HW_ClassSearch

September 27, 2024

1 PRACTICE - iSchool Class Search

1.1 The Problem

You have been hired to build an interactive data product for the iSchool that makes it easier for students to find classes. Your task is to read in a schedule of classes and create a user interface that allows someone to search the classes by:

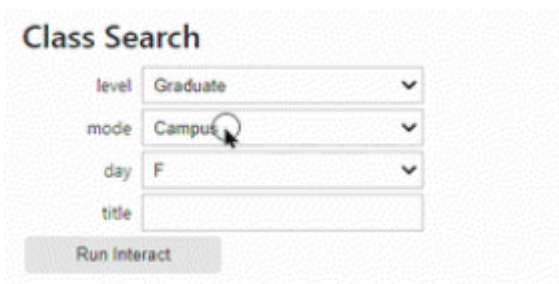
```
- Level: Gradudate / Undergraduate
- Mode: Campus / Online
- Meeting: MW / TuTh / MWF / etc...
- Contents of the Course title
```

The program should then output a dataframe of the courses which match the selected criteria.

If this were a real-world project there would be two key steps - building the data pipeline to aquire the necessary data and then - building the user interface around the data.

The data can be found at the following URL: `https://raw.githubusercontent.com/mafudge/datasets/master/`

See the final product layout here:



1.2 Approach:

This assignment is broken up into parts. We will use problem simplification to solve this problem and take a bottom up approach, making the components, then assembling them together.

- Load the data into a pandas dataframe
- Data Cleanup
- Engineer the Columns we Need
- Building data for our widgets
- Assemble the final program from its parts

1.3 Part 1: Code Solution

You may write your code in several cells, but place the complete, final working copy of your code solution.

1.3.1 You Code 2.1: Load the data into a pandas dataframe

In this first step, import the `pandas` and `numpy` libraries and then write code to load the dataset from the url found in the instructions at the top. Load into a Pandas DataFrame.

Also for your own sanity, you should probably ignore Pandas `filterwarnings`. We did this in the lab.

use the `print()` function to display the first 10 classes. The code checker will scan this output so if you want this to pass you will need to use `print()` instead of `display()`

Just know that you can use `display()` while figuring it out, but if you want to pass the code checks, you'll habe to switch to `print()`

[1]:
```
# SOLUTION CELL 2.1

# PRINT FIRST 10 ROWS: using head(10) to generate first 10 rows.
import pandas as pd
classes_data=pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/
 ↪master/classes/ischool-schedule-fall2015.csv")
classes_data.head(10)
```

[1]:
```
   Course Section  Class  Credits                          Title  \
0  IST990    M001  29957      1.0               Independent Study
1  IST999    M001  21631      1.0                    Dissertation
2  IST625    M001  21661      3.0      Enterprise Risk Management
3  IST625    M002  21902      3.0      Enterprise Risk Management
4  IST627    M002  21755      3.0             What's the Big Idea
5  IST639    M001  21696      3.0         Enterprise Technologies
6  IST641    M001  21711      3.0               User-Based Design
7  IST645    M001  21664      3.0  Managing Info Systems Projects
8  IST645    M002  21665      3.0  Managing Info Systems Projects
9  IST645    M800  21666      3.0  Managing Info Systems Projects

                             Instructor(s)               Time   Day  \
0                                      NaN  12:00am - 12:00am   NaN
1                                      NaN  12:00am - 12:00am   NaN
2                        Michelle L. Brown   9:30am - 12:15pm     W
3                         Frank Jr Marullo    5:00pm - 7:50pm    Tu
4  William C Padgett  Marcene S. Sonneborn    2:00pm - 3:20pm  TuTh
5                          P Douglas Taber    5:15pm - 8:05pm     W
6                          Michael S Nilan   12:30pm - 3:15pm    Th
7                         Arthur P. Thomas    5:00pm - 7:45pm    Tu
8                                  Tom Uva    5:15pm - 8:05pm     M
9                        Robert A Emborski  12:00am - 12:00am   NaN
```

```
                          Room(s)
0                             NaN
1                             NaN
2                 Hinds Hall 021
3                 Newhouse 1 101
4                 Hinds Hall 021
5  Hinds Hall 027 Hinds Hall 111
6                 Hinds Hall 120
7                 Hinds Hall 021
8                 Hinds Hall 117
9                         Online
```

1.3.2 You Code 2.2: Data Cleanup

If you look over the data with `info()` you will notice there are missing values in the `Instructor(s)`, `Day`, and `Room(s)` columns. We need to clean this data up before presenting it as the missing values showing `"NaN"` will be confusing to the users of our program.

Specifically do the following:

- in the `Instructor(s)` column replace all `NaN` with `"Staff"`. Its common for universities to use this label when the instructor is to be determined.
- in the `Room(s)` column replace `NaN` with `"TBA"`. Its common for universities to use this label when the room will be announced later TBA == To be Announced.
- in the `Day` column replace `NaN` with `"N/A"`. N/A means not applicable.

TIPS :

- use the column-selector then boolean filter approach : `df[col][boolean-index-selector] = value`
- your boolean index selector cannot compare the column to `np.nan` e.g. `col == np.nan` this is not the way to find nulls in a series!
- if you screw up your dataframe, don't fret just run 2.1 to reload it!

[2]:
```
# SOLUTION CELL 2.2

# COPY CODE FROM 2.1 INCLUDE THE IMPORTS!
# YOUR CLEANUP CODE

# FOR CHECKER: PRINT ROWS
```

[3]:
```
#understanding data using info

classes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202 entries, 0 to 201
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   Course          202 non-null    object
 1   Section         202 non-null    object
 2   Class           202 non-null    int64
 3   Credits         202 non-null    float64
 4   Title           202 non-null    object
 5   Instructor(s)   196 non-null    object
 6   Time            202 non-null    object
 7   Day             158 non-null    object
 8   Room(s)         182 non-null    object
dtypes: float64(1), int64(1), object(7)
memory usage: 14.3+ KB
```

[4]:
```
#replacing nan data in Instructor to Staff. and ensuring the replacement using .
↪sum which returns 0
classes_data.loc[classes_data['Instructor(s)'].isna(), 'Instructor(s)'] =␣
↪'Staff'
classes_data['Instructor(s)'].isna().sum()
```

[4]: 0

[5]:
```
classes_data.loc[classes_data['Room(s)'].isna(), 'Room(s)'] = 'TBA'
classes_data['Room(s)'].isna().sum()
```

[5]: 0

[6]:
```
classes_data.loc[classes_data['Day'].isna(), 'Day'] = 'N/A'
classes_data['Day'].isna().sum()
```

[6]: 0

1.3.3 You Code 2.3 - Engineer the columns we need

Next we need to engineer two columns. Here are the criteria:

- Column name `Level`, value is:
  - `"Graduate"` number part of the course is >=500 , e.g. `IST625` (`625>=500`)
  - `"Undergraduate"` number part of the course is <500 e.g. `IST256` (256<500‘)
-
- Column name `Mode`, value is:
  - `"Online"` 2nd character in Section is an `"8"` e.g. `M800`
  - `"Campus"` 2nd character in Section is not an `"8"` e.g. `M012`

TIPS :

- Again, use the column-selector then boolean filter: `df[col][boolean-index-selector] = value`

[7]:
```python
# SOLUTION CELL 2.3

# COPY CODE FROM 2.1 INCLUDE THE IMPORTS!

# ENGINEER COLUMNS

# FOR CHECKER: PRINT SLICE FROM 100 to 106
```

[8]:
```python
def level_apply(course_code):
    if float(course_code[-3:]) >=500:
        return 'Graduate'
    else:
        return 'Undergraduate'

classes_data['Level'] = classes_data['Course'].apply(level_apply)
```

[9]:
```python
#checking for graduate students
classes_data.head(3)
```

[9]:
```
   Course Section  Class  Credits                       Title  \
0  IST990    M001  29957      1.0           Independent Study
1  IST999    M001  21631      1.0                Dissertation
2  IST625    M001  21661      3.0  Enterprise Risk Management

       Instructor(s)               Time  Day         Room(s)     Level
0              Staff  12:00am - 12:00am  N/A             TBA  Graduate
1              Staff  12:00am - 12:00am  N/A             TBA  Graduate
2  Michelle L. Brown   9:30am - 12:15pm    W  Hinds Hall 021  Graduate
```

[10]:
```python
#inserting Mode column based on 2nd charecter in section
def level_apply(section_code):
    if float(section_code[1]) ==8:
        return 'Online'
    else:
        return 'Campus'

classes_data['Mode'] = classes_data['Section'].apply(level_apply)
```

[11]:
```python
#checking for undergraduate students
classes_data.tail(3)
```

[11]:
```
     Course Section  Class  Credits                          Title  \
199  IST263    M004  28864      3.0             Web Design and Mgmt
200  IST300    M001  21671      1.0      Information Studies Skills
201  IST323    M001  21673      3.0  Intro to Information Security

                   Instructor(s)               Time   Day         Room(s)  \
```

```
199  Michael McCafferty Clarke    3:30pm - 4:50pm  TuTh  Hinds Hall 027
200              Julie L Huynh  12:00am - 12:00am   N/A             TBA
201               Joon S. Park    2:15pm - 3:35pm    MW  Hinds Hall 021

             Level    Mode
199  Undergraduate  Campus
200  Undergraduate  Campus
201  Undergraduate  Campus
```

1.3.4 You Code 2.4 Buidling data for our widgets

Next we need to build the data for our dropdown input widgets. We need three:

1. a sorted list of unique non NaN values in the `Mode` Series, call this variable `modes`
2. a sorted list of unique non NaN values in the `Level` Series, call this variable `levels`
3. a sorted list of unique non NaN values in the `Day` Series, call this variable `days`

TIPS :

- Take the approach we used in the homework.
- You do not need to create a custom widget here, just the Python lists of unique values.

[12]:
```
# SOLUTION CELL 2.4

# COPY CODE FROM 2.1, 2.2 and 2.3

# CREATE LISTS

# FOR CHECKER: PRINT EACH LIST
```

[13]:
```
#1. a sorted list of unique non NaN values in the `Mode` Series, call this␣
↪variable `modes`
modes=classes_data[~classes_data['Mode'].isna()]['Mode'].unique().tolist()
modes.sort()
#2. a sorted list of unique non NaN values in the `Level` Series, call this␣
↪variable `levels`
levels=classes_data[~classes_data['Level'].isna()]['Level'].unique().tolist()
levels.sort()
#3.a sorted list of unique non NaN values in the Day Series, call this variable␣
↪days
days=classes_data[~classes_data['Day'].isna()]['Day'].unique().tolist()
days.sort()

#3.a sorted list of unique non NaN values in the Day Series, call this variable␣
↪days
titles=classes_data[~classes_data['Title'].isna()]['Title'].unique().tolist()
days.sort()
```

[14]:
```
days
```

[14]: ['F', 'M', 'MW', 'MWF', 'N/A', 'SaSu', 'Th', 'Tu', 'TuTh', 'W', 'WF']

1.3.5 You Code 2.5 Assemble the final program as an interact

With all the components built, its time to consider the complete program.

TIPS:

- As you write your algorithm remember you will peform steps 2.1 - 2.4 **before** you accept any input.
- Since there are 4 inputs, there should be 4 arguments to `@interact_manual` and the `on_click()` function.
- use `display()` to print the filtered dataframe.

[15]:
```
# SOLUTION CELL 2.5
import pandas as pd
import numpy as np
import warnings
from IPython.display import display, HTML
from ipywidgets import interact_manual
warnings.filterwarnings('ignore')
```

[16]:
```
from ipywidgets import interact_manual, widgets
from IPython.display import display
import pandas as pd
import numpy as np

#creating different dropdowns for title, day, mode, levels
#creating a search dropdown for superman movies
days_dropdown = widgets.Dropdown(options=days, description="Day")
level_dropdown = widgets.Dropdown(options=levels, description="Level")
mode_dropdown = widgets.Dropdown(options=modes, description="Mode")
title_input = widgets.Text(
    value='',
    placeholder='Enter course title',
    description='Title:',
)


#creating on click interact function to pass the input values of movies and the␣
↪composite scores to get the filtered list of movies
@interact_manual(day=days_dropdown,level=level_dropdown,mode=mode_dropdown,title=title_input)
def on_click(day,level,mode,title):
    filtered_courses = classes_data[
         (classes_data["Title"].str.contains(title, case=False, na=False)) &
        (classes_data["Mode"] == mode) &
```

```
        (classes_data["Level"] ==level)&
        (classes_data["Day"]==day)
    ]
    display(filtered_courses)
```

```
interactive(children=(Dropdown(description='Day', options=('F', 'M', 'MW',␣
↪'MWF', 'N/A', 'SaSu', 'Th', 'Tu', '…
```

1.4 Part 3: Metacognition

These questions are designed to prompt you to reflect on your learning. Reflection is part of the assignment grade so please take time to answer the questions thoughtfully.

3.1 List at least 3 things you learned this week and/or throughout the process of completing this assignment?

1. Building interactive input forms using python
2. Applying data operations of data frame directly using Pandas
3. Cleaning and engineering data using python built in functions

3.2 What were the challenges or roadblocks (if any) you encountered on the way to completing it? I had to see the syntax of taking the input for the title of the course. Implementing data engineering directly on the columns of data frames

3.3 Were you prepared for this assignment? What can you do to be better prepared? I was prepared. I could have known the syntax of interactive components better before starting

3.4 Did someone (or something such as AI) help you? Did You help someone? Provide details. No

3.5 Now that you have completed the assignment rate your comfort level with this week's material. This should be an honest assessment of your ability: **1** ==> I don't understand this at all yet and need extra help. If you choose this please try to articulate that which you do not understand to the best of your ability in the questions and comments section below.
2 ==> I can do this with help or guidance from other people or resources. If you choose this level, please indicate HOW this person helped you in the questions and comments section below.
3 ==> I can do this on my own without any help.
4 ==> I can do this on my own and can explain/teach how to do it to others.

```
ENTER A NUMBER 1-4 IN THE CELL BELOW
```

4