

ModifiedCollectingDataUsingInteractiveJupyterWidgets

June 18, 2022

1 Title: Collecting data using interactive Jupyter widgets

Author details: *Author:* B210741. *Contact details:* s2271734@edu.ac.uk.

Notebook and data info: This Notebook provides an example of using interactive jupyter-widgets and to collect the NHS England accident and emergency attendances and admissions (ae_attendances) data.

Data: Data consists of date, numerical data and character data from NHSRdatasets package.

Copyright statement: This Notebook is the product of The University of Edinburgh.

2 Data

The data are from the NHSRdatasets package. The dataset I have chosen to manage from the NHSRdatasets package is the NHS England accident and emergency (A&E) attendances and admissions (ae_attendances) data. The ae_attendances data includes reported attendances, four-hour breaches and admissions for all A&E departments in England for 2016/17 through 2018/19 (Apr-Mar).

I selected a subset of the variables needed for my data capture tool, including period, attendances and breaches, and subsetting the data into test and training data. I will use interactive Jupyter-widgets from the *ipywidgets* package to collect all data types from the ae_attendances data. The R script “./RScripts/LoadingNHSRdatasets_fulldata.R” was used to subset the full ae_attendances data into test and training data.

2.0.1 Loading the data

Loading the *panda* package to import the data.

```
[ ]: #Load the 'pandas' package
import pandas as pd
testData=pd.read_csv("../Data/ae_attendances_ENG_4hr_perform_test_full.csv")
testData
```

Data type Using the dtypes function from the Python *pandas* package to query the data types in the testData.

```
[ ]: result = testData.dtypes
print("Output:")
```

```
print(result)
```

The data type object is a string

Setting up an empty data frame Setting up an empty data frame in the working data folder to collect the data captured by the Jupyter widgets.

```
[ ]: dfTofill = pd.DataFrame({'index': [0], # Integer
                             'period': [pd.Timestamp('20000101')], # Date
                             'org_code': ['NA'], # String
                             'type': ['NA'], # String
                             'attendances': [0], # Integer
                             'breaches': [0], # Integer
                             'admissions': [0], # Integer
                             'performance': [0.0], # Float
                             'consent': [False]}) # Boolean

dfTofill
```

Save the empty data frame to your working 'Data' folder:

```
[ ]: #dfTofill.to_csv('../Data/CollectedData.csv', index=False)
```

Reading in the empty data frame to collect the data from the Jupyter-widgets.

```
[ ]: CollectData=pd.read_csv("../Data/CollectedData.csv")
CollectData
```

Index variable I will use indexing to add the index number to the 'dfTofill' file. The first variable contains the index number, indexing allows us to connect the test data to the original data set "../RawData/ae_attendances.csv".

```
[ ]: index_number=1155
dfTofill.iloc[0,0]=index_number
dfTofill
```

2.0.2 Setting up Widgets

Widgets are interactive Python objects that have a representation in the browser. A widget is a graphical user interface element, such as a button, dropdown or textbox.

Importing the *ipywidgets* Python package.

```
[ ]: #Load the 'ipywidgets' package
import ipywidgets as widgets
```

Importing the *IPython.display* package

The *IPython.display* package is used to display different objects in Jupyter.

```
[ ]: #Load the 'IPython.display' package
from IPython.display import display
```

3 Consent

Consent is a vital area for data protection compliance. Consent means giving data subjects genuine choice and control over how you process their data. If the data subject has no real choice, consent is not freely given, and it will be invalid. The [General Data Protection Regulation](#) sets a high standard for consent and contains significantly more detail than previous data protection legislation. Consent is defined in Article 4 as: “Consent of the data subject means any freely given, specific informed and unambiguous indication of the data subject’s wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her”.

Before we collect any data, we need to get consent from the end-user to process and share the data we will collect with the data capture tool.

3.1 Boolean widgets

Boolean widgets are designed to display a boolean value.

3.1.1 Checkbox widget

```
[ ]: a = widgets.Checkbox(
    value=False,
    description='I consent for the data I have provided to be processed and
    ↪shared in accordance with data protection regulations with the purpose of
    ↪improving care service provision across the UK.',
    disabled=False
)

[ ]: display(a)

[ ]: dfTofill.iloc[0,8]=a.value
dfTofill
```

4 The period variable

The period variable includes the month this activity relates to, stored as a date (1st of each month).

Data type We now need to check the data type in the testData data frame. Let us use the dtypes function from the Python *pandas* package to query the data types in the testData. The dtypes function returns the data types in the data frame.

```
[ ]: print(result[1])
#String data type
```

4.0.1 DatePicker widget

We next need to set up a DatePicker widget to collect the period data.

```
[ ]: b = widgets.DatePicker(  
    description='Period',  
    disabled=False  
)  
display(b)
```

```
[ ]: dfTofill.iloc[0,1]=b.value  
dfTofill
```

5 The org_code variable

The `org_code` variable includes the Organisation data service (ODS) code for the organisation. The ODS code is a unique code created by the Organisation data service within [NHS Digital](#), and used to identify organisations across health and social care. ODS codes are required in order to gain access to national systems like NHSmail and the Data Security and Protection Toolkit. If you want to know the organisation associated with a particular ODS code, you can look it up from the following address: <https://odsportal.digital.nhs.uk/Organisation/Search>. For example, the organisation associated with the ODS code 'AF003' is [Parkway health centre](#).

Data type Using the `dtypes` function from the Python *pandas* package to query the data types in the `testData`.

```
[ ]: print(result[2])  
#String data type
```

Using the `describe()` function from the *numpy* Python package to calculate summary statistics for the `testData` data frame.

```
[ ]: #Load the 'numpy' package  
import numpy as np  
testData.describe(include='all')
```

Using the *pandas* package `unique()` function to get the unique Organisation data service (ODS) codes in the test data.

```
[ ]: org_code=list(testData['org_code'].unique())  
org_code
```

5.1 Selection widgets

Several widgets can be used to display single selection lists. You can specify the selectable options by passing a list.

```
[ ]: c=widgets.Select(
    options=org_code,
    value='C82010',
    rows=len(org_code),
    description='ODS code:',
    disabled=False
)
display(c)
```

```
[ ]: dfTofill.iloc[0,2]=c.value
dfTofill
```

6 The type variable

The type variable contains the department type for this activity, either

- * **1:** Emergency departments are a consultant-led 24-hour service with full resuscitation facilities and designated accommodation for the reception of accident and emergency patients,
- * **2:** Consultant-led mono speciality accident and emergency service (e.g. ophthalmology, dental) with designated accommodation for the reception of patients, or
- * **other:** Other type of A&E/minor injury activity with designated accommodation for the reception of accident and emergency patients. The department may be doctor-led or nurse-led and treats at least minor injuries and illnesses and can be routinely accessed without an appointment. A service mainly or entirely appointment-based (for example, a GP Practice or Outpatient clinic) is excluded even though it may treat a number of patients with minor illnesses or injury. Excludes NHS walk-in centres. ([National Health Service, 2020](#))

Data type Using the `dtypes` function from the Python *pandas* package to query the data types in the `testData`.

```
[ ]: print(result[3])
#String data type
```

Applying pandas `unique()` function Using the pandas package `unique()` function to get the unique department type in the test data.

```
[ ]: type=list(testData['type'].unique())
type
```

6.0.1 RadioButtons Widget

```
[ ]: d=widgets.RadioButtons(
        options=type,
        #     value='other',
        description='Type:',
        disabled=False
    )
    display(d)
```

```
[ ]: dfTofill.iloc[0,3]=d.value
    dfTofill
```

7 The attendances variable

The attendances variable includes the number of attendances for this department type at this organisation for this month.

Data type Using the `dtypes` function from the Python *pandas* package to query the data types in the `testData`.

```
[ ]: print(result[4])
    #int64
```

7.1 Numeric widgets

There are many widgets distributed with `ipywidgets` that are designed to display numeric values. Widgets exist for displaying integers and floats, both bounded and unbounded. The integer widgets share a similar naming scheme to their floating point counterparts. By replacing `Float` with `Int` in the widget name, you can find the Integer equivalent.

7.1.1 IntText Widget

```
[ ]: e=widgets.IntText(
        value=0,
        description='Attendances:',
        disabled=False)
    display(e)
```

```
[ ]: dfTofill.iloc[0,4]=e.value
    dfTofill
```

8 The breaches variable

The breaches variable includes the number of attendances that breached the four hour target.

Data type Using the `dtypes` function from the Python *pandas* package to query the data types in the `testData`.

```
[ ]: print(result[5])
      #int64
```

8.0.1 IntText Widget

```
[ ]: f=widgets.IntText(
      value=0,
      description='Breaches:',
      disabled=False)
      display(f)
```

```
[ ]: dfTofill.iloc[0,5]=f.value
      dfTofill
```

9 The admissions variable

The admissions variable includes the number of attendances that resulted in an admission to the hospital. (Chris Mainey, 2021)

Data type Using the `dtypes` function from the Python *pandas* package to query the data types in the `testData`.

```
[ ]: print(result[6])
      #It is an integer variable.
```

9.0.1 IntText Widget

```
[ ]: g=widgets.IntText(
      value=0,
      description='Admissions:',
      disabled=False)
      display(g)
```

```
[ ]: dfTofill.iloc[0,6]=g.value
      dfTofill
```

10 The performance variable

The performance variable was calculated for the whole of England as $(1 - \text{breaches}) / \text{attendances}$.

Data type Using the `dtypes` function from the Python *pandas* package to query the data types in the `testData`.

```
[ ]: print(result[7])
      #It is a float variable.
```

10.0.1 FloatText Widget

```
[ ]: h=widgets.FloatText(  
    value=0.0,  
    description='Performance:',  
    disabled=False  
)  
display(h)
```

```
[ ]: dfTofill.iloc[0,7]=h.value  
dfTofill
```

11 Adding data to each widget.

For each record, I add the selected widgets to capture test data

```
[ ]: index_number=2059  
dfTofill.iloc[0,0]=index_number  
dfTofill  
  
display(a)  
dfTofill.iloc[0,8]=a.value  
dfTofill  
  
display(b)  
dfTofill.iloc[0,1]=b.value  
dfTofill  
  
display(c)  
dfTofill.iloc[0,2]=c.value  
dfTofill  
  
display(d)  
dfTofill.iloc[0,3]=d.value  
dfTofill  
  
display(e)  
dfTofill.iloc[0,4]=e.value  
dfTofill  
  
display(f)  
dfTofill.iloc[0,5]=f.value  
dfTofill  
  
display(g)  
dfTofill.iloc[0,6]=g.value  
dfTofill
```



```
display(h)
dfTofill.iloc[0,7]=h.value
dfTofill
```

```
[ ]: index_number=3468
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill
```

```
[ ]: index_number=4153
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
```

```

dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill

```

```

[ ]: index_number=4820
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)

```

```

dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill

```

```

[ ]: index_number=7243
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

```

```
display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill
```

```
[ ]: index_number=8057
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill
```

```
[ ]: index_number=8957
dfTofill.iloc[0,0]=index_number
dfTofill
```

```

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill

```

```

[ ]: index_number=10214
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value

```

```

dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill

```

```

[ ]: index_number=10328
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)

```

```
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill
```

```
[ ]: index_number=11761
dfTofill.iloc[0,0]=index_number
dfTofill

display(a)
dfTofill.iloc[0,8]=a.value
dfTofill

display(b)
dfTofill.iloc[0,1]=b.value
dfTofill

display(c)
dfTofill.iloc[0,2]=c.value
dfTofill

display(d)
dfTofill.iloc[0,3]=d.value
dfTofill

display(e)
dfTofill.iloc[0,4]=e.value
dfTofill

display(f)
dfTofill.iloc[0,5]=f.value
dfTofill

display(g)
dfTofill.iloc[0,6]=g.value
dfTofill

display(h)
dfTofill.iloc[0,7]=h.value
dfTofill
```

11.1 Concatenating the collected data to the CollectData data frame.

Using the `concat()` function from the Python *pandas* package to append the `CollectData` and `dfTofill` data frames. The `concat()` function is used to concatenate *pandas* objects.

```
[ ]: # CollectData is the first data frame
      # dfTofill is the second data frame
      CollectData = pd.concat([CollectData, dfTofill])
      display(CollectData)
```

11.2 Ensuring we have consent to save the data

Before we save our data to file, we must make sure we have consent to do so. The following line of code, will ensure that you have consent to save data.

```
[ ]: CollectData=CollectData[CollectData['consent'] == True]
      display(CollectData)
```

11.3 Saving the CollectData data frame

Saving the data collected by the data-capture tool to the working data folder:

```
[ ]: CollectData.to_csv('../Data/CollectedData.csv', index=False)
```

That is the `CollectData` data frame saved to the working 'Data' folder. You need to iterate through this Notebook until you have collected all of your test data and then save the captured test data to your 'RawData' folder.

```
[ ]: CollectData.to_csv('../RawData/CollectedDataFinal.csv', index=False)
```

That is the final `CollectData` data frame saved to the 'RawData' folder.