Hyperiondev

TASK

Data Analysis I

Visit our website

Introduction

WELCOME TO THE DATA ANALYSIS I TASK!

We live in a digital age. Therefore, vast amounts of data are produced and stored every day. It is extremely important to be able to organise, manipulate and interpret this data in meaningful ways. In the next few tasks, you will learn to analyse data. In this task, we'll focus on data cleaning and sanitisation.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

INTRODUCTION

Cleaning data is a very important aspect of any Data Scientist's job! Without data cleaning, it becomes very hard to be able to create models or come up with assumptions from the data. Data that hasn't been appropriately cleaned and formatted can also result in incorrect assumptions and poor decisions by the people presented with the data. Therefore, data scientists spend a lot (some say the majority) of their time cleaning data. Cleaning data could involve actions such as making sure that all data is formatted in the same way, removing nonsensical outliers and identifying incorrect data for it to be changed or removed.

DROPPING COLUMNS IN A DATAFRAME

Often, you'll find that not all the variables in a data set are useful to you, or are complete enough to be used in the analysis. For example, you might have a data set containing student information (shown in the image below), but you want to focus on analysing student grades. In such a scenario, the 'address' or 'parents' names' categories are not important. Thus, it may be unnecessary (or even inefficient) to store all these columns.

first_name	last_name	student_id	gender	math_score	english_scor	science_scor	Mother	Father	Address
Theresa	Imore	1	F	89	78	50	Jane Claremint	Michael Scofield	90 East Buckingham
Brook	Gratrex	2	M	67	56	65	Joesephine Brow	Lincon Burrows	60 Fairway Ave.
Jerrold	Isenor	3	M	98	67	42	Claire Mint	Ballack Benet	749 West Kingston Street
Jo	Pretsel	4	M	56	72	87	Alexis Mahone	Duke Harry	Lawrence Township, NJ

When cleaning data, save the data that you need. What you need will depend on the goal of your data analysis. For example, if we wanted to do a study of the population characteristics of the students, then data such as the student's address would be important, but information about the student's scores may not be necessary.

REPLACING VALUES

Sometimes it is important to replace a value from your data set with another value. For example, if you store data about a person's 'Nationality' but your data describes British nationals as either 'British', 'English', 'UK' or 'United Kingdom', you may want to replace all other values with the term 'British'. To do this, you could create a function that will look for all instances of the phrase (or phrases) and replace it (or them) with the desired output. You could choose to handle Null values in a similar manner. Null values in a dataset are data points that are empty. For example, for the variable 'Nationality' if we have an empty entry in a record, how would you

suggest we handle that? We can replace all Null values with a default value such as 'Other' under the assumption that some people did not want to disclose their nationality.

USING AN INDEX TO ACCESS RECORDS

Sometimes it is useful to have a value that uniquely identifies a record as its index. Back to the example of the students, the index of the observations can be a series of 1,2,3 …. N, or it could be the student identification number. It can be expected that when a user searches for a record, he or she may input the unique identifier (values in the Identifier column) to get the student's records.

In the code example below, we make sure that the column called 'identifier' is unique. We then set this column as the index. The **.loc() function** is used to access a group of rows and columns by the index.

```
df['Identifier'].is_unique
df = df.set_index('Identifier')
df.loc[76244]
```

CREATING DIFFERENT DATASETS

As you have learned in the previous level of this Bootcamp, it is possible to make a copy of either a complete Python list or a subset of a list by using the slice operator. For example, consider the array nums with 9 numbers (depicted below), you can get a subset of that by indexing the range you want - i.e. `nums[2:7]`

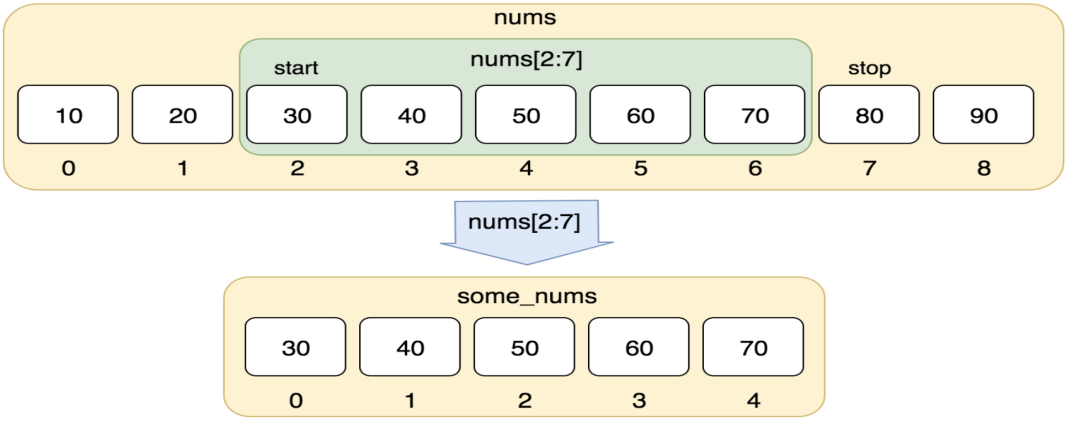

Image source: Boiko, 2018

This can also be used in a data set. One very important use of such a function in data modelling is having a training set and a test set. We will explain this further in the next level tasks but this is a percentage split of the data set, mostly 80:20 percent, that can be done using slicing. If the dataset has 100 records, we can do the split by indexing:

```
# to slice records using loc
   df_train = df.loc[1:80]
   df_test = df.loc[81:100]
```

We can never fully exhaust all methods to clean your data. Every data set will come with a different set of unique problems and you will have to get creative to work it out.

Compulsory Task 1

Follow these steps:

- Within this task folder, you will find a Jupyter Notebook named **Data Sources.ipynb**. You can open it by going to Jupyter's home screen and double-clicking on the notebook. The notebook will contain the rest of the content and requirements for this Task.

Completed the task(s)?

Ask your mentor to review your work!

Review work

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:

Agarwal, M. (n.d.). Pythonic Data Cleaning With NumPy and Pandas. Retrieved April 23, 2019, from Real Python: **https://realpython.com/python-data-cleaning-numpy-pandas/**

Boiko, S. (2018, October 3). Python for Machine Learning: Indexing and Slicing for Lists, Tuples, Strings, and other Sequential Types. Retrieved from Railsware.com: **https://railsware.com/blog/python-for-machine-learning-indexing-and-slicing-for-lists-tuples-strings-and-other-sequential-types/**