# Classification

CS3300 Data Science

RJ Nowling

# Readings

- Sections 9.6-9.7, 7.4.1-7.4.3

# Common Forms of Machine Learning

- Supervised Learning
  - Regression – predicting a continuous output
  - Classification – predicting a categorical output
- Unsupervised Learning
  - Clustering – grouping similar records

# Classification Problems

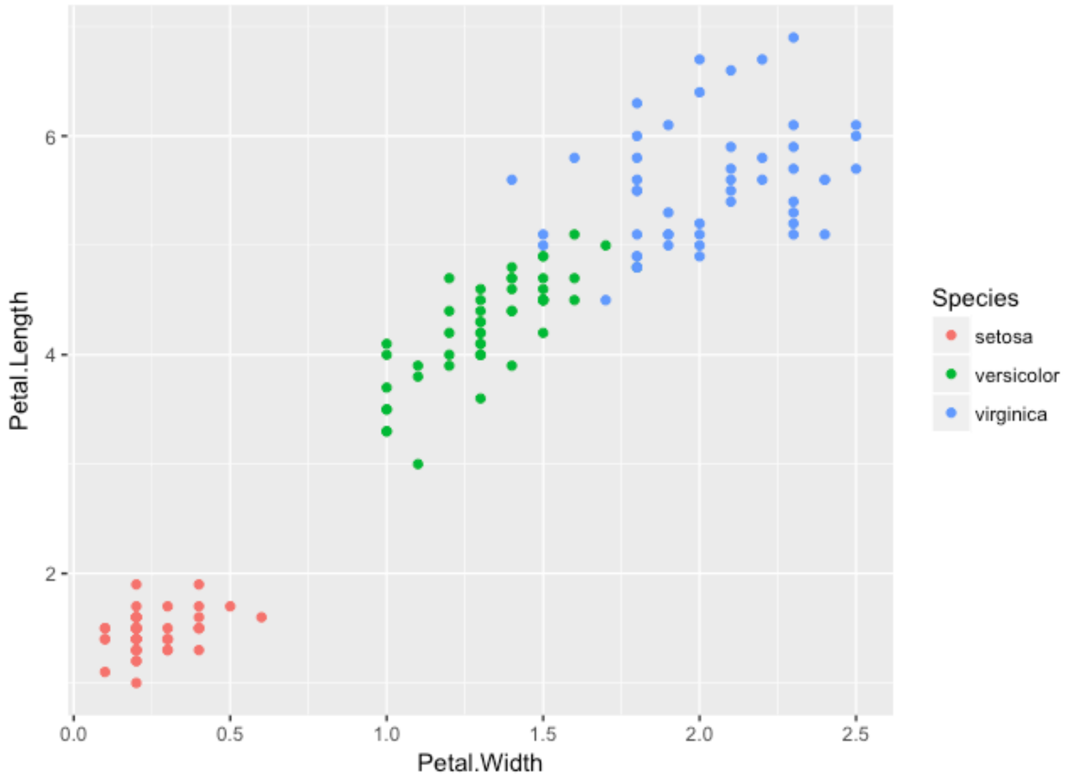
- Is the animal in the picture a cat or a dog?
- Is the customer likely to default on their credit card?
- What character is in the image?

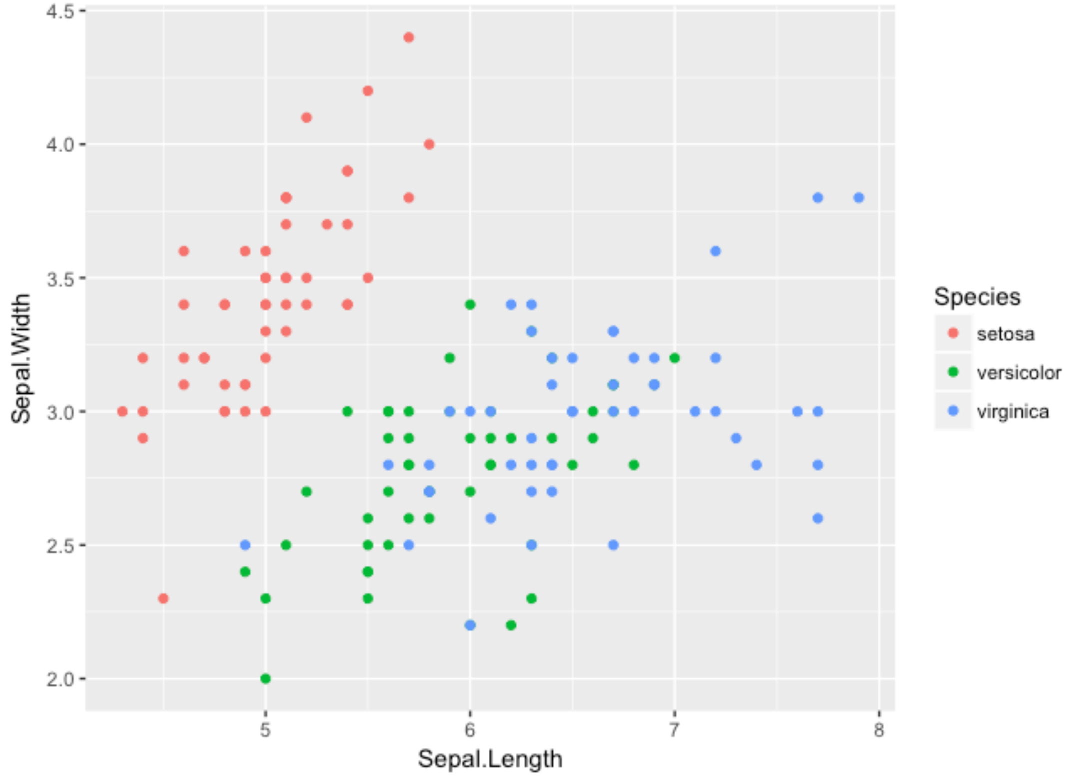
# Iris Data Set

- Trying to classify irises as one of three species
- Response: species
- Features (predictors) are:
  - Petal width
  - Petal height
  - Sepal width
  - Sepal height

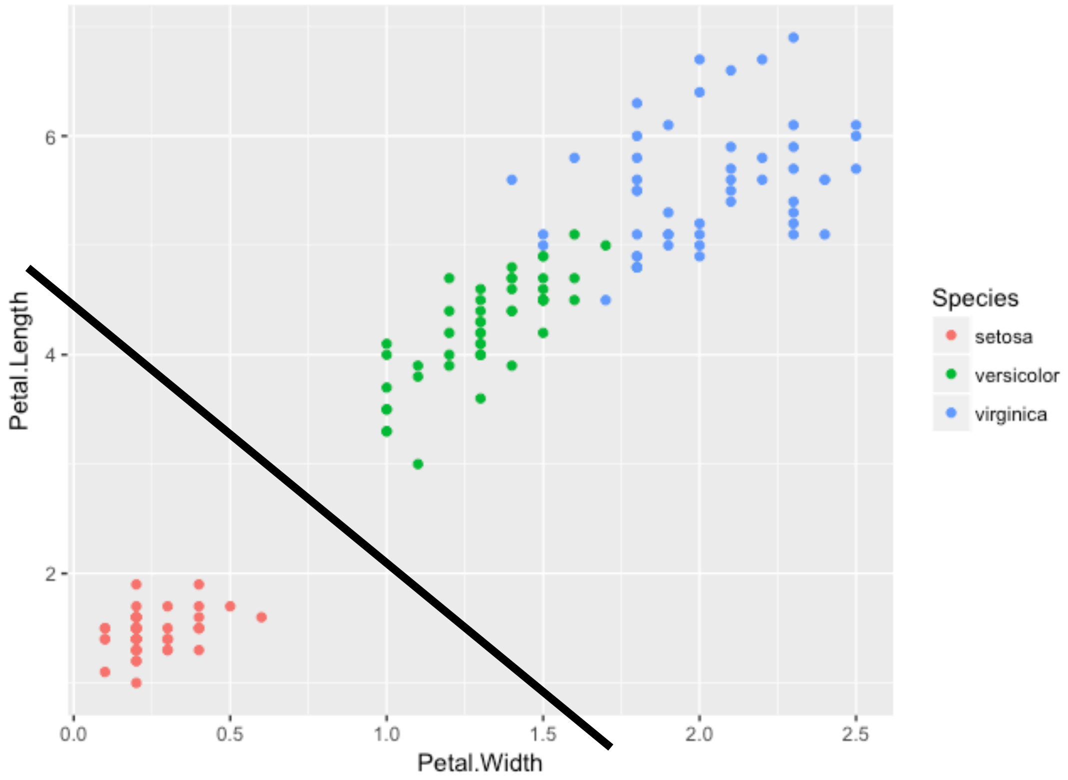
https://www.americanmeadows.com/flower-bulbs/iris-flower-bulbs/dutch-iris-bulbs-sapphire-beauty

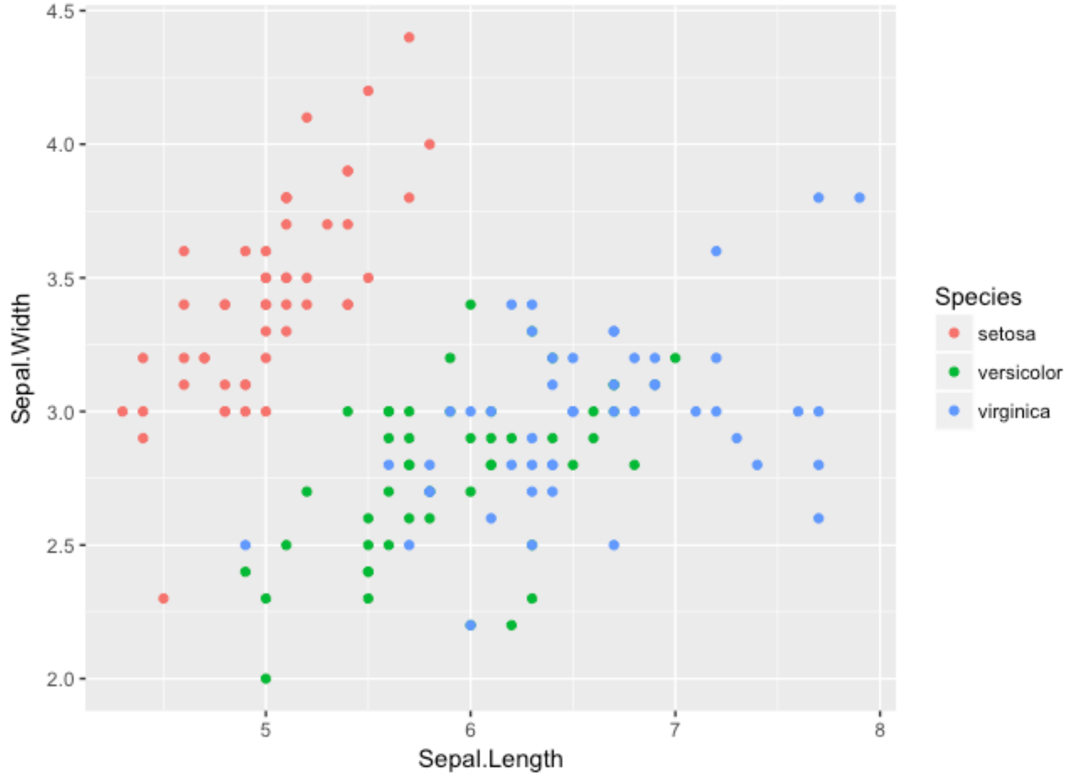
# Iris Data Set



# Iris Data Set



# Logistic Regression

- Despite its name, Logistic Regression is a method for classification, not regression
- Part of a larger class of models called Generalized Linear Models
  - Linear Regression
  - Logistic Regression

# Logistic Regression

- Logistic Regression assumes that there are two outputs (e.g., positive and negative)
- Logistic Regression actually outputs a probability that the data point is in the positive class:

  $$P(y = 1)$$
- We threshold this probability. If $P(y=1) >= 0.5$, we predict $y=1$. Otherwise, we predict $y=0$.

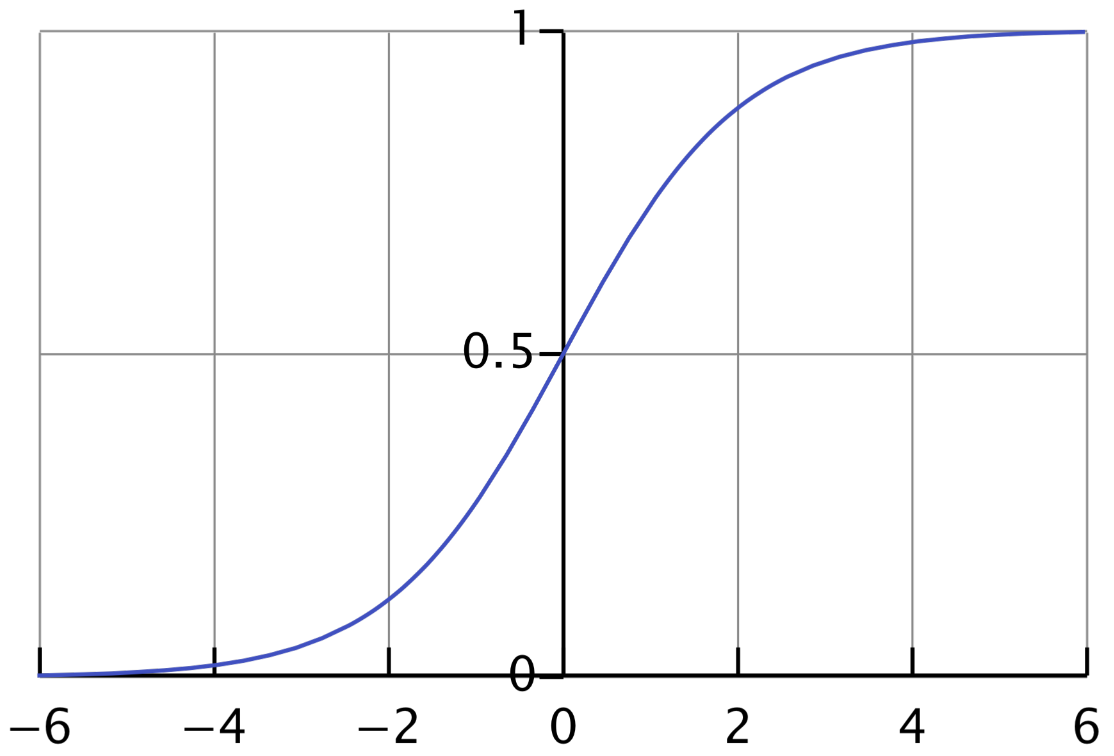
# Logistic Regression

$$P(y = 1) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)$$

where P is the predicted probability, σ is the sigmoid function, $p$ is the number of features, $x_i$ are the features, and $\beta_i$ are the feature weights.

# Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



# Likelihood Function

$$L\left(\beta_0, \beta_1, \dots, \beta_p \middle| y, x_1, x_2, \dots, x_p\right) = \prod_{i=1}^{n} p(y_i = 1)^{y_i} (1 - p(y = 1))^{1-y_i}$$

# Logistic Regression in Scikit Learn

- Scikit-Learn contains two main LR implementations:
  - LogisticRegresion
  - SGDClassifier
- We will use SGDClassifier for the examples in this class

# Fit LR Model

```
features = iris.data[:, 2:],
sgd = SGDClassifier(max_iter=1000, tol=1e-3, loss="log")
sgd.fit(features, setosa_labels)
pred_labels = sgd.predict(feature)
pred_prob = sgd.predict_proba(features)
```

# Fitted Model

The model has two features ($x_1$ petal length and $x_2$ petal width) and an intercept:

$$P(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

Scikit Learn found parameters that optimized the likelihood for the training set:

$$P(y=1) = \frac{1}{1+e^{-(32.0\ -9.4x_1\ -8.0x_2)}}$$

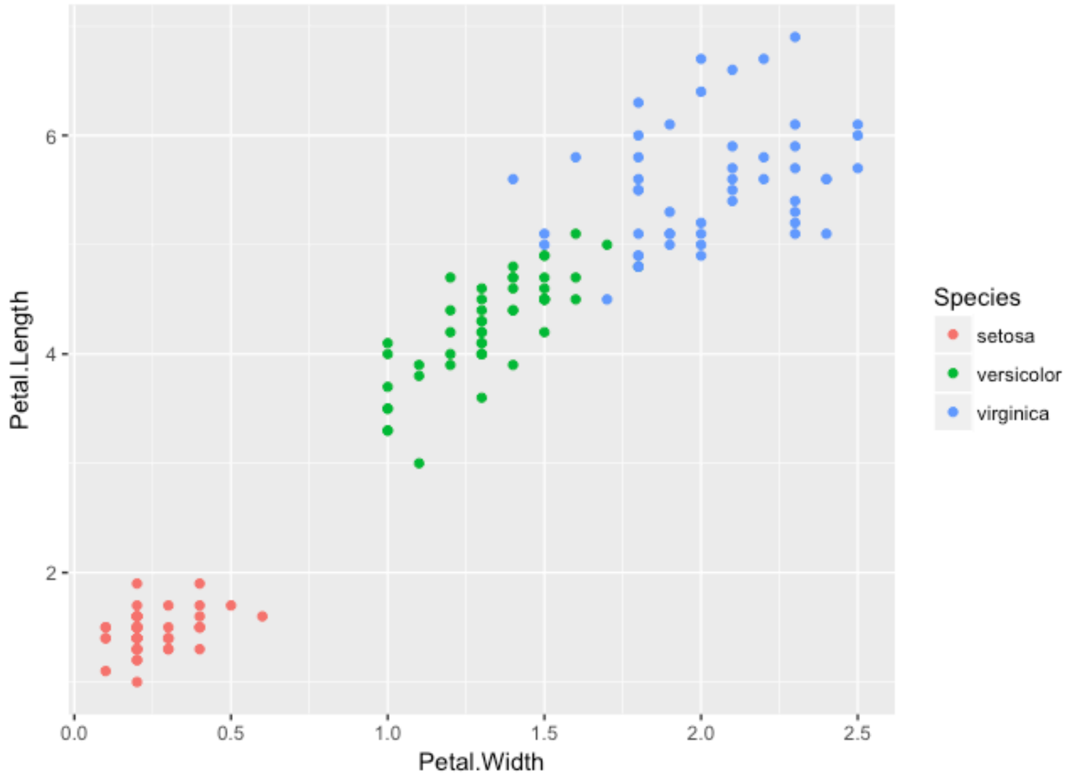
# Model Interpretation

Let's evaluate some points and calculate the resulting probabilities of being setosa:

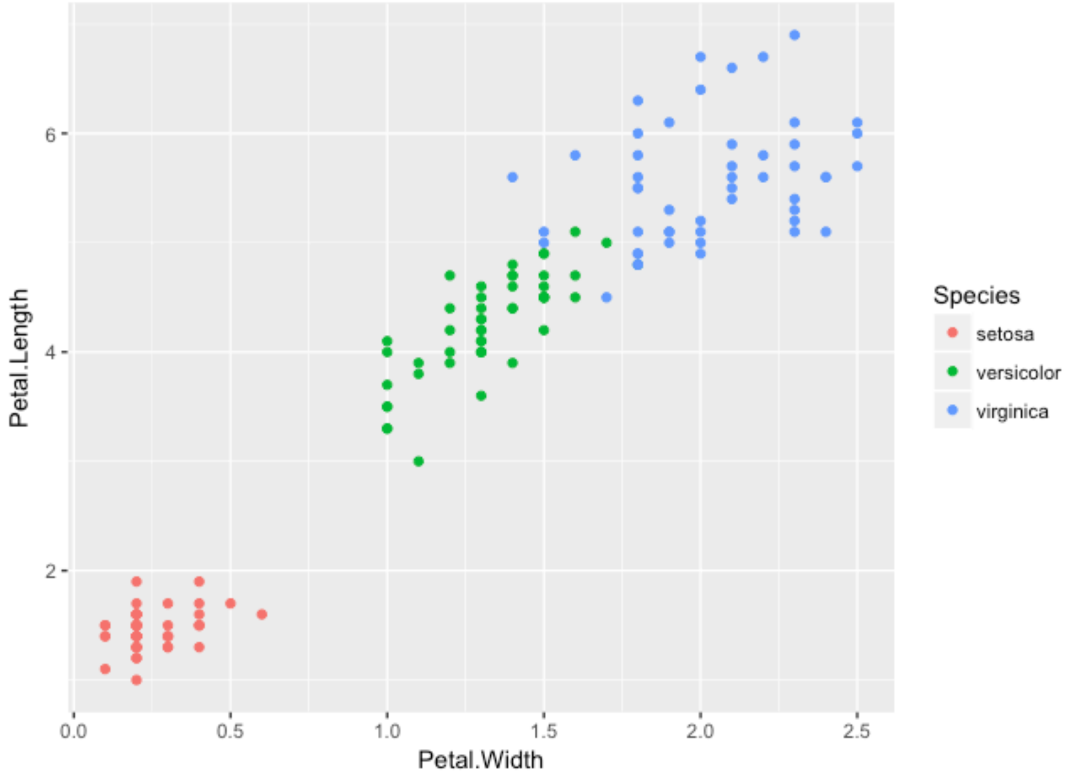
$$P(y = 1) = \frac{1}{1 + e^{-(32.0 - 9.4x_1 - 8.0x_2)}}$$

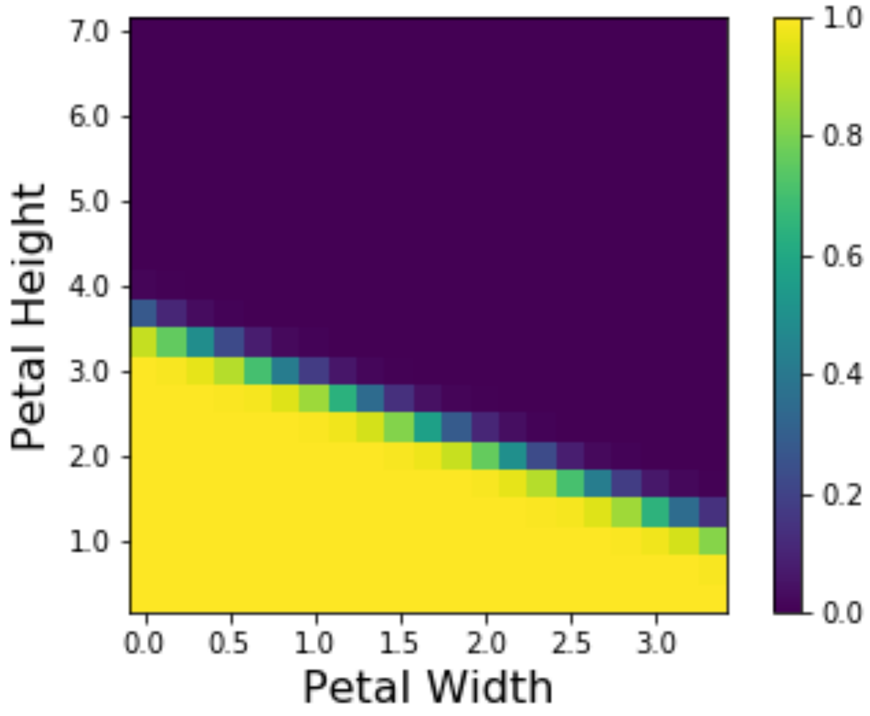
$x_1$ petal length

$x_2$ petal width



# Model Interpretation -- Probabilities



# Model Evaluation

- Experimental Setup
  - Training Set
  - Testing Set
- Evaluation metrics compare:
  - Predicted labels for testing set from model
  - True labels for the testing set

# Definitions

- Positive: sample from the positive class
- Negative:  sample from the negative class
- True Positive: positive sample correctly predicted as positive
- True Negative: negative sample correctly predicted as negative
- False Positive: positive sample incorrectly predicted as negative
- False Negative: positive sample incorrectly predicted as positive

# Metrics for Evaluating Classification Models

- Accuracy – fraction of correct predictions over total samples

$$Accuracy = \frac{tp + tn}{p + n}$$

# Recall and Precision

- Recall – fraction of positive samples that have been correctly predicted
- Precision -- fraction of positive predictions that are correct

$$\text{precision} = \frac{tp}{tp + fp},$$

$$\text{recall} = \frac{tp}{tp + fn},$$

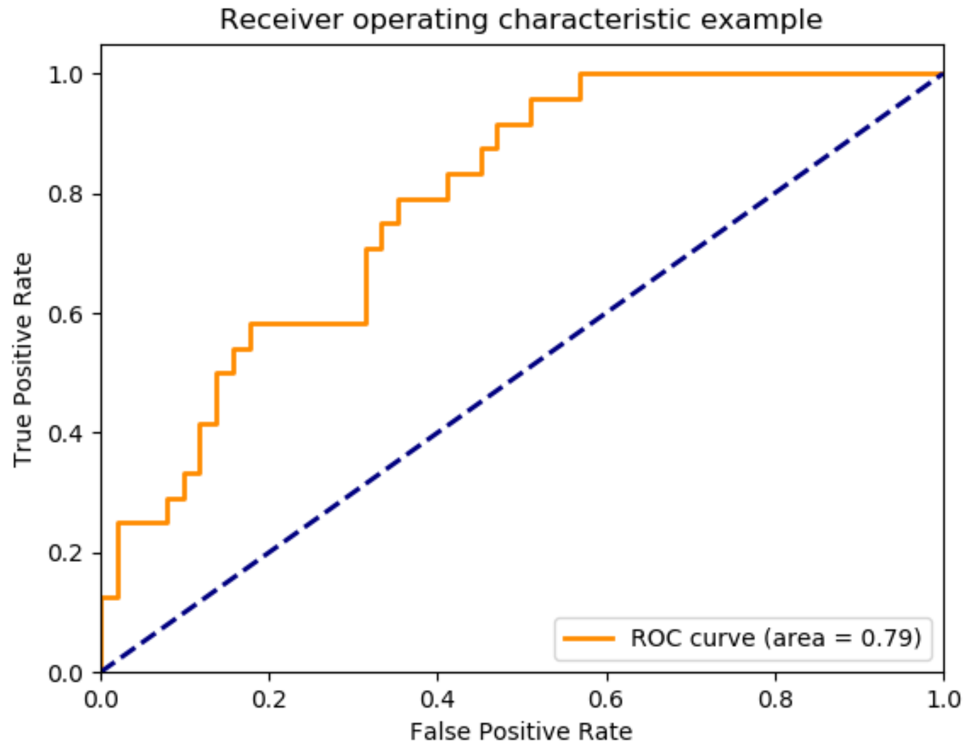
# Confusion Matrix

- Matrix of counts of sample true classes and predicted classes
- Class exercise: Calculate the following metrics using the confusion matrix:
  - Accuracy
  - Precision
  - Recall

| | | **True** | |
|---|---|---|---|
| | | Positive | Negative |
| **Predicted** | Positive | 53 | 90 |
| | Negative | 10 | 47 |

# Receiver-Operator Characteristics (ROC) Curve

- ROC curves tell us evaluate the trade off between true positive rate (sensitivity) and false positive rate (specificity)
- Good classifiers have lines near the upper left
- A random classifier produces a diagonal line
- ROC curves can help us choose a threshold other the default 0.5



# Scikit Learn

Scikit Learn provides a `metrics` module:

```
acc = metrics.accuracy_score(true_labels, pred_labels)
prec = metrics.precision_score(true_labels, pred_labels)
recall = metrics.recall_score(true_labels, pred_labels)
cm = metrics.confusion_matrix(true_labels, pred_labels)

tpr, fpr, thresholds = metrics.roc_curve(true_labels,
pred_prob[:, 1])
```

# Logistic Regression with Multiple Classes

- One vs all scheme
  - Build a model for each class predicting whether a sample is in that class or not
  - Whichever model predicts the highest probability for its class is used to make the categorical prediction
  - Done for us by Scikit-learn when we pass a vector of labels with more than 2 values
- Multinomial
  - Extension of Logistic Regression to multiple classes
  - We won't use this

# Evaluating Multi-Class Predictions

- Accuracy – fraction of correct predictions over total samples

$$Accuracy = \frac{correct\ predictions}{number\ of\ samples}$$

- Confusion matrices can be created for multiple classes
- Precision, Recall, and ROC curves are harder to use for multi-class problems