hw2

September 30, 2022

1 Homework 2: Programming with `Numpy` and `Pandas`

1.0.1 About this assignment:

The main purpose of this assignment is to practice your Python skills. This assignment covers two python packages, `numpy` and `pandas`, which we'll be using throughout the course. For some of you, Python/numpy/pandas will be familiar; for others, it will be new. While ECE 4420/6420 is a machine learning course rather than a programming course, programming will be an essential but less challenging part of it.

Also, as part of this assignment you will likely need to consult the documentation for various Python packages we're using. This is, of course, totally OK and in fact strongly encouraged. Reading and interpreting documentation is an important skill, and in fact is one of the skills this assignment is meant to assess.

1.1 Recommending tutorials for HW 2

1. Python NumPy Tutorial for Beginners
2. Python Pandas Data Science Tutorial

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

1.2 Exercise 1: Loading files with Pandas

rubric={points:20}

When working with tabular data, you will typically be creating Pandas dataframes by reading data from .csv files using `pd.read_csv()`. The documentation for this function is available here.

In the "data" folder in this homework repository there are 2 different .csv files named `wine_#.csv/.txt`. Look at each of these files and use `pd.read_csv()` to load these data so that they resemble the following:

Bottle	Grape	Origin	Alcohol	pH	Colour	Aroma
1	Chardonnay	Australia	14.23	3.51	White	Floral
2	Pinot Grigio	Italy	13.20	3.30	White	Fruity
3	Pinot Blanc	France	13.16	3.16	White	Citrus
4	Shiraz	Chile	14.91	3.39	Red	Berry

Bottle	Grape	Origin	Alcohol	pH	Colour	Aroma
5	Malbec	Argentina	13.83	3.28	Red	Fruity

Hint: 1. Read two files and assign them to `df1` and `df2` accordingly. 2. You may use pandas.read_csv.

You are provided with tests that use `df.equals()` to check that all the dataframes are identical. If you're in a situation where the two dataframes look identical but `df.equals()` is returning `False`, it may be an issue of types - try checking `df.index`, `df.columns`, or `df.info()`.

[2]:
```
df1 = None
df2 = None

# BEGIN YOUR CODE HERE
df1 = pd.read_csv(".\data\wine_1.csv")
df2 = pd.read_csv(".\data\wine_2.txt",sep="\t")
# END YOUR CODE HERE

assert df1.equals(df2), f"df1 not equal to df2"
print("All tests passed.")
```

```
All tests passed.
```

1.3 Exercise 2: The Titanic dataset

Rubric={points:52}

The file *titanic.csv* contains data of 1309 passengers who were on the Titanic's unfortunate voyage. For each passenger, the following data are recorded:

- survival - Survival (0 = No; 1 = Yes)
- class - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- name - Name
- sex - Sex
- age - Age
- sibsp - Number of Siblings/Spouses Aboard
- parch - Number of Parents/Children Aboard
- ticket - Ticket Number
- fare - Passenger Fare
- cabin - Cabin
- embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
- boat - Lifeboat (if survived)
- body - Body number (if did not survive and body was recovered)

In this exercise you will perform a number of wrangling operations to manipulate and extract subsets of the data.

Note: many popular datasets have sex as a feature where the possible values are male and female. This representation reflects how the data were collected and is not meant to imply that, for example, gender is binary.

2(a) rubric={points:3}

Load the `titanic.csv` dataset into a pandas dataframe named `titanic_df`.

[3]:
```
titanic_df = None

# BEGIN YOUR CODE HERE
titanic_df=pd.read_csv(".\\data\\titanic.csv")
# END YOUR CODE HERE
```

[4]:
```
assert set(titanic_df.columns) == set([
    "pclass",
    "survived",
    "name",
    "sex",
    "age",
    "sibsp",
    "parch",
    "ticket",
    "fare",
    "cabin",
    "embarked",
    "boat",
    "body",
    "home.dest",
]), "All required columns are not present"
assert len(titanic_df.index) == 1309, "Wrong number of rows in dataframe"
print("Success")
```

```
Success
```

2(b) rubric={points:3}

The column names `sibsp` and `parch` are not very descriptive. Use df.rename() to rename these columns to `siblings_spouses` and `parents_children` respectively.

[5]:
```
# BEGIN YOUR CODE HERE
titanic_df = titanic_df.rename(columns={"sibsp":"siblings_spouses","parch":
 ↪"parents_children"})
# END YOUR CODE HERE
```

[6]:
```
assert set(["siblings_spouses", "parents_children"]).issubset(
    titanic_df.columns), "Column names were not changed properly"
print("Success")
```

```
Success
```

2(c) rubric={points:3}

We will practice indexing different subsets of the dataframe in the following questions.

Select the column `age` using single bracket notation `[]`. What type of object is returned?

[7]:
```
# BEGIN YOUR CODE HERE
titanic_df["age"]
print(f"The type of object returned is {titanic_df['age'].dtype}.")
print(f"The type of object returned is float64.")
# END YOUR CODE HERE
```

```
The type of object returned is float64.
The type of object returned is float64.
```

2(d) rubric={points:3}

Now select the `age` using double bracket notation `[[]]`. What type of object is returned?

[8]:
```
# BEGIN YOUR CODE HERE
titanic_df[["age"]]
print(f"The type of object returned is Dataframe.")
# END YOUR CODE HERE
```

```
The type of object returned is Dataframe.
```

2(e) rubric={points:3}

Select the columns `pclass`, `survived`, and `age` using a single line of code.

[9]:
```
# BEGIN YOUR CODE HERE
titanic_df[["pclass","survived","age"]]
# END YOUR CODE HERE
```

[9]:

	pclass	survived	age
0	1	1	29.0000
1	1	1	0.9167
2	1	0	2.0000
3	1	0	30.0000
4	1	0	25.0000
…	…	…	…
1304	3	0	14.5000
1305	3	0	NaN
1306	3	0	26.5000
1307	3	0	27.0000
1308	3	0	29.0000

[1309 rows x 3 columns]

2(f) rubric={points:4}

Use the iloc method to obtain the first 5 rows of the columns `name`, `sex` and `age` using a single line of code.

[10]:
```
# BEGIN YOUR CODE HERE
titanic_df[["name","sex","age"]].iloc[0:5]
# END YOUR CODE HERE
```

[10]:

	name	sex	age
0	Allen, Miss. Elisabeth Walton	female	29.0000
1	Allison, Master. Hudson Trevor	male	0.9167
2	Allison, Miss. Helen Loraine	female	2.0000
3	Allison, Mr. Hudson Joshua Creighton	male	30.0000
4	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000

2(g) rubric={points:3}

Now use the loc method to obtain the first 5 rows of the columns `name`, `sex` and `age` using a single line of code.

[11]:
```
# BEGIN YOUR CODE HERE
titanic_df[["name","sex","age"]].loc[0:4]
# END YOUR CODE HERE
```

[11]:

	name	sex	age
0	Allen, Miss. Elisabeth Walton	female	29.0000
1	Allison, Master. Hudson Trevor	male	0.9167
2	Allison, Miss. Helen Loraine	female	2.0000
3	Allison, Mr. Hudson Joshua Creighton	male	30.0000
4	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000

2(h) rubric={points:2}

How many passengers survived (`survived = 1`) the disaster?

Hints: 1. try using df.query() or `[]` notation to subset the dataframe and then df.shape to check its size.

[12]:
```
# BEGIN YOUR CODE HERE
# df.shape will give number of rows and columns. Since we are only interested␣
 ↪in number of passesnger i.e. number of rows,
# we can use len(). However, I have tried to solve the problem using both len␣
 ↪and shape.

titanic_df.query('survived == 1').shape
print(f"{titanic_df.query('survived == 1').shape} is the number of passengers␣
 ↪survived.")
```

```
print(f"{len(titanic_df.query('survived == 1'))} is the number of passengers␣
 ↪survived.")

# OR

# sur = titanic_df.query('survived == 1')
# sur.shape

# END YOUR CODE HERE
```

```
(500, 14) is the number of passengers survived.
500 is the number of passengers survived.
```

2(i) rubric={points:4}

How many passengers that survived the disaster (`survived = 1`) were over 60 years of age?

[13]:
```
# BEGIN YOUR CODE HERE
titanic_df.query('survived == 1' and 'age>60').shape
print(f"{titanic_df.query('survived == 1' and 'age>60').shape}")
print(f"{len(titanic_df.query('survived == 1' and 'age>60'))} is the number of␣
 ↪people over the age of 60 years who survived.")
# END YOUR CODE HERE
```

```
(33, 14)
33 is the number of people over the age of 60 years who survived.
```

2(j) rubric={points:4}

What was the lowest and highest fare paid to board the titanic? Store your answers as floats in the variables `lowest` and `highest`.

Hints: 1. `min` and `max` can return the smallest and largest values.

[14]:
```
lowest = None
highest = None

# BEGIN YOUR CODE HERE
lowetst = titanic_df[["fare"]].min(skipna=True,numeric_only=True)
highest = titanic_df[["fare"]].max(skipna=True,numeric_only=True)
print(f"The lowest fare was 0 or None and the highest fare was 512.3292")
# END YOUR CODE HERE

print(lowest, highest)
```

```
The lowest fare was 0 or None and the highest fare was 512.3292
None fare    512.3292
dtype: float64
```

2(k) rubric={points:4}

Sort the dataframe by fare paid (most to least) and show the sorted result.

Hints: 1. try df.sort_values

[15]:
```
# BEGIN YOUR CODE HERE
titanic_sort_df = titanic_df.sort_values('fare',ignore_index=True)

# END YOUR CODE HERE

titanic_sort_df
```

[15]:
```
      pclass  survived                                             name  \
0          2         0                            Campbell, Mr. William
1          1         0                Parr, Mr. William Henry Marsh
2          1         1                      Ismay, Mr. Joseph Bruce
3          3         1                 Tornquist, Mr. William Henry
4          3         0                          Johnson, Mr. Alfred
…        …       …                                                …
1304       1         1                             Ward, Miss. Anna
1305       1         1  Cardeza, Mrs. James Warburton Martinez (Charlo…
1306       1         1           Cardeza, Mr. Thomas Drake Martinez
1307       1         1                       Lesurer, Mr. Gustave J
1308       3         0                           Storey, Mr. Thomas

          sex   age  siblings_spouses  parents_children    ticket      fare  \
0        male   NaN                 0                 0    239853    0.0000
1        male   NaN                 0                 0    112052    0.0000
2        male  49.0                 0                 0    112058    0.0000
3        male  25.0                 0                 0      LINE    0.0000
4        male  49.0                 0                 0      LINE    0.0000
…      …   …                 …                 …       …       …
1304   female  35.0                 0                 0  PC 17755  512.3292
1305   female  58.0                 0                 1  PC 17755  512.3292
1306     male  36.0                 0                 1  PC 17755  512.3292
1307     male  35.0                 0                 0  PC 17755  512.3292
1308     male  60.5                 0                 0      3701       NaN

             cabin embarked boat   body  \
0              NaN        S  NaN    NaN
1              NaN        S  NaN    NaN
2     B52 B54 B56        S    C    NaN
3              NaN        S   15    NaN
4              NaN        S  NaN    NaN
…            …      …  …    …
1304           NaN        C    3    NaN
1305  B51 B53 B55        C    3    NaN
```

```
1306  B51 B53 B55        C    3    NaN
1307         B101        C    3    NaN
1308          NaN        S  NaN  261.0

                                          home.dest
0                                           Belfast
1                                           Belfast
2                                         Liverpool
3                                               NaN
4                                               NaN
…                                               …
1304                                            NaN
1305                    Germantown, Philadelphia, PA
1306  Austria-Hungary / Germantown, Philadelphia, PA
1307                                            NaN
1308                                            NaN

[1309 rows x 14 columns]
```

2(l) rubric={points:3}

Save the sorted dataframe to a .csv file called 'titanic_fares.csv' using to_csv().

[16]:
```
# BEGIN YOUR CODE HERE
titanic_sort_df.to_csv("titanic_fares.csv",index=False)

# END YOUR CODE HERE
```

2(m) rubric={points:5}

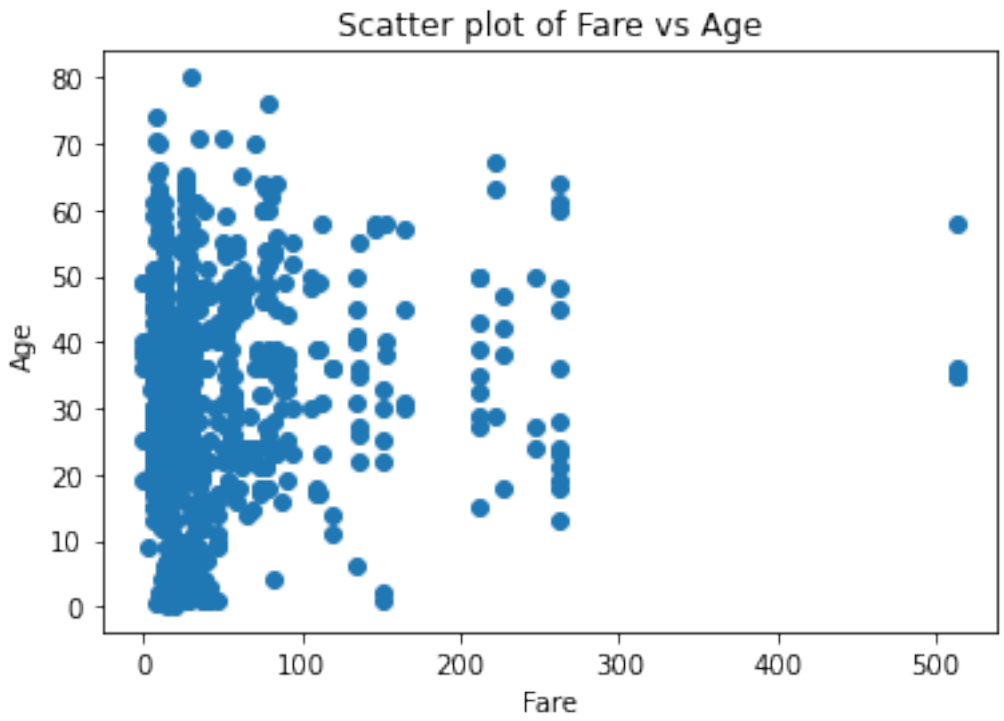
Create a scatter plot of fare (y-axis) vs. age (x-axis).

Hints - matplotlib.pyplot.scatter

[17]:
```
# BEGIN YOUR CODE HERE
# plt.scatter(titanic_sort_df[['fare']],titanic_sort_df[['age']])
plt.scatter(titanic_df[['fare']],titanic_df[['age']])
plt.xlabel('Fare')
plt.ylabel('Age')
plt.title('Scatter plot of Fare vs Age')

# END YOUR CODE HERE
```

[17]:
```
Text(0.5, 1.0, 'Scatter plot of Fare vs Age')
```



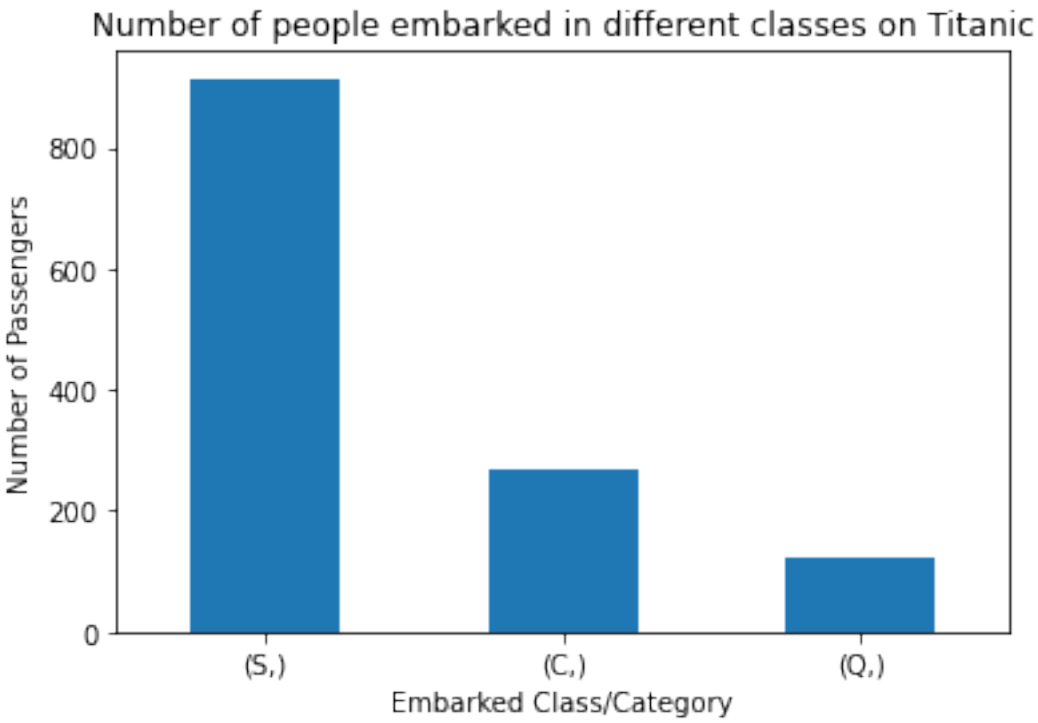
2(n) rubric={points:5}

Create a bar plot of `embarked` values.

Hints - Make sure to name the axes and give a title to your plot. - You may read value_counts and matplotlib.pyplot.bar.

[18]:
```
# BEGIN YOUR CODE HERE
A = titanic_df[['embarked']].value_counts()
# print(f"{A}")
# plt.bar(A)
A.plot.bar(rot=0)
plt.xlabel("Embarked Class/Category")
plt.ylabel("Number of Passengers")
plt.title("Number of people embarked in different classes on Titanic")
# END YOUR CODE HERE
```

[18]:
```
Text(0.5, 1.0, 'Number of people embarked in different classes on Titanic')
```



1.4 Exercise 3: Treasure Hunt

Rubric={points:18}

In this exercise, we will generate various collections of objects either as a list, a tuple, or a dictionary. Your task is to inspect the objects and look for treasure, which in our case is a particular object: **the character "T"**.

Your tasks:

For each of the following cases, index into the Python object to obtain the "T" (for Treasure).

> Please do not modify the original line of code that generates `x` (though you are welcome to copy it). You are welcome to answer this question "manually" or by writing code - whatever works for you. However, your submission should always end with a line of code that prints out `'T'` at the end (because you've found it).

[19]:
```
import string

letters = string.ascii_uppercase
```

The first one is done for you as an example.

Example question

[20]:
```
x = ("nothing", {-i: l for i, l in enumerate(letters)})
x
```

[20]:
```
('nothing',
 {0: 'A',
  -1: 'B',
  -2: 'C',
  -3: 'D',
  -4: 'E',
  -5: 'F',
  -6: 'G',
  -7: 'H',
  -8: 'I',
  -9: 'J',
  -10: 'K',
  -11: 'L',
  -12: 'M',
  -13: 'N',
  -14: 'O',
  -15: 'P',
  -16: 'Q',
  -17: 'R',
  -18: 'S',
  -19: 'T',
  -20: 'U',
  -21: 'V',
  -22: 'W',
  -23: 'X',
  -24: 'Y',
  -25: 'Z'})
```

Example answer:

[21]:
```
x[1][-19]
```

[21]:
```
'T'
```

> Note: In these questions, the goal is not to understand the code itself, which may be confusing. Instead, try to probe the types of the various objects. For example `type(x)` reveals that `x` is a tuple, and `len(x)` reveals that it has two elements. Element 0 just contains "nothing", but element 1 contains more stuff, hence `x[1]`. Then we can again probe `type(x[1])` and see that it's a dictionary. If you `print(x[1])` you'll see that the letter "T" corresponds to the key -19, hence `x[1][-19]`.

3(a) rubric={points:6}

Return the position(index) of 'T' when you find 'T'.

[22]:
```
# Do not modify this cell
x = [
    [letters[i] for i in range(26) if i % 2 == 0],
    [letters[i] for i in range(26) if i % 2 == 1],
]
x
```

[22]:
```
[['A', 'C', 'E', 'G', 'I', 'K', 'M', 'O', 'Q', 'S', 'U', 'W', 'Y'],
 ['B', 'D', 'F', 'H', 'J', 'L', 'N', 'P', 'R', 'T', 'V', 'X', 'Z']]
```

[23]:
```
# BEGIN YOUR CODE HERE
for i in range(0,2):
    for j in range(0,len(x[1])):
        if x[i][j] == 'T':
            print(f"The position of the x is x[{i}][{j}]")
        else:
            continue
# print(f"{x[1][9]}") #Checking if the answer is correct.
# END YOUR CODE HERE
```

```
The position of the x is x[1][9]
```

3(b) rubric={points:6}

Return the position(index) of 'T' when you find 'T'.

[24]:
```
# Do not modify this cell
np.random.seed(1)
x = np.random.choice(list(set(letters) - set("T")),
                     size=(100, 26),
                     replace=True)
x[np.random.randint(100), np.random.randint(26)] = "T"
```

[25]:
```
# BEGIN YOUR CODE HERE

for i in range(0,len(x)):
    for j in range(0,len(x[1])):
        if x[i][j] == 'T':
            print(f"The position of the x is x[{i}][{j}]")
        else:
            continue
# print(f"{x[95][2]}")#Checking if the answer is correct.
# END YOUR CODE HERE
```

```
The position of the x is x[95][2]
```

3(c) rubric={points:6}

[26]:
```python
# Do not modify this cell
n = 26
x = dict()
for i in range(n):
    x[string.ascii_lowercase[i]] = {
        string.ascii_lowercase[(j + 1) % n]:
        [[letters[j]] if j - 2 == i else None]
        for j in range(n)
    }
```

[27]:
```python
# BEGIN YOUR CODE HERE
letters = string.ascii_lowercase
for i in letters:
    for j in letters:
        A =x[i][j][0]
        if A != None and A[0] == 'T':
            print(f"The letter T can be found at keys {i} and {j}.")

# print(f"{x['r']['u'][0][0]}") # checking answer
# END YOUR CODE HERE
```

```
The letter T can be found at keys r and u.
```

1.5 Submission instructions

rubric={points:10}

PLEASE READ: When you are ready to submit your assignment do the following:

1. Run all cells in your notebook to make sure there are no errors by doing `Kernel -> Restart Kernel and Clear All Outputs` and then `Run -> Run All Cells`.
2. Notebooks with cell execution numbers out of order or not starting from "1" will have marks deducted. Notebooks without the output displayed may not be graded at all (because we need to see the output in order to grade your work).
3. Upload the assignment at Canvas.
4. Finish the corresponding reflection survey.