Comma Separated Files

This lesson focuses on CSV type files. It gives a complete explanation about how to read data from CSV files using the Pandas library of Python.

WE'LL COVER THE FOLLOWING

Introduction to CSV file

Comma-separated files (CSV) are common in machine learning. These files have a **row** of data per line of the file and each line is a *comma-separated list* in which each element is a **column**. Pandas makes it easy to read this data.

Reading CSV file with Pandas

The documentation can be found here. Before reading a CSV file, there are three parameters that should be known:

- `sep` - this defaults to a comma, but we can specify anything we want. For example, CSV format is poor if some of your columns contain commas. A better option might be a `|`.
- `header` - which row (if any) have the column names.
- `names` - column names to use.

If your CSV is well-formatted where the first row is the column names, then the default parameters should work well.

It is important to note that while it might sound simple to read in a CSV file without Pandas, CSV files are often very messy and reading them appropriately can often consist of handling many edge cases. Pandas module handles many of those edge cases right out of the box and has many parameters that you can change to handle messier CSV files.

parameters that you can change to handle messier CSV files.

Let's see an example with code.

```
import pandas as pd

# Define the column names as a list
names = ['age', 'workclass', 'fnlwgt', 'education', 'educationnum', 'maritalstatus', 'occupat
        'sex', 'capitalgain', 'capitalloss', 'hoursperweek', 'nativecountry', 'label']
# Read in the CSV file from the webpage using the defined column names
df = pd.read_csv("adult.data", header=None, names=names)

print(df.head())
```

Let's deconstruct the code. First, we have a CSV of data that lives at this page.

Go to the page, click on **Data Folder** and then select **adult.data** file. It will download the small dataset for you. Once downloaded, open it in Excel or any text editor. You will see rows of data with each column separated by commas.

You will notice that this CSV doesn't have column names. Fortunately, we know what the names should be and supplied them to the `names` parameter at **line 2**. Since Pandas assumes the first row is the `header` (columns), `header=None` was used at **line** 7, so it would read the first row as data.

The data is read into a Pandas dataframe.

A Pandas dataframe is very similar to a matrix of data, but with some additional benefits (usually at the cost of performance). For example, you get named columns and rows as well as the ability to store different types of data in each column. For example, one column could be integers and another text.

You might imagine a dataframe as an Excel sheet of data. For example, if you had a sheet with a column of dates and a column of temperatures on those dates, you could easily represent it as a dataframe.

We can see the first 5 rows of data by calling the `head()` function on the dataframe. In the next section on describing data, we will go more in-depth on how to use a dataframe.

Next, let's look at reading in JSON files.