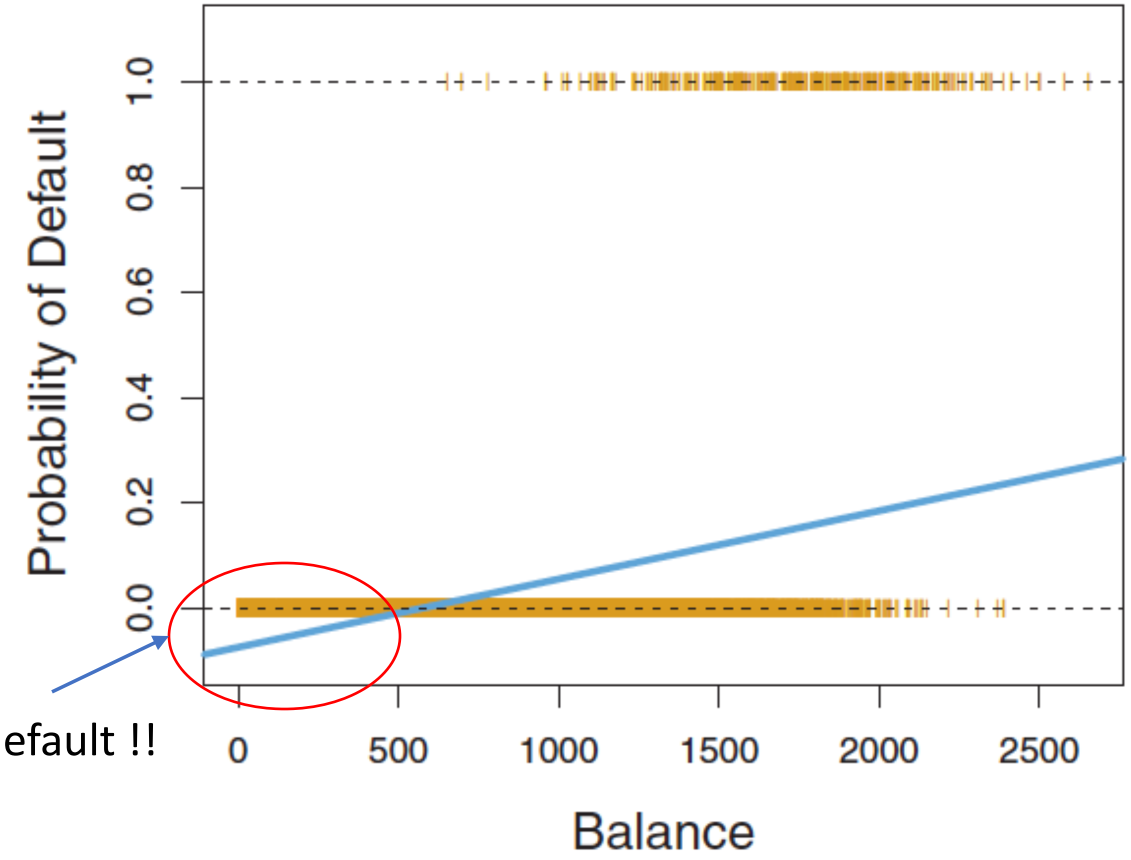
- This conditional probability is in the range [0,1]

# Logistic Model

- Assume two possible classes, and denote: $p(X) = \Pr(Y = 1|X)$
- Now we can assume that this probability is the response we want to predict.
- Can we use the linear regression model?

$$p(X) = \beta_0 + \beta_1 X$$

- Linear regression may result in values less than 0 or greater than 1
  - Not estimate of probability



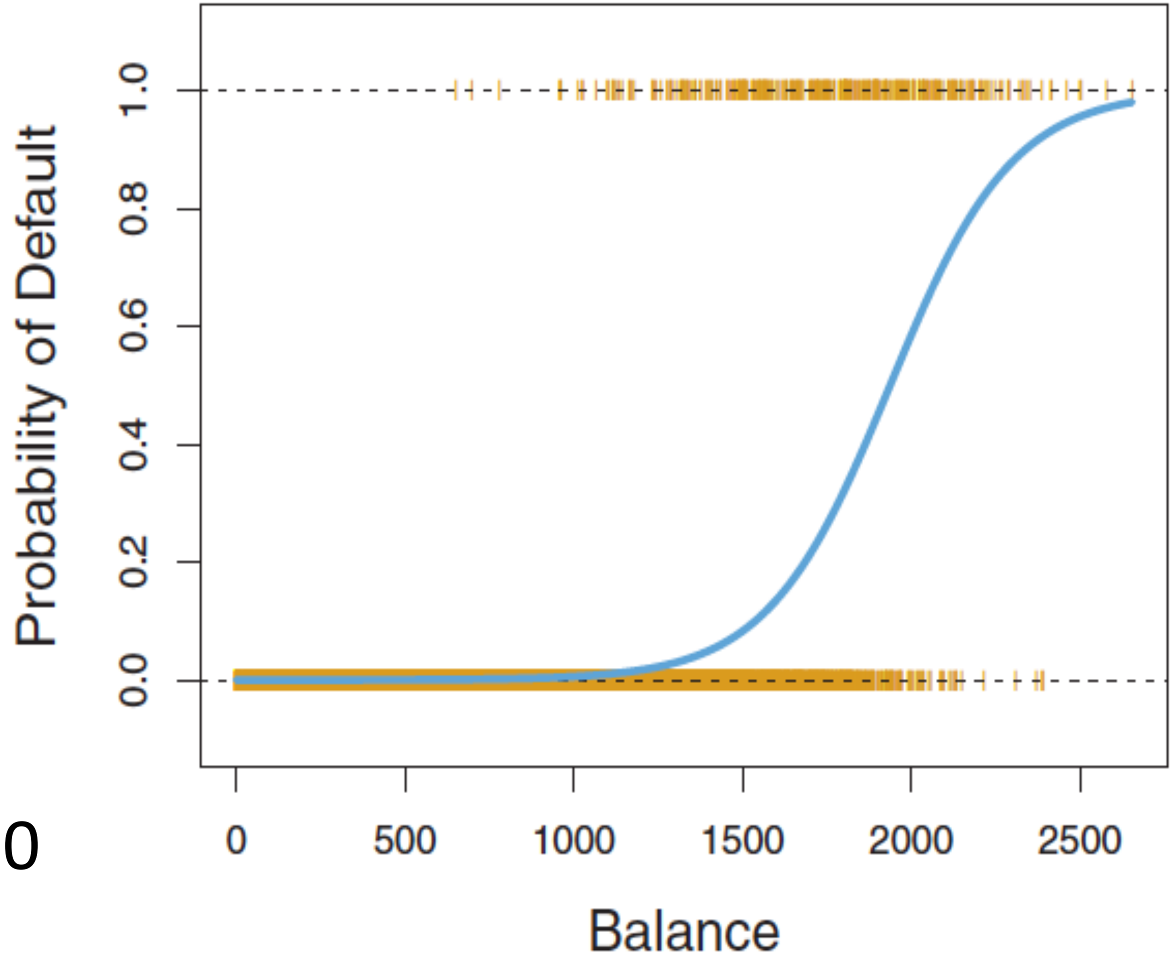

Predict negative probability of default !!

# Logistic Model

- To solve this problem the logistic regression uses the form:

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

*e = 2.71828* is a mathematical constant




- *p(X) is always in the range [0,1]*
- Large X ($\infty$): logistic regression output 1
- Small X ($-\infty$): logistic regression output 0

- We can rearrange the terms, and get logit $\log(p(x)/(1-p(X))$:

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

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

Logistic regression has a **logit** that is **linear** with X

# Estimating Coefficients – Maximum Likelihood

- It is more common to use maximum likelihood to estimate coefficients
- likelihood gives the probability of the observed zeros and ones in the training data.

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

Note that $p(x_i)$is the probability $\Pr(Yi = 1|xi)$..
Therefore, 1- $p(x_i)$ = $\Pr(Yi = 0|xi)$.

- Get coefficients that maximizes the likelihood, then use them for predictions

# Note

- Maximizing the likelihood function is equivalent to minimizing the cost function $J(\beta)$ defined as

$$J(\beta) = -\sum_{i=1}^{n}[y_i log(P(y_i = 1|x)) + (1 - y_i)log(1 - P(y_i = 1|x))]$$

Try to prove!

- Similar aspects to linear regression: accuracy of coefficient estimate and p-value
- Here we have Z-statistics (instead of t-statistics): $\hat{\beta}_1 / SE(\hat{\beta}_1)$

Credit card Default data using Balance

| | Coefficient | Std. Error | Z-statistic | P-value |
|---|---|---|---|---|
| Intercept | -10.6513 | 0.3612 | -29.5 | $< 0.0001$ |
| balance | 0.0055 | 0.0002 | 24.9 | $< 0.0001$ |

# Predictions

- After estimating coefficients we can make predictions.
- Example: estimated probability of default for someone with balance \$1000 is

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

Coefficient values are in the previous table

# Multiple Features

Y is 1 or zero, and is predicted using multiple features

$p(X)$is the probability $\Pr(Y = 1|X)$

$$p(X) = \frac{e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}{1 + e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}$$

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1X_1 + \cdots + \beta_pX_p$$

# Example

| | Coefficient | Std. Error | Z-statistic | P-value |
| --- | --- | --- | --- | --- |
| Intercept | -10.8690 | 0.4923 | -22.08 | $< 0.0001$ |
| balance | 0.0057 | 0.0002 | 24.74 | $< 0.0001$ |
| income | 0.0030 | 0.0082 | 0.37 | 0.7115 |
| student[Yes] | -0.6468 | 0.2362 | -2.74 | 0.0062 |

- Student with card balance of $1,500 and an income of $40,000 has an estimated probability of default:

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times 1{,}500+0.003\times 40-0.6468\times 1}}{1+e^{-10.869+0.00574\times 1{,}500+0.003\times 40-0.6468\times 1}} = 0.058$$

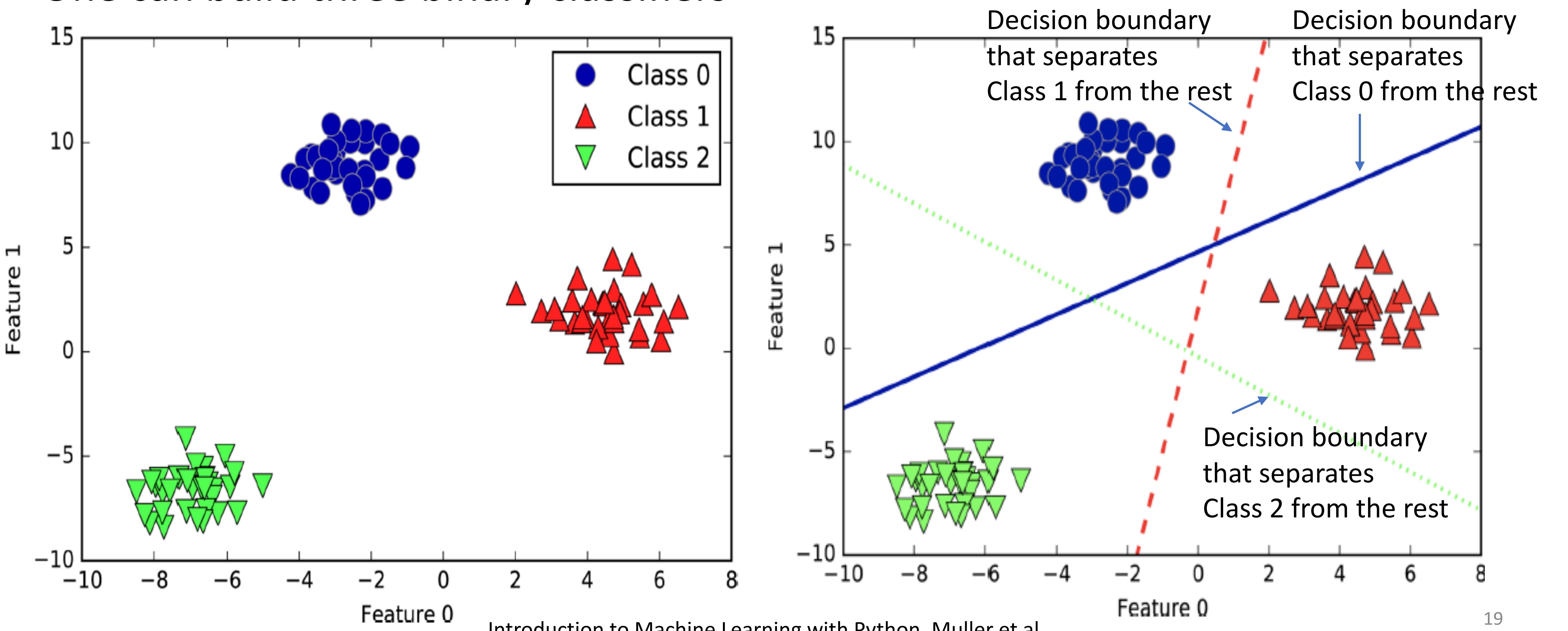
# Multiple Classes

- One vs all
  - Predict whether response is from: class 1 or not, class 2 or not,..
- Train classifier for each class j to predict y=j
- Pick class that has max P(y=j|X)

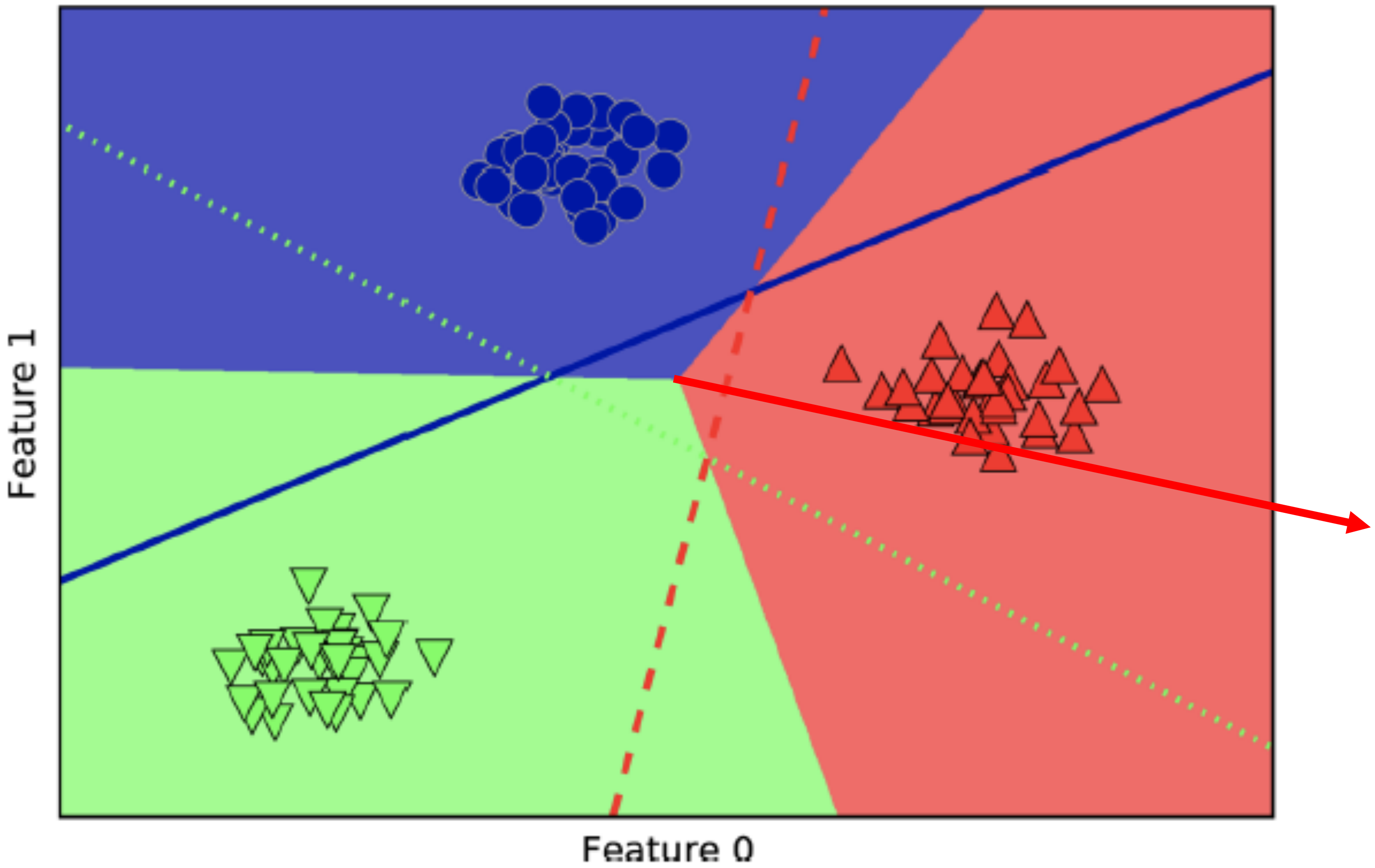
- We'll talk about other methods that are more popular for multiclass classification

# One vs. All for Multiclass Classification

- Assume two features and three classes shown in the Figure
- One can build three binary classifiers




Introduction to Machine Learning with Python, Muller et al.




The region (triangle) where the classification is "rest" by all three binary classifiers: decision is made to assign to the class with the closest boundary

Introduction to Machine Learning with Python, Muller et al

# Logistic Regression in Python

From sklearn.linear_model import LogisticRegression

LogRegModel= LogisticRegression( )

- Use .fit and .score as before.

# Regularization in Logistics Regression

- Regularization can be applied to logistic regression
- Same principles as before:
  - Ridge: All coefficients shrink towards zero
  - Lasso: Shrinks coefficients, and some coefficients will be forced to be zero (feature selection)

# Logistic Regression in Python - Ridge

From **sklearn.linear_model** import **LogisticRegression**
LogRegModel= LogisticRegression(C=100 )

Regularization strength, Ridge
Large C => less regularization

- By **default** logistic regression implements **Ridge regularization** to the classification problem, with strength defined by parameters C.
  - By **default C=1** in LogisticRegression( )
  - **Large C** means **less regularization** strength
    - very large C means is close to the no regularization case
  - **Small C** means **more regularization** and coefficients will be close to zero
  - Note that C is opposite to alpha or $\lambda$

# Logistic Regression in Python - Lasso

- We can implement Lasso regularization (also called L1 regularization) which limits the model to few features

  LogRegModel= LogisticRegression(C=0.1, penalty=”$\ell$1”)

# Finding Class probabilities in Python

- predict_proba: gives the probability of each of the classes given the observation
  - First row is: [probability that first observation is in class 1, probability that the first observation is in class 2,..]
- In python:

```
FittedLogRegModel1= LogisticRegression().fit(X_train_transformed,Y_train)
Probabilities=FittedLogRegModel1.predict_proba(X_test_transformed)
```