REGRESSION

Dr. Brian Mc Ginley

INDEPENDENT/DEPENDENT VARIABLES

- Short note on *independent* and *dependent* variables
- **Example:**
  - The Value of a House Over Time
  - Here we have two variables:
    - *Value of House (V)*
    - *Time (t)*
  - Since they both change, they are Variables
    - Value is a dependent variable, since the value of a house **depends on** the **time** you bought it
    - Time is independent.
  - Plotting we typically leave the horizontal axis to represent the independent variable and the vertical axis to represent the dependent variable

X

Y

LINEAR RELATIONSHIPS

- If this course, we will mostly look at linear relationships to make our predictions.

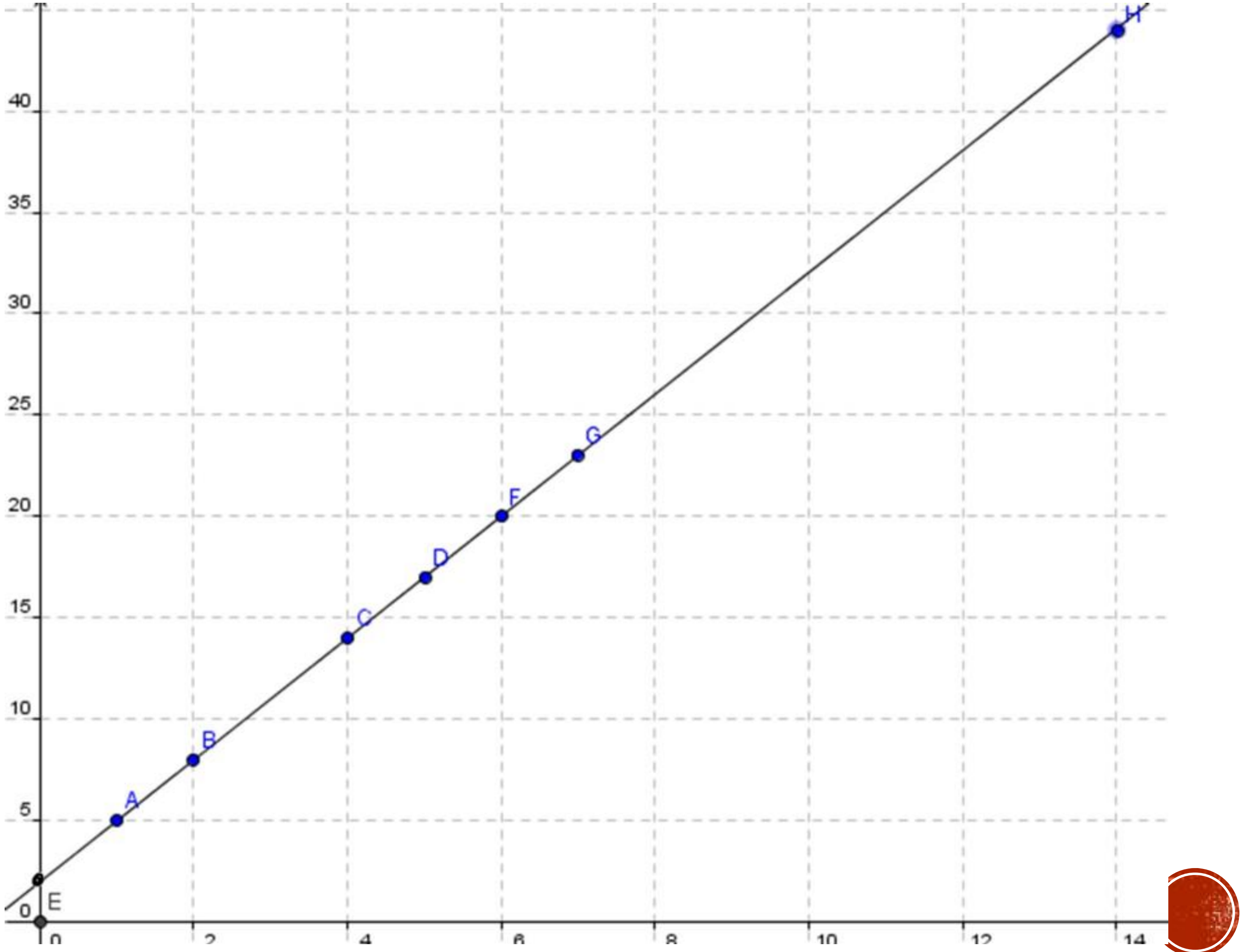
- **Definition:**
  - A Linear relationship: a change in the value of x always produces the same proportionate change in the value of y.

- Consider the following table:
  - What is the value of y when x = 3?
  - What value of x will give y = 44?

x	1	2	3	4	5	6				
y	5	8		14	17	20				44

LINEAR RELATIONSHIPS

- Once we are convinced that some relationship does exist, we can establish the precise nature of that relationship and use it to predict values of one variable that would correspond to any given values of the other.
- The exactness of the relationship can be seen in the diagram. The plotted points lie along an imaginary straight line. If we were to **draw a straight line through them, we could use it for making exact predictions without even using the formula.**



LINEAR RELATIONSHIPS

- Can you come up with a formula for the previous graph?

$$y = mx + c$$

$$y = 3x + 2$$

for this particular line. We can now use this formula to "predict" values.

- This very early taught equation of a line is the basis for regression and lots of machine learning analysis

REGRESSION

- This is where we try to zone in on a continuous value(s) rather than trying to predict a class or category.
- **Example: How to Cook a Turkey**
  - There is a rule of thumb when cooking a turkey.
  - Put it in a pre-heated oven for 20 minutes per pound plus 20 minutes.
  - Where did this idea come from?
    - Is it correct?
    - What could happen if it was wrong?

EXAMPLE: HOW TO COOK A TURKEY

- This is a linear model which is likely based on linear regression analysis of different weights of turkeys. The mathematical model is

$$t = mw + c$$

- where t is the cooking time, w is the weight in pounds, m is the slope of the line and c is the y -intercept.

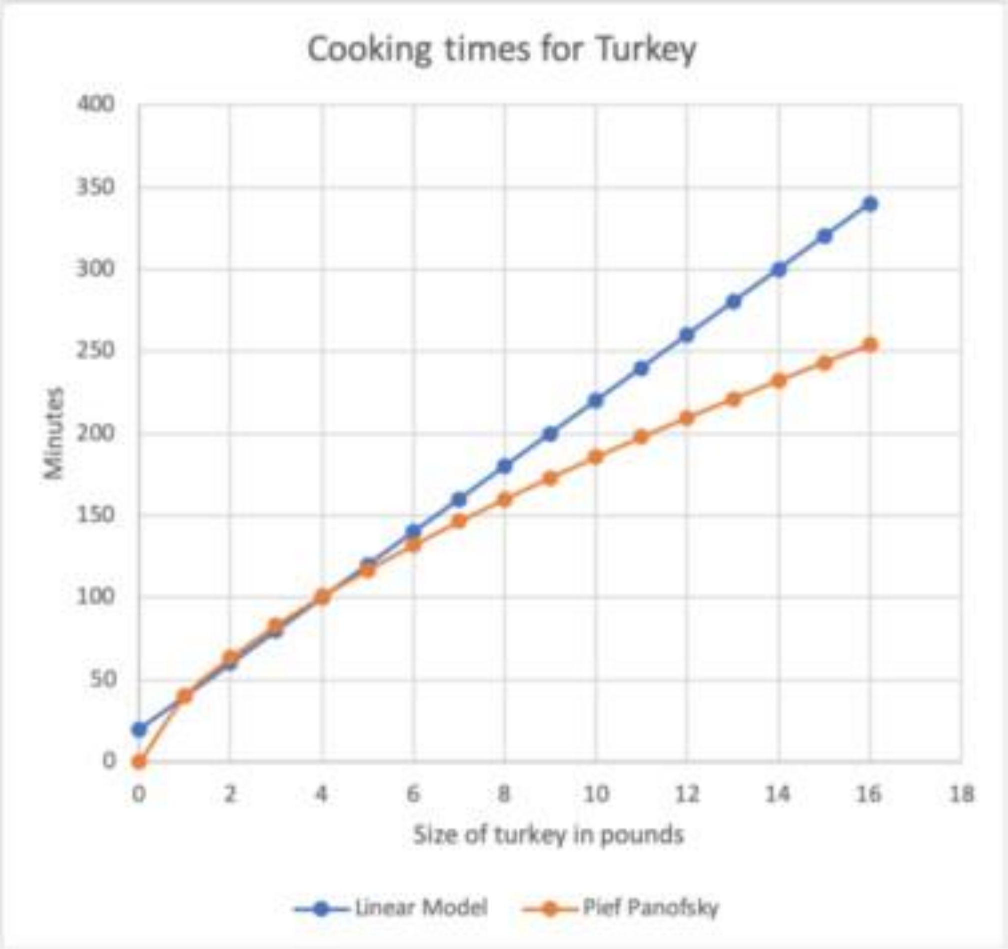
- Pief Panofsky disagreed and used the following model.

$$t = w^{(2/3)}/1.5$$

- Let’s analyse this.

EXAMPLE: HOW TO COOK A TURKEY

- First notice that for the linear model you have to cook an imaginary turkey for 20 minutes.
- Between 1 pound and 6 pounds, the models are very similar. There is 90 minutes difference between them at 16 pounds.
  - Which is Safer?
  - Which tastes better?
- Use a meat thermometer.



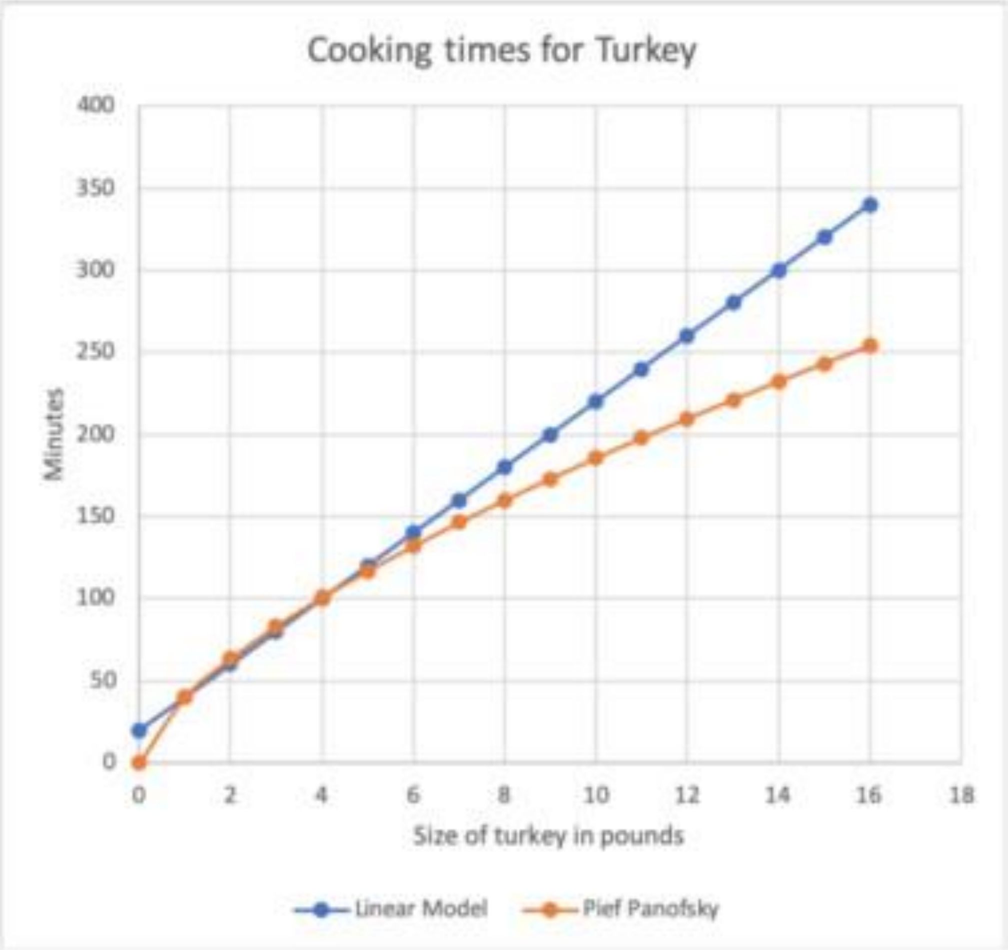
FITTING A LINEAR MODEL

- With the above example we see the result of a linear model but not how it was arrived at.
- To do that we have to collect some data and fit the best line to it.
- In the above case, this would mean cooking many turkeys of various weights and cooking them all using some calibrated method, such as a meat thermometer (or an expert cook with experience).
- We collect the “right” answer and use this to build the model i.e. figure out the parameters.
- Let’s look at some data.

EXAMPLE: HOW TO COOK A TURKEY

- Given the data to the left, if you had a nine-pound turkey, how long should you cook it for?
- Is this a task for supervised learning or unsupervised learning?
- If we assume a linear model, what are the parameters that we need to learn?

$$t = mw + c$$



EXAMPLE: HOW TO COOK A TURKEY

Weight in Pounds	1.1	1.3	1.7	2	2.1	2.3	…
Time taken to cook	39.4	48.8	55.8	63.0	69.5	67.7	…

- So *w* is our input variable/feature.
- *t* is our output and can be considered our regression target that we will use to train the model.
- We need to learn *m* and *c*.
- We will also use the notation of *M* for number of training samples.
  - *(w, t)* one training example.
  - $(w^{(i)}, t^{(i)})$ the *i*-th training example.

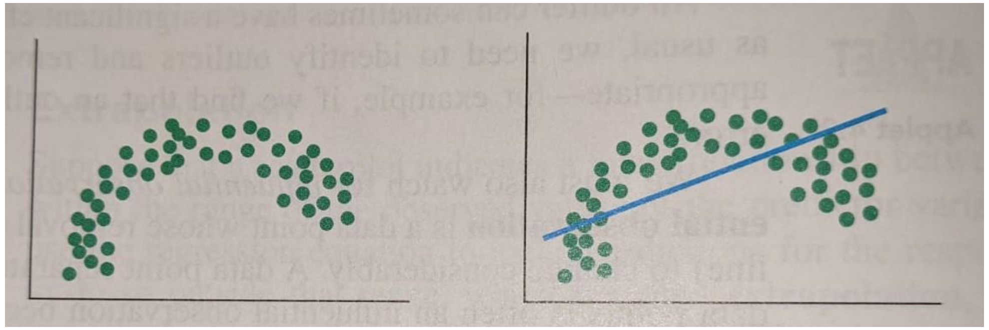
NOTATION

- You will notice in the previous slides that we used variable names that either match the problem or that are used regularly in Mathematics.
- However, we will not always be dealing with linear models so we are going to use notation that can be scaled up to larger models.
- There isn't a set notation, so where possible we will use whatever is used in scikit-learn
- Therefore, we will use w to represent parameters/weights that we are going to train. We will most often use y as our output and x as input.
- So, our above equation will now be

$$y = w_0 + w_1 x$$

WARNING ON USE OF LINEAR REGRESSION

- Linear Regression is based on the assumption that the data points are scattered about a line. However, often data points are scattered on a curve, e.g.



- You can still compute the coefficients (w_0 and w_1) which will give the line above. However, we can see it is an inappropriate fit as the line suggests the y-values keep increasing while the curve shows that they will actually decrease after a point
- **Criterion for Finding a Regression Line**
  - Before finding a regression line for a set of data points, draw a scatterplot. If the data points do not appear to be scattered about a line, do not determine a regression line.

OTHER REGRESSIONS

- This was an example of a Simple Linear Regression model. There is only one independent feature, but of course the real world doesn't work like that.
- There can be polynomial regression

$$y = w_0 + w_1 x + w_2 x^2 + \dots w_n x^n$$

- Multiple Linear Regression, where instead of x being a single feature it is a vector of features $\boldsymbol{x} = (x_1, x_2, x_3, x_4, \dots)^T$

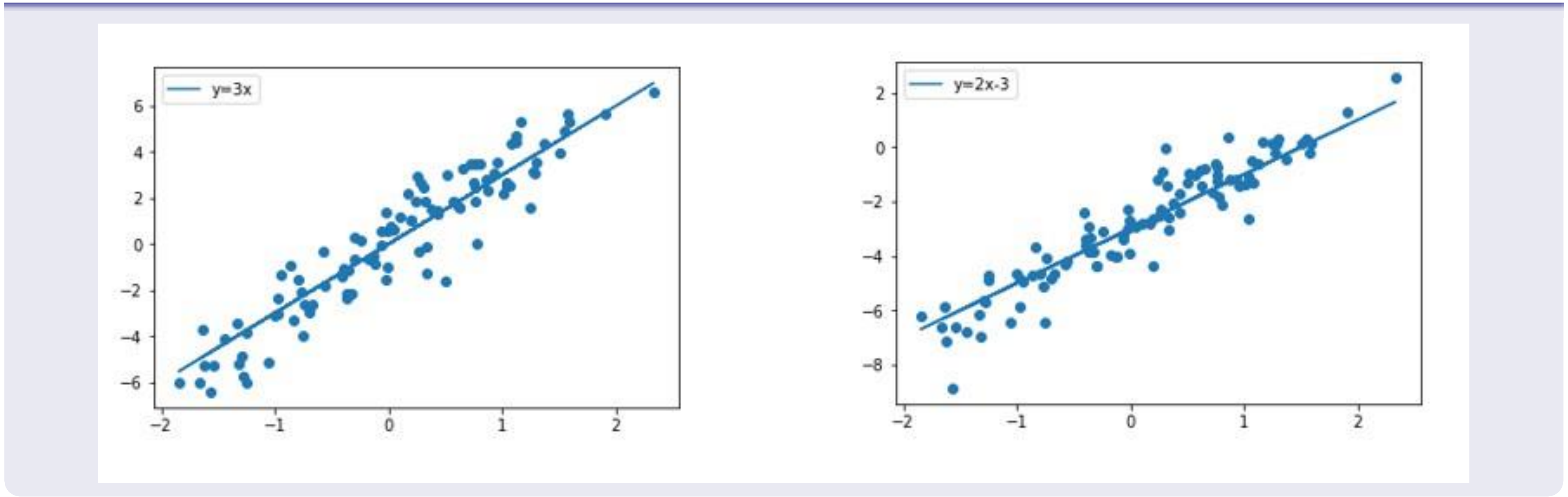
$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \dots$$

- We will look at those in some detail later, but keep in mind the weights (w) are still linear in both instances. We can also write the weights as a vector

$$w = (w_0, w_1, w_2, w_3, w_4, \dots)^T$$

ERRORS

- The first example I put up, had all the data points (training data) fall exactly on the line. This is not how it works really. The prediction is not going to be exactly the same as what we fed into our model to train. Think of the example I put up about the Friends vs Daily Minutes Online
- **Example:**

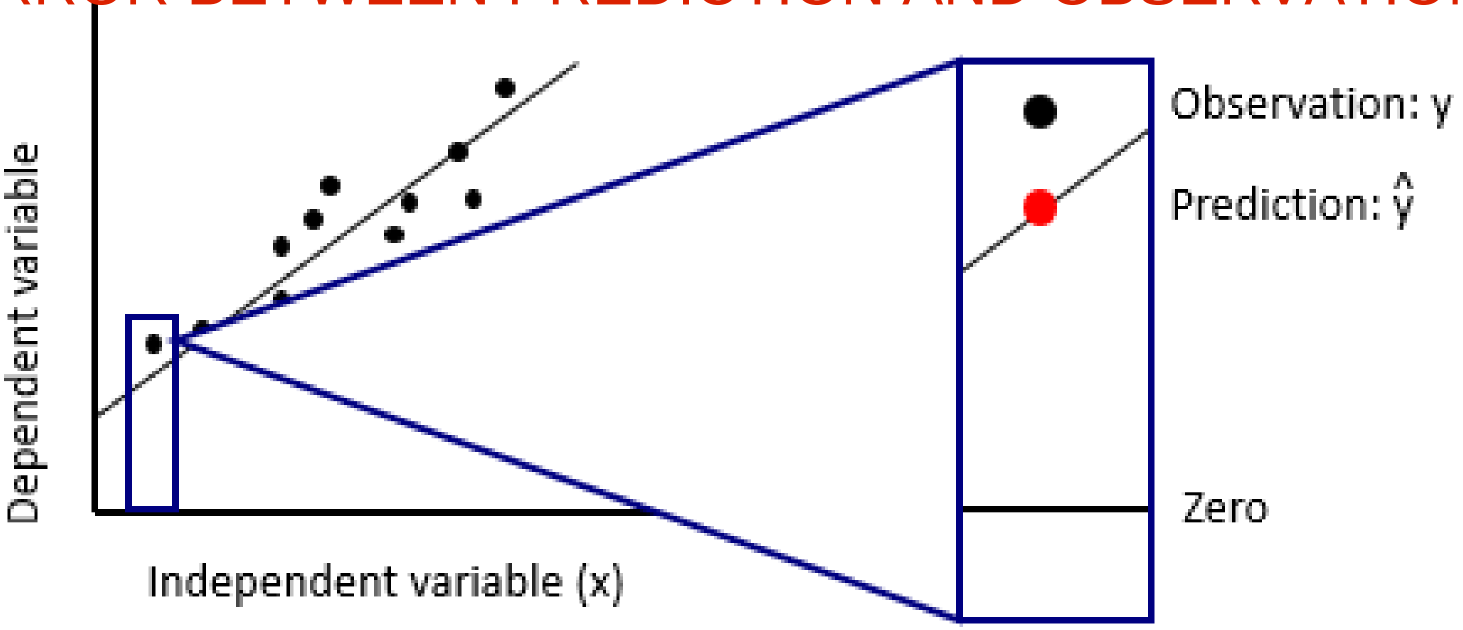


- So, there is some amount of error on the model vs the real data.

SIMPLE LINEAR REGRESSION

- The function will make a prediction for each observed data point.
- The observation is denoted by y and the prediction is denoted by $\hat{y}$.
- The line with the smallest amount of error and that will be the model we want!

ERROR BETWEEN PREDICTION AND OBSERVATION



TRAINING THE MODEL – MINIMISING ERROR

- Training the model means, find the best line that minimises the error for our training sample. This depends on the *Loss/Error* function.
- Measuring the difference between what the model predicts the value should be and what the training sample actually was.
- There are multiple loss functions to choose and choosing the wrong one can be like choosing the wrong type of model.
- Typical example is the mean squared error:

$$L = \frac{1}{2m}\sum_{i=0}^{m}(\hat{y}_i - y_i)^2$$

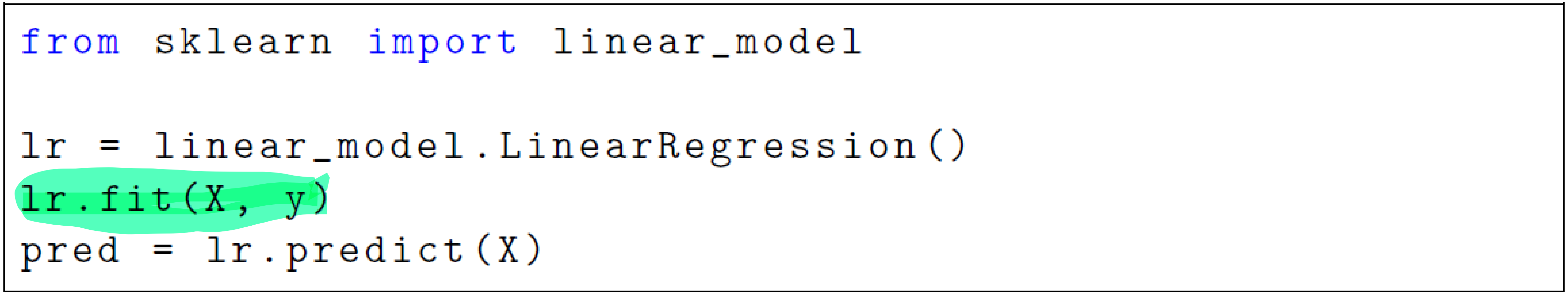
- where *m* is the number of samples, $\hat{y}_i$ is the prediction for the *i*-th sample and y_i is the correct answer for that sample.
- Note: the *1/2m* is just chosen for convenience as the Maths will look “nicer”. Minimising *L* without the fraction will give us the same values for *w*.

TRAINING THE MODEL

- So, what we want are the particular parameters (say w_0 and w_1 in the simplest case we've seen) that give us the smallest value for L.
- This is actually quite easy for the linear regression examples we've had so far. There's a known formula, or we can use some algebra.
- But we want the idea to work for much more complicated models too, and with more parameters.
- The idea does work in further ML algorithms, get the parameters $w_0, w_1, w_2, \ldots$ that give us the smallest L, just the technique for finding the parameters might take a bit longer in more complicated models. This is even true of deep learning (Neural Networks).
- Usually, a method called gradient descent is used to solve this (later!), that does multiple iterations to find the best parameters.
  - When we use the .fit() methods in scikit-learn and Tensorflow this is what happens.

BUILDING A MODEL

- We have a set of data, that is a pair of information. We know the correct answer for the set.
  - X is the set of all inputs (a matrix).
  - y is the set of all corresponding response variables.
  - A particular y_i is the result for a particular x_i.



```
from sklearn import linear_model

lr = linear_model.LinearRegression()
lr.fit(X, y)
pred = lr.predict(X)
```