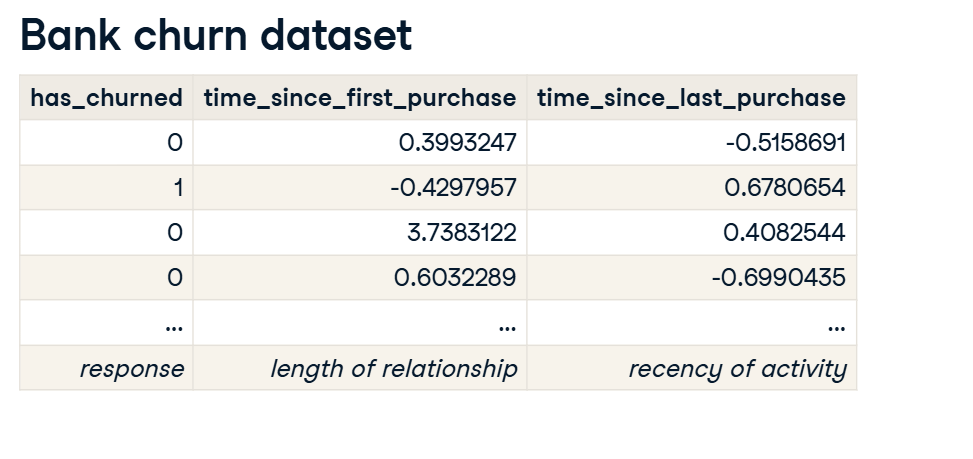
**Why you need logistic regression**

The datasets you've seen so far all had a numeric response variable. Now we'll explore the case of a binary response variable.

**2. Bank churn dataset**

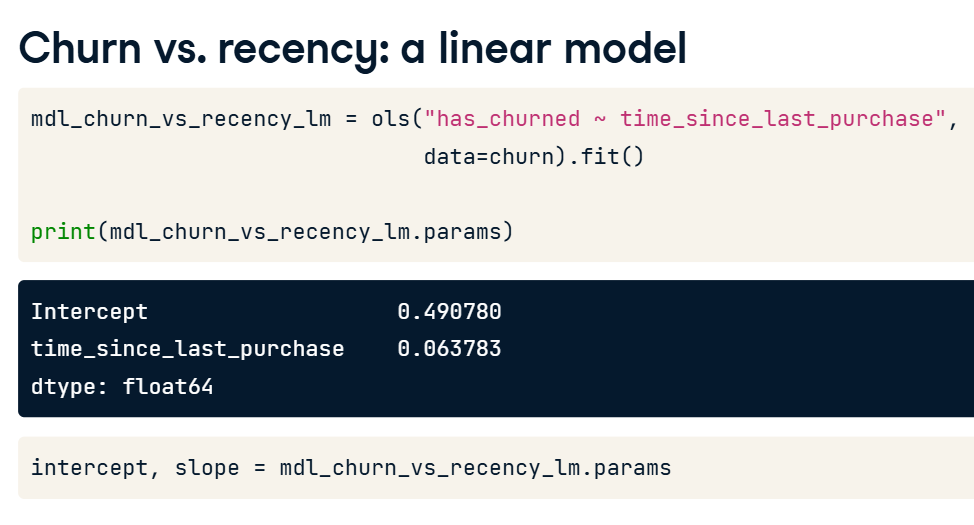
Consider this dataset on churn at a European financial services company in 2006. There are 400 rows, each representing a customer. If the customer closed all accounts during the time period, they were considered to have churned, and that column is marked with a one. If they still had an open account at the end of the time period, has\_churned is marked with a zero. Using one and zero for the response instead of a logical variable makes the plotting code easier. The two explanatory variables are the time since the customer first bought a service and the time since they last bought a service. Respectively, they measure the length of the relationship with the customer and the recency of the customer's activity. The time columns contain negative values because they have been standardized for confidentiality reasons.

1. 1 <https://www.rdocumentation.org/packages/bayesQR/topics/Churn>



**3. Churn vs. recency: a linear model**

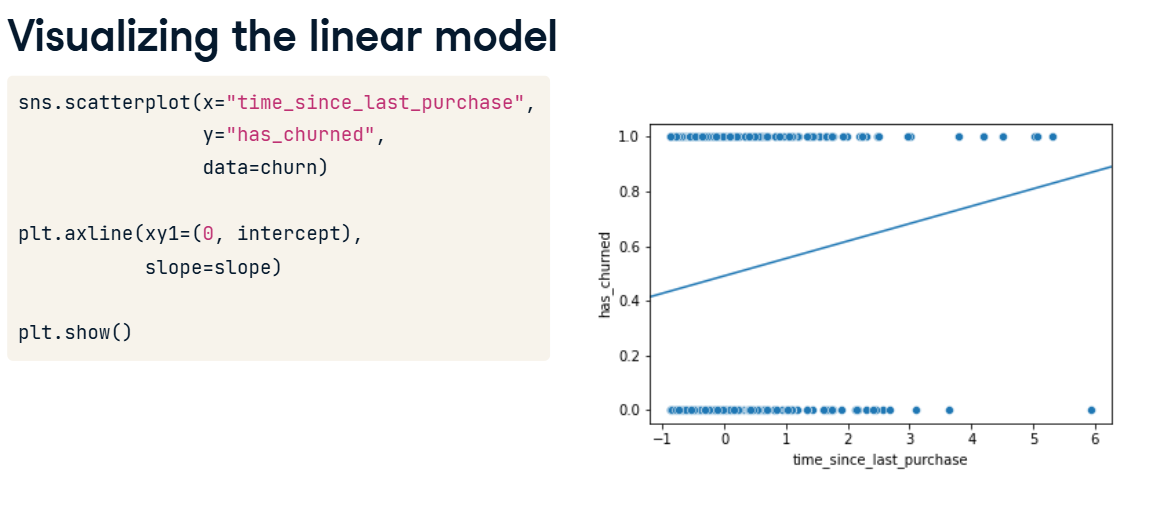
Let's run a linear model of churn versus recency and see what happens. We can use the params attribute to pull out the intercept and slope. The intercept is about point-five and the slope is slightly positive at zero-point-zero-six.



**4. Visualizing the linear model**

01:22 - 01:45

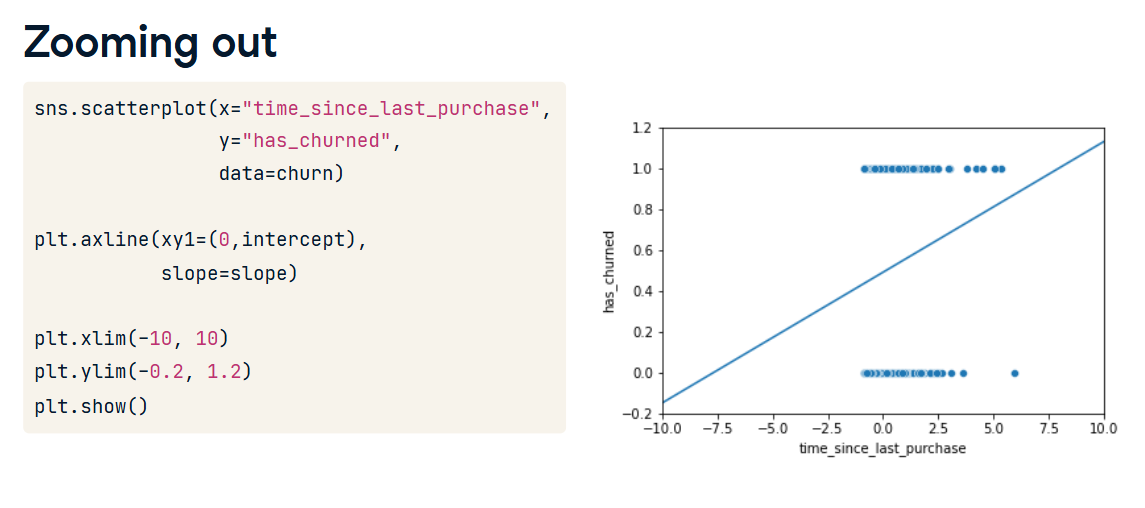
Here's a plot of the data points with the linear trend. I used plt dot axline rather than sns dot regplot so the line isn't limited to the extent of the data. All the churn values are zero or one, but the model predictions are fractional. You can think of the predictions as being probabilities that the customer will churn.



**5. Zooming out**

01:45 - 02:04

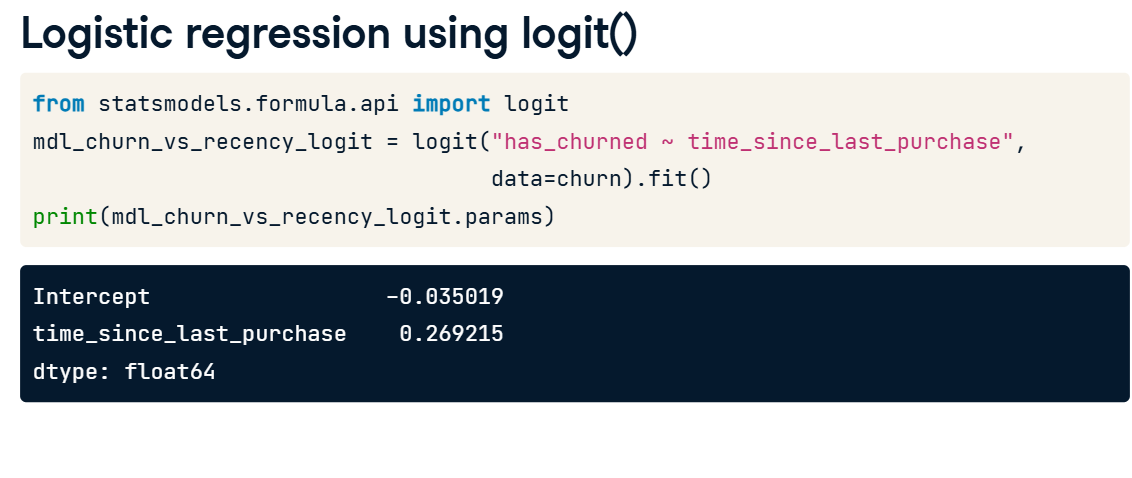
Zooming out by setting axis limits with xlim and ylim shows the problem with using a linear model. In the bottom-left of the plot, the model predicts negative probabilities. In the top-right, the model predicts probabilities greater than one. Both situations are impossible.



**6. What is logistic regression?**

02:04 - 02:25

The solution is to use logistic regression models, which are a type of generalized linear model, used when the response variable is logical. Whereas linear models result in predictions that follow a straight line, logistic models result in predictions that follow a logistic curve, which is S-shaped.



**7. Logistic regression using logit()**

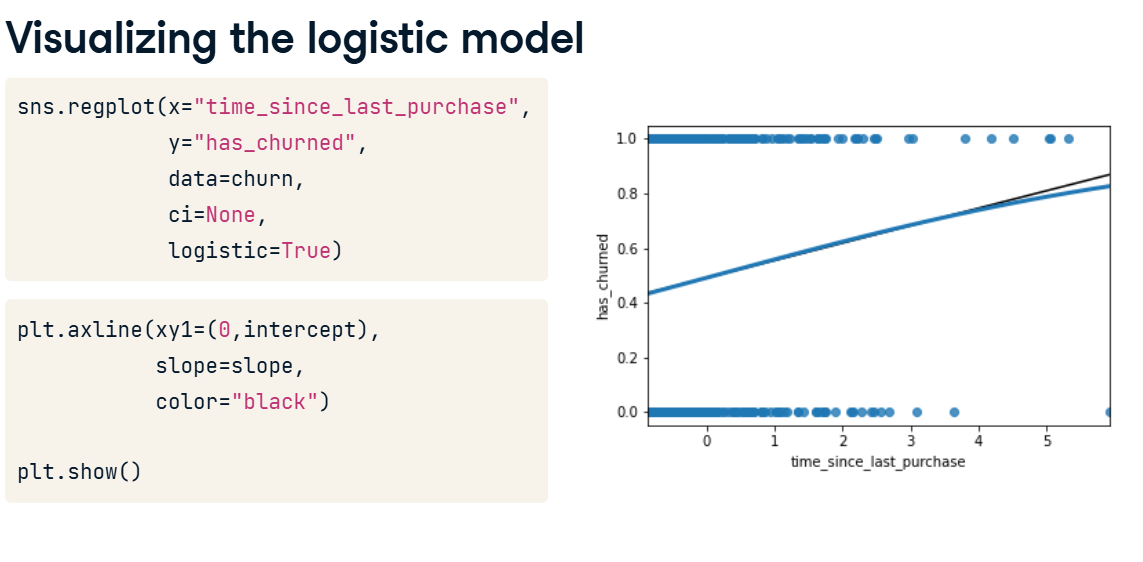
02:25 - 03:06

To run a logistic regression, you need a new function from statsmodels. From the same statsmodels dot formula dot api package, import the logit function. This function begins the process of fitting a logistic regression model to your data. The function name is the only difference between fitting a linear regression and a logistic regression: the formula and data argument remain the same, and you use the dot fit method to fit the model. As before, you get two coefficients, one for the intercept and one for the numerical explanatory variable. The interpretation is a little different; we'll come to that later.

**8. Visualizing the logistic model**

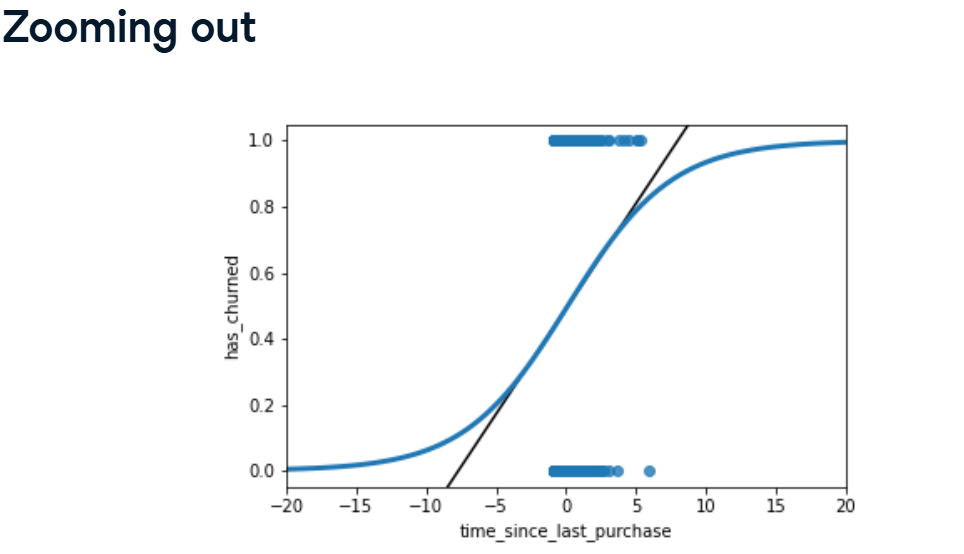
03:06 - 03:31

Let's add the logistic regression predictions to the plot. regplot will draw a logistic regression trend line when you set the logistic argument to True. Notice that the logistic regression line, shown in blue, is slightly curved. Especially when there's a longer time since the last purchase values, the blue trend line no longer follows the black, linear trend line anymore.



**9. Zooming out**

Now zooming out shows that the logistic regression curve never goes below zero or above one. To interpret this curve, when the standardized time since last purchase is very small, the probability of churning is close to zero. When the time since last purchase is very high, the probability is close to one. That is, customers who recently bought things are less likely to churn.



**Predictions and odds ratios**

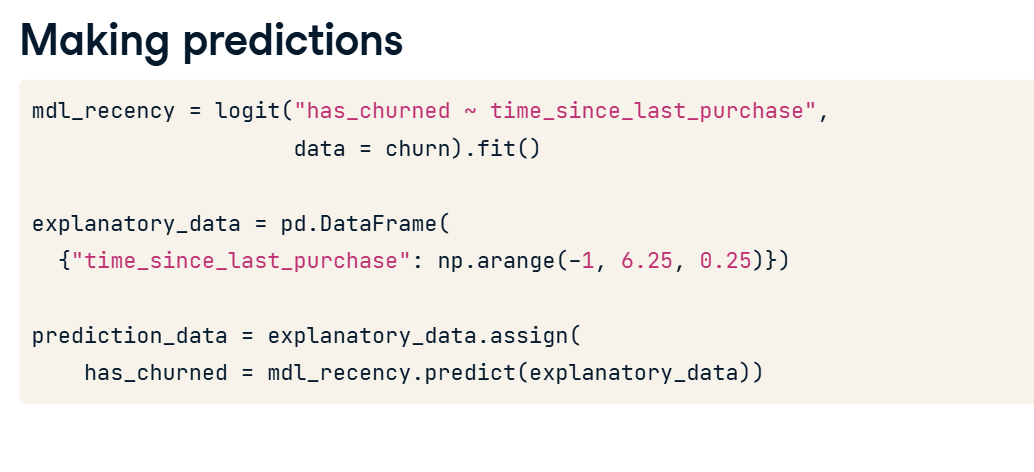
Let's see how to make predictions with your logistic regression model.

**The regplot() predictions**

You've already seen how regplot will give you a logistic regression trend line.

**Making predictions**

To make a prediction with a logistic model, you use the same technique as for linear models. Create a DataFrame of explanatory variable values. Then add a response column calculated using the predict method.

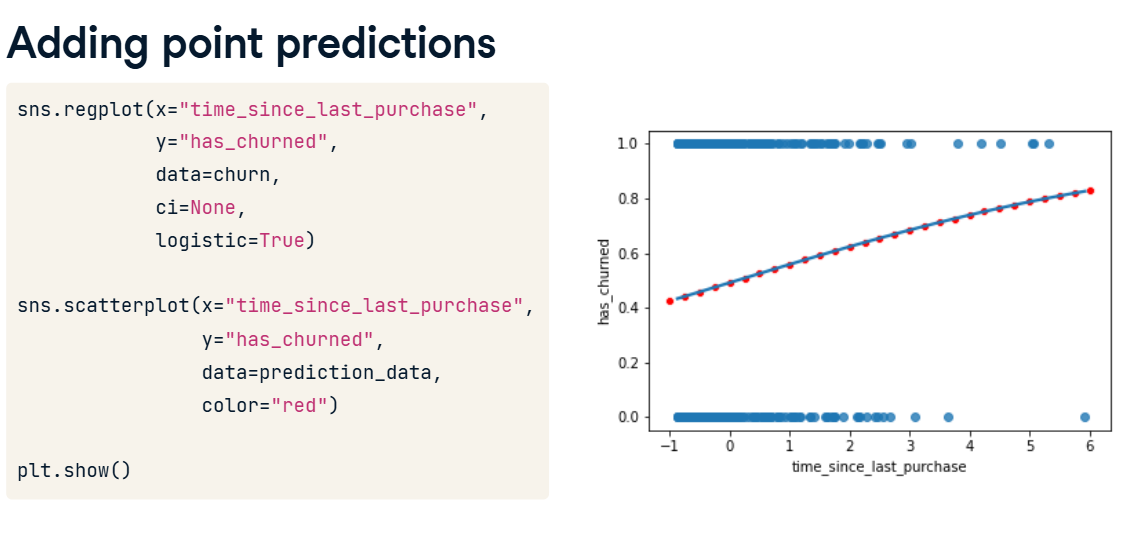


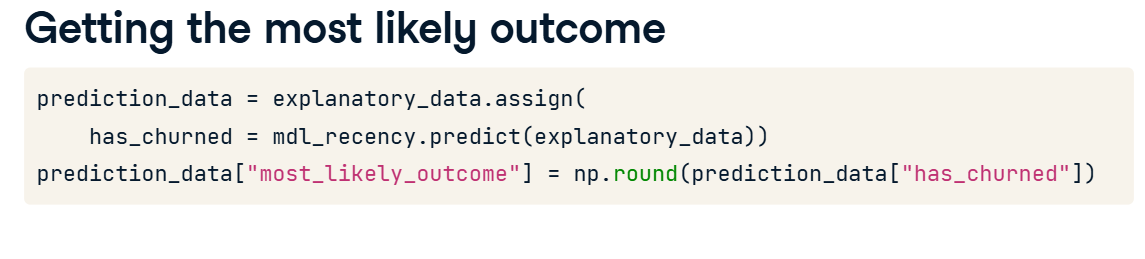
**Adding point predictions**

As with the linear case, we can add those predictions onto the plot by creating a scatter plot with prediction\_data as the data argument. As expected, these points follow the trend line.

**Getting the most likely outcome**

One simpler prediction you can make, rather than calculating probabilities of a response, is to calculate the most likely response. That is, if the probability of churning is less than 0-point-5, the most likely outcome is that they won't churn. If their probability is greater then 0-point-5, it's more likely that they will churn. To calculate this, simply round the predicted probabilities using numpy's round() function.





**Visualizing most likely outcome**

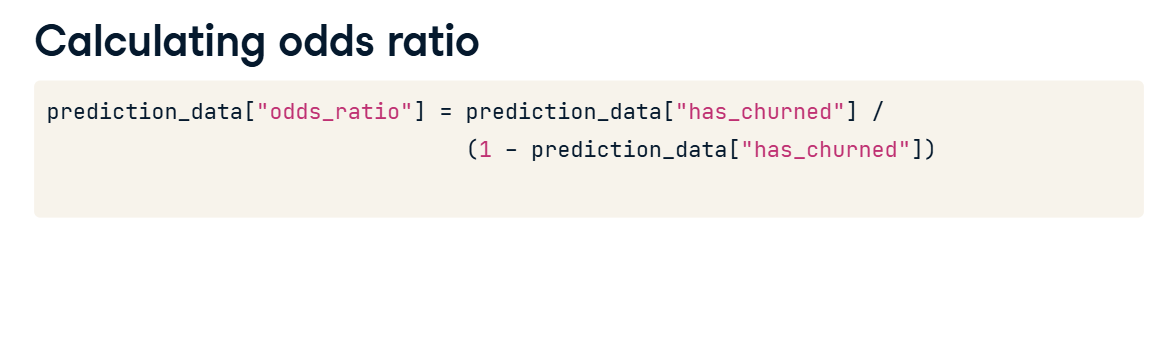
We can plot the most likely outcome by using the prediction data with the numbers we just calculated. For recently active customers, the most likely outcome is that they don't churn. Otherwise, the most likely outcome is that they churn.

**Odds ratios**

There is another way to talk about binary responses, commonly used in gambling. The odds ratio is the probability that something happens, divided by the probability that it doesn't. For example, a probability of zero-point-two-five is the same as the odds of "three to one against", because the probability of the event not happening is zero-point-seven-five, which is three times as much. The plot shows the relationship between the two terms.

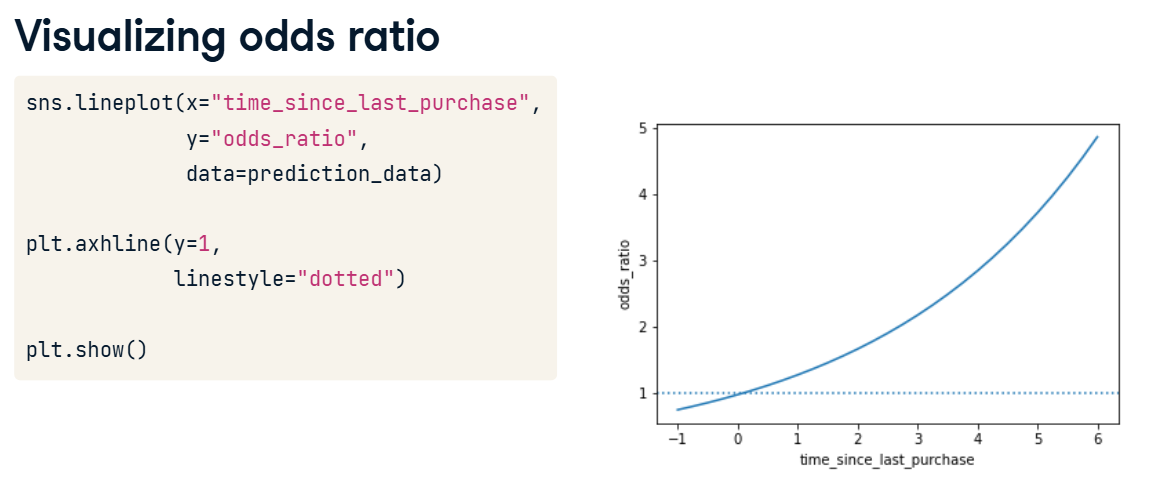
**Calculating odds ratio**

We can calculate the odds ratio by dividing the predicted response probability by one minus that number.



**Visualizing odds ratio**

It doesn't make sense to visualize odds with the original data points, so we need a new plot. To create a plot with a continuous line, we can use seaborn's lineplot function. Here, the dotted line where the odds ratio is one indicates where churning is just as likely as not churning. This has been added by using the axhline function. In the bottom-left, the predictions are below one, so the chance of churning is less than the chance of not churning. In the top-right, the chance of churning is about five times more than the chance of not churning.



**Visualizing log odds ratio**

One nice property of logistic regression odds ratios is that on a log-scale, they change linearly with the explanatory variable. This plot adds a logarithmic y scale.

**Calculating log odds ratio**

This nice property of the logarithm of odds ratios means log-odds ratio is another common way of describing logistic regression predictions. In fact, the log-odds ratio is also known as the logit, hence the name of the function you've been using to model logistic regression.

**All predictions together**

Here are all the values calculated in the prediction dataset. Some column names are abbreviated for better printing.

**Comparing scales**

Each way of describing responses has different benefits. Most likely outcome is easiest to understand because the answer is always yes or no, but this lacks precision. Probabilities and odds ratios are still fairly easy to understand for a data literate audience. However, the non-linear predictions make it hard to reason about how changes in the explanatory variable will change the response. Log odds ratio is difficult to interpret for individual values, but the linear relationship with the explanatory variables makes it easy to reason about changes.

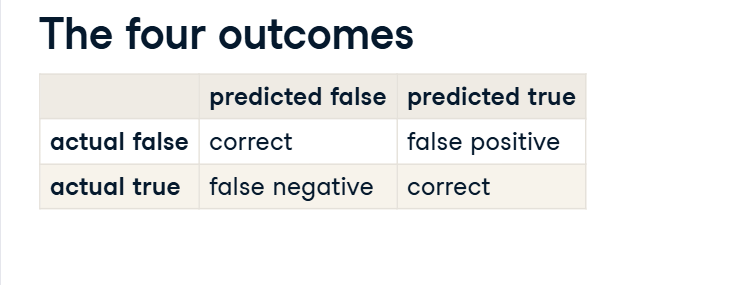
**Quantifying logistic regression fit**

In this last lesson, we'll assess the performance of logistic regression models. The diagnostic plots we drew for linear models are less useful in the logistic case. Instead, we'll look at confusion matrices.

**2. The four outcomes**

00:15 - 00:46

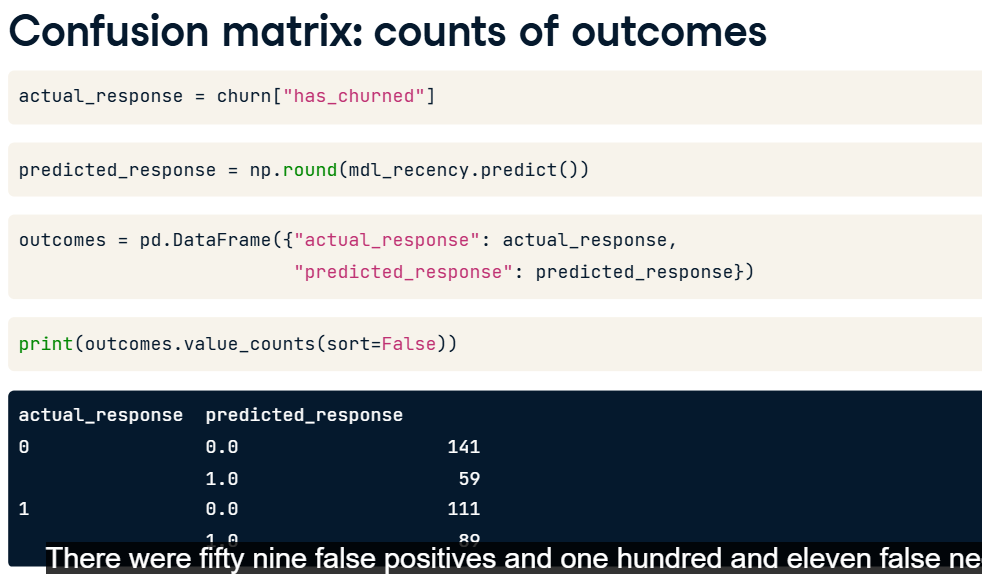
A logical response variable leads to four possible outcomes. If the customer didn't churn and we predicted they wouldn't, or if they did churn and we predicted that, the model did well. There are two bad cases. Predicting the customer churned when they didn't is called a false positive. Predicting the customer didn't churn when they did is called a false negative. The counts of each outcome are called a confusion matrix.



**3. Confusion matrix: counts of outcomes**

00:46 - 01:47

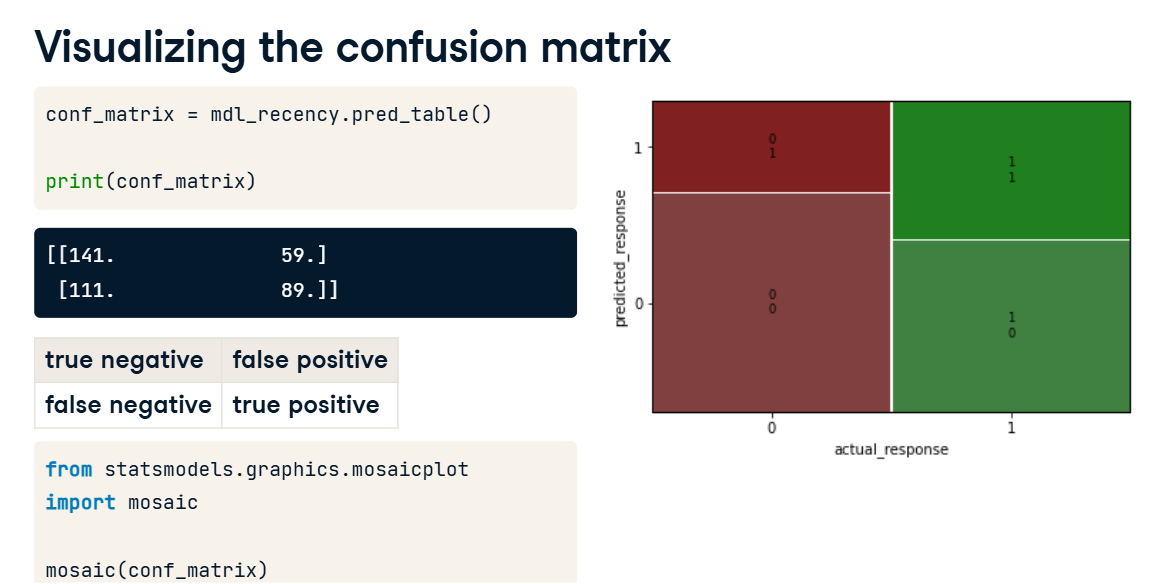
Recall the model of churn versus recency. Getting the counts of model outcomes required some data manipulation. First, we get the actual responses from the has\_churned column of the dataset. Next we get the predicted responses from the model. Calling the predict method on the fitted logistic regression model returns the predicted values of each observation in the dataset. These predicted values are probabilities. To get the most likely outcome, we need to round the values to zero or one. We then combine actual and predicted responses in a DataFrame, and use the value\_counts method to get the counts of each combination of values. This is the confusion matrix mentioned earlier. We correctly predicted that one hundred and forty one customers didn't churn and eighty nine customers did churn. There were fifty nine false positives and one hundred and eleven false negatives.



**4. Visualizing the confusion matrix**

01:47 - 02:56

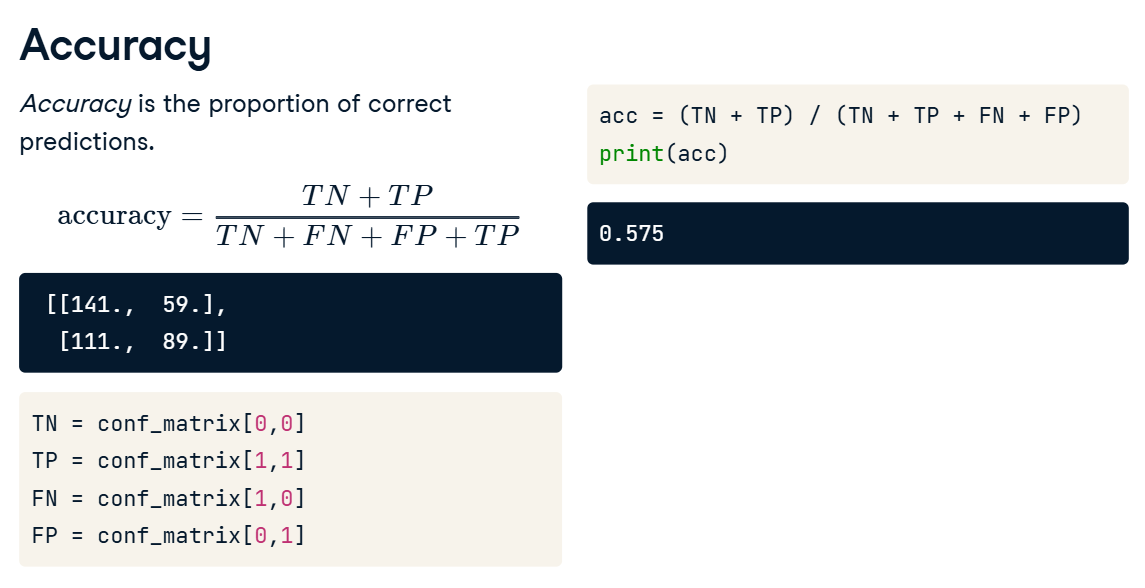
The confusion matrix can also be created automatically with the pred\_table method. Calling pred\_table on the fitted model object will return an array. The true negatives and true positives are on the main diagonal of the matrix, the false negatives and false positives are on the second diagonal of the matrix. These values are the same as what we calculated on the previous slide. The mosaic function from the statsmodels package lets you easily plot the confusion matrix. To interpret this, start by looking at the column widths. The width of each column is proportional to the fraction of observations in each category of actual values. Here, there are two hundred actual churns and two hundred actual not churns, so each column has the same width. Then each column displays the fraction of predicted observations with each value. Here, just over a quarter of the actual not churns were predicted to be churns, so the block in the upper left is just over a quarter of the height of the first column.



**5. Accuracy**

02:56 - 03:30

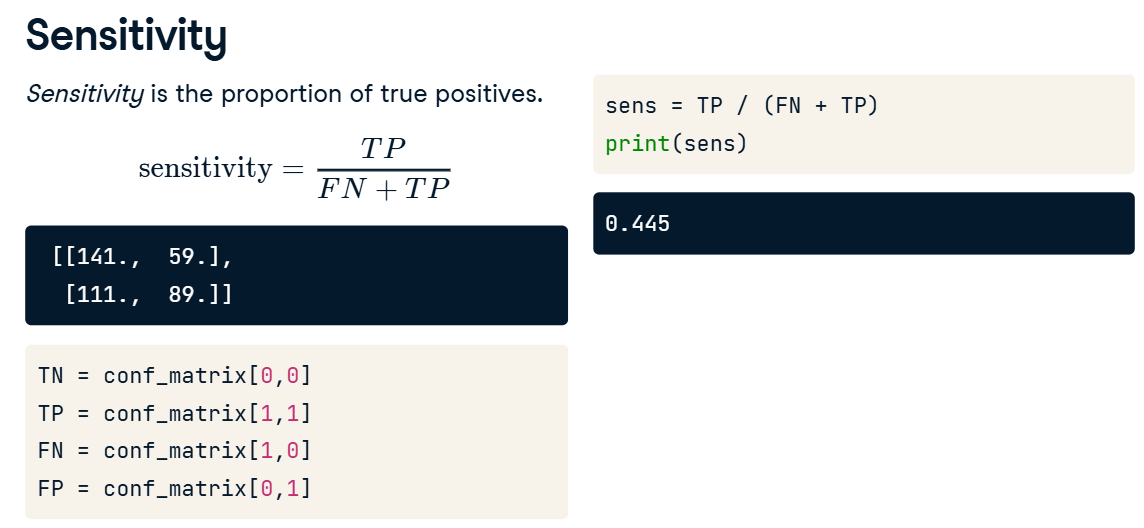
Now let's look at ways of quantifying model fit using performance metrics. The first metric is the model accuracy. This is the proportion of correct predictions. That is, the number of true negatives plus the true positives, divided by the total number of observations. Higher accuracy is better. The total number of correct observations is one hundred and forty one plus eighty nine. We divide this total by the total number of observations, which is the sum of all four numbers.



**6. Sensitivity**

03:30 - 03:58

The second metric is sensitivity. This is the proportion of observations where the actual response was true where the model also predicted that they were true. That is, the number of true positives divided by the sum of the false negatives and true positives. Higher sensitivity is better. Here, eighty nine of the two hundred customers who churned were correctly predicted to churn.



**7. Specificity**

03:58 - 04:33

The third metric is specificity. This is the proportion of observations where the actual response was false where the model also predicted that they were false. That is, the number of true negatives divided by the sum of the true negatives and false positives. Again, higher specificity is better, though there is often a trade-off where improving specificity will decrease sensitivity, or increasing sensitivity will decrease specificity. Here, one hundred and forty one of the two hundred customers who didn't churn were correctly predicted to not churn.

