EMORY
UNIVERSITY

Emory Continuing Education

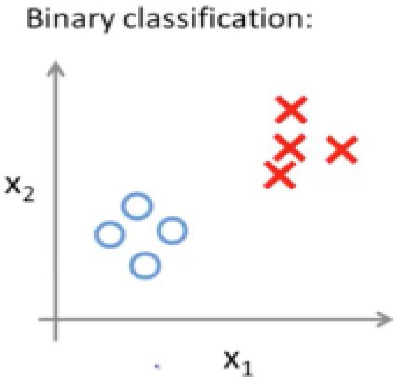
Predictive Analytics with Python (Classification)

Sridhar Palle, Ph.D.

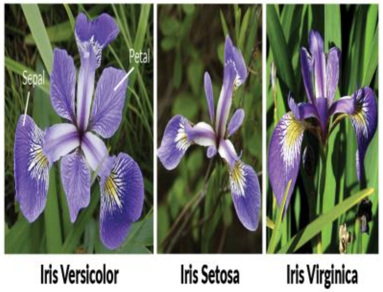
Consort Institute

Classification

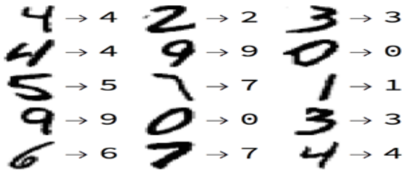
- Goal is to predict a ‘label’ or (‘target variable’) which is discrete (not continuous)

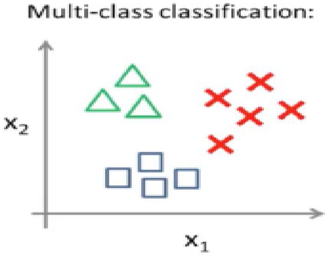


- Types of Classification
  - Binary Classification
    - Pass/fail
    - Yes/no
    - Customer selects a product or not
    - Disease or no disease



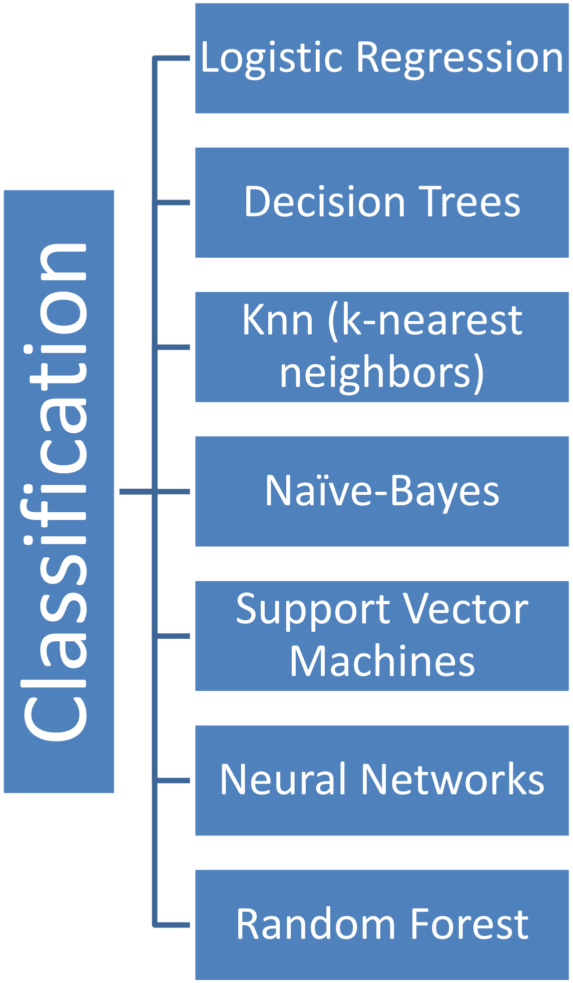
  - Multiclass classification
    - Identify types of flowers
    - Digits recognition
    - Predicting wine types
    - Classify several diseases





Classification Algorithms

– There are a number of ML classification algorithms:



Classification: Logistic Regression

Ex: Predicting if a website link is Phishing or not

Y - Actual Value of variable (0,1,0,1,0,0,0,1-Discrete)

$$Z = \beta_0 + \beta_1 x$$

$$\overset{\Lambda}{Y} = Sigmoid(Z) = \frac{1}{1 + e^{-Z}}$$

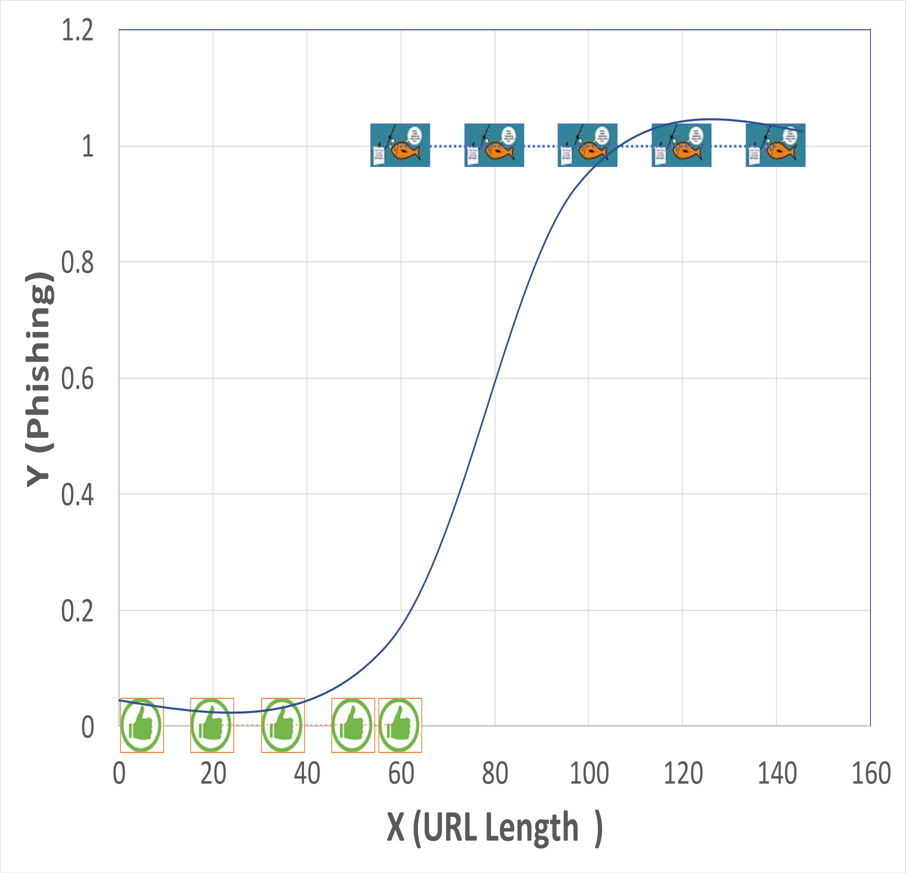
Predict $\overset{\Lambda}{Y}$ Such that for the best fit, difference between $\overset{\Lambda}{Y}$ and Y is minimized.

If Sigmoid(Z)> 0.5, Predict $\overset{\Lambda}{Y} = 1$

If Sigmoid(Z) < 0.5, Predict $\overset{\Lambda}{Y} = 0$

Y = 1 (Phishing)

Y = 0 (Normal)



Algorithm

1. Initialize β_0, β_1

<u>Loop over some iterations or until min Error or (Cost) {</u>

2. Compute $Z = \beta_0 + \beta_1 x$, $\overset{\Lambda}{Y} = Sigmoid(Z) = \dfrac{1}{1+\mathrm{e}^{-Z}}$

3. Calculate Error

$$Error = \sum \left[y \log(Sigmoid(z)) + (1-y) * \log(1 - Sigmoid(z)) \right]$$

4. Minimize Error or (Cost) on β_0, β_1
   - Gradient Descent

$$\beta_0 = \beta_0 - \alpha \frac{\partial(Error)}{\partial \beta_0} \beta_1 = \beta_1 - \alpha \frac{\partial(Error)}{\partial \beta_1}$$

5. Repeat step 2

Source: Andrew Ng

Interpreting Classification Model Output

- Confusion Matrix (or Classification matrix or Error Matrix)

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

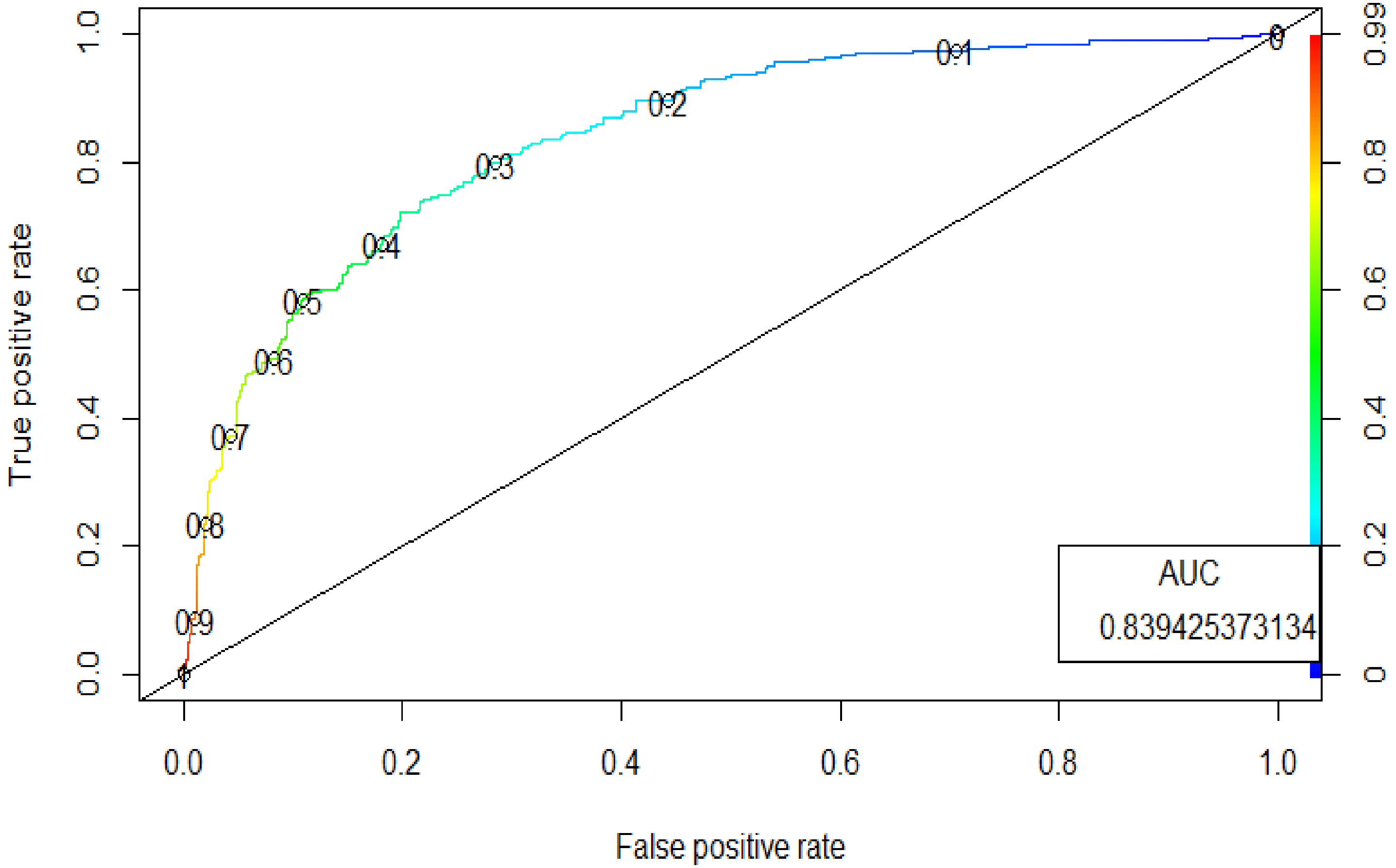
Overall Accuracy = (TN+TP)/(Total Observations)
Overall Error Rate = (FN + FP)/(Total Observations)

True Positive Rate (TPR) (Sensitivity, **Recall**) = TP/(FN+TP), **Precision** = TP/(TP+FP)

False Negative Rate (FNR) = FN/(FN+TP) = 1-TPR
True Negative Rate (TNR) (Specificity) = TN/(TN+FP)
False Positive Rate (FPR) = FP/(TN+FP) = 1-TNR

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

ROCR curves


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
0
0.2
0.4
0.6
0.8
0.99
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
AUC
0.839425373134

Predictive Analytics with Python (Classification)

- Please go to the link below
  - Tinyurl.com/ece-bdata-python/
- Download the folder
  - SML-2
- Open Jupyter notebooks related to Classification using Anaconda

Thank You ☺

Any questions?