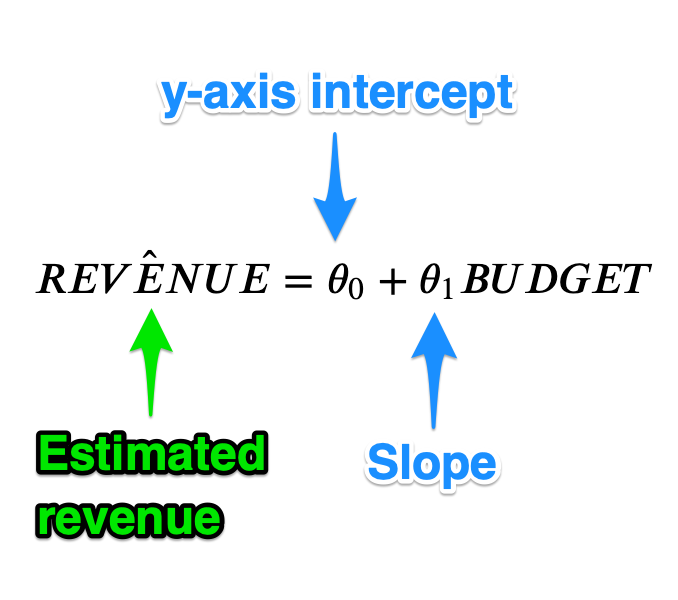
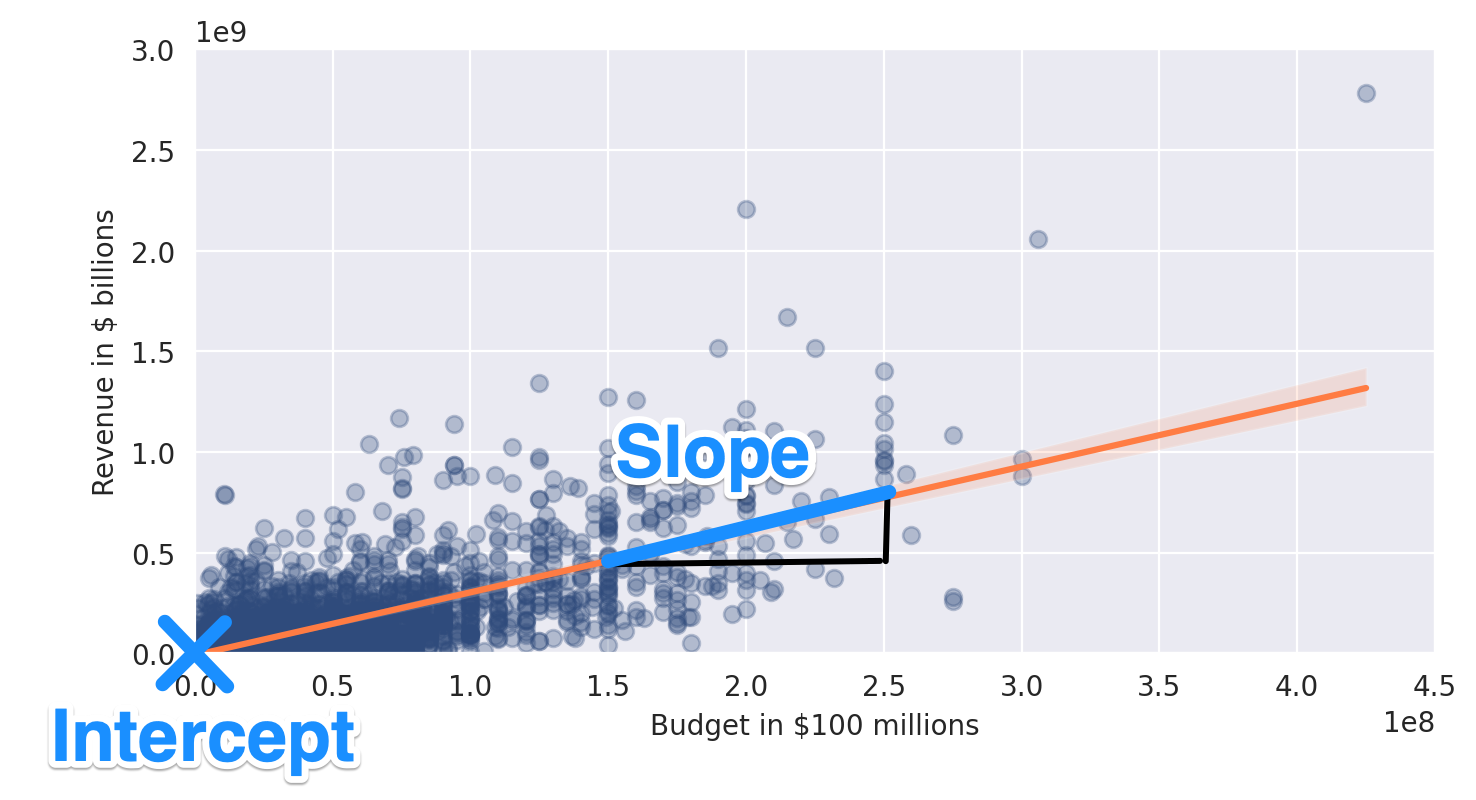
**Use scikit-learn to Run Your Own Regression**

Let's dive into our linear regression model a bit more. We are using a **univariate** regression. This is a regression with a single **explanatory variable** (our movie BUDGET). Explanatory variables are also referred to as **features** in machine learning terminology.

Using our data on budgets, the linear regression estimates the best possible line to fit our movie revenues. The regression line has the following structure:



To find the best possible line, our regression will estimate the y-intercept ("theta zero") and the slope ("theta one"). The line's **intercept** on the y-axis tells us how much revenue a movie would make if the budget was 0. The **slope** tells us how much extra revenue we get for a $1 increase in the movie budget.

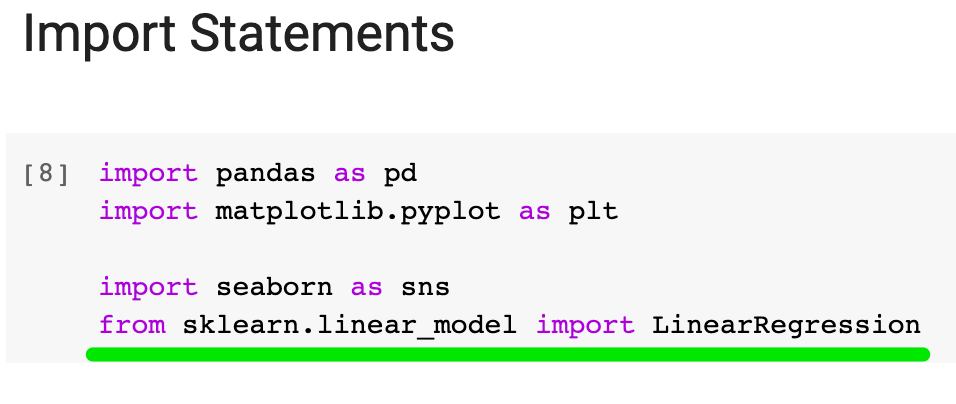


So how can we find out what our model's estimates are for theta-one and theta-zero? And how can we run our own regression, regardless of whether we want to visualise it on a chart? For that, we can use [scikit-learn](https://scikit-learn.org/stable/).



**Import scikit-learn**

Let's add the LinearRegression from scikit-learn to our notebook.



Now we can run a [LinearRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \t "_blank). First, let's create a LinearRegression object that will do the work for us.

1. regression = LinearRegression()

Now we should specify our features and our targets (i.e., our response variable). You will often see the features named capital X and the target named lower case y:

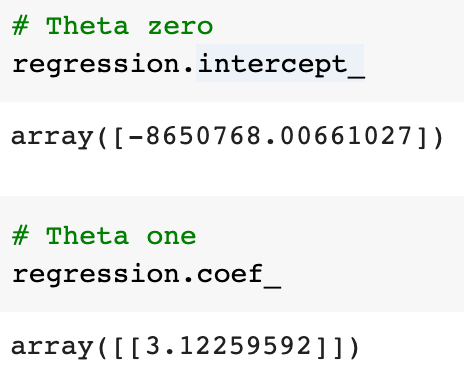
1. # Explanatory Variable(s) or Feature(s)
2. X = pd.DataFrame(new\_films, columns=['USD\_Production\_Budget'])
4. # Response Variable or Target
5. y = pd.DataFrame(new\_films, columns=['USD\_Worldwide\_Gross'])

Our LinearRegression does not like receiving Pandas Series (e.g., new\_films.USD\_Production\_Budget), so I've created some new DataFrames here.

Now it's time to get to work and run the calculations:

1. # Find the best-fit line
2. regression.fit(X, y)

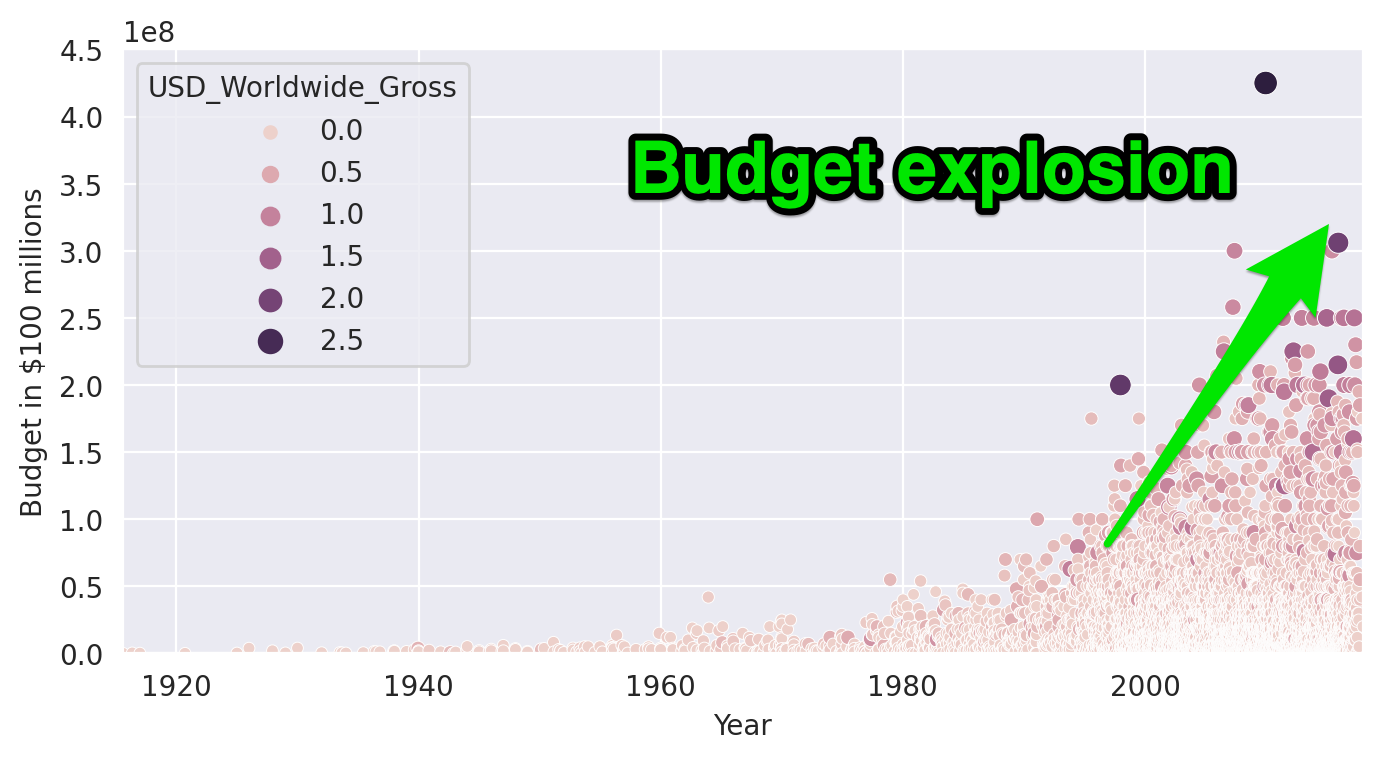
That's it. Now we can look at the values of theta-one and theta-zero from the equation above.



Both intercept\_ and coef\_ are simply [attributes of the LinearRegression object](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html). Don't worry about the underscores at the end, these are simply part of the attribute names that the scikit-learn developers have chosen.

How do we interpret the y-intercept? Literally, means that if a movie budget is $0, the estimated movie revenue is -$8.65 million. Hmm... so this is clearly unrealistic. Why would our model tell us such nonsense? Well, the reason is that we are specifying what the model should be ahead of time - namely a straight line - and then finding the best straight line for our data. Considering that you can't have negative revenue or a negative budget, we have to be careful about interpreting our very simple model too literally. After all, it's just an estimate and this estimate will be the most accurate on the chart where we have the most data points (rather than at the extreme left or right).

What about the slope? The slope tells us that for every extra $1 in the budget, movie revenue increases by $3.1. So, that's pretty interesting. That means the higher our budget, the higher our estimated revenue. If budgets are all that matter to make lots of money, then studio executives and film financiers should try and produce the biggest films possible, right? Maybe that's exactly why we've seen a massive increase in budgets over the past 30 years.



**R-Squared: Goodness of Fit**

One measure of figuring out how well our model fits our data is by looking at a metric called r-squared. This is a good number to look at in addition to eyeballing our charts.

1. # R-squared
2. regression.score(X, y)

We see that our r-squared comes in at around 0.558. This means that our model explains about 56% of the variance in movie revenue. That's actually pretty amazing, considering we've got the simplest possible model, with only one explanatory variable. The real world is *super complex*, so in many academic circles, if a researcher can build a simple model that explains over 50% or so of what is actually happening, then it's a pretty decent model.

Remember how we were quite sceptical about our regression looking at the chart for our old\_films?

Challenge

Run a linear regression for the old\_films. Calculate the intercept, slope and r-squared. How much of the variance in movie revenue does the linear model explain in this case?

.

.

..

...

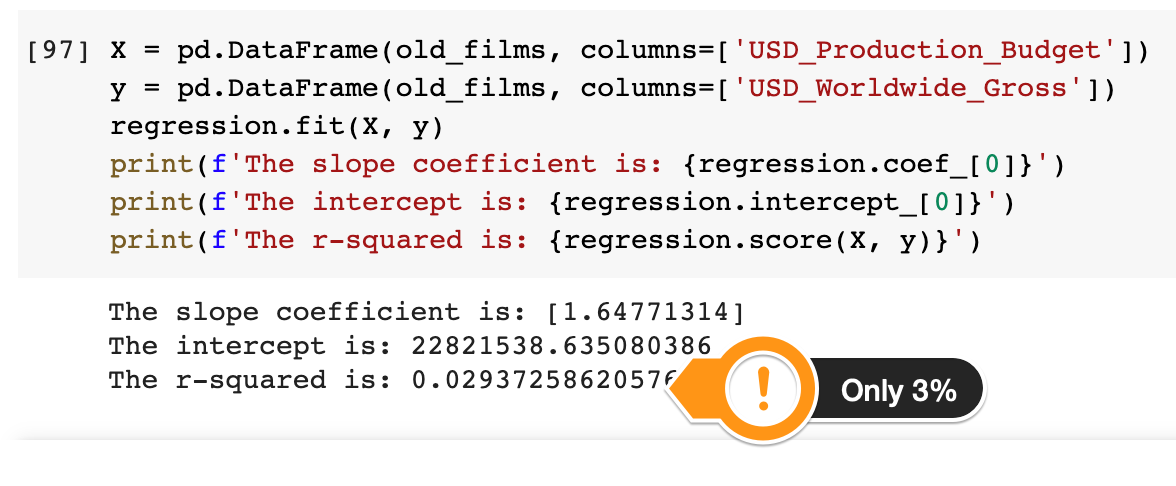
..

.

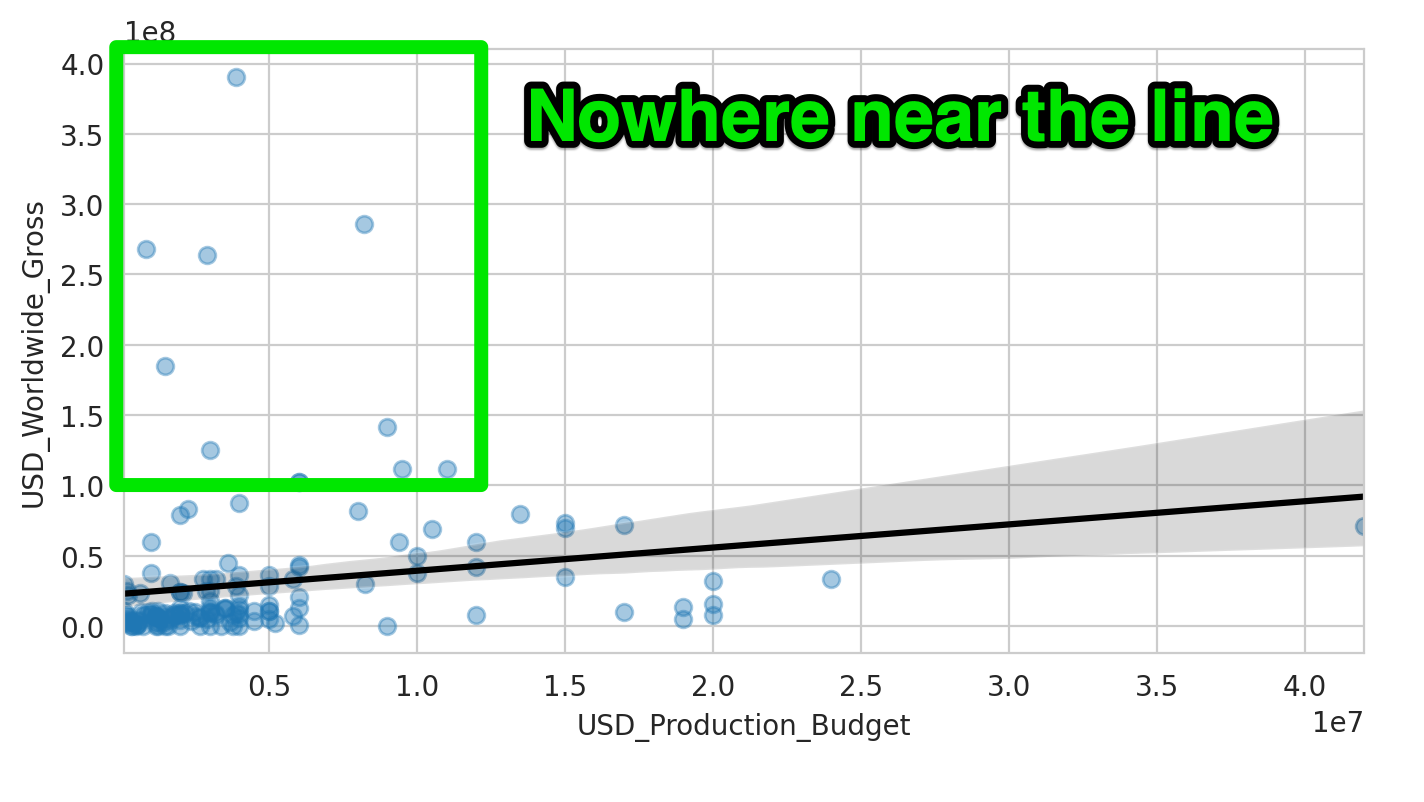
.

**Solution: A bad fit**

Running the numbers this time around, we can confirm just how inappropriate the linear model is for the pre-1970 films. We still see a positive relationship between budgets and revenue, since the slope (our theta-one) is 1.6, but the r-squared is very low.



This makes sense considering how poorly our data points aligned with our line earlier.



Challenge

You've just estimated the intercept and slope for the Linear Regression model. Now we can use it to make a prediction! For example, how much global revenue does our model estimate for a film with a budget of $350 million?

.

.

..

...

..

.

.

**Solution: Using the model to make a prediction**

For a $350 million budget film, our model predicts a worldwide revenue of around $600 million! You can calculate this as follows:

1. 22821538 + 1.64771314 \* 350000000

Or, using the regression object, you could also work it out like this:

1. budget = 350000000
2. revenue\_estimate = regression.intercept\_[0] + regression.coef\_[0,0]\*budget
3. revenue\_estimate = round(revenue\_estimate, -6)
4. print(f'The estimated revenue for a $350 film is around ${revenue\_estimate:.10}.')

(The colon : and dot . in a print statement is quite handy for controlling the number of digits you'd like to show up in the output)