TASK

Data Analysis - Data Cleaning

Visit our website

Copyright © 2021 HyperionDev. All rights reserved.

Introduction

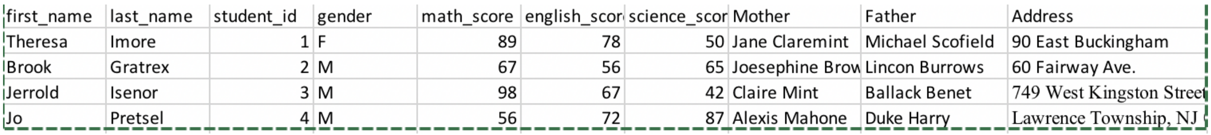
WELCOME TO THE DATA ANALYSIS - DATA CLEANING TASK!

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.

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 dataset are useful to you, or are not complete enough to be used in the analysis. For example, you might have a dataset 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. To drop columns in pandas, the syntax is:

```
school_data.drop(["math_score", "english_score", "science_score"], axis=1,
inplace=True)
```

Copyright © 2021 HyperionDev. All rights reserved.

Note the axis parameter is set to 1 (**`axis=1`**). This drops a specific column (and not a row). In addition, **`inplace=True`** tells the method to change the DataFrame itself (and not return a copy).

REPLACING VALUES

Sometimes it is important to replace a value from your dataset 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 one term e.g. '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. This would look something like this:

```
my_data.Nationality.fillna('Other', inplace=True)
```

A way to specify the column name is by using `.Nationality`. The **`fillna`** method fills all blank values with the specified value. To do more general replacements, such as replacing 'UK' with 'United Kingdom", use **`replace()`**:

```
my_data.Nationality.replace('UK', 'United Kingdom', inplace=True)
```

ACCESSING SPECIFIC RECORDS

Sometimes it is helpful 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. When a user searches for a record, they 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. Then, the **`.loc()` function** is used to access a specific row by the index value for that row.

Copyright © 2021 HyperionDev. All rights reserved.

```
# Check all entries in 'Identifier' are unique
df['Identifier'].is_unique

# Set 'Identifier' column as the index
df = df.set_index('Identifier')

# Access a specific row
df.loc[76244]
```

Recall, it is possible to make a copy of a complete Python list or a subset of a list using the slice operator. For example, consider the array `nums` with nine numbers (depicted below). You can get a subset of that by indexing the range you want - i.e. `nums[2:7]`

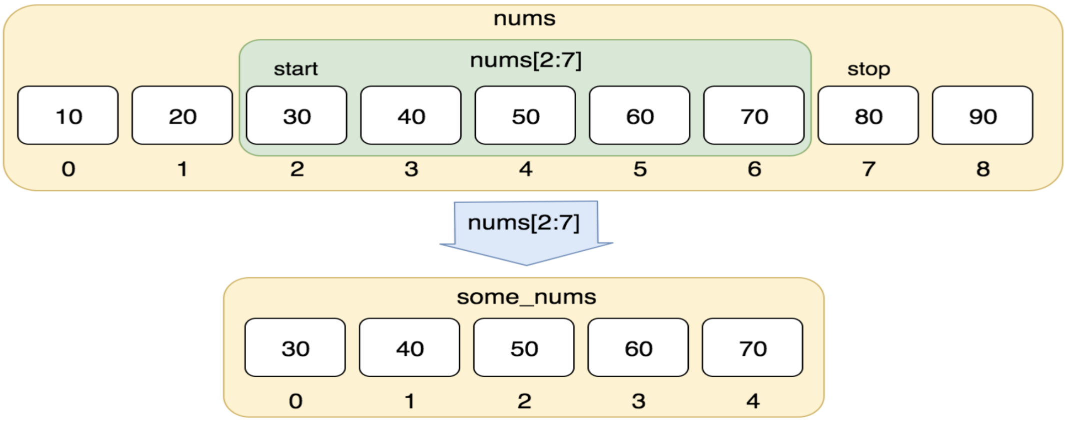


Image source: Boiko, 2018

This approach can also be used with a dataset.

```
# Slice records using loc
 subset_data = df.loc[1:80]
```

Extra resource

It is also possible to access columns by their label. Explore examples to **access DataFrame columns** using column labels or positions.

Copyright © 2021 HyperionDev. All rights reserved.

We can only suggest some methods to clean your data. Every dataset will come with a different set of unique problems and you will have to get creative to work it out.

Instructions

In this compulsory task, you will need the following libraries:

- `fuzzywuzzy`
- `chardet`

Within this task folder, you will find a Jupyter Notebook named **data_cleaning_example.ipynb**. Work through this notebook before moving on to the compulsory task.

Compulsory Task 1

In this compulsory task, you will clean the country column and parse the "`date_measured`" column in the **store_income_data_task.csv** file. Use the examples given in **data_cleaning_example.ipynb** to guide you.

Complete the following in **store_income_task.ipynb**:

- Load **store_income_data_task.csv**.
- Take a look at all the unique values in the "`country`" column. Then, convert the column entries to lowercase and remove any trailing white spaces.
- Clean up the "`country`" column so that there are three distinct countries.
- Create a new column called ``days_ago`` in the DataFrame that is a copy of the "`date_measured`" column but instead shows a number that represents the number of days ago that it was measured. Note that the current date can be obtained using `datetime.date.today()`.

Copyright © 2021 HyperionDev. All rights reserved.

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/**

Copyright © 2021 HyperionDev. All rights reserved.