Python102

Python for Data Science Bootcamp

(6.1) Machine Learning Basics with Python Part 1

AIAT Academy

Machine Learning Basics

- **Machine Learning Basic Part 1**
  - **Machine Learning with Python using Scikit Learn**
  - **Linear Regression**
  - **Logistic Regression**
- Machine Learning Basic Part 2
  - Support Vector Machine (SVM)
  - K means Clustering
- Machine Learning Basic Part 3
  - Natural Language Processing (NLP)
  - Neural Network and Deep Learning

Machine Learning with Python using Scikit Learn

Machine Learning with Python using Scikit Learn

- **Scikit Learn** will be used for practically learning ML by using Python in this course
  - The most popular machine learning package for python with a lot of algorithms built-in

Scikit Learn Installation

- To install Scikit Learn, just going to your terminal or command prompt and typing

```
conda install scikit-learn
```

or

```
pip install scikit-learn
```

Scikit Learn Import

- Every algorithm is exposed in scikit-learn so you will need to import the model

```
from sklearn.<family> import <Model>
```

For example:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(normalize=True)
```

Scikit Learn Example

AiAT ACADEMY

```
import numpy as np
from sklearn.model_selection import train_test_split
X, y = np.arange(10).reshape((5,2)), range(5)
X
# array([[0, 1],
#        [2, 3],
#        [4, 5],
#        [6, 7],
#        [8, 9]])
y
range(0, 5)
```

Scikit Learn Example

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
X_train
# array([[4, 5],
#        [2, 3],
#        [0, 1]])
y_test
[2, 1, 0]
X_test
# array([[8, 9],
#        [6, 7]])
y_test
[4, 3]
```

Scikit Learn Example

- Now we have split the data so we can train (fit) our model on the training data
- This is done by the model.fit() method:

```
model.fit(Z_train, y_train)
```

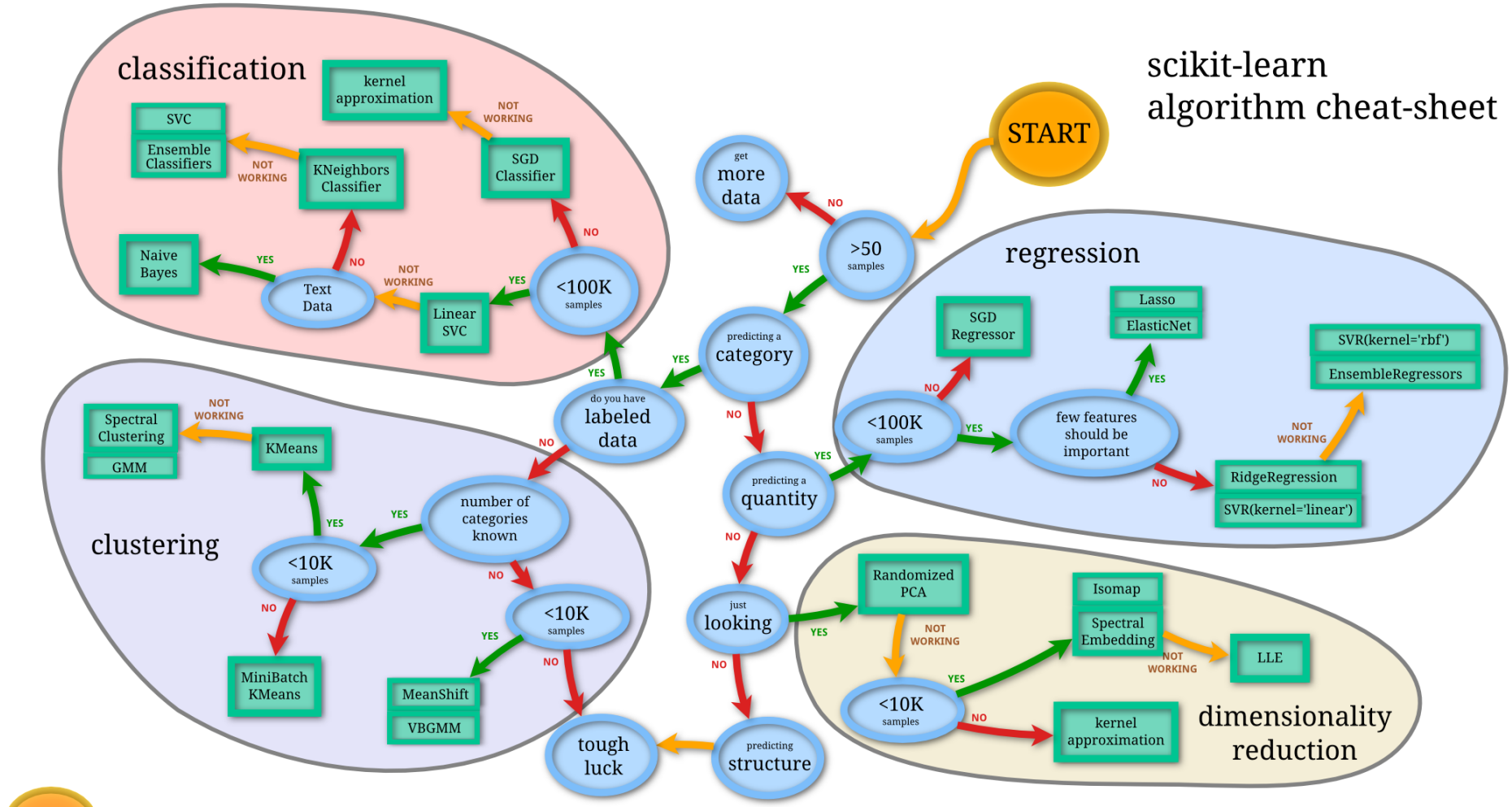
- We get predicted values using the predict method

```
predictions = model.predict(X_test)
```

Scikit Learn Overview

- `model.fit()`: Fit training data **(Supervised/Unsupervised)**
- **Supervised**
  - `model.predict()`: Predict the new data
  - `model.predict_prob()`: Predict the new data and return probability
  - `model.score()`: Get an accuracy of the model.predict() results
- **Unsupervised**
  - `model.transform()`: Transform new data into the new basis (Grouping)

Scikit Learn Overview

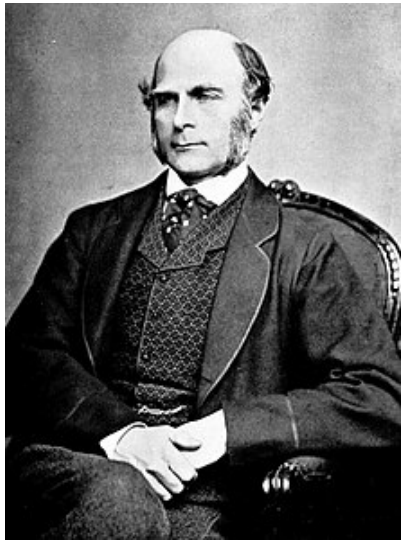


AiAT ACADEMY

Linear Regression

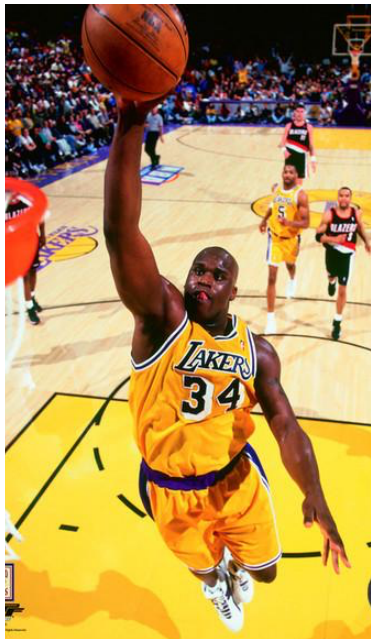
Linear Regression Example

- In 1800s, **Francis Galton** was studying the relationship between parents and their children
- He particularly investigated the relationship between the heights of fathers and their sons
- We discovered that the **son's height tended to be closer to the overall average** height of all people



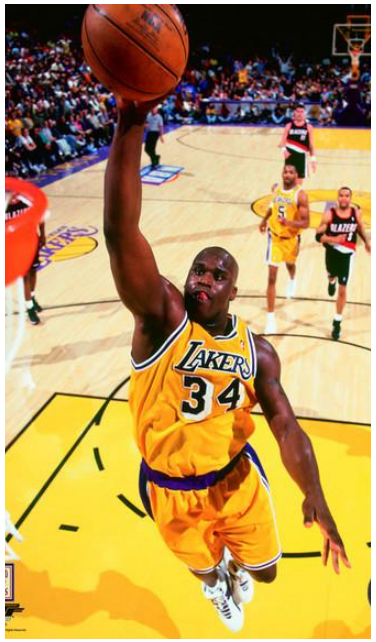
Linear Regression Example

- Let's take Shaquille O'Neal as an example.
  - Shaq is really tall: 6ft 11in
- If he has a son, chances are son will be pretty tall too.
- However, Shaq is very tall so there is also a good chance that **his son will not be tall as Shaq**



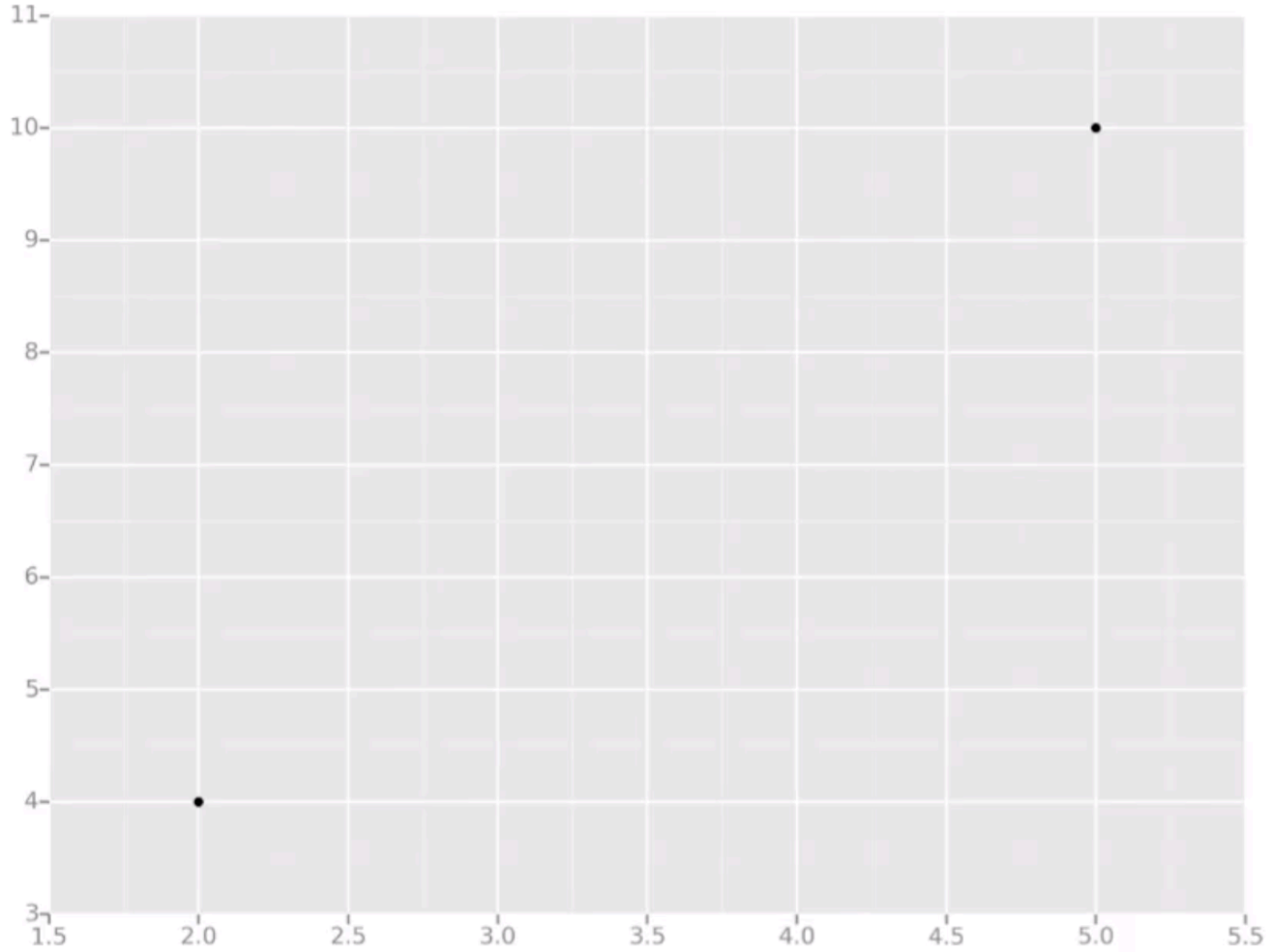
Linear Regression Example

- Turns out this is the case: Shaq's son is tall (6ft 7in)
  - but not as tall as his dad
- Galton called this phenomena **Regression**
  - A father's son's height tends to regress (or move closer) to the mean (average) height



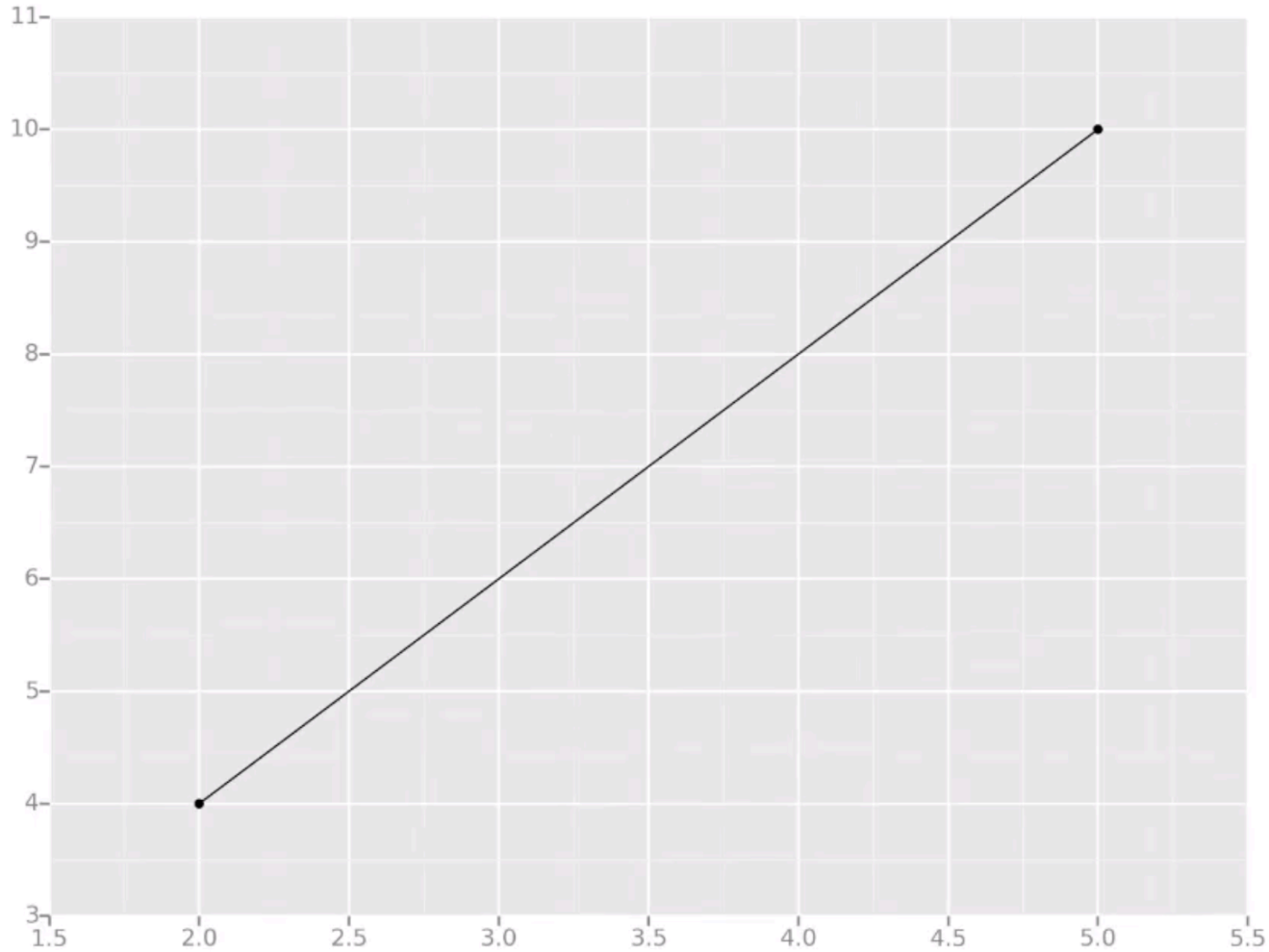
Linear Regression Example

- Let's take a simplest example:
  - Calculate a regression with only 2 data points
  - (2, 4), (5, 10)



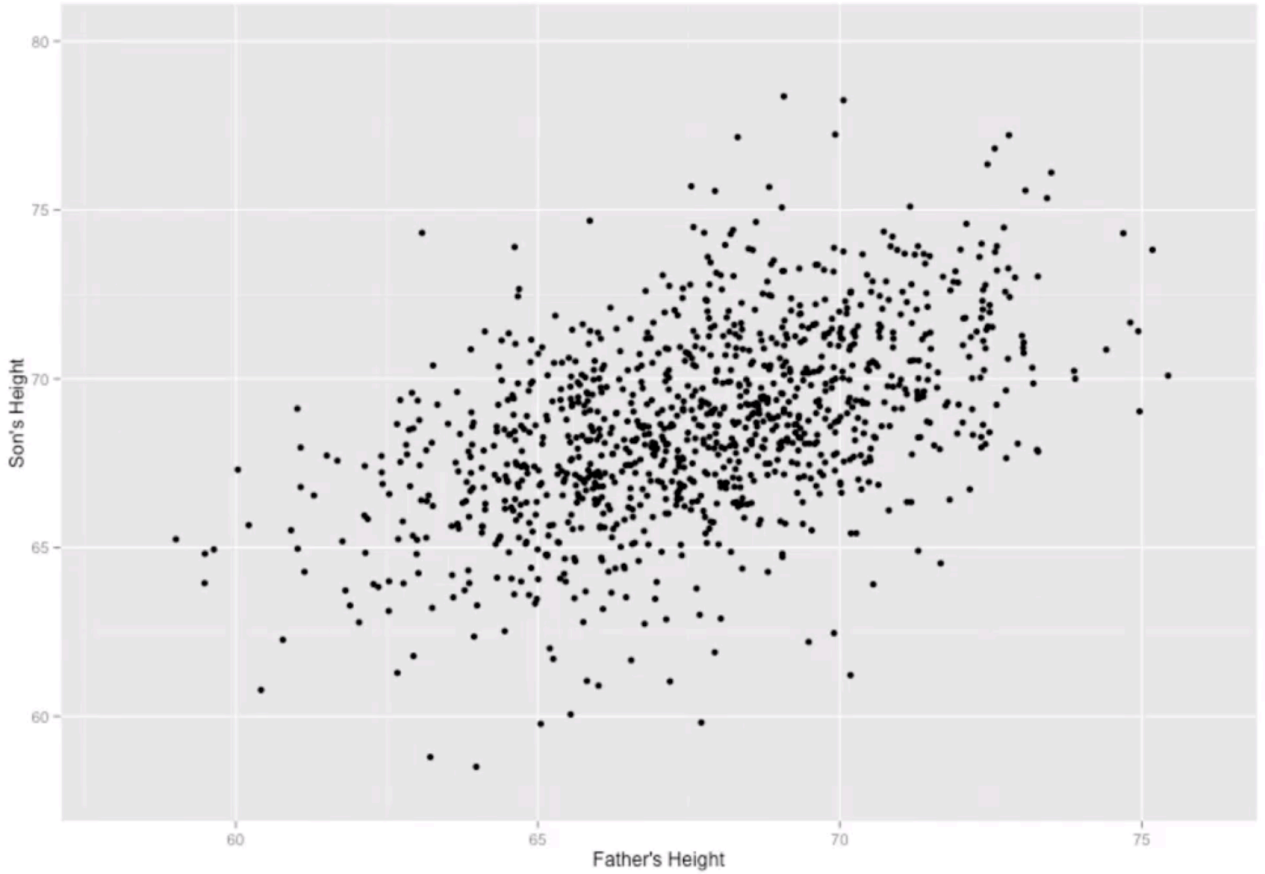
Linear Regression Example

- All we are trying to do when we calculate a regression is
  - Drawing a line that is as close to every dot as possible



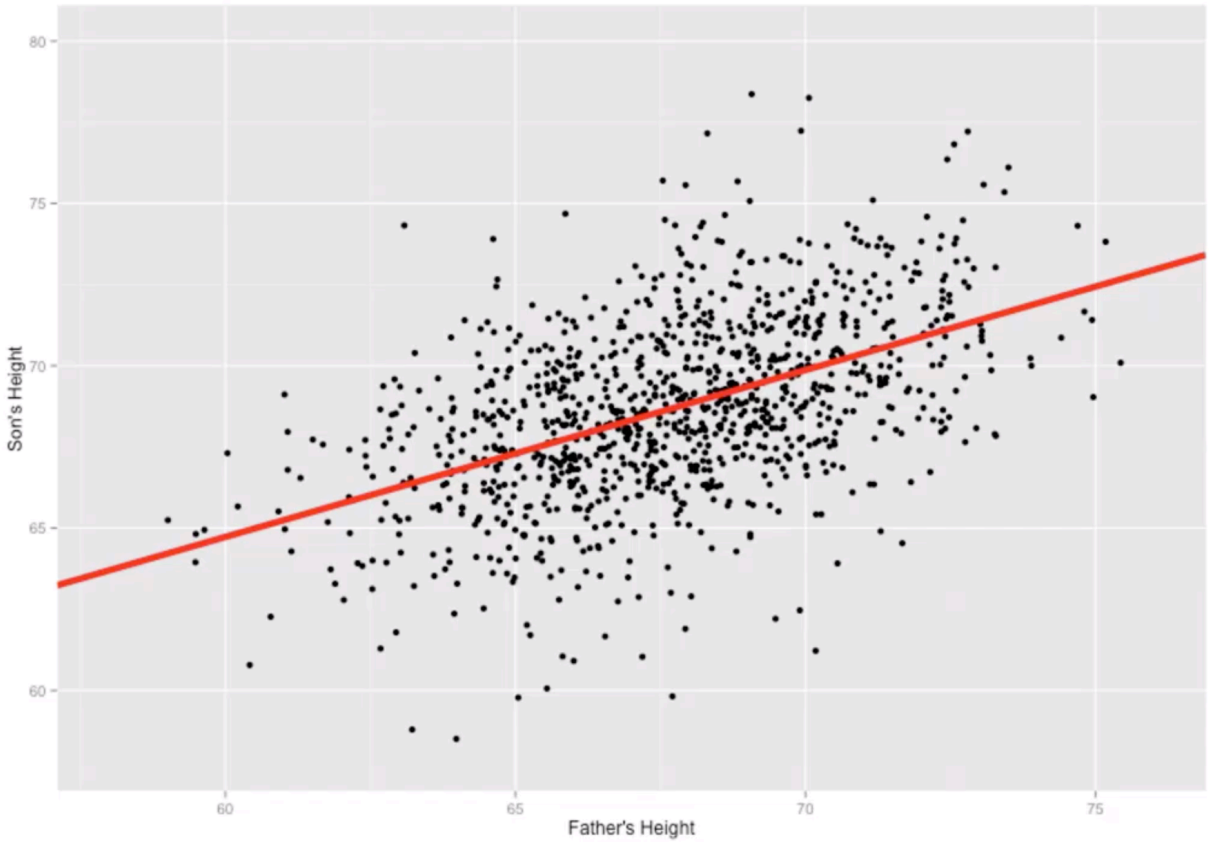
Linear Regression Example

- Now if we apply this concept to a graph with more than just 2 data points
- By doing this, we could take multiple men and their son's height for **telling how tall we expect his son to be.. before he even has a son**



Linear Regression Example

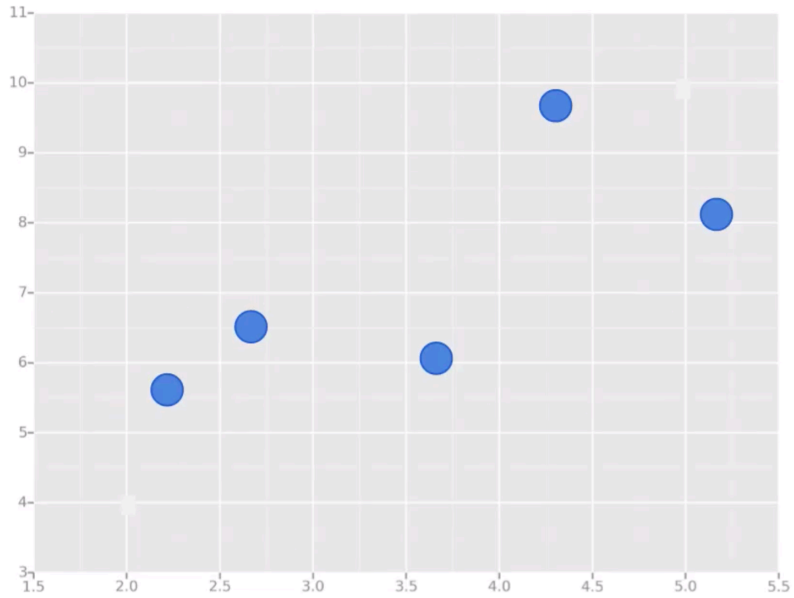
- Our goal with linear regression is to **minimize the vertical distance** between all the data points and our line
- For the **best line**, we are attempting to minimize the distance between **all** the points and their distance to our line
  - Sum of squared errors
  - Sum of absolute errors
  - etc.

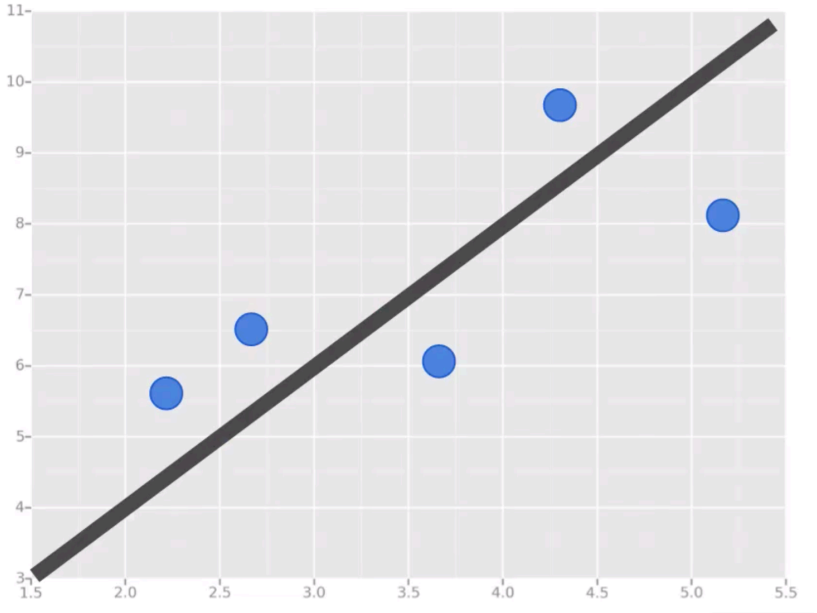


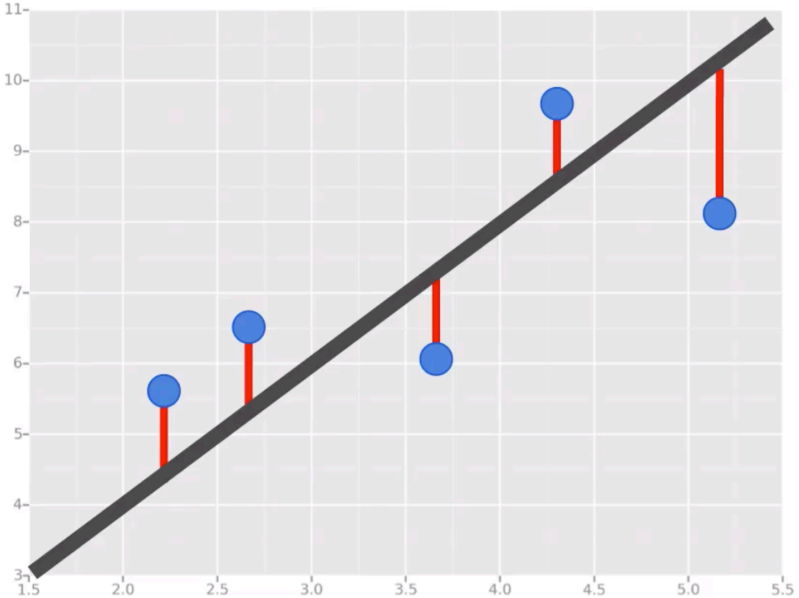
$$SS_{Total} = \sum (y_i - \bar{y})^2$$

- SS_{Total}: Sum Squared Total Error
- $\sum$: Sum Over All The Data Points
- y_i: Each Data Point
- $\bar{y}$: Mean Value
- $(\cdot)^2$: Square The Result

Linear Regression Example







Linear Regression with Python
Colab

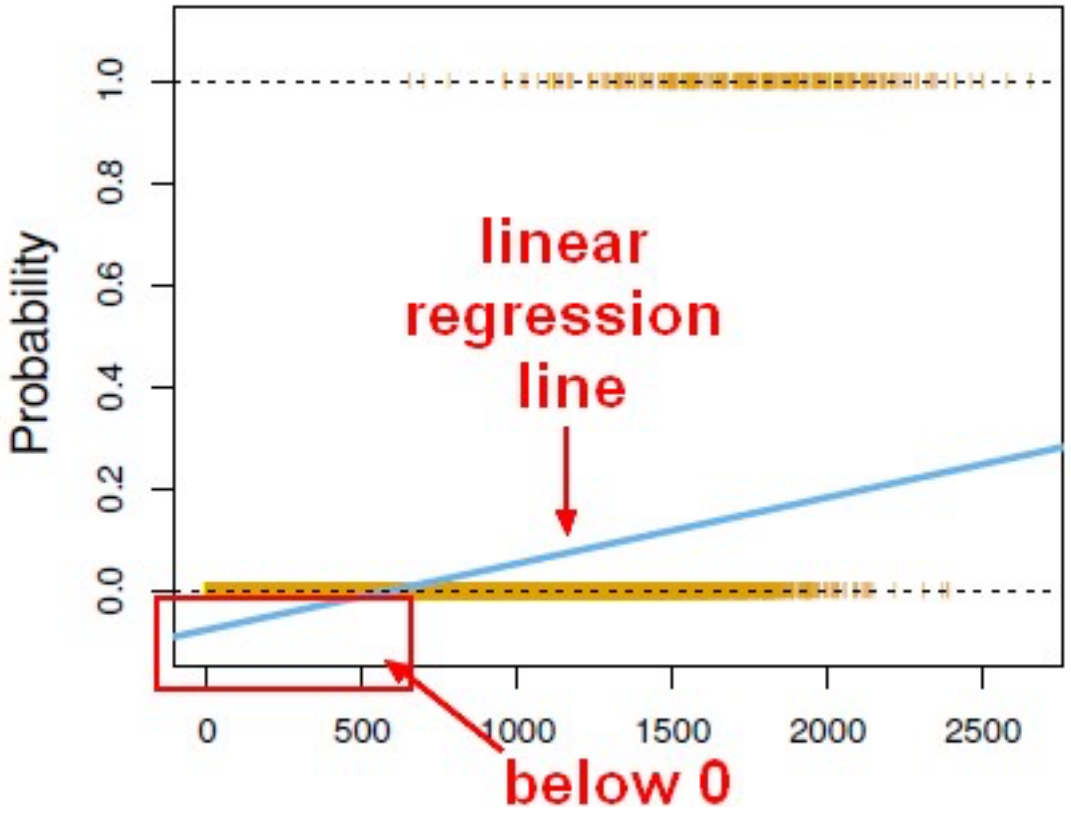
Logistic Regression

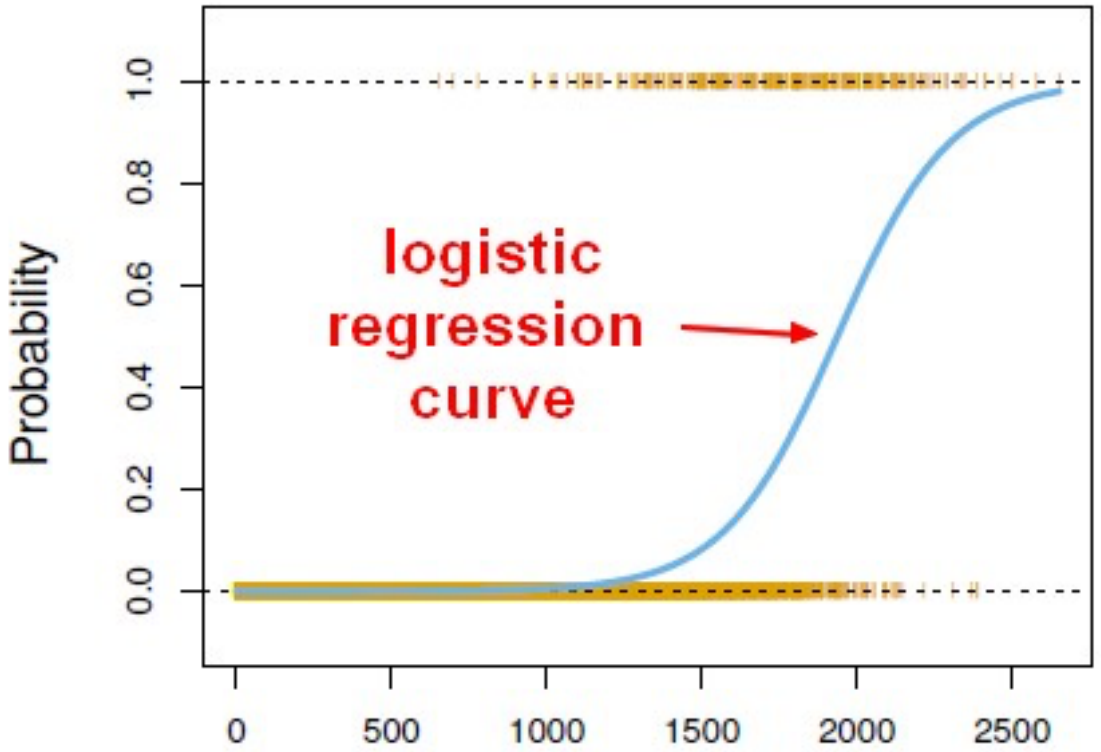
Logistic Regression

- Logistic Regression is normally used as a classification method
- Example of classification problems
  (Binary[0/1] Classification)
  - Spam mail filtering
  - Loan Default (Yes/No)
  - Disease Diagnosis

Linear Regression VS. Logistic Regression

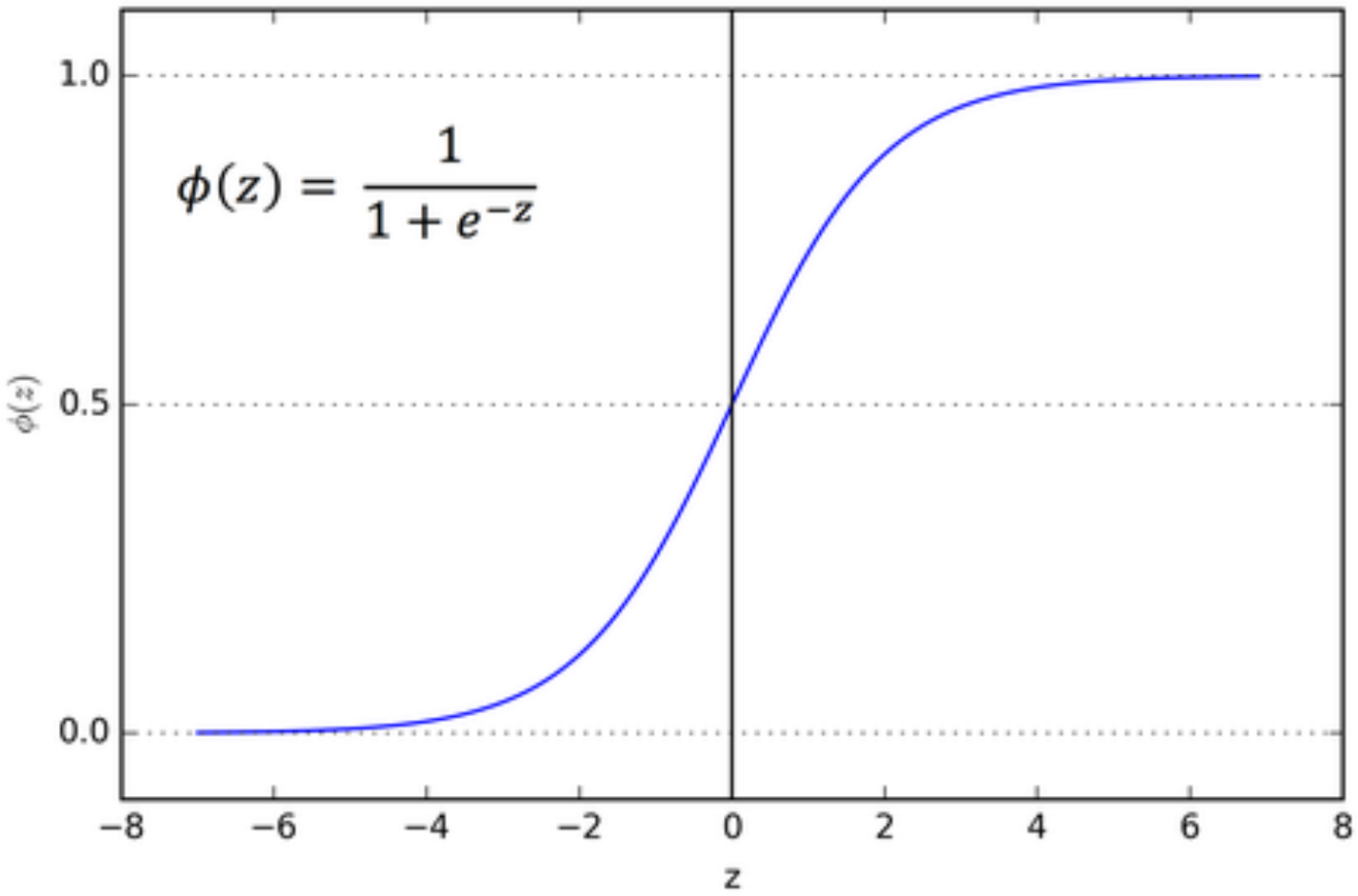
- **Linear Regression** for predicting continuous values (-inf/inf)
- **Logistic Regression** for predicting in binary values (yes/no)





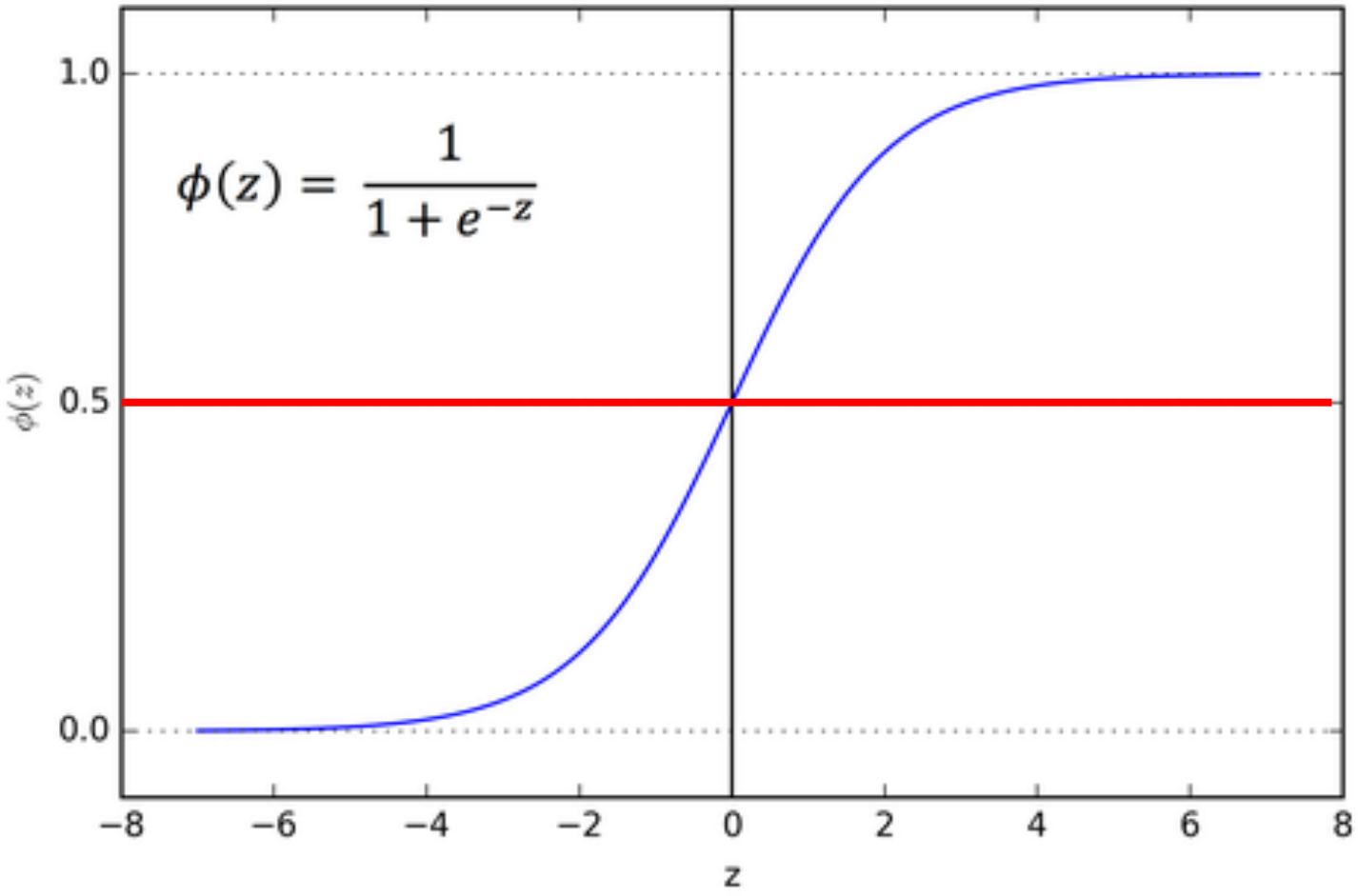
A Key to Logistic Regression

- **Sigmoid Function** (aka Logistic)
  - Function takes in any value and outputs it between 0 and 1



A Key to Logistic Regression

- We can set the cutoff point at 0.5, anything below it results in class 0, anything above is class 1



Model Evaluation (Confusion Matrix)

- After you train a logistic regression model on some training data, you will evaluate your model's performance on some test data
- You can use **confusion matrix** to evaluate classification models
- For example

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Test for presence of disease

NO = Negative test = False = 0

YES = Positive test = True = 1

Basic Terminology

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

Model Evaluation (Confusion Matrix)

- After you train a logistic regression model on some training data, you will evaluate your model's performance on some test data
- You can use confusion matrix to evaluate classification models
- For example

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy

- Overall, how often is it **correct**?
- (TP+TN)/total
- (50+100)/165 = **0.91 (91%)**

Misclassification Rate (Error)

- Overall, how often is it **wrong**?
- (FP+FN)/total
- (10+5)/165 = **0.09 (9%)**

Model Evaluation (Confusion Matrix)

- After you train a logistic regression model on some training data, you will evaluate your model's performance on some test data
- You can use confusion matrix to evaluate classification models
- For example

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

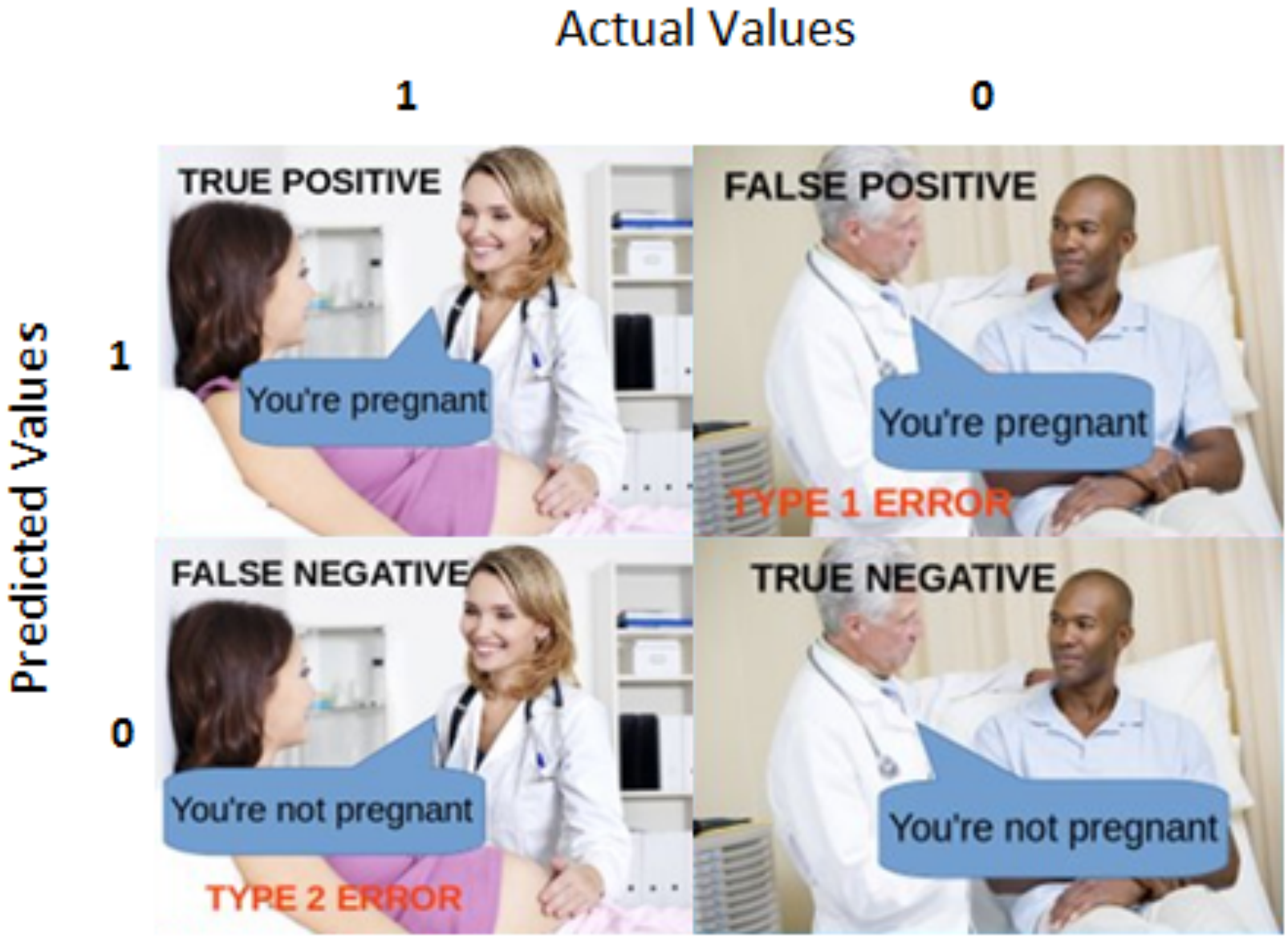
Accuracy

- Overall, how often is it **correct**?
- (TP+TN)/total
- (50+100)/165 = **0.91 (91%)**

Misclassification Rate (Error)

- Overall, how often is it **wrong**?
- (FP+FN)/total
- (10+5)/165 = **0.09 (9%)**

Model Evaluation (Confusion Matrix)



Logistic Regression with Python

Colab