. Predicting data over time

In the third chapter we'll shift our focus from classification to regression. Regression has several features and caveats that are unique to timeseries data. We'll begin by visualizing and predicting timeseries data. Then, we'll cover the basics of cleaning the data, and finally, we'll begin extracting features that we can use in our models.

2. Classification vs. Regression

The biggest difference between regression and classification is that regression models predict continuous outputs whereas classification models predict categorical outputs. In the context of timeseries, this means we can have more fine-grained predictions over time.

3. Correlation and regression

Both Regression and correlation reflect the extent to which the values of two variables have a consistent relationship (either they both go down or up together, or they have an inverse relationship). However, regression results in a "model" of the data, while correlation is just a single statistic that describes the data. Regression models have more information about the data, while correlation is easier to calculate and interpret.

4. Correlation between variables often changes over time

When running regression models with timeseries data, it's important to visualize how the data changes over time. You can either do this by plotting the whole timeseries at once, or by directly comparing two segments of time.

5. Visualizing relationships between timeseries

Here we show two ways to compare timeseries data. On the left, we'll make two line plots with the x-axis encoding time. On the right, we'll make a single scatterplot, with color encoding time.

6. Visualizing two timeseries

Here is the visualization. In this case, it seems like these two timeseries are uncorrelated at first, but then move in sync with one another. We can confirm this by looking at the brighter colors on the right. We see that brighter datapoints fall on a line, meaning that for those moments in time, the two variables had a linear relationship.

7. Regression models with scikit-learn

Fitting regression models with scikit-learn works the same way as classifiers - the consistency in API is one of scikit-learn's greatest strengths. There are, however, a completely different subset of models that accomplish regression. We'll begin by focusing on LinearRegression, which is the simplest form of regression. Here we see how you can instantiate the model, fit, and predict on training data.

8. Visualize predictions with scikit-learn

Here we visualize the predictions from several different models fit on the same data. We'll use Ridge regression, which has a parameter called "alpha" that causes coefficients to be smoother and smaller, and is useful if you have noisy or correlated variables. We loop through a few values of alpha, initializing a model with each one and fitting it on the training data. We then plot the model's predictions on the test data,

9. Visualize predictions with scikit-learn

which lets us see what each model is getting right and wrong. For more information on Ridge regression, refer to DataCamp's introductory course on scikit-learn.

10. Scoring regression models

Visualizing is useful, but not quantifiable. There are several options for scoring a regression model. The simplest is the correlation coefficient, whereas the most common is the coefficient of

determination, or R squared.

11. Coefficient of Determination (R^2)

The coefficient of determination can be summarized as the total amount of error in your model (the difference between predicted and actual values) divided by the total amount of error if you'd built a "dummy" model that simply predicted the output data's mean value at each timepoint. You subtract this ratio from "1", and the result is the coefficient of determination. It is bounded on top by "1", and can be infinitely low (since models can be infinitely bad).

12. R^2 in scikit-learn

In scikit-learn, we can import the r2_score function which calculates the coefficient of determination. It takes the predicted output values first, and the "true" output values second, to calculate r-square.