COMP41680

Modelling and Prediction

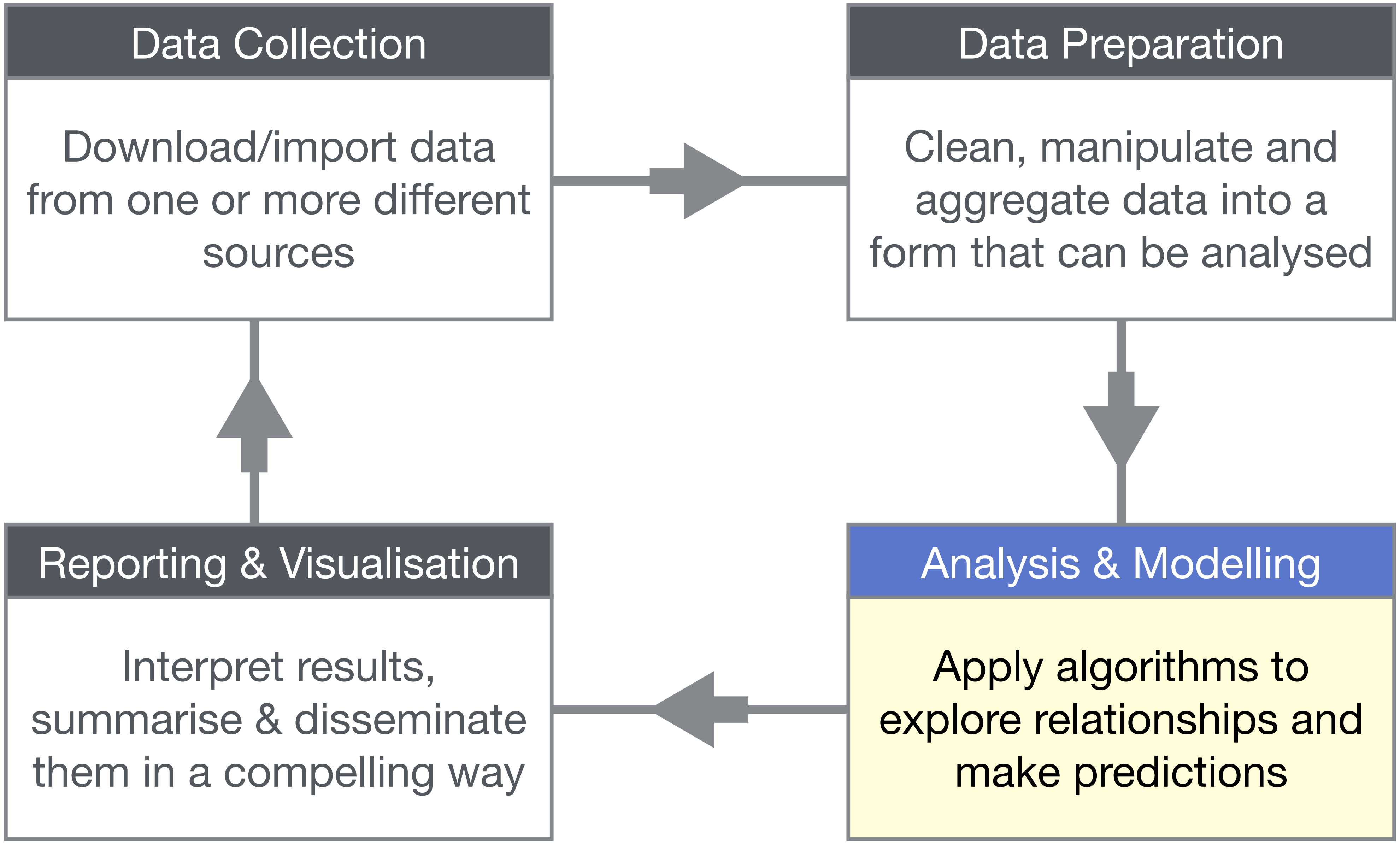
Derek Greene

UCD School of Computer Science

Spring 2023

Reminder: Data Science Pipeline

- Recall the stages of the basic data science pipeline…
- Most complex component relates to data mining and modelling.



Data Collection

Download/import data from one or more different sources

Data Preparation

Clean, manipulate and aggregate data into a form that can be analysed

Analysis & Modelling

Apply algorithms to explore relationships and make predictions

Reporting & Visualisation

Interpret results, summarise & disseminate them in a compelling way

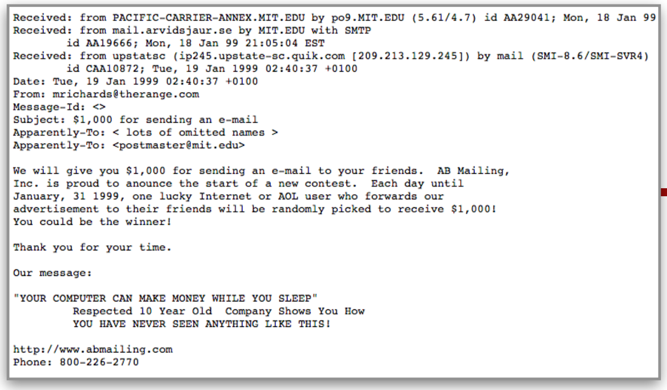
Modelling and Prediction Tasks

- Predictive modelling uses statistics to predict outcomes, based on historic data. Also referred to as supervised machine learning.
- **Examples:**
  - Spam filtering: predict if a new email is spam or non-spam, based on annotated examples of past spam / non-spam.
  - Car insurance: assign risk of accidents to policy holders and potential customers.
  - Healthcare: predict disease which a patient has, based on their symptoms.
  - Algorithmic trading: predictive models can be built for different assets like stocks, futures, currencies, etc, based on historic data and company information.

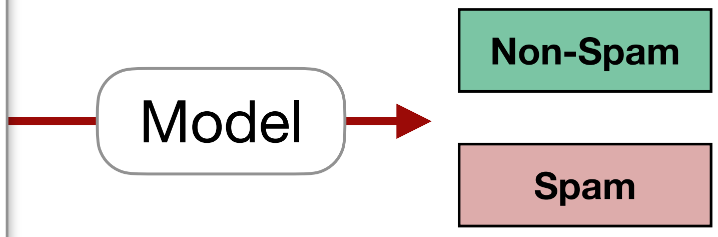
Supervised Learning

- Classification:
  Learn from a labelled training set to make a prediction to assign a new "unseen" example to one of a fixed number of classes.

Unseen
Input
Email



```
Received: from PACIFIC-CARRIER-ANNEX.MIT.EDU by po9.MIT.EDU (5.61/4.7) id AA29041; Mon, 18 Jan 99
Received: from mail.arvidsjaur.se by MIT.EDU with SMTP
        id AA19666; Mon, 18 Jan 99 21:05:04 EST
Received: from upstatsc (ip245.upstate-sc.quik.com [209.213.129.245]) by mail (SMI-8.6/SMI-SVR4)
        id CAA10872; Tue, 19 Jan 1999 02:40:37 +0100
Date: Tue, 19 Jan 1999 02:40:37 +0100
From: mrichards@therange.com
Message-Id: <>
Subject: $1,000 for sending an e-mail
Apparently-To: < lots of omitted names >
Apparently-To: <postmaster@mit.edu>

We will give you $1,000 for sending an e-mail to your friends.  AB Mailing,
Inc. is proud to anounce the start of a new contest.  Each day until
January, 31 1999, one lucky Internet or AOL user who forwards our
advertisement to their friends will be randomly picked to receive $1,000!
You could be the winner!

Thank you for your time.

Our message:

"YOUR COMPUTER CAN MAKE MONEY WHILE YOU SLEEP"
         Respected 10 Year Old  Company Shows You How
         YOU HAVE NEVER SEEN ANYTHING LIKE THIS!

http://www.abmailing.com
Phone: 800-226-2770
```



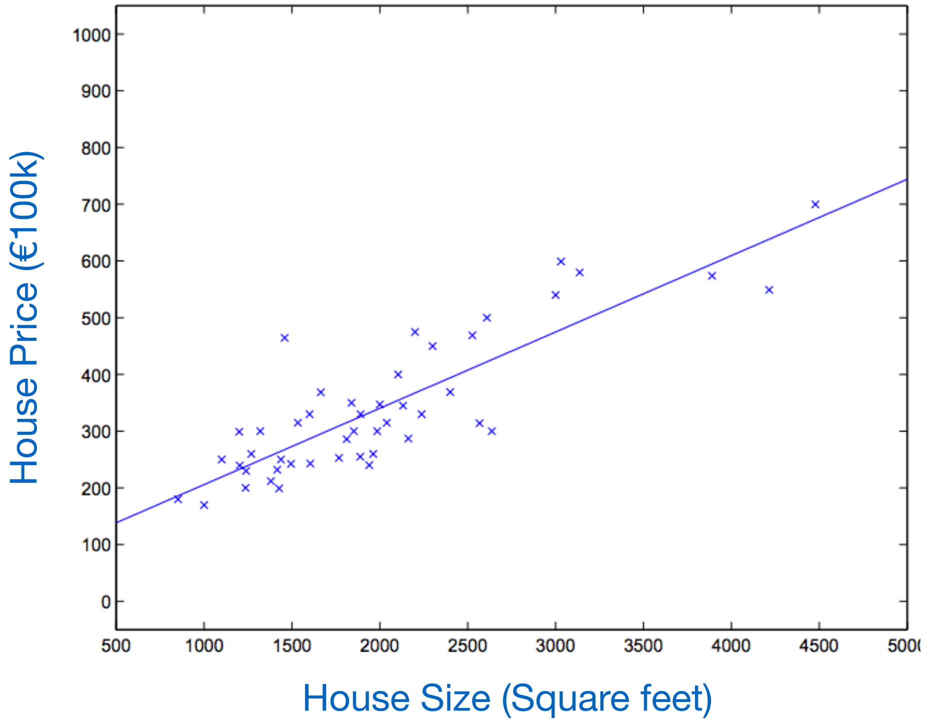
Model

Non-Spam

Spam

2 classes

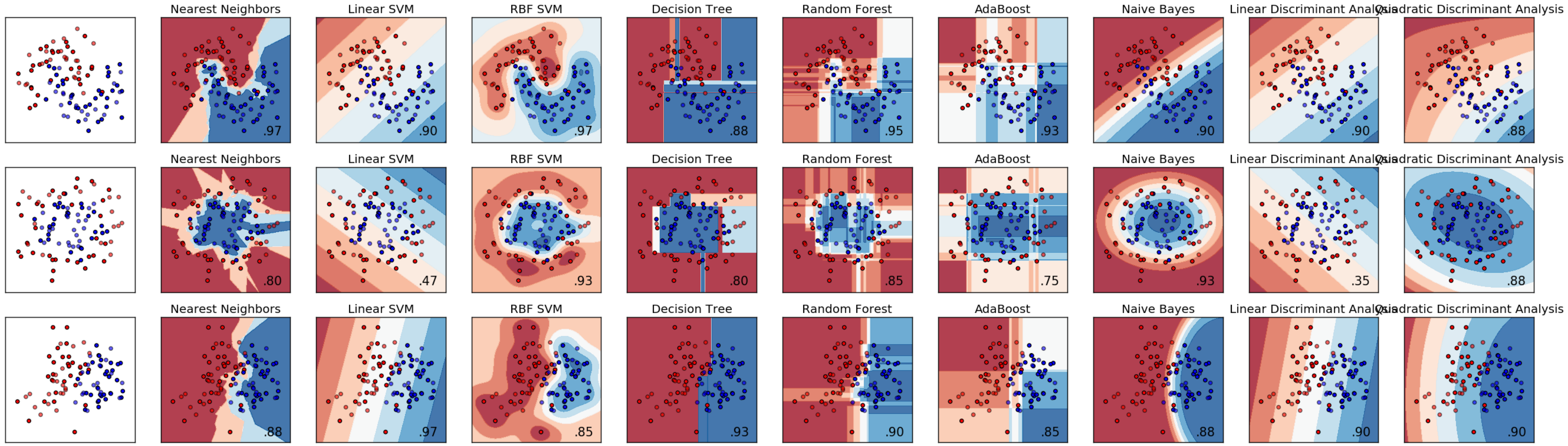
- Regression:
  Learn from an existing training set to decide the value of a continuous output variable (i.e. the output is a number).



Scikit-Learn Package

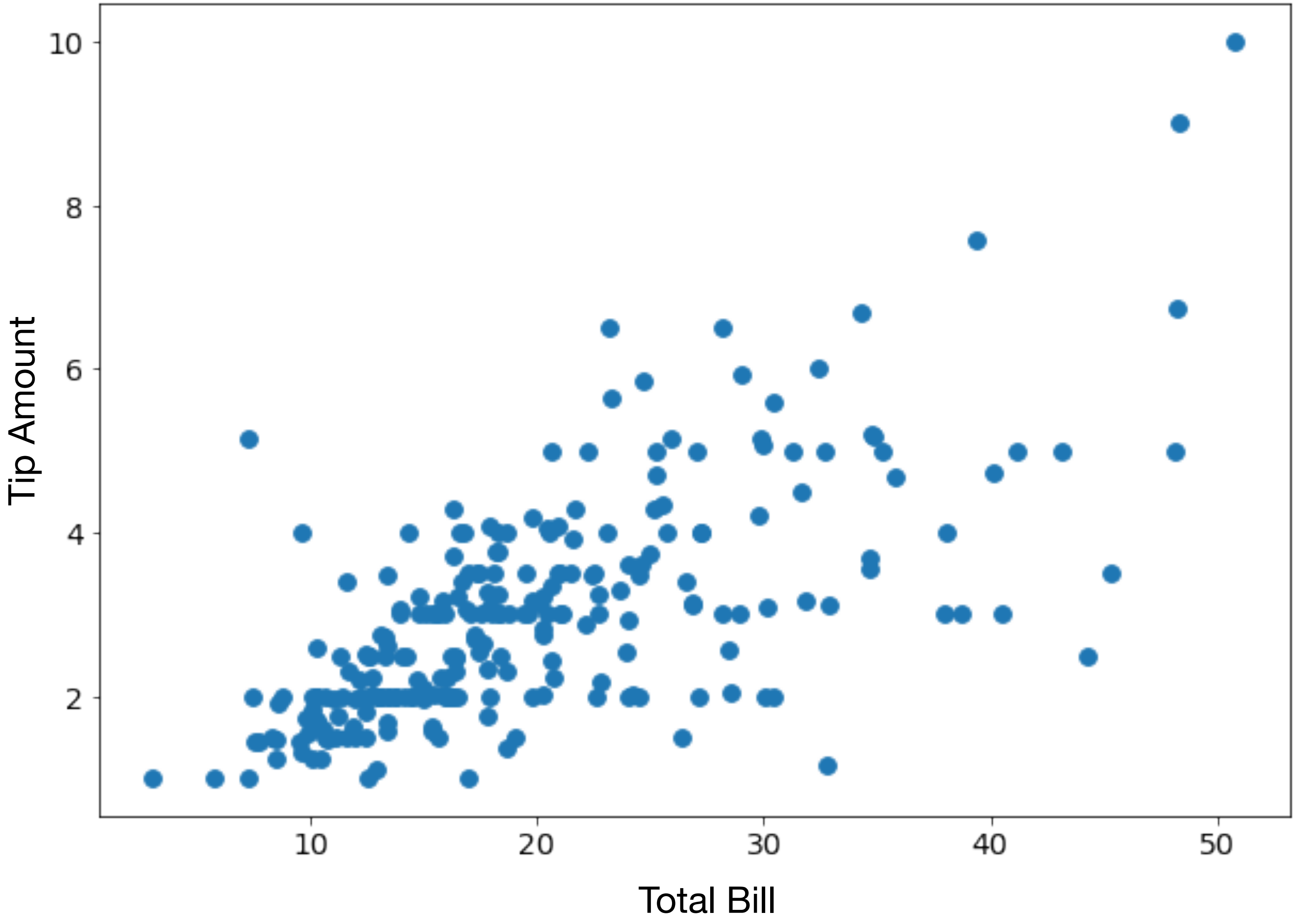
- Scikit-learn is a comprehensive open source Python package for machine learning and data analysis: http://scikit-learn.org
- Anaconda includes Scikit-learn as part of its free distribution.
- Scikit-learn algorithm inputs and outputs are usually represented as NumPy arrays, although we can also work with input data as Pandas DataFrames.

```
import sklearn
```



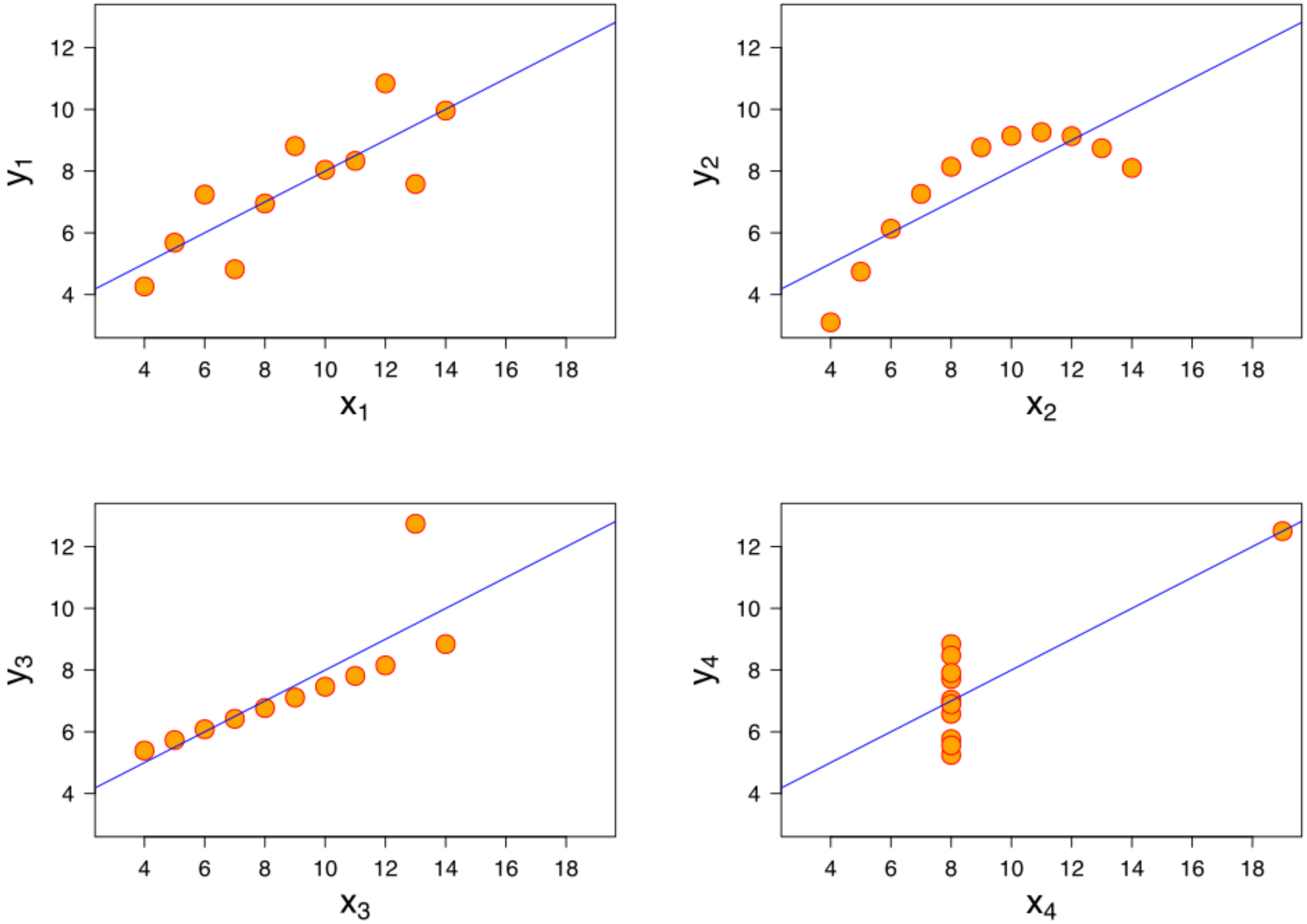
Finding Patterns in Data

- When analysing a new dataset, as a first step we might visualise the data to identify any obvious patterns.
- **Example:** Dataset of 244 meals, with details of total meal bill and tip amount. What can we say about the data?



Anscombe's Quartet

- 4 datasets with nearly identical statistical properties. Yet, each expresses quite different relationships between Y and X.
- Important to look at the data visually before building a model.

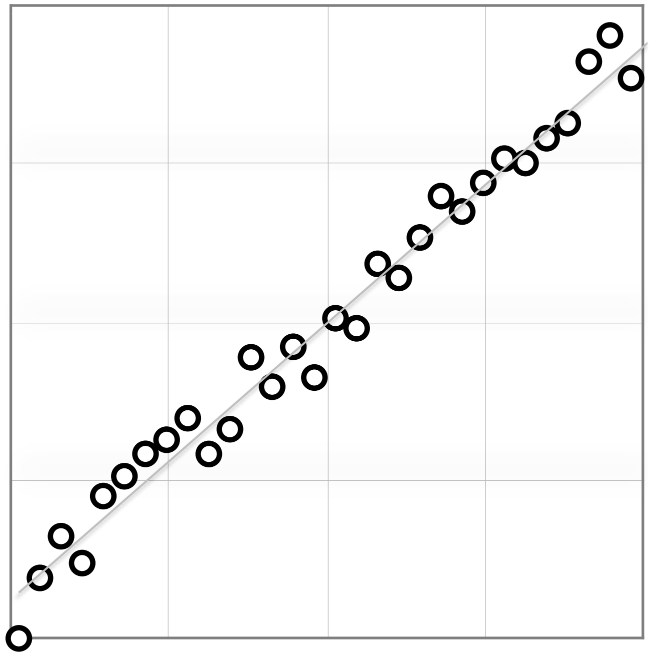


Wikipedia

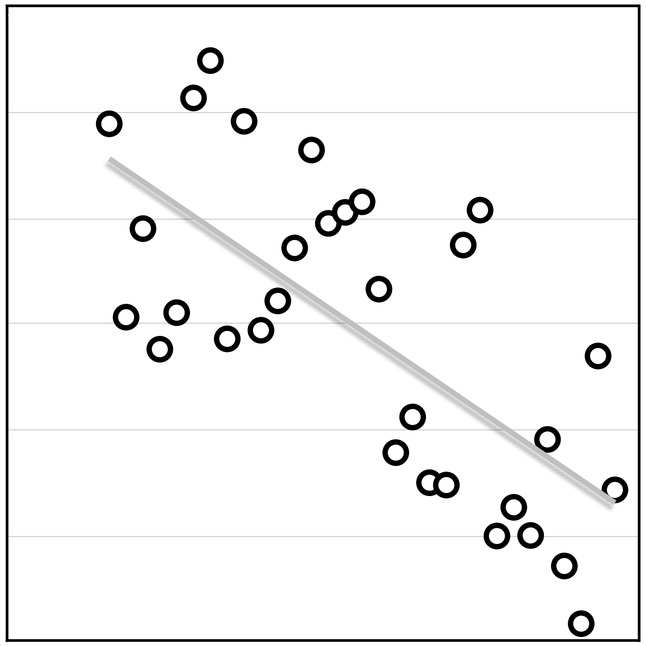
Correlation

- Visualising data using scatter plots can provide us with a sense of the relationship between two variables.
- Relationships can have different directions (positive or negative) and different strengths (weak or strong).
- How can we quantify this numerically?

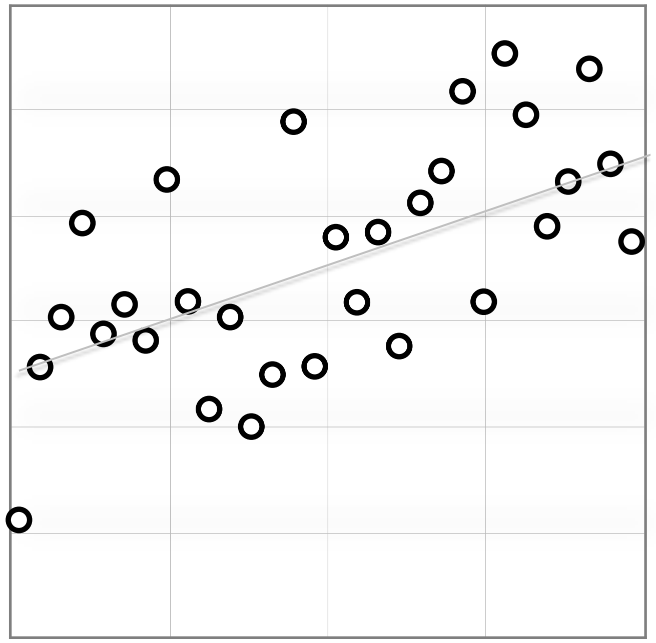
Strong positive relationship



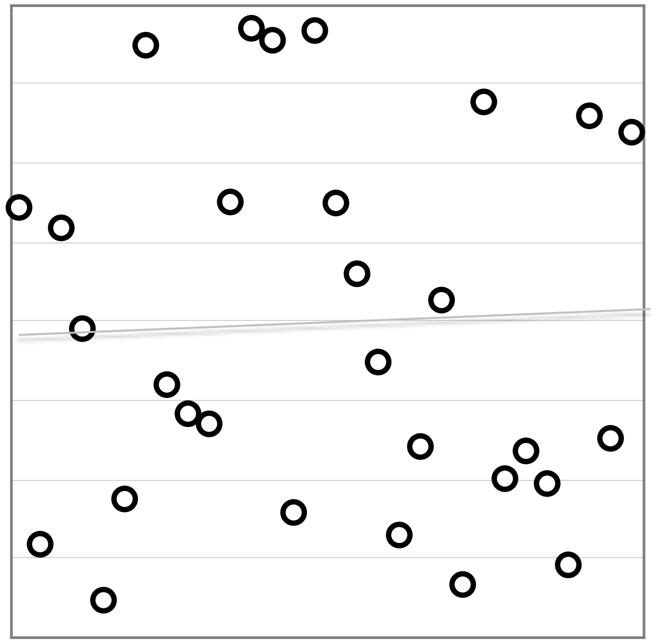
Weak negative relationship



Weak positive relationship



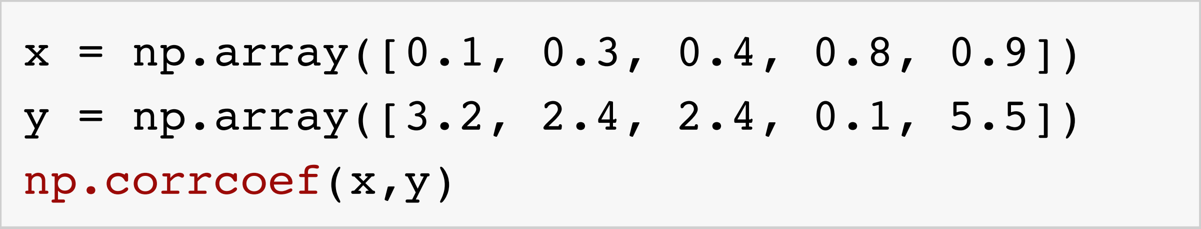
No relationship



Correlation in Python

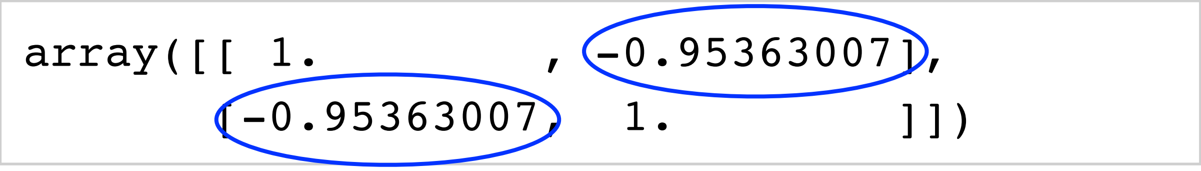
- Often useful to know the strength of relationship between Y and X, but independent of the units of measurement.
- The Correlation between Y and X is a statistical measure of how strongly two variables are related. It is dimensionless, i.e. a unit-free measure of the relationship between variables.
- Takes a value in [-1,+1], where 1 is total positive correlation, 0 is no correlation, –1 is total negative correlation.

$$Cor(Y,X) = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{y_i-\bar{y}}{s_y}\right)\left(\frac{x_i-\bar{x}}{s_x}\right) \quad \text{where} \quad s_y^2 = \frac{\sum_i^n (y_i-\bar{y})^2}{n-1}$$



```
x = np.array([0.1, 0.3, 0.4, 0.8, 0.9])
y = np.array([3.2, 2.4, 2.4, 0.1, 5.5])
np.corrcoef(x,y)
```

Calling the NumPy `corrcoef(x,y)` function will create a 2x2 Pearson correlation coefficient matrix.

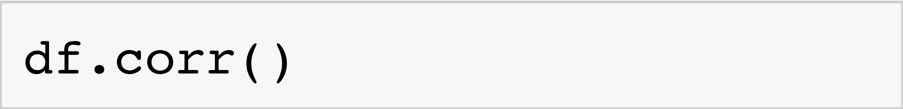


```
array([[ 1.        , -0.95363007],
       [-0.95363007,  1.        ]])
```

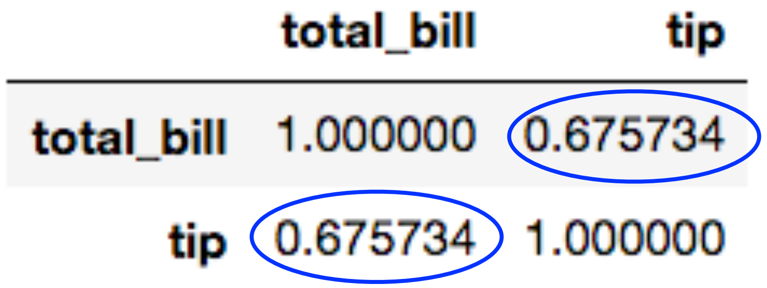
The off-diagonals indicate strength of relationships.

Correlation in Python

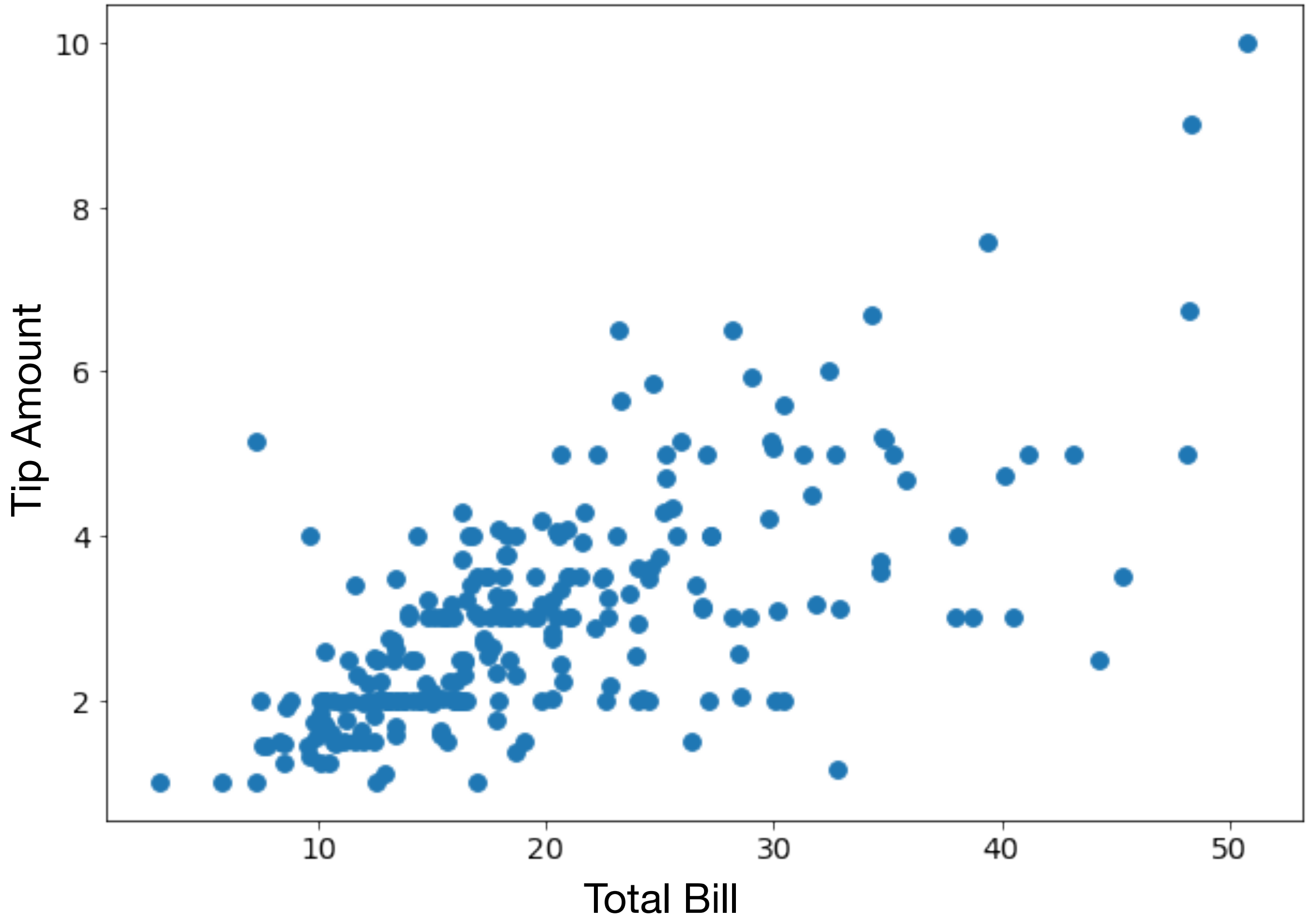
- If our data is stored in a Pandas DataFrame, we can also use the `df.corr()` function.

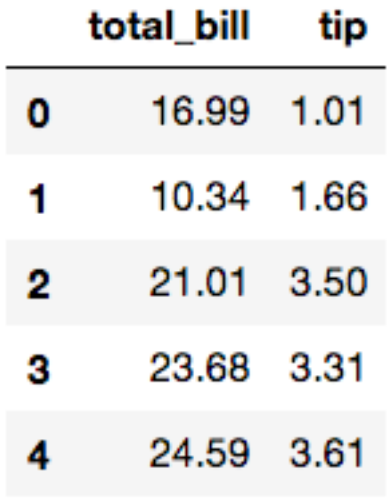


```
df.corr()
```



	total_bill	tip
total_bill	1.000000	0.675734
tip	0.675734	1.000000





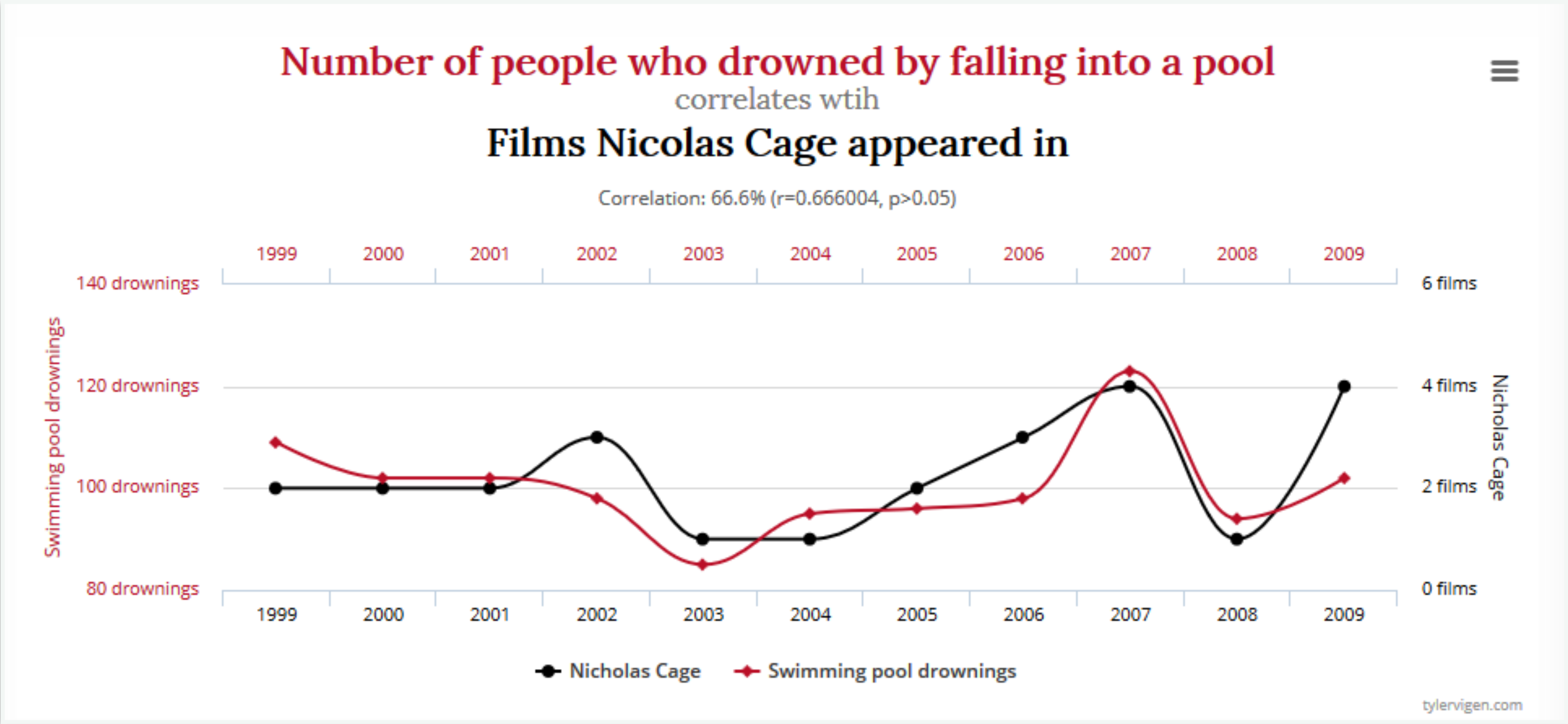
	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61

The off-diagonals indicate strength of relationships.

Remember, 1 is total positive correlation, 0 is no correlation, –1 is total negative correlation.

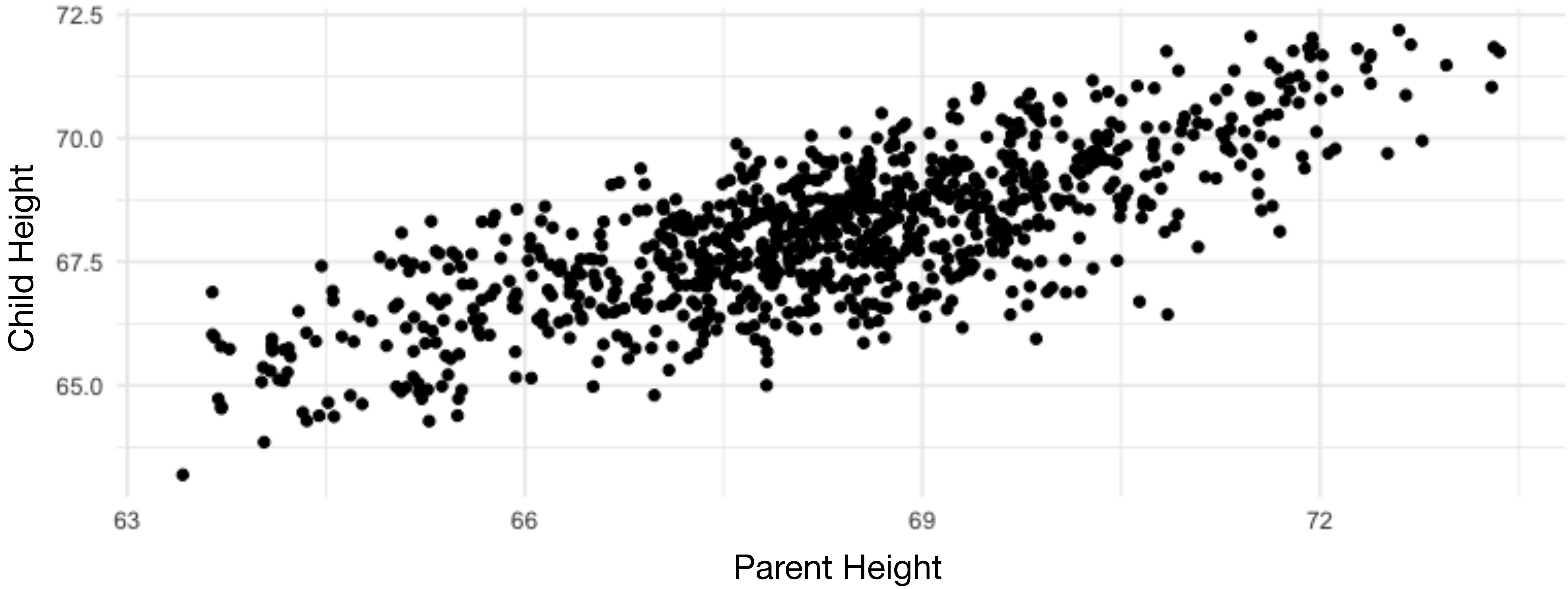
Correlation vs Causation

- Causation: indicates that one event is the result of the occurrence of the other event - i.e. there is a causal relationship between the two events.
- But a correlation between variables does not automatically mean that the change in one variable is the cause of the change in the values of the other variable.



Regression

- Regression analysis: A common statistical process for estimating the relationships between variables. This can allow us to make numeric predictions based on past data.
- **Simple example**: Can we predict a child's height, based on their parent's height?

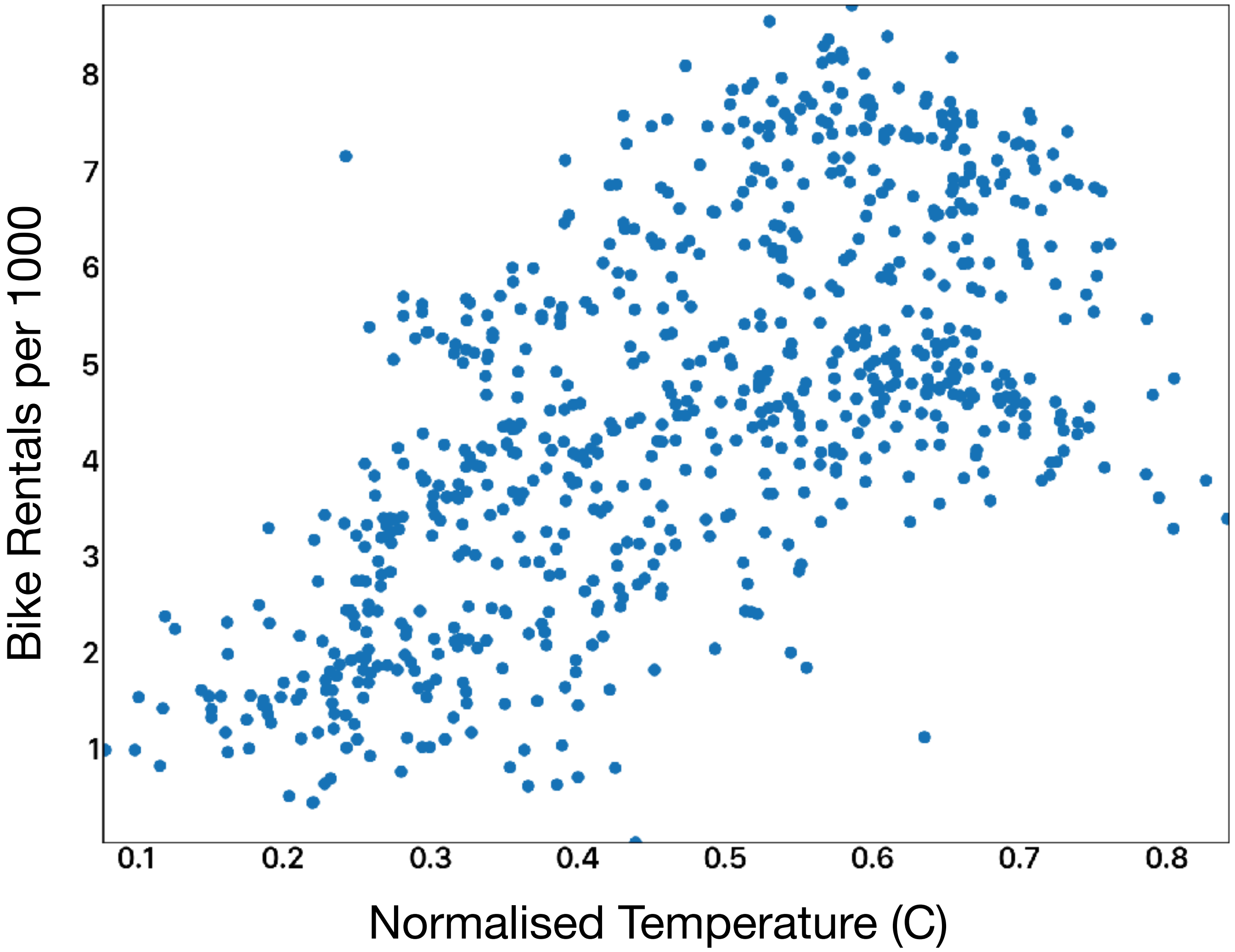


https://web.stanford.edu

Regression

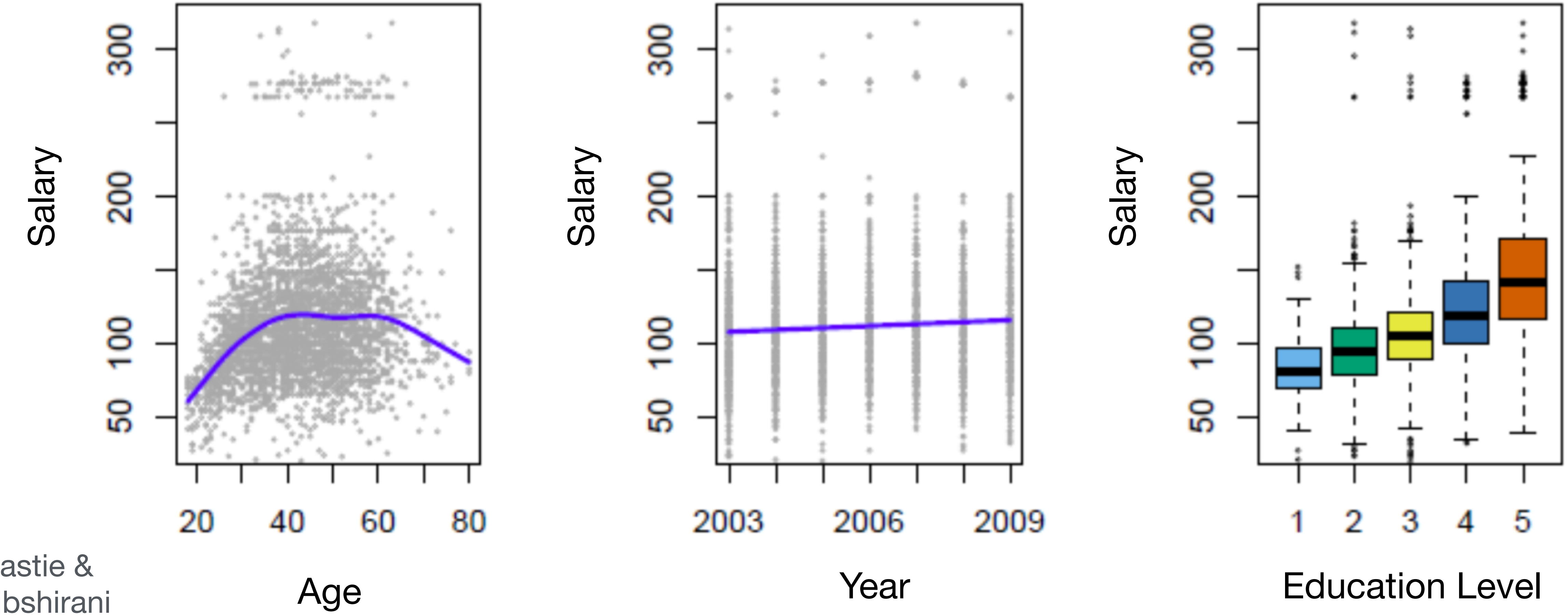
- Regression analysis: A common statistical process for estimating the relationships between variables. This can allow us to make numeric predictions based on past data.

- **Example**: Bike sharing data
- From the data it looks like there are more rentals on warmer days.
- Can we predict the number of bike rentals per day, based on temperature forecast information?



Regression

- Regression analysis: A common statistical process for estimating the relationships between variables. This can allow us to make numeric predictions based on past data.
- **Example**: Can we establish a relationship between salary level and demographic variables in population survey data?



Hastie & Tibshirani

Regression

- Regression analysis: A common statistical process for estimating the relationships between variables. This can allow us to make numeric predictions based on past data.
- **Approach**: Can we determine the relationship between two given variables (X and Y) based on an existing set of examples?

Simple examples:

```
(X,Y) = (1,1), (2,2), (4,4), (100,100), (20,20)
```

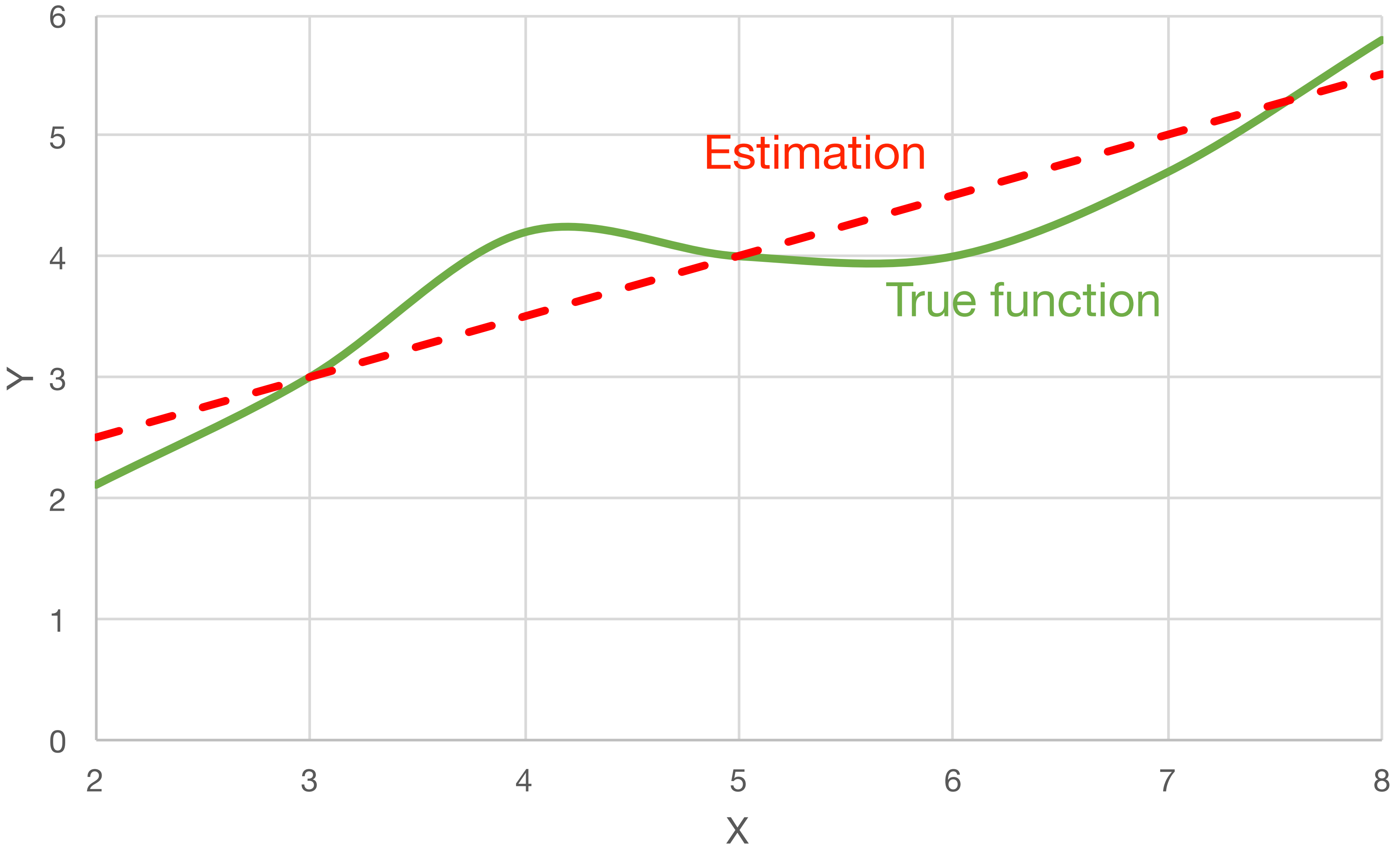
What will the value of Y be at X=5? $Y = 5 \ \ (Y = X)$

```
(X,Y) = (1,1), (2,4), (4,16), (100,10000), (20,400)
```

What will the value of Y be at X=5? $Y = 25 \ \ (Y = X^2)$

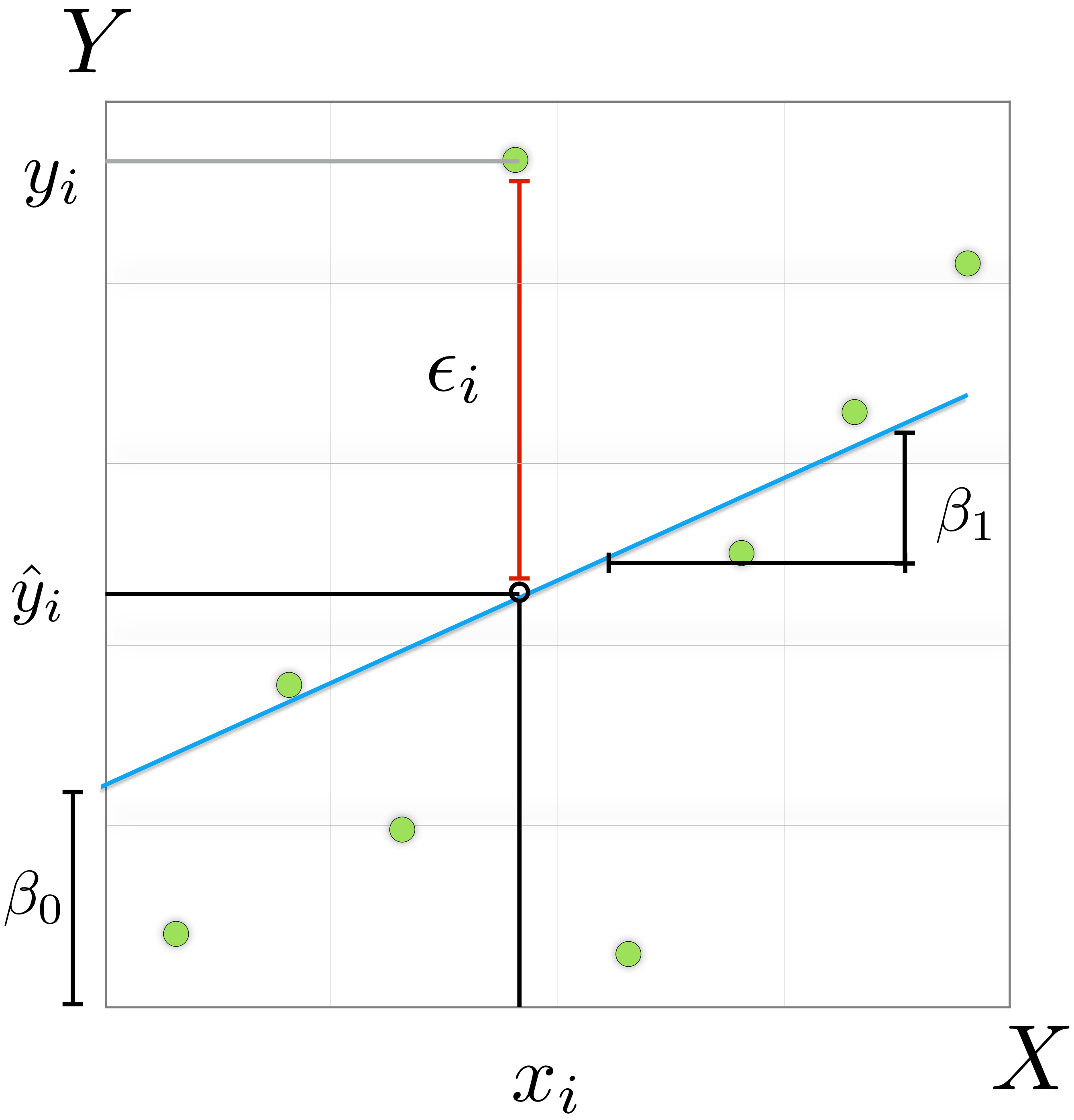
Linear Regression

- Linear Regression: a simple approach to predictive modelling. It assumes that the dependence of a response (dependent) variable *Y* on input (independent) variables X_1, X_2, … is linear.
- While true regression functions are never linear, simple linear regression is still often useful.



Linear Regression: Terminology

X	independent variable(s)
Y	dependent variable
y_i	observed value of Y
$\hat{y}_i$	predicted value of Y
β_0	intercept
β_1	slope
ϵ_i	error



Simple Linear Regression

- Simple Linear Regression: Method for predicting a numeric response using a single input variable (feature).
- The model is: $y = \beta_0 + \beta_1 x$

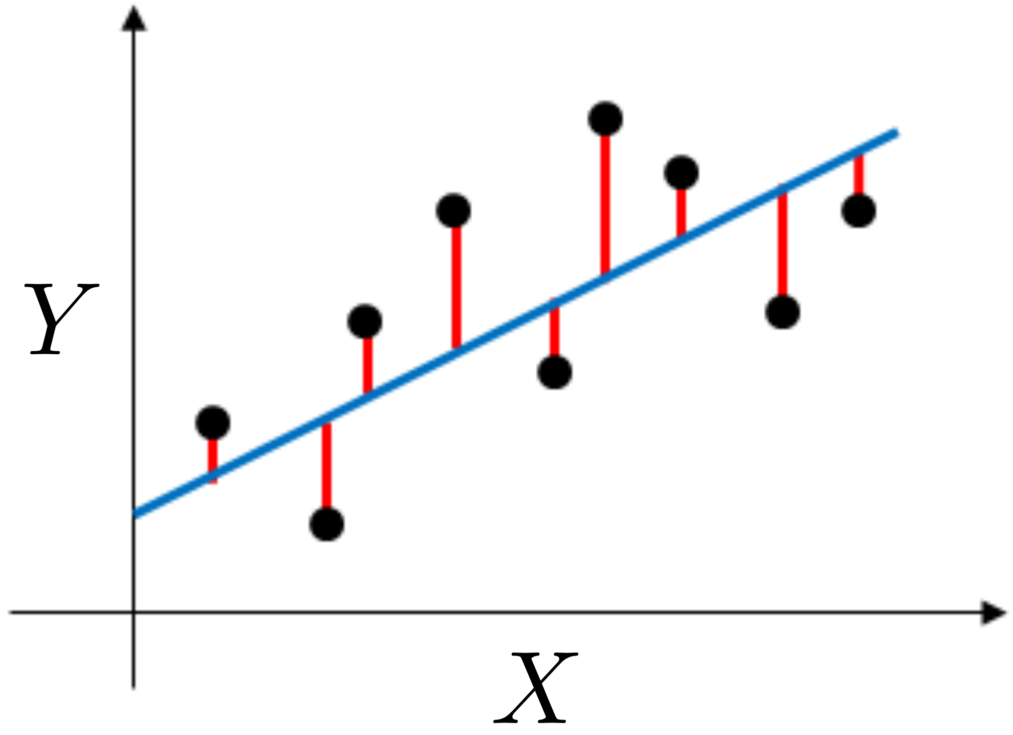
y : the response variable

x : the input feature

β_0, β_1 : the model coefficients (intercept and slope)

Goal is to learn the model coefficients from existing data.

Once we have learned the model, we can make future predictions.

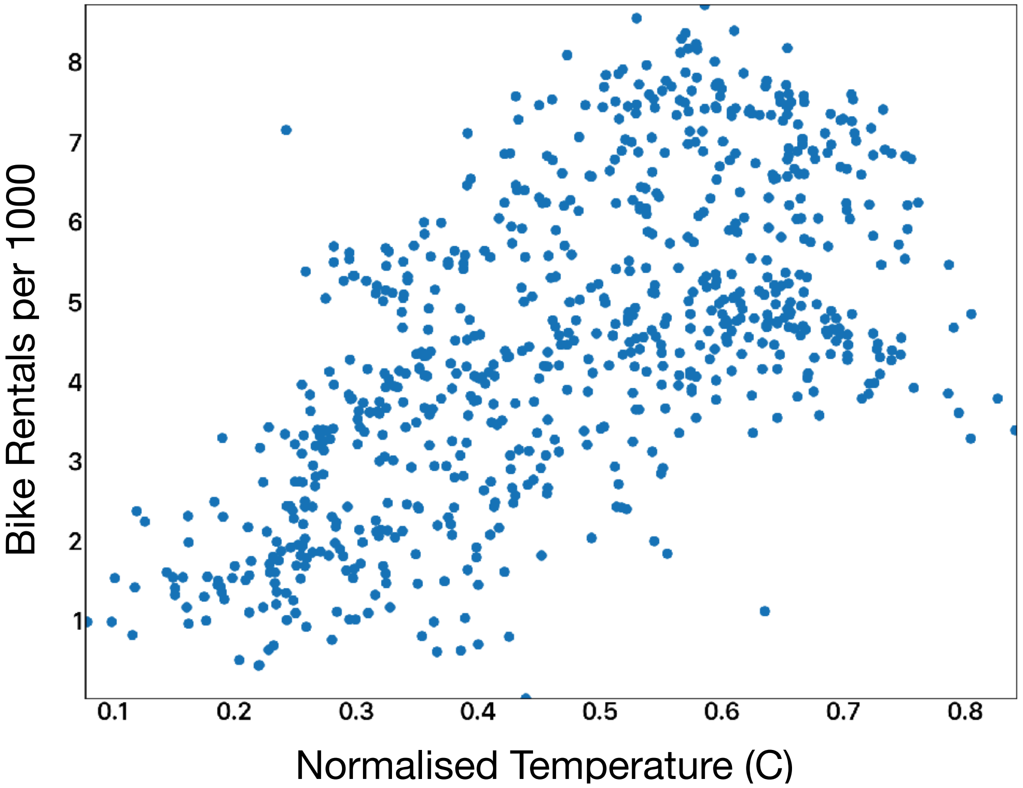


We learn the model by finding the best line (coefficients) which minimises the squared distance between our examples and the line.

Example: Simple Linear Regression

Q. Can we predict number of bike rentals, based on daily temperature?

- Independent variable: the temperature X
- Dependent variable: number of bike rentals Y



- Regression allows us to build a linear model of the form:

$$\text{rentals} = 945.824 + 7501.8339 \times \text{temperature}$$

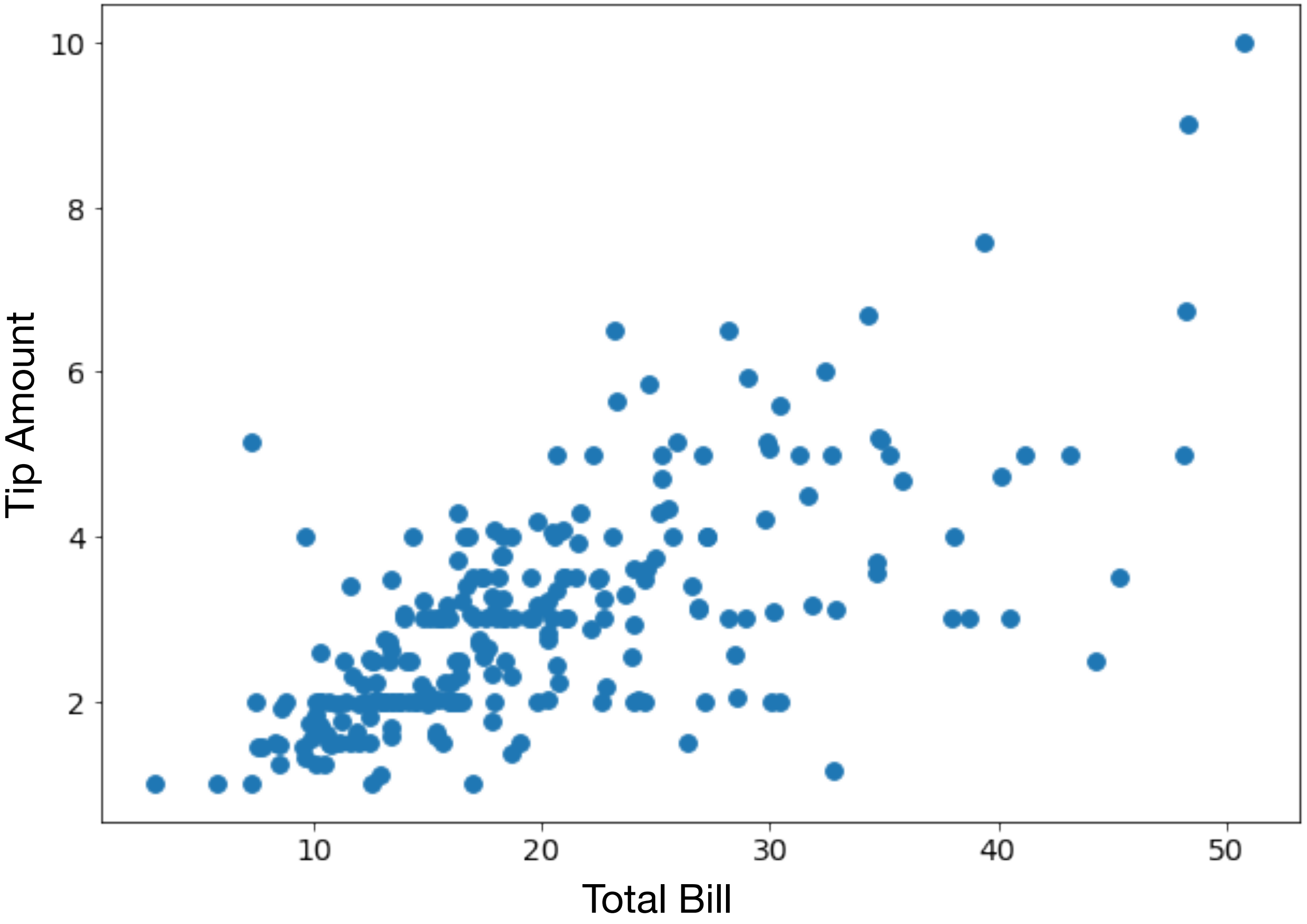
$$y = \beta_0 + \beta_1 x$$

For every 1 degree increase in the temperature, we would expect the bike rentals to increase by ~7501

Example: Simple Linear Regression

Q. Can we predict the tip amount, from the total bill?

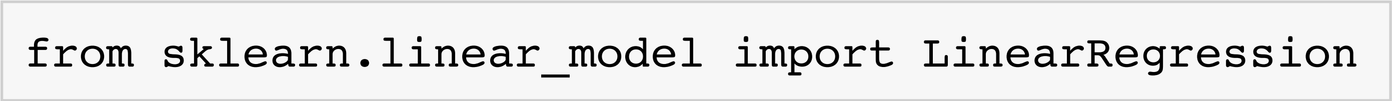
- Independent variable: the total bill amount X
- Dependent variable: the tip amount Y ("response" variable)



Use our historic dataset of 244 meals as the training data to build a new regression model which can then be used to make predictions.

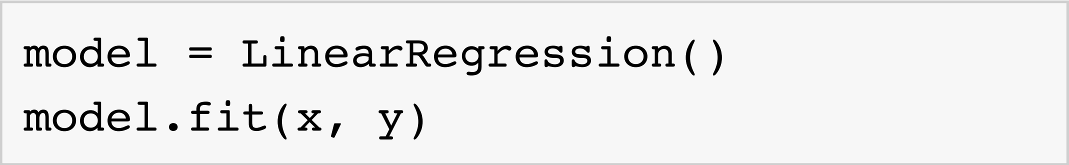
Example: Simple Linear Regression

- Scikit-Learn provides functions to apply linear regression to NumPy arrays. To build a model for input *x* and response *y*:



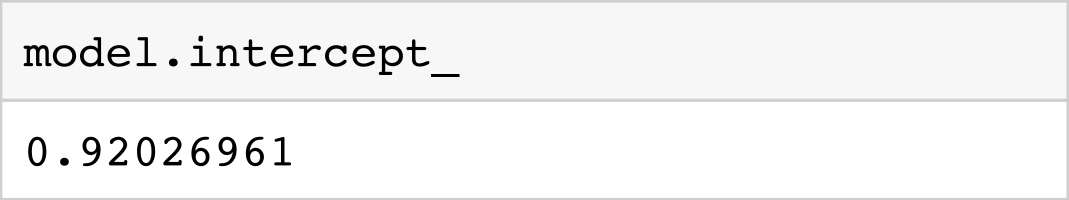
```
from sklearn.linear_model import LinearRegression
```

Create and fit the model based on the training data



```
model = LinearRegression()
model.fit(x, y)
```

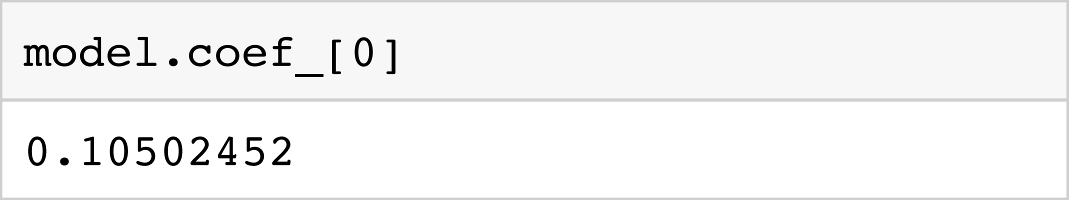
Get the intercept coefficient



```
model.intercept_
```

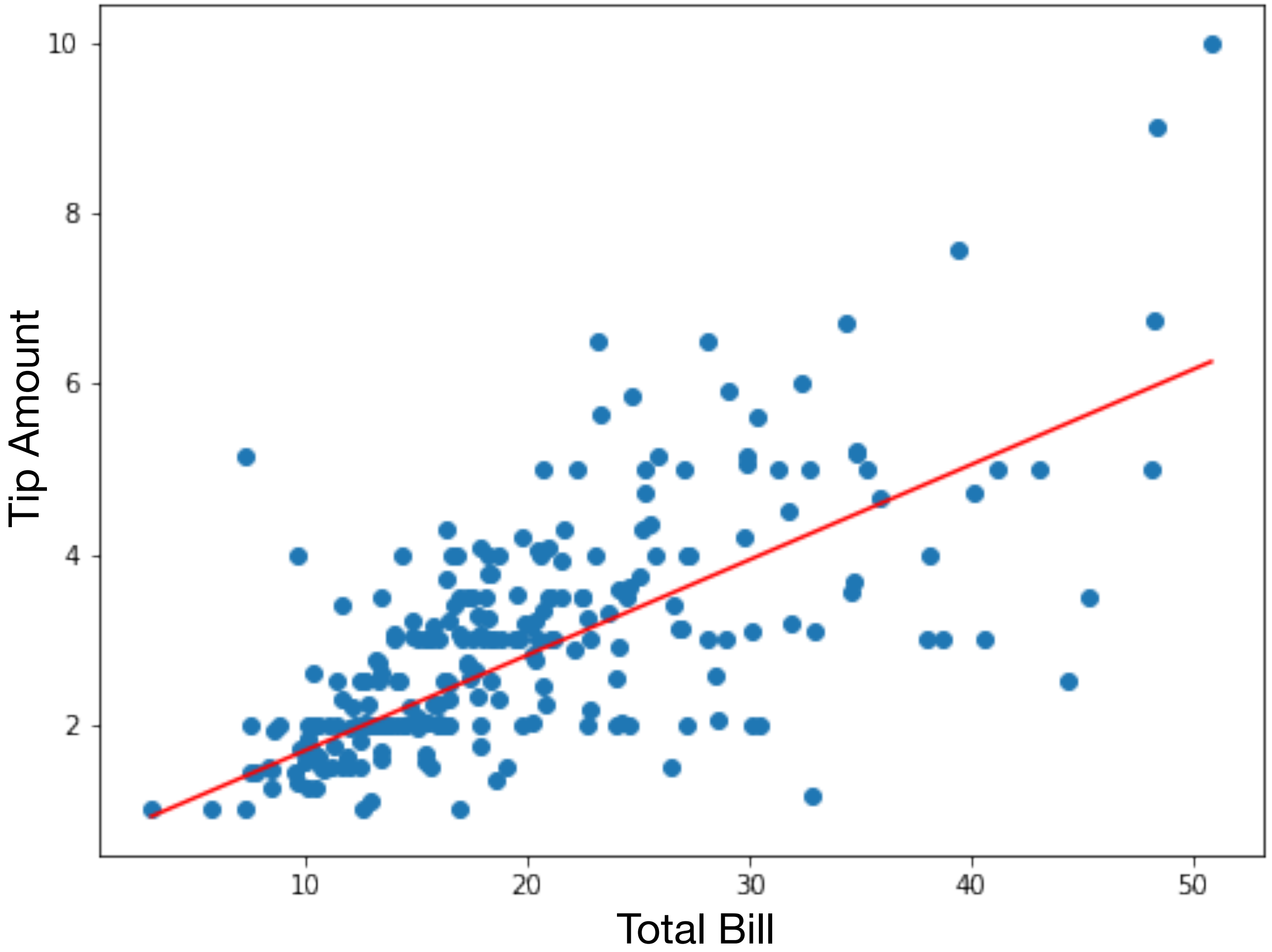
```
0.92026961
```

Get the slope coefficient



```
model.coef_[0]
```

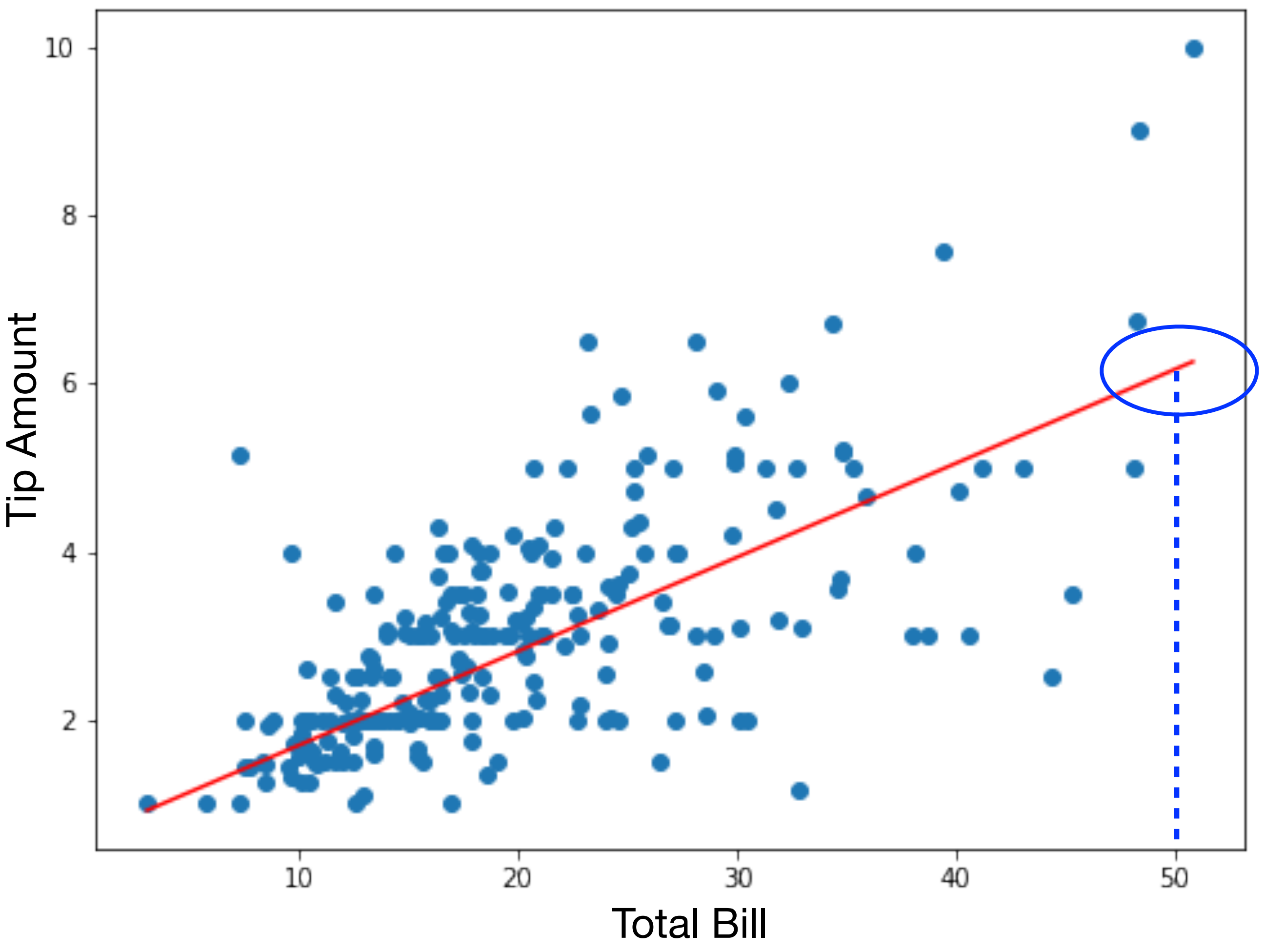
```
0.10502452
```



Example: Simple Linear Regression

- Now that we have a built our regression model, we can use it to make predictions - i.e. predict the tip amount, based on some specified amount for the total bill.

Bill	*Predicted Tip*
€10.00	€1.97
€15.00	€2.50
€20.00	€3.02
€25.00	€3.55
€30.00	€4.07
€35.00	€4.60
€40.00	€5.12
€45.00	€5.65
€50.00	€6.17
€55.00	€6.70
€60.00	€7.22
€65.00	€7.75

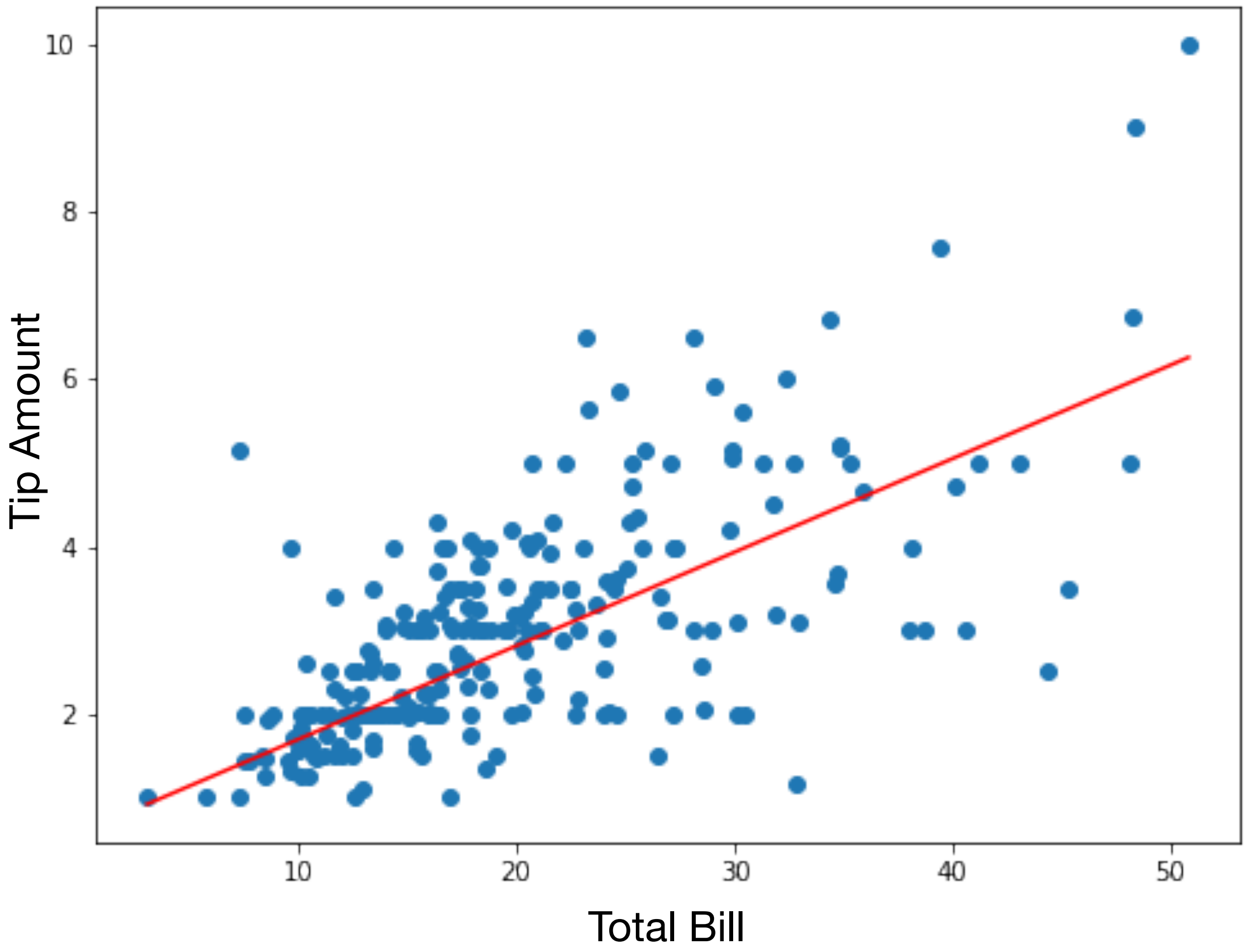


e.g. Predicted tip for €50 meal is €6.17

Example: Simple Linear Regression

- We can also compare the output of our model with the original data to see if it agrees. When we look at the first 10 rows of the training data, we see there are some errors. Our regression model does not fit the data perfectly.

Bill	*Predicted Tip*	*Actual Tip*
€16.99	€2.70	€1.01
€10.34	€2.01	€1.66
€21.01	€3.13	€3.50
€23.68	€3.41	€3.31
€24.59	€3.50	€3.61
€25.29	€3.58	€4.71
€8.77	€1.84	€2.00
€26.88	€3.74	€3.12
€15.04	€2.50	€1.96
€14.78	€2.47	€3.23



Example: Simple Linear Regression

- **Example:** For a startup company, what is the relationship between budget spent in different categories and profit generated?

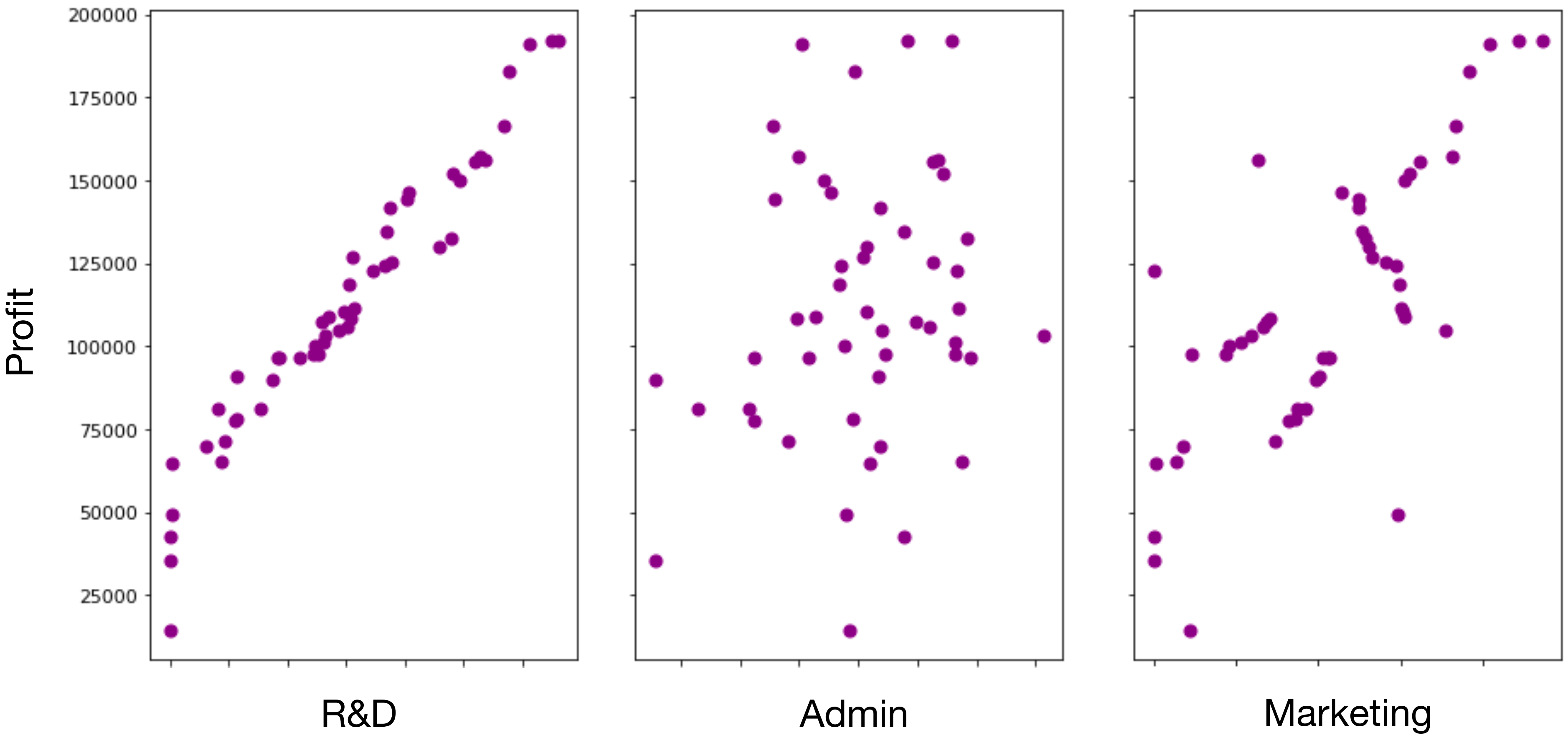
R&D	Admin	Marketing	Profit
1000.00	124153.00	1904.00	64926.00
78389.00	153773.00	299737.00	111313.00
20230.00	65948.00	185265.00	81229.00
78013.00	121598.00	264346.00	126993.00
0.00	135427.00	0.00	42560.00

Input Features:

1. Spending on Research & Development
2. Spending on Administration activities
3. Spending on Marketing Activities

Response:

- Profit generated

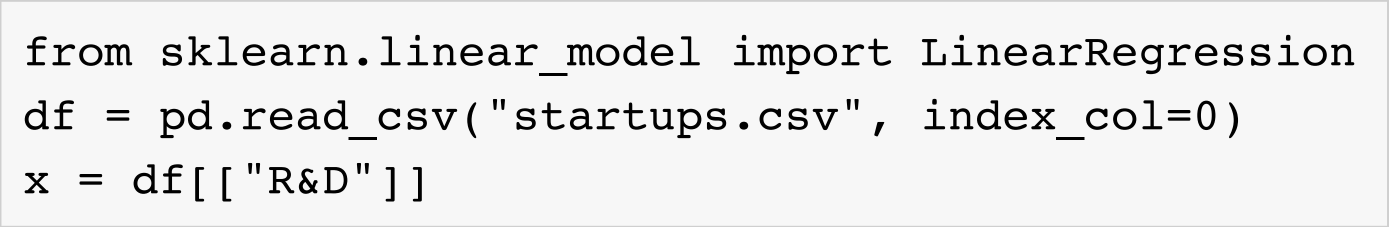


Which spending category has the greatest impact on profit?

Can we predict future profit based on existing data?

Example: Simple Linear Regression

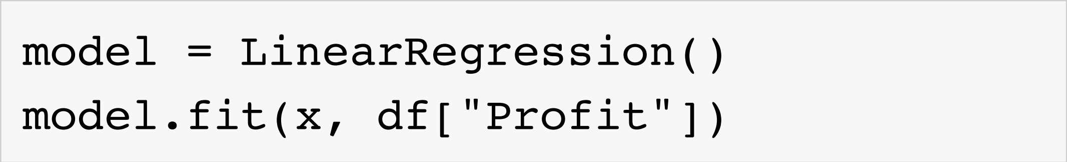
- **Example:** For a startup company, what is the relationship between budget spent in different categories and profit generated?



```
from sklearn.linear_model import LinearRegression
df = pd.read_csv("startups.csv", index_col=0)
x = df[["R&D"]]
```

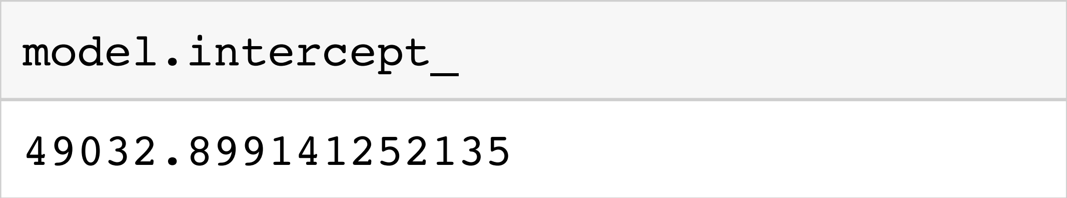
Let's examine the relationship between R&D spend and profit

Create and fit the model



```
model = LinearRegression()
model.fit(x, df["Profit"])
```

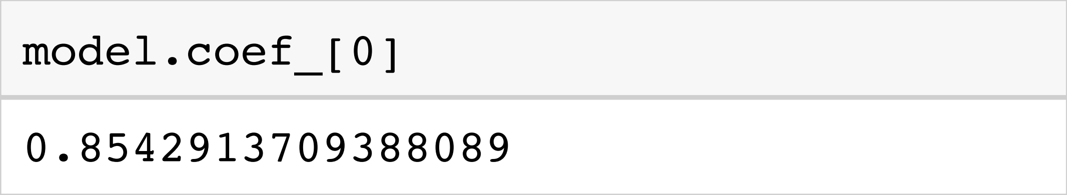
Get the intercept coefficient



```
model.intercept_
```

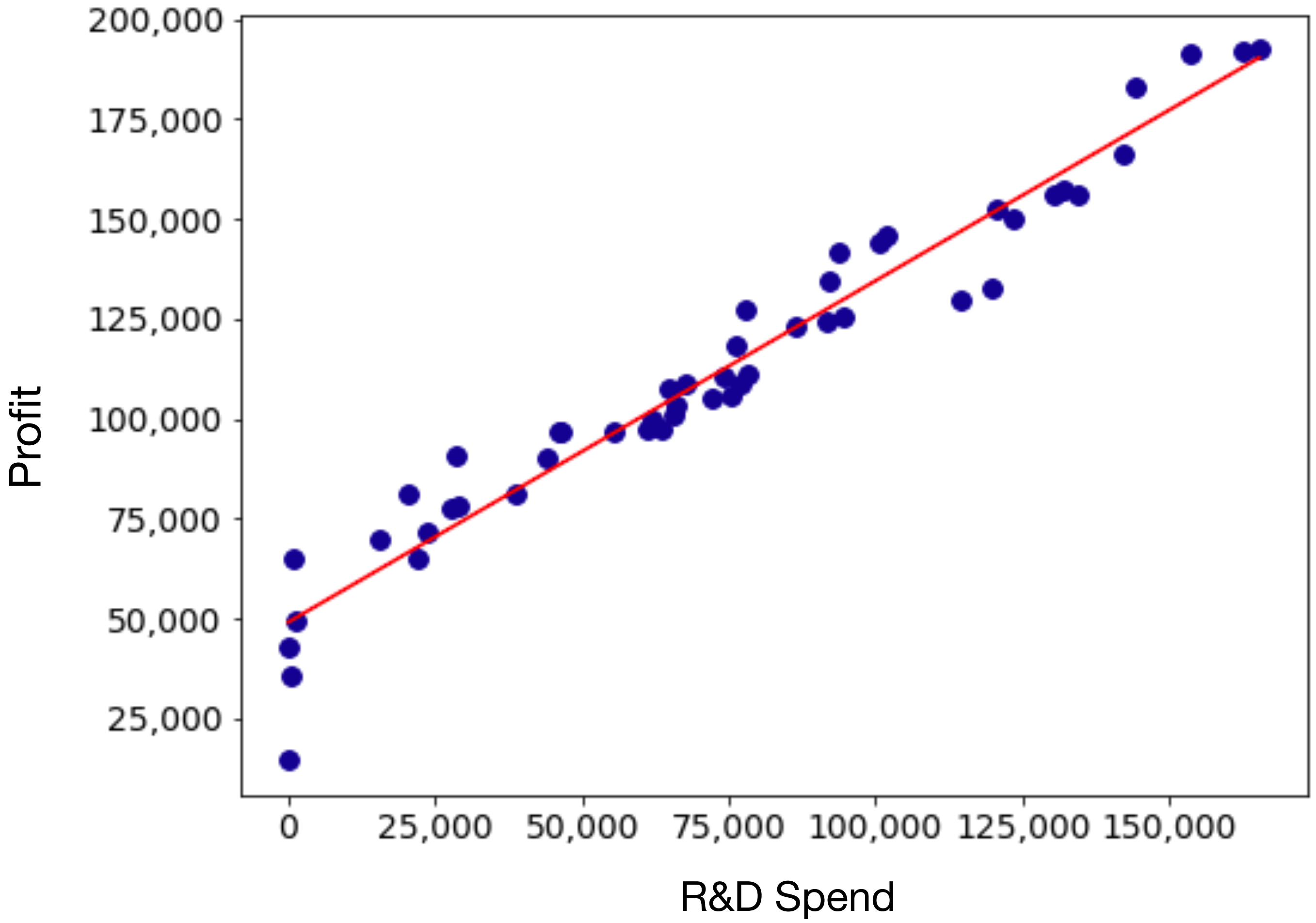
```
49032.899141252135
```

Get the slope coefficient



```
model.coef_[0]
```

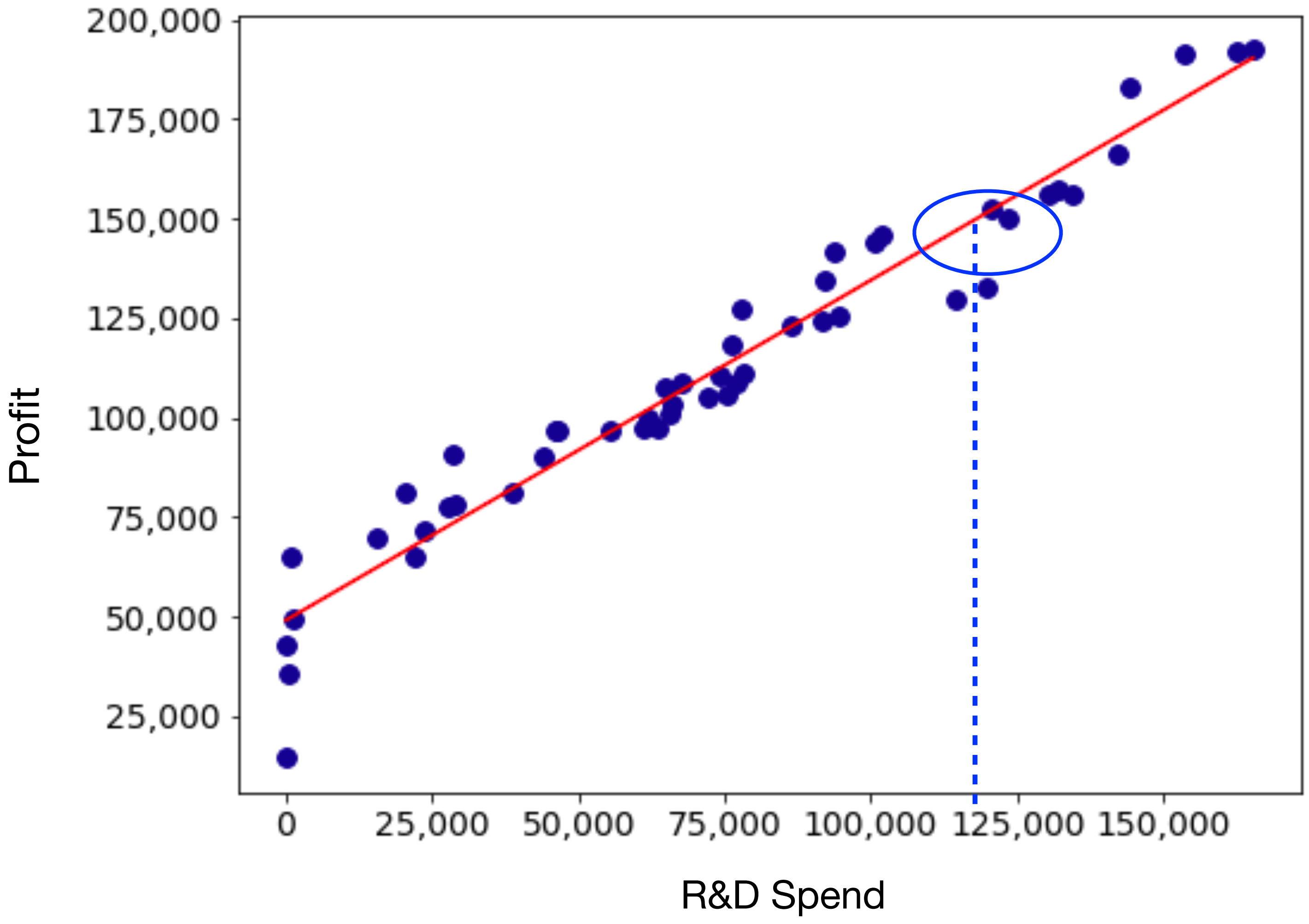
```
0.8542913709388089
```



Example: Simple Linear Regression

- Now that we have a built our regression model, we can use it to make predictions - i.e. predict profit levels, based on R&D budget spend.

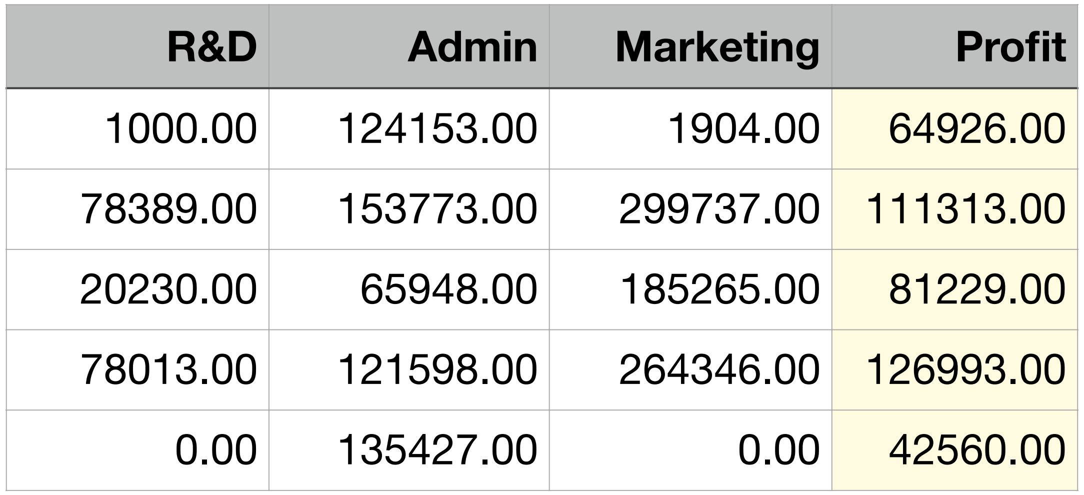
R&D Spend	*Predicted Profit*
0.00	49032.90
20000.00	66118.73
40000.00	83204.55
60000.00	100290.38
80000.00	117376.21
100000.00	134462.04
120000.00	151547.86
140000.00	168633.69
160000.00	185719.52



e.g. Predicted profit for 120,000 R&D spend is 151,547

Multiple Regression

- Multiple linear regression: Simple linear regression can easily be extended to include multiple features, where we try to learn a model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$
- Each feature x_i has its own coefficient β_i

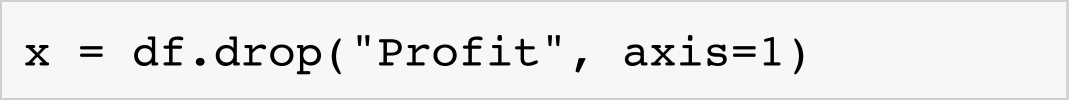


R&D	Admin	Marketing	Profit
1000.00	124153.00	1904.00	64926.00
78389.00	153773.00	299737.00	111313.00
20230.00	65948.00	185265.00	81229.00
78013.00	121598.00	264346.00	126993.00
0.00	135427.00	0.00	42560.00

e.g. Predict profit based on 3 features

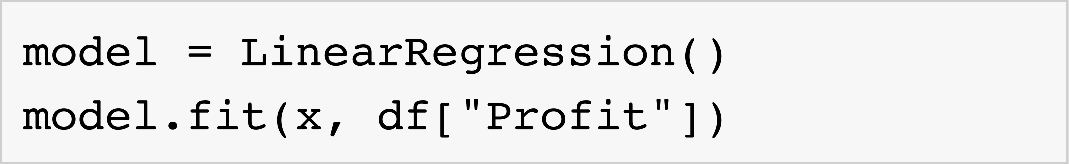
$$\begin{aligned} y = \beta_0 + \beta_1 &\times \text{R\&D} \\ +\beta_2 &\times \text{Admin} \\ +\beta_3 &\times \text{Marketing} \end{aligned}$$

Remove column we want to predict



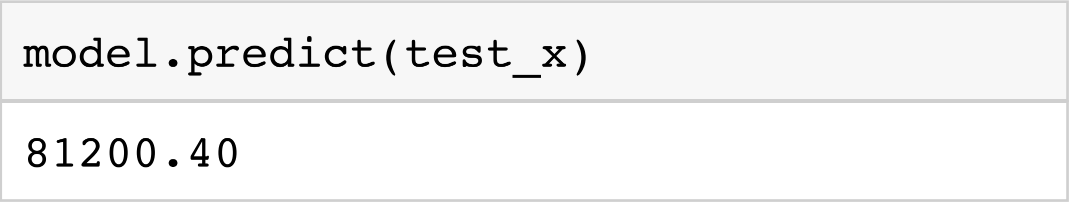
```
x = df.drop("Profit", axis=1)
```

Fit the model based on all 3 features



```
model = LinearRegression()
model.fit(x, df["Profit"])
```

We can now use the model to make profit predictions for unseen values of *(R&D, Admin, Marketing)*

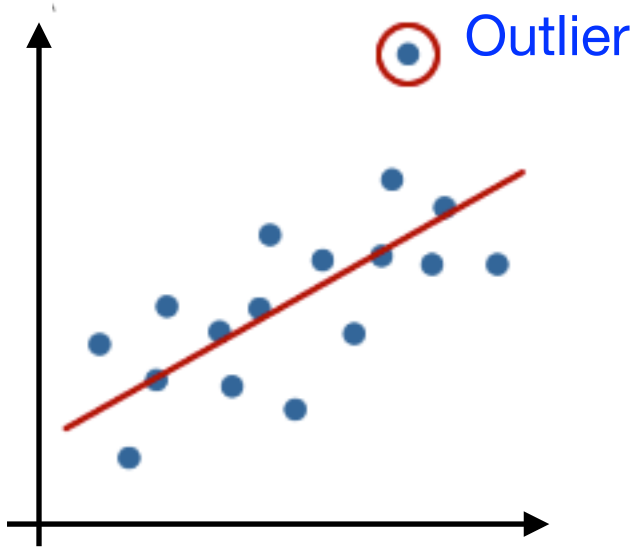


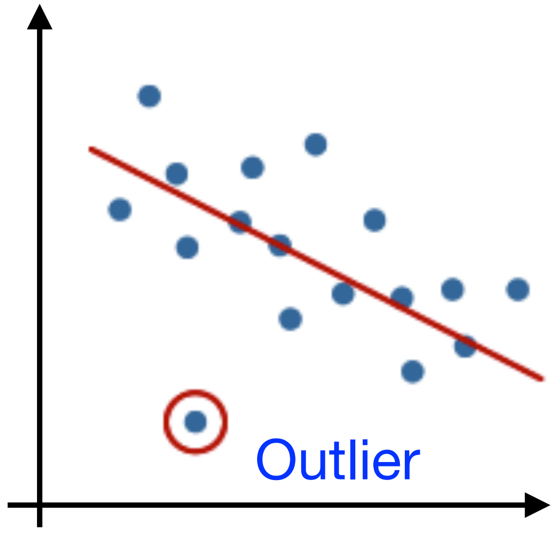
```
model.predict(test_x)
```

```
81200.40
```

Outliers in Regression

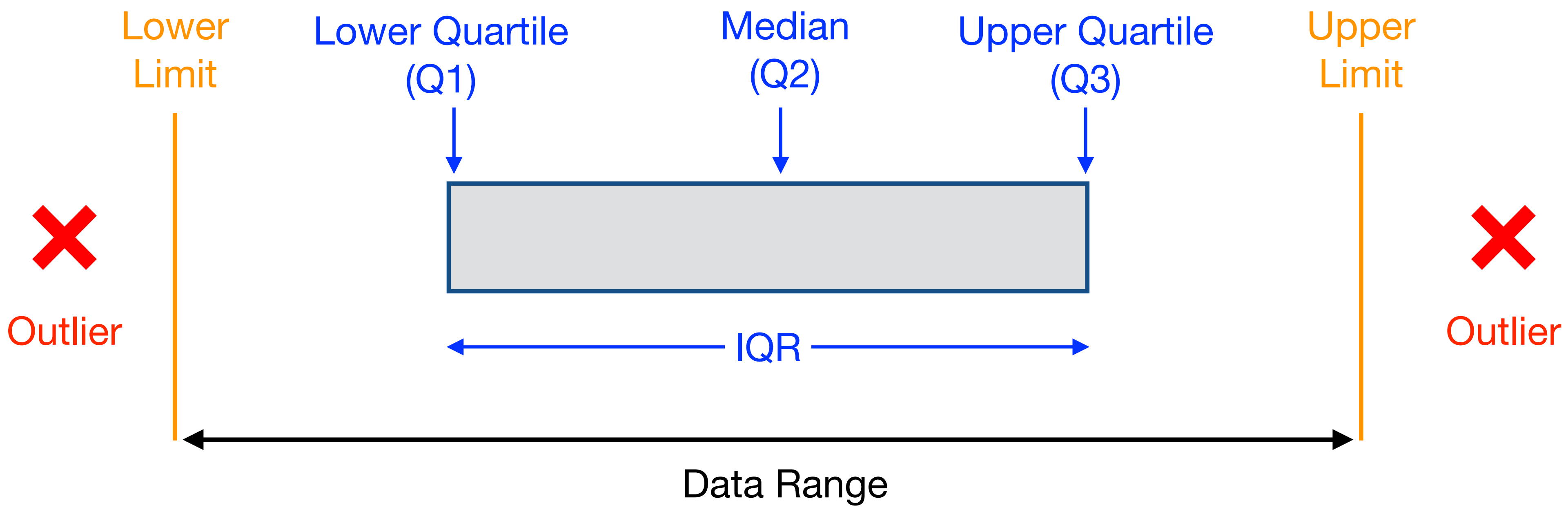
- Outlier: An observed data point that has a dependent variable value which is very different to the value predicted by the regression equation.
- Linear regression models assume there should be no significant outliers, as they can lead to a very poor regression fit.
- In some cases we might drop the outlier and recalculate the regression model. But it is always important to investigate the nature of the outlier before deciding whether to drop.





Finding Outliers

- Box plot diagram: a graphical method which helps to define reasonable lower and upper limits for a dataset, beyond which any data points will be considered as outliers.
- Median (Q2): the middle value of the dataset.
- Lower quartile (Q1): the median of the lower half of the dataset.
- Upper quartile (Q3): the median of the upper half of the dataset.
- Interquartile range (IQR): the spread of the middle 50% of the data values = Q3 - Q1



Box Plots in Python

- We can create box plots in Python to visualise the distributions of values for different variables, and look for any potential outliers.

Create a box plot with Matplotlib:

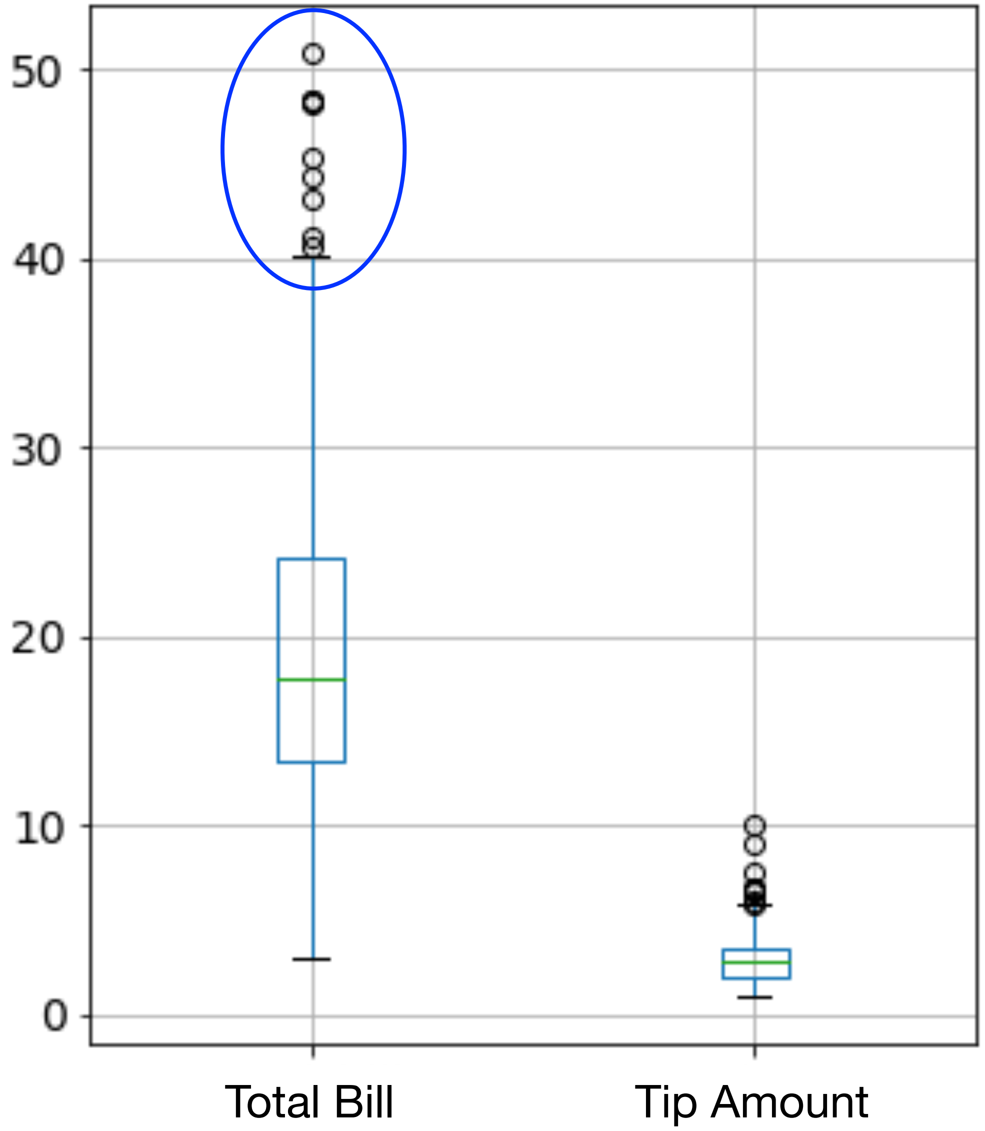
```
plt.figure()
plt.boxplot(data)
```

Create a box plot from all of the columns in a Pandas DataFrame:

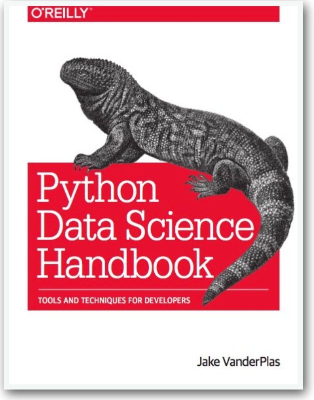
```
df.boxplot()
```

- Remember, it is always important to investigate the nature of an outlier before deciding whether to drop it.

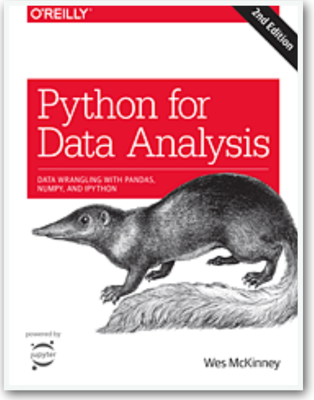
Possible outliers?



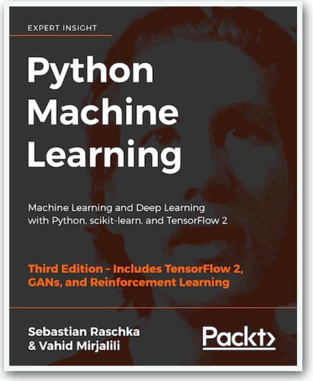
Further Reading


O'REILLY
Python Data Science Handbook
TOOLS AND TECHNIQUES FOR DEVELOPERS
Jake VanderPlas

Python Data Science Handbook
Jake VanderPlas
http://shop.oreilly.com/product/0636920034919.do



Python for Data Analysis, 2nd Edition
William McKinney
http://shop.oreilly.com/product/0636920050896.do


EXPERT INSIGHT
Python Machine Learning
Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2
Third Edition - Includes TensorFlow 2, GANs, and Reinforcement Learning
Sebastian Raschka & Vahid Mirjalili
Packt>

Python Machine Learning, 3rd Edition
Sebastian Raschka
https://github.com/rasbt/python-machine-learning-book-3rd-edition