Cool Kids Coding School

Problem Solving with Python
Lesson 15: Introduction to AI (Logistic Regression)

CKCS
The future is made of code
http://www.coolkidscodingschool.com

> 1.0 What is a regression?

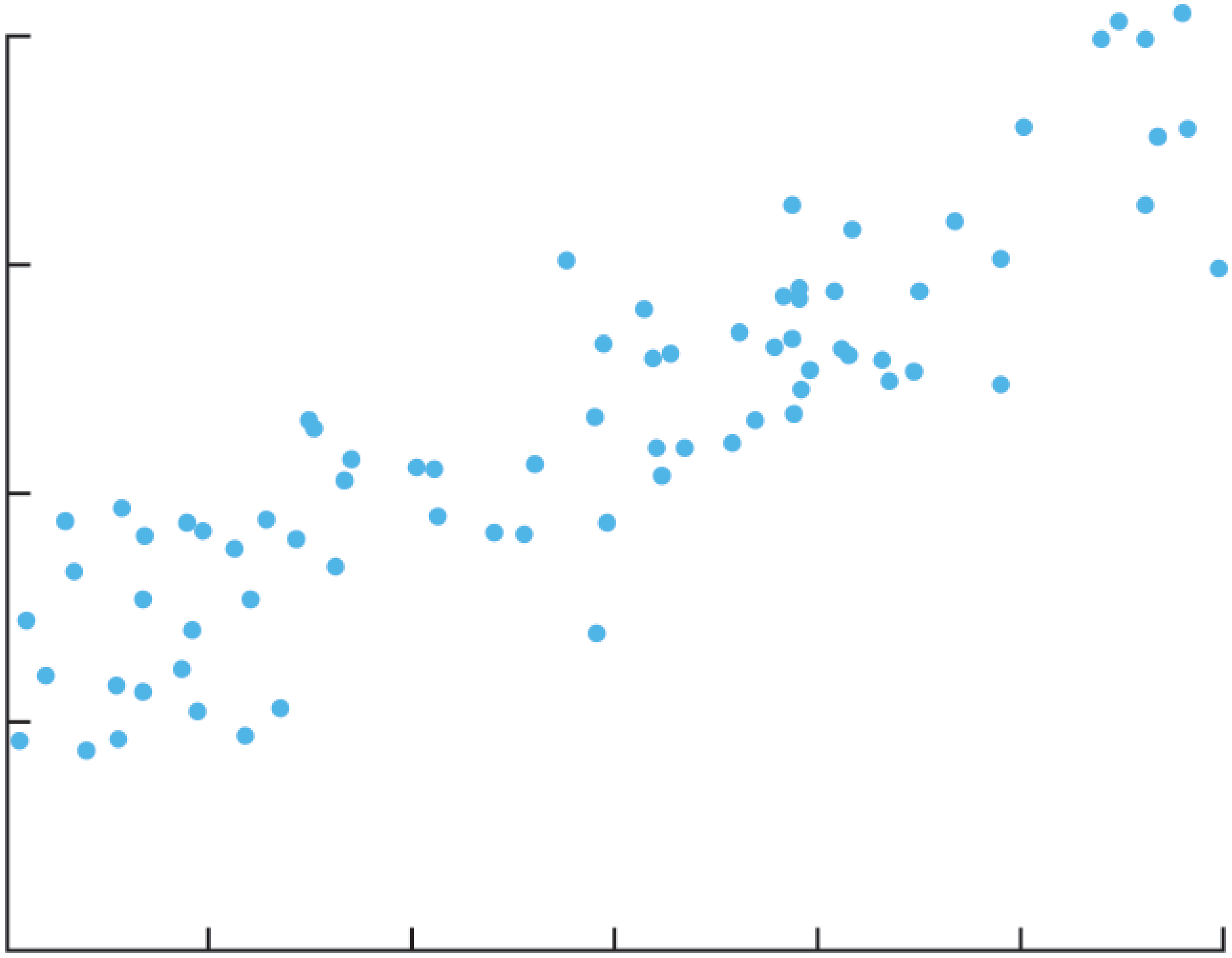
Suppose you're a sales manager trying to predict next month's numbers. You know that dozens, perhaps even hundreds of factors from the weather to a competitor's promotion to the rumor of a new and improved model can impact the number.

Regression analysis is a way of mathematically sorting out which of those variables does indeed have an impact. It answers the questions: Which factors matter most? Which can we ignore? How do those factors interact with each other? And, perhaps most importantly, how certain are we about all of these factors?

In regression analysis, those factors are called independent variables. The factors that you are trying to predict are called dependent variables. In the example above, the dependent variable is monthly sales. And then you have your independent variables — the factors you suspect have an impact on your dependent variable.

Is There a Relationship Between These Two Variables?

Plotting your data is the first step in figuring that out.



SOURCE HBR.ORG © HBR.ORG

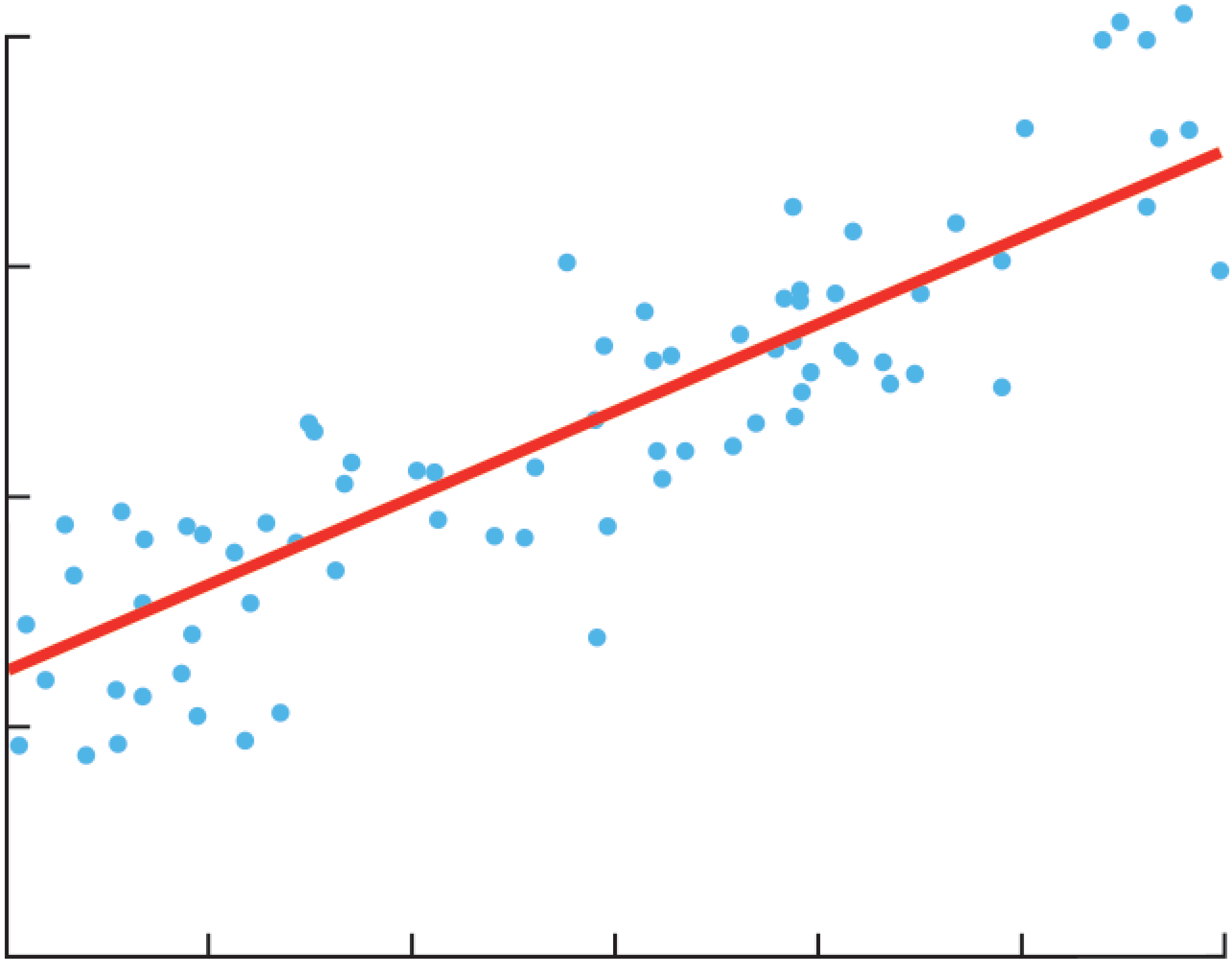
The y-axis is the amount of sales and the x-axis is the total rainfall. Each blue dot represents one month’s data —how much it rained that month and how many sales you made that same month.

Glancing at this data, you probably notice that sales are higher on days when it rains a lot. That’s interesting to know, but by how much? If it rains 3 inches, do you know how much you’ll sell? What about if it rains 4 inches?

Now imagine drawing a line through the chart above, one that runs roughly through the middle of all the data points. This line will help you answer, with some degree of certainty, how much you typically sell when it rains a certain amount.

Building a Regression Model

The line summarizes the relationship between x and y.



SOURCE HBR.ORG

> 2.0 What is a Logistic Regressions

Supervised machine learning algorithms define models that capture relationships among data. Classification is an area of supervised machine learning that tries to predict which class or category some entity belongs to, based on its features. Last week we discussed a classification algorithm that could identify what type of flower we were looking at.

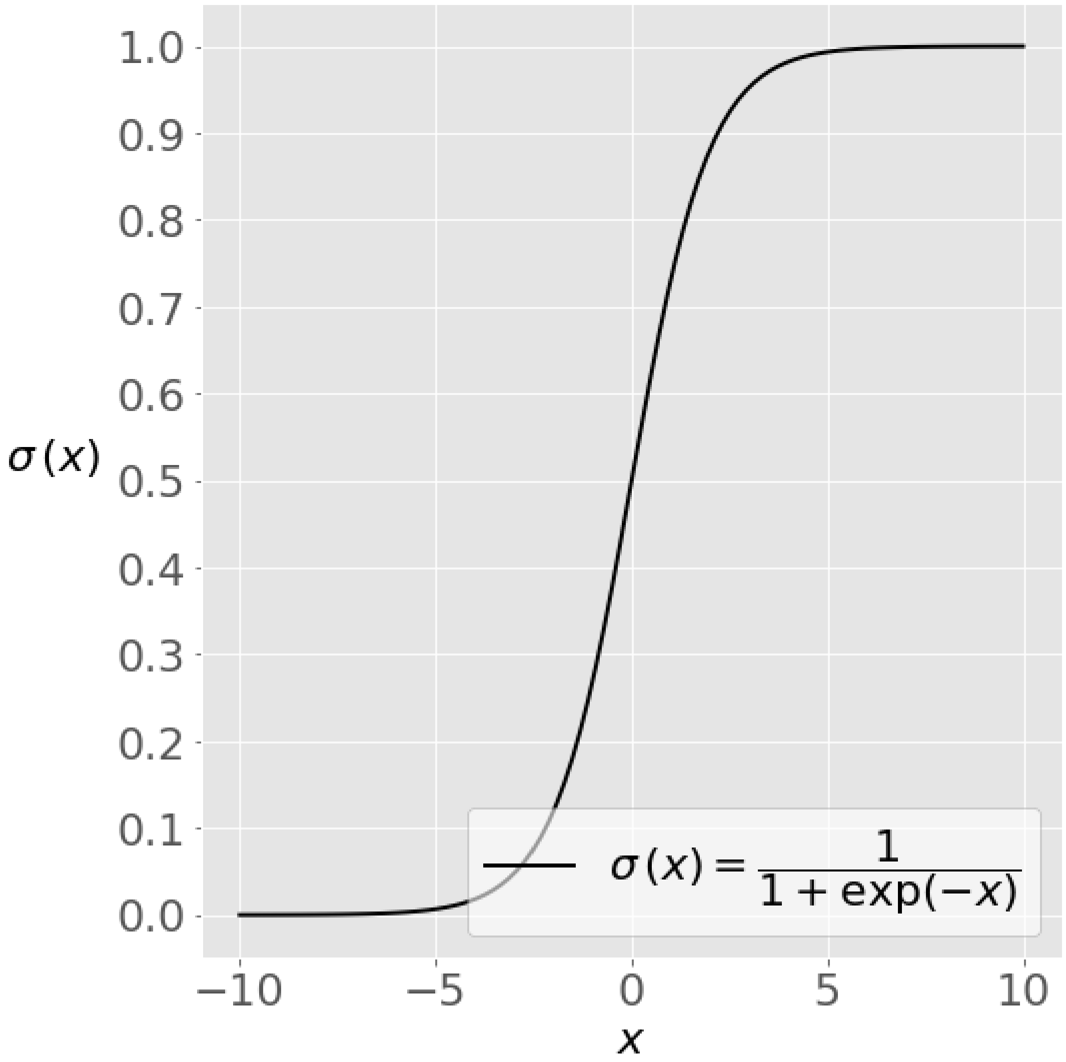
The nature of the dependent variables differentiates regression and classification problems. Linear regression problems have continuous and usually unbounded outputs. An example is when you're estimating the salary as a function of experience and education level. On the other hand, classification problems have discrete and finite outputs called classes or categories. For example, predicting if an employee is going to be promoted or not (true or false) is a classification problem.

Logistic regression is a type of classification algorithm that we are going to learn about today.

> 3.0 Math Background

A logistic regression works because of *sigmoid* functions.

This image shows a sigmoid function (or S curve).



The sigmoid function has values very close to either 0 or 1 across most of its domain. This fact makes it suitable for application in classification methods.

Logistic regression is a linear classifier, so you'll use a linear function $f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$, also called the logit. The variables b_0, b_1, ..., b_r are the estimators of the regression coefficients, which are also called the predicted weights or just coefficients.

The process of calculating the best weights using available observations is called model training or fitting.

4.0 Model Performance

Classification has four possible types of results:

- True negatives: correctly predicted negatives (zeros)
- True positives: correctly predicted positives (ones)
- False negatives: incorrectly predicted negatives (zeros)

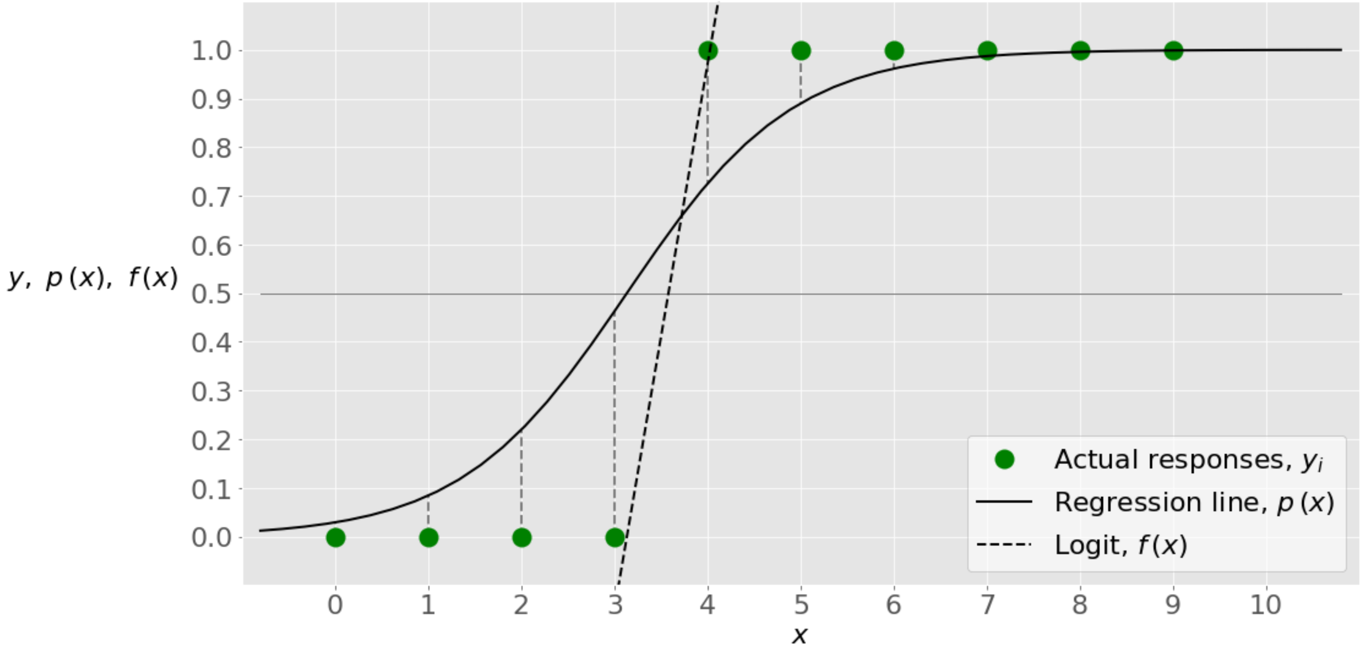
- False positives: incorrectly predicted positives (ones)

You usually evaluate the performance of your classifier by comparing the actual and predicted outputs and counting the correct and incorrect predictions.

5.0 Example

Single-variate logistic regression is the most straightforward case of logistic regression. There is only one independent variable (or feature), which is $\mathbf{x} = x$.

This figure illustrates single-variate logistic regression:



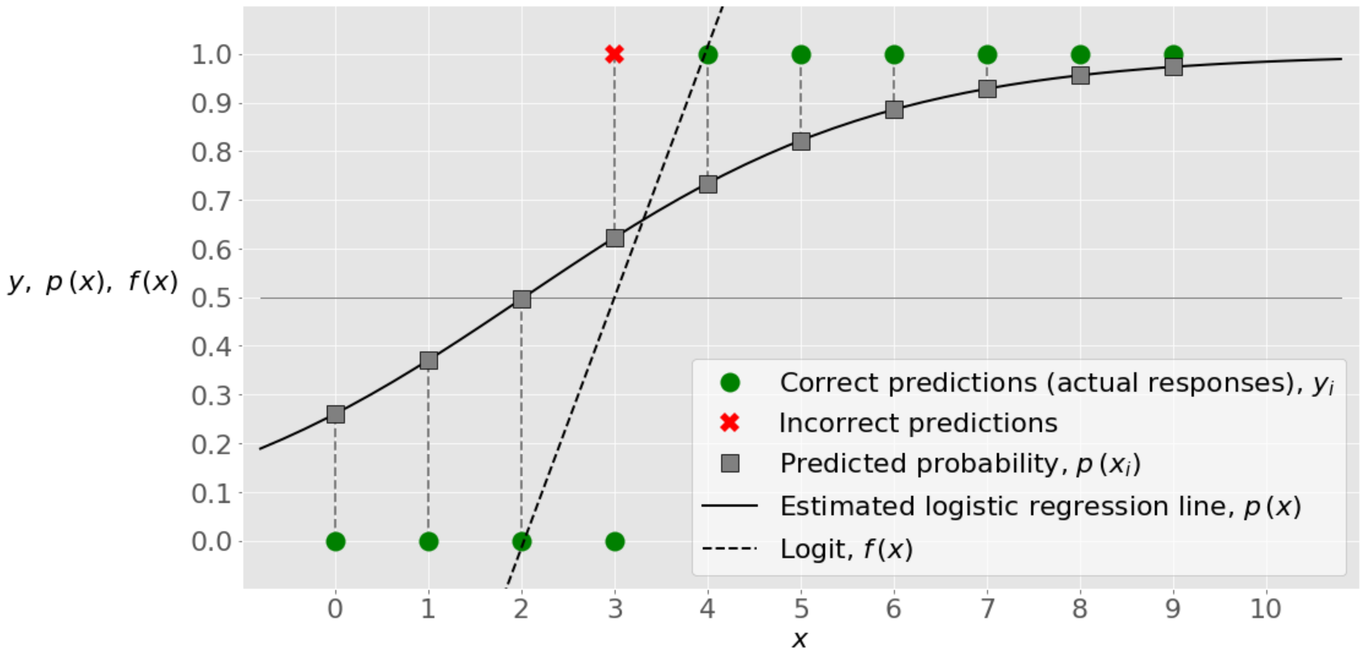
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

x = np.arange(10).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

model = LogisticRegression(solver='liblinear', random_state=0).fit(x, y)
```

Checking the model we can see the classes that it will create as well as the intercept and coefficients

```
model.classes_

model.intercept_

model.coef_
```

We can check the predicted probabilities for the projections created by this logit.

```
model.predict_proba(x)

model.predict(x)
```



The green circles represent the actual responses as well as the correct predictions. The red × shows the incorrect prediction. The full black line is the estimated logistic regression line $p(x)$. The grey squares are the points on this line that correspond to x and the values in the second column of the probability matrix. The black dashed line is the logit $f(x)$.

The value of x slightly above 2 corresponds to the threshold $p(x)$=0.5, which is $f(x)$=0. This value of x is the boundary between the points that are classified as zeros and those predicted as ones.

For example, the first point has input x=0, actual output y=0, probability p=0.26, and a predicted value of 0. The second point has x=1, y=0, p=0.37, and a prediction of 0. Only the fourth point has the actual output y=0 and the probability higher than 0.5 (at p=0.62), so it's wrongly classified as 1. All other values are predicted correctly.

When you have nine out of ten observations classified correctly, the accuracy of your model is equal to 9/10=0.9, which you can obtain with .score():

```
model.score(x,y)
```

You can get more information on the accuracy of the model with a confusion matrix. In the case of binary classification, the confusion matrix shows the numbers of the following:

- True negatives in the upper-left position
- False negatives in the lower-left position
- False positives in the upper-right position

- True positives in the lower-right position

To create the confusion matrix, you can use confusion_matrix() and provide the actual and predicted outputs as the arguments:

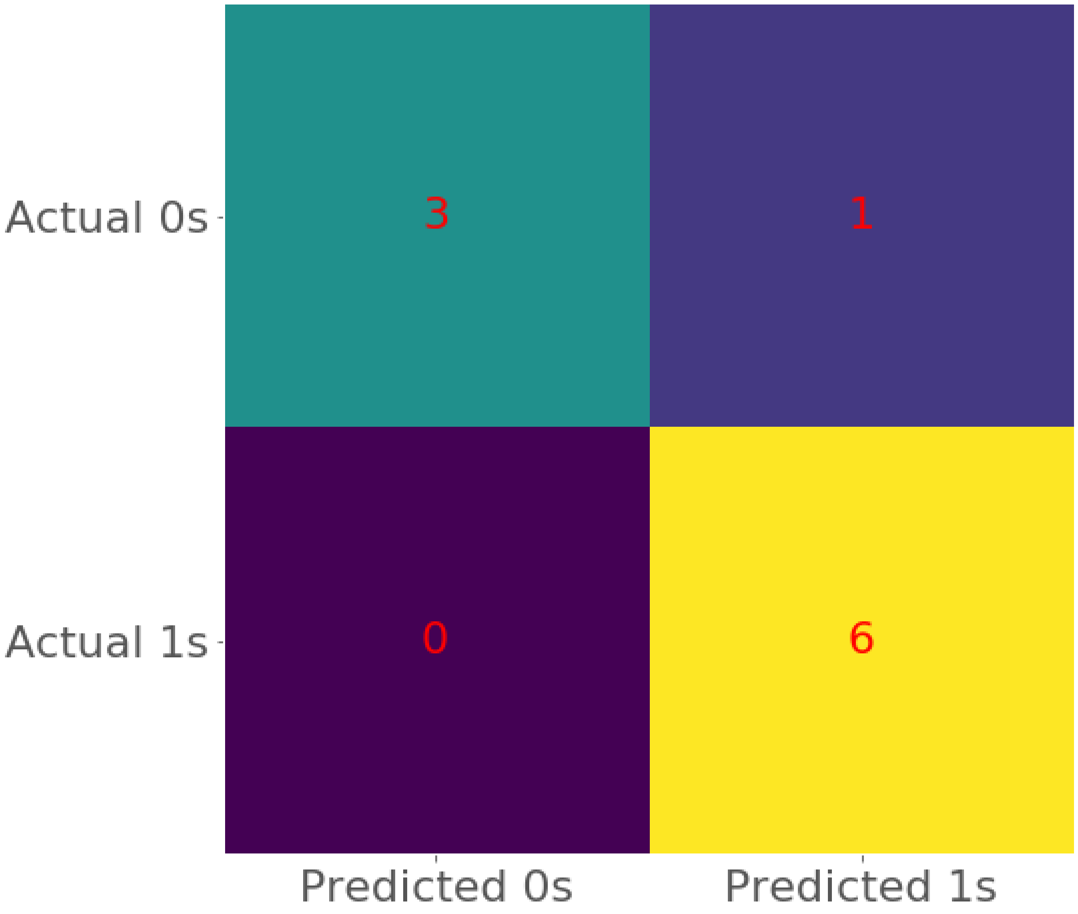
```
confusion_matrix(y, model.predict(x))
```

The obtained matrix shows the following:

- Three true negative predictions: The first three observations are zeros predicted correctly.
- No false negative predictions: These are the ones wrongly predicted as zeros.
- One false positive prediction: The fourth observation is a zero that was wrongly predicted as one.
- Six true positive predictions: The last six observations are ones predicted correctly.

It's often useful to visualize the confusion matrix. You can do that with .imshow() from Matplotlib, which accepts the confusion matrix as the argument:

```
cm = confusion_matrix(y, model.predict(x))

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
plt.show()
```



5.0 Problem

In today's lesson we are going to use data from the Titanic passenger list. Specifically we are going to use data that identified the passenger and whether or not that passenger survived. We are going to use this data to build a model that can predict whether a given passenger profile would survive or not.

Any Questions?

for any questions contact hw_help@coolkidscodingschool.com