M6-04: Linear Regression in Python

Part of the "Towards Machine Learning" Learning Badge

Video Walkthrough: https://discovery.cs.illinois.edu/m6-04/

Building a Linear Regression in Python

All models in the scikit-learn (sklearn) Python library follow the same basic design. The simplest model is the linear regression model:

Step 1: Initialize the Machine Learning Model

The first step is to always create an instance of the machine learning model, which provides Python an object to use to store the parameters of the model. For all models, we must import the package and then initialize the model:

Python:	`# Import sklearn's LinearRegression model:`
	`from sklearn.linear_model import LinearRegression`
	
	`# Create a new instance of a LinearRegression model:`

Description:	Imports and creates an instance of sklearn's linear regression model.

Step 2: Train the Model using `.fit(...)`

All machine learning models require training, and sklean uses the fit function that "fits" the model to the data by training the model to the data. Linear regression is a supervised learning model, so we **must** give it:

- A list of input data columns (we call these the "independent variables")
- The known output for each input (we call this the "dependent variable")

The most obvious prediction of price would be the size of the diamond. To create a linear regression model that uses the size of the diamond (carat weight) to predict the price, we know that:

- The list of independent variable column names is: ____________________

- The dependent variable column name is: ____________________

Writing this in Python:

Python:	
Description:	Fits the linear regression model with the data in the DataFrame df.

Step 3: Use the Model using `.predict(...)`

All machine learning models will predict an expected output value when given input values -- sklean uses the predict function to produce the output from any model. In all models, a list of all independent variables that you want to predict is required and the model will provide a predicted value for the dependent variable.

In our model, our list of independent variables was simple: `["carat"]`

To predict the price of a 1ct diamond, our input of independent variables must be: `[1]`, representing that we want to predict the price of a single 1ct diamond. The predict function requires a list of all the diamonds we want to predict, resulting in the following Python:

Python:	`model.predict( [ [1] ] )`
Description:	Uses the machine learning model to predict the price of a 1ct diamond.

Puzzle #1: What is the prediction of a 2 carat diamond?

Python:	
Description:	Uses the machine learning model to predict the price of a 2ct diamond.

Puzzle #2: What is the prediction of a **20** carat diamond?

Python:	
Description:	Uses the machine learning model to predict the price of a 20ct diamond.

Analysis: Is this answer reasonable? What's the real price of a 20ct diamond?

What is the reason for this?

M6-04: Linear Regression in Python

Part of the "Towards Machine Learning" Learning Badge

Video Walkthrough: https://discovery.cs.illinois.edu/m6-04/

Machine Learning and DataFrames

Once a model is trained, it is extremely useful to use the trained model to predict the value on a DataFrame. Using the DataFrame **df**:

Python:	`model.predict( df[ ["carat"] ] )`
Description:	Predicts the price of every diamond in the dataset, when a model has been trained using the independent variable list of ["carat"].

It's most useful when we add this prediction to our DataFrame:

Python:	`df["price_predict"] = model.predict( df[ ["carat"] ] )`
Description:	Adds a new column to our DataFrame, "price_predict", that contains the model-predicted price of every diamond in the dataset.

Puzzle #1:

When using a model on the dataset itself, this allows us to calculate a **prediction error** -- or the amount the machine learning model was off in its prediction. Given that the true price in stored in the variable "price" and we just added a new column "price_predicted", calculate the error for each diamond:

Python:	
Description:	Calculates the error between the "price" and the "price_predicted", adding the error as a new column called ____________________.

Analysis: Is the prediction too high, too low, or both?

What is the total prediction error across all diamonds?