Hyperiondev

TASK

Datasets and DataFrames

Visit our website

Introduction

WELCOME TO THE DATASETS AND DATAFRAMES TASK!

In the context of this task, when we refer to a dataset, we are referring to a collection of related data. This data can be manipulated in various ways programmatically. In this task, you will be using pandas DataFrames to manipulate data.

IMPORTING DATA USING PANDAS

Recall you can create a pandas DataFrame but passing in a NumPy array or dictionary of objects, however most of the time you will import existing data from a file. Remember you can read data from a .csv file into a DataFrame using the `read_csv()` function as shown below:

```
insurance_df = pd.read_csv("insurance.csv")
```

There are also other functions that can be used to read data from different sources into a pandas DataFrame. For example, **`read_excel()`** can be used to read data from an Excel spreadsheet file and **`read_sql()`** can be used to load data from a SQL database. Sometimes it is easier to extract data from other sources into a .csv file and then read it into a DataFrame.

SELECTING COLUMNS IN PANDAS

There are many ways to specify columns in pandas. The simplest way is to use dictionary notation for specific columns. In essence, pandas Dataframes can be thought of as dictionaries. The key is the column name and the value is the corresponding column values.

```
# Select only the species column
just_the_species = iris_df['species']
just_the_species.sample(5)
```

To select multiple columns, you simply need to specify a list of strings with each column name:

```
# Select columns with sepal and petal information
sepal_and_petal_info = iris_df[['sepal_length', 'sepal_width',
'petal_length', 'petal_width']]
sepal_and_petal_info.sample(5)
```

You can also choose specific values to be included in your search (i.e. omit certain rows from the results).

```
# Filter for specific values in a column
small_sepal_length = iris_df[iris_df['sepal_length'] < 4.8]
small_sepal_length.sample(5)
```

In essence, we are filtering the dataset for all entries where the **sepal_length** is less than 4.8.

BUILT-IN DATAFRAME METHODS

When attempting to gain insight into your data, it is often useful to leverage built-in methods to process your data. For example, finding the mean or total of a column.

Here is a list of common build-in computational and statistical methods in pandas:

- **mean()**: mean for each column
- **min()**: minimum for each column
- **max()**: maximum for each column
- **std()**: standard deviation for each column
- **var()**: variance for each column
- **nunique()**: number of unique values in each column
- **count()**: number of cells for each column or row that are not empty or undefined (e.g. NaN)
- **sum()**: sum of values for each column or row

For a list of all methods relating to general computations and descriptive statistics, explore the **pandas documentation**.

GROUPING IN PANDAS

Data analysis can sometimes get complicated, and more advanced functionality is needed. Let's say you want to average the insurance charges of all people between the ages of 30 and 35. This can be done quite easily using:

```
# Get people in the 30-35 age group
between_30_and_35 = insurance_df[(insurance_df['age'] > 30) &
(insurance_df['age'] < 35 )]
```

```
# Print mean charges for all people in the 30-35 age group
print(between_30_and_35['charges'].mean())
```

Alternatively, you can also use the pandas `DataFrame.query()` method which takes boolean strings as an argument as shown below:

```
# Use the query method to get people in the 30-35 age group
between_30_and_35 = insurance_df.query("age > 30 and age < 35")

# Print mean charges for all people in the 30-35 age group
print(between_30_and_35['charges'].mean())
```

Now let's say you want to average the insurance charges of every person in each age group. This can still be done with the syntax you know, but it will take a lot of lines of code. This is bad because we want to keep our code simple and concise. Thankfully, pandas provides us with something that allows us to do this with one line of code:

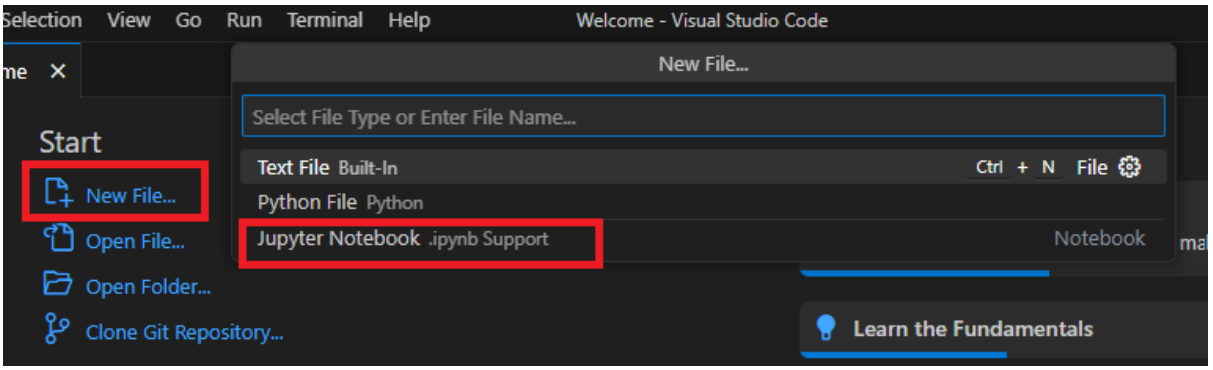
```
# Get the mean charges for each age
print(insurance_df.groupby('age')['charges'].mean())
```

This `groupby()` method tells the aggregation to work separately on each unique group specified.

JUPYTER NOTEBOOK

For the data science tasks in this bootcamp you will use Jupyter Notebook, an IDE often used by data scientists. The notebooks generated using Jupyter Notebook can contain executable code, text, and visualisations. Combining all these elements allows you to include rich explanations about data analysis and model building alongside the code that is producing the outputs. The notebooks are also easy to share with others in non-technical teams in formats such as PDF or HTML.

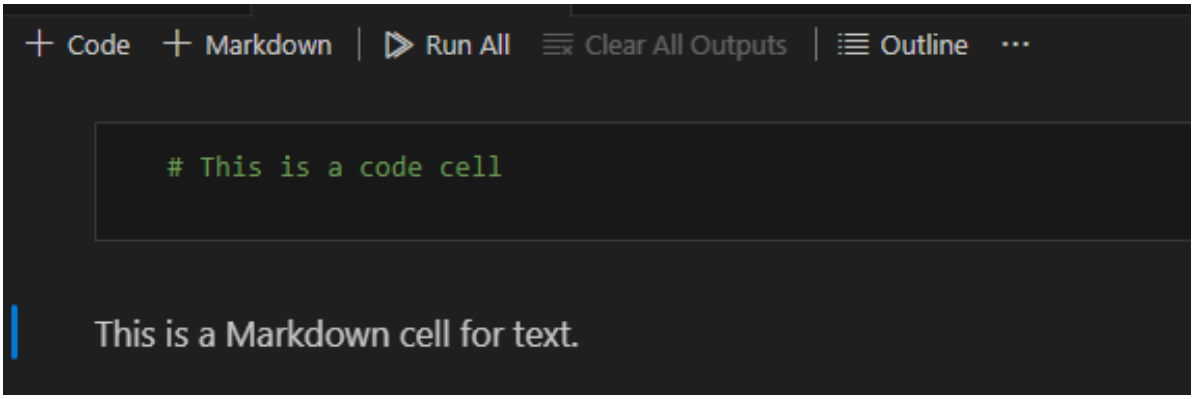
You can open Jupyter notebooks in VS Code just like the Python files you have worked with. To create a new notebook use the file extension **.ipynb** or click on New file > Jupyter Notebook, as shown below.



Extra resource

Jupyter notebooks can also be opened in your browser using the **classic Jupyter Interface**, **JupyterLab**, or **Google Collaboratory**. These options require a different installation setup and are not supported on this bootcamp, however, they are common tools you should be aware of.

In Jupyter notebooks, you can specify whether a cell contains code or Markdown. Markdown is a lightweight Markup language used to embed documentation or other textual information between code cells.



Extra resource

For more information about working with Jupyter, please consult the first chapter ("**IPython: Beyond Normal Python**") in the book entitled, "**Python Data Science Handbook**" by Jake VanderPlas.

Instructions

- Within this task folder, you will find examples in Jupyter notebook files
- .You can open and explore them in VS Code.
    - **Dataset Examples.ipynb** for Compulsory Task 1.
    - **Report_Example.ipynb** for Compulsory Task 2.

Compulsory Task 1

Open the **Datasets Compulsory task.ipynb** file and complete the following tasks in the notebook. Save your notebook to your task folder for submission.

1. Write the code that performs the action described in the following statements.
    a. Select the 'Limit' and 'Rating' columns of the first five observations
    b. Select the first five observations with 4 cards
    c. Sort the observations by 'Education'. Show users with a high education value first.
2. Write a short explanation in the form of a comment for the following lines of code.
    a. `df.iloc[:,:]`
    b. `df.iloc[5:,5:]`
    c. `df.iloc[:,0]`
    d. `df.iloc[9,:]`

Compulsory Task 2

Open and run the example file for this task before attempting this task. Follow these steps:

- Open the **Report.ipynb** in this folder.
- Create a DataFrame that contains the data in **balance.txt**.
- Write the code needed to produce a report that provides the following information:
    - Compare the average income based on ethnicity.
    - On average, do married or single people have a higher balance?
    - What is the highest income in our dataset?
    - What is the lowest income in our dataset?
    - How many cards do we have recorded in our dataset? (Hint: use `sum()`)
    - How many females do we have information for vs how many males? (Hint: use `count()`)

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved? Do you think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCES

Lynn, S. (2018). The Pandas DataFrame – loading, editing, and viewing data in Python. Retrieved from Shane Lynn: Pandas Tutorials: **https://www.shanelynn.ie/using-pandas-dataframe-creating-editing-viewing-data-in-python/**

Jupyter Team. (2015). Running the Notebook — Jupyter Documentation 4.1.1 alpha documentation. Retrieved 18 August 2020, from **https://test-jupyter.readthedocs.io/en/latest/running.html**

Petrou, T. (2017, October 27). Dissecting the anatomy of a DataFrame. (Packt>) Retrieved from Pandas Cookbook: Recipes for Scientific Computing, Time Series Analysis and Data Visualization using Python: **https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781784393878/1/ch01lvl1sec12/dissecting-the-anatomy-of-a-dataframe**