CPSC 330 - Applied Machine Learning

Homework 1: Programming with Python

Due date: Thursday May 19, 2022 at 18:00.

About this assignment:

The main purpose of this assignment is to check whether your programming knowledge is adequate to take CPSC 330. 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. Either way, ***if you find this assignment very difficult then that could be a sign that you will struggle later on in the course***. While CPSC 330 is a machine learning course rather than a programming course, programming will be an essential 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. For Python refresher, check out Python notes and Python resources.

Set-up

In order to do this assignment, you will need to set up the CPSC 330 software stack, which is Python and Jupyter. For software install help, see here. However, please ***note*** that for cloning the repository, instead of the git command mentioned in the link above (`git clone https://github.com/UBC-CS/cpsc330.git`), you should use the following command, which clones the repository for the current course offering:

```
git clone https://github.com/UBC-CS/cpsc330-2022s.git
```

Once you have the software stack installed, you should be able to run the next cell, which imports some packages needed for the assignment.

Imports

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

Points

Each question or sub-question will have a number of points allocated to it, which is indicated right below the question name.

Instructions

rubric={points:6}

Follow the CPSC 330 homework instructions, which include information on how to do your assignment and how to submit your assignment.

Exercise 1: Loading files with Pandas

rubric={points:12}

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 6 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
5	Malbec	Argentina	13.83	3.28	Red	Fruity

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()`.

In [2]:
```
df1 = pd.read_csv('data/wine_1.csv')
df2 = pd.read_csv('data/wine_2.csv', header=1)
df3 = pd.read_csv('data/wine_3.csv', skiprows=[6, 7])
df4 = pd.read_csv('data/wine_4.txt', sep='\t')
df5 = pd.read_csv('data/wine_5.csv', usecols=range(7))
df6 = pd.read_csv('data/wine_6.txt', sep='\t', header=1, skiprows=[7, 8], usecols=range(1,8))
```

In [3]:
```
for i, df in enumerate([df2, df3, df4, df5, df6]):
    assert df1.equals(df), f"df1 not equal to df{i + 2}"
print("All tests passed.")
```

```
All tests passed.
```

In [4]:
```
df
```

Out[4]:

	Bottle	Grape	Origin	Alcohol	pH	Colour	Aroma
0	1	Chardonnay	Australia	14.23	3.51	White	Floral
1	2	Pinot Grigio	Italy	13.20	3.30	White	Fruity
2	3	Pinot Blanc	France	13.16	3.16	White	Citrus
3	4	Shiraz	Chile	14.91	3.39	Red	Berry
4	5	Malbec	Argentina	13.83	3.28	Red	Fruity

Exercise 2: The Titanic dataset

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:1}

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

In [5]:
```
titanic_df = pd.read_csv('data/titanic.csv')
```

In [6]:
```
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:2}

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

In [7]:
```
titanic_df.rename(columns={'sibsp': 'siblings_spouses', 'parch': 'parents_children'}, inplace=Tr
```

In [8]:
```
assert set(["siblings_spouses", "parents_children"]).issubset(
    titanic_df.columns
), "Column names were not changed properly"
print("Success")
```

```
Success
```

2(c)

rubric={points:2}

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?

In [9]:
```
type(titanic_df['age'])
```

Out[9]:
```
pandas.core.series.Series
```

2(d)

rubric={points:2}

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

In [10]:
```
type(titanic_df[['age']])
```

Out[10]:
```
pandas.core.frame.DataFrame
```

2(e)

rubric={points:1}

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

In [11]:
```
titanic_df[['pclass', 'survived', 'age']]
```

Out[11]:

	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 × 3 columns

2(f)

rubric={points:2}

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

In [12]:
```
titanic_df.iloc[:5, 2:5]
```

Out[12]:

	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:2}

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

In [13]:
```
titanic_df.loc[:4, 'name':'age']
```

Out[13]:

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

In [14]:
```
# (titanic_df[titanic_df['survived'] == 1]).shape[0]
(titanic_df.query('survived == 1')).shape[0]
```

Out[14]:
```
500
```

2(i)

rubric={points:1}

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

In [15]:
```
# (titanic_df.query('survived == 1 & age > 60')).shape[0]
(titanic_df[(titanic_df['survived'] == 1) & (titanic_df['age'] > 60)]).shape[0]
```

Out[15]:
```
8
```

2(j)

rubric={points:2}

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

In [16]:
```
lowest = titanic_df['fare'].min()
highest = titanic_df['fare'].max()
lowest, highest
```

Out[16]:
```
(0.0, 512.3292)
```

2(k)

rubric={points:1}

Sort the dataframe by fare paid (most to least).

In [17]:
```
titanic_df.sort_values(by=['fare'], ascending=False, inplace=True)
titanic_df.head(10)
```

Out[17]:

	pclass	survived	name	sex	age	siblings_spouses	parents_children	ticket	fare	cabin	emba
183	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	
302	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	
49	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	
50	1	1	Cardeza, Mrs. James Warburton Martinez (Charlo...	female	58.0	0	1	PC 17755	512.3292	B51 B53 B55	
113	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.0000	C23 C25 C27	
114	1	0	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000	C23 C25 C27	
115	1	0	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	C23 C25 C27	
116	1	1	Fortune, Mrs. Mark (Mary McDougald)	female	60.0	1	4	19950	263.0000	C23 C25 C27	
111	1	1	Fortune, Miss. Alice Elizabeth	female	24.0	3	2	19950	263.0000	C23 C25 C27	
112	1	1	Fortune, Miss. Ethel Flora	female	28.0	3	2	19950	263.0000	C23 C25 C27	

2(l)

rubric={points:1}

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

In [18]:
```
titanic_df.to_csv('data/titanic_fares.csv')
```

2(m)

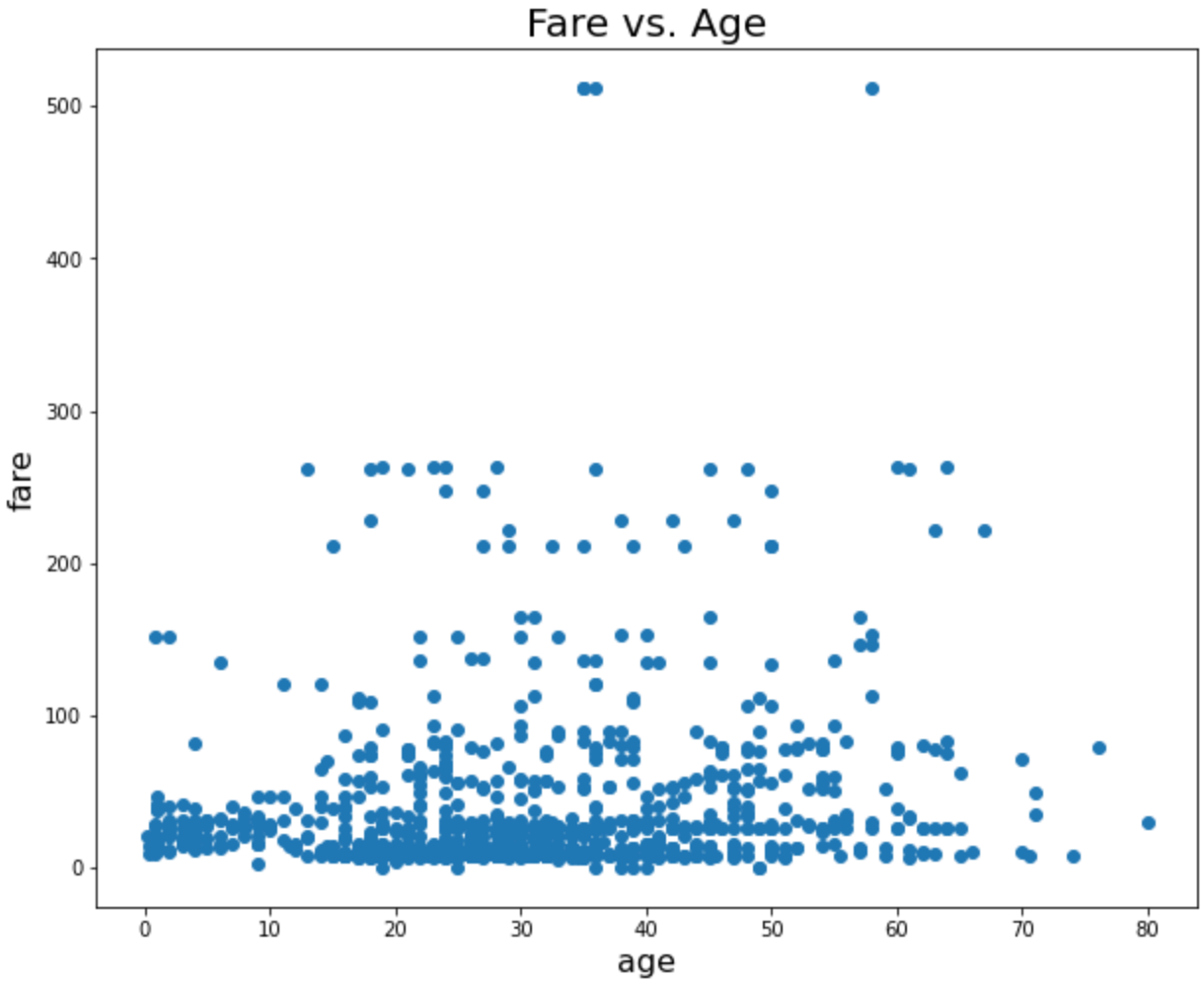
rubric={points:3}

Create a scatter plot of fare (y-axis) vs. age (x-axis). Make sure to follow the guidelines on figures. You are welcome to use pandas built-in plotting or `matplotlib`.

In [19]:
```
_, ax = plt.subplots(figsize=(10,8))
ax.scatter('age', 'fare', data=titanic_df)
```

```
ax.set_xlabel('age', fontsize=16)
ax.set_ylabel('fare', fontsize=16)
ax.set_title('Fare vs. Age', fontsize=20)
```

Out[19]:
```
Text(0.5, 1.0, 'Fare vs. Age')
```



2(n)

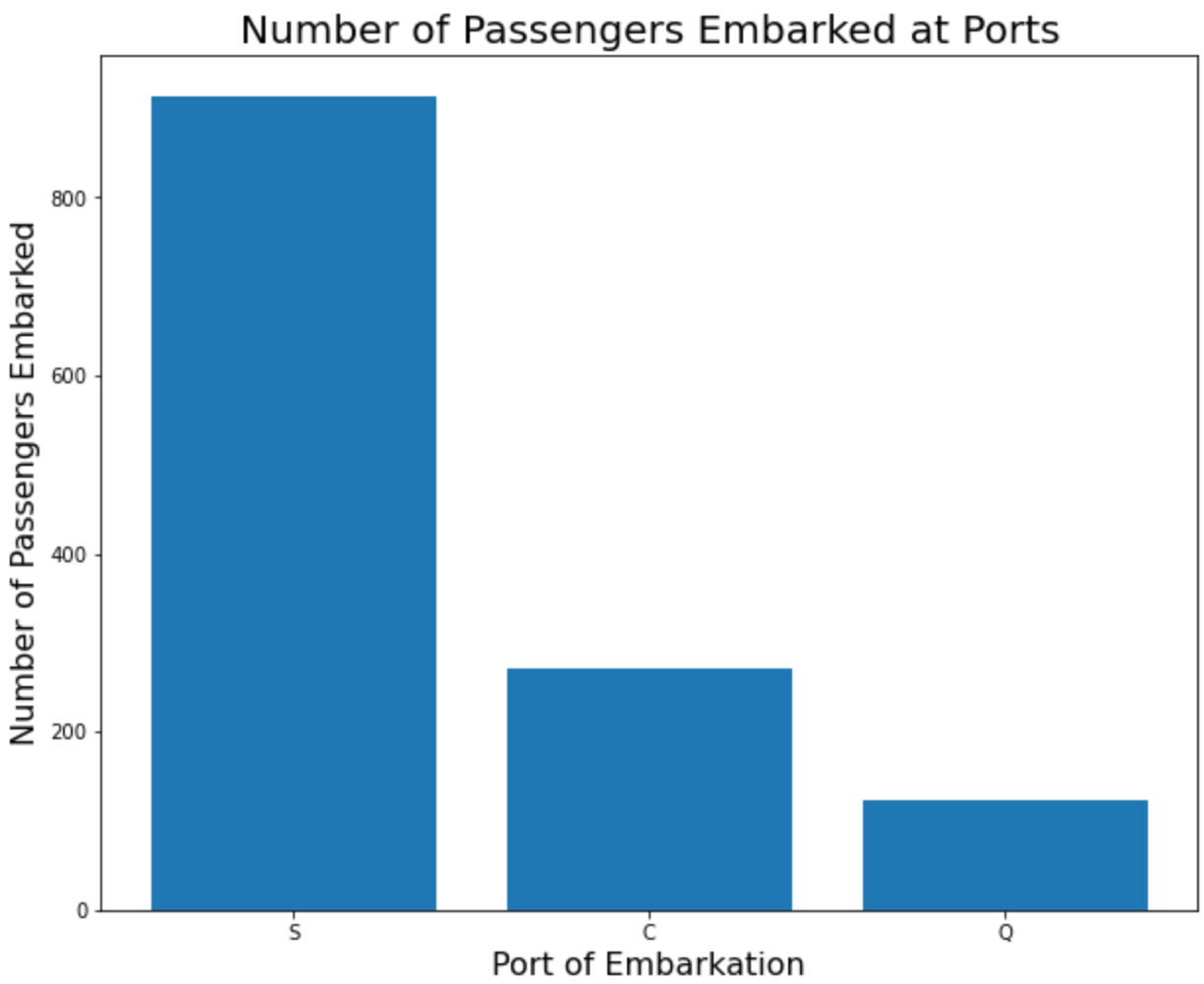
rubric={points:3}

Create a bar plot of `embarked` values.

> Make sure to name the axes and give a title to your plot.

In [20]:
```
counts = titanic_df['embarked'].value_counts()
_, ax = plt.subplots(figsize=(10,8))
ax.bar(counts.index, counts.values)
ax.set_xlabel('Port of Embarkation', fontsize=16)
ax.set_ylabel('Number of Passengers Embarked', fontsize=16)
ax.set_title('Number of Passengers Embarked at Ports', fontsize=20)
```

Out[20]:
```
Text(0.5, 1.0, 'Number of Passengers Embarked at Ports')
```



Exercise 3: Treasure Hunt

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).

In [21]:
```
import string

letters = string.ascii_uppercase
```

The first one is done for you as an example.

Example question

In [22]:
```
x = ("nothing", {-i: l for i, l in enumerate(letters)})
x
```

Out[22]:
```
('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:

In [23]:
```
x[1][-19]
```

Out[23]:
```
'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:2}

In [24]:
```
# 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],
]
```

In [25]:
```
i = letters.find('T')
x[i % 2][i // 2]
```

Out[25]:
```
'T'
```

3(b)

rubric={points:2}

In [26]:
```
# 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"
```

In [27]:
```
# for i in range(100):
#     for j in range(26):
#         if x[i][j] == 'T':
#             break
#     else:
#         continue
#     break

x[95][2]
```

Out[27]:
```
'T'
```

3(c)

rubric={points:3}

In [28]:
```
# 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)
    }
```

In [29]:
```
j = letters.find('T')
x[string.ascii_lowercase[j-2]][string.ascii_lowercase[j+1]][0][0]
```

Out[29]:
```
'T'
```

Submission instructions

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 using Gradescope's drag and drop tool. Check out this Gradescope Student Guide if you need help with Gradescope submission.

In [ ]: