tu11_re_PandasReview_HW

February 28, 2023

1 Pandas Review Homework

Import pandas

[4]:
```
import pandas as pd
```

1.1 1. Make a data frame from a Python dictionary.

Create a Python dictionary containing

- the names of four of your friends (real or imaginary)
- their ages
- the year they started college
- their majors

[2]:
```
roomies = {
    'name': ['Marie','Ryan','Finny','Domenica'],
    'age': ['22','22','22','20'],
    'year': ['2020','2019','2019','2022'],
    'major':['art','biochemistry','physics','physics']
}
```

Make a pandas data frame from your dictionary.

[6]:
```
roomies_df = pd.DataFrame(roomies)
```

Show your new data frame.

[7]:
```
roomies_df
```

[7]:

	name	age	year	major
0	Marie	22	2020	art
1	Ryan	22	2019	biochemistry
2	Finny	22	2019	physics
3	Domenica	20	2022	physics

Fetch the ages of all your friends.

[8]:
```
roomies_df['age']
```

[8]:
```
0    22
1    22
2    22
3    20
Name: age, dtype: object
```

Fetch the name of your fourth friend.

[10]:
```
roomies_df['name'][3]
```

[10]:
```
'Domenica'
```

Fetch the age of your third friend.

[11]:
```
roomies_df['age'][2]
```

[11]:
```
'22'
```

Compute and show the average age of your friends.

[16]:
```
roomies_df['age'].mean() #this is not working hm
```

[16]:
```
5555555.0
```

1.2 2. Find a table of data on Wikipedia and import it.

Go to Widepedia and find a table of data. It can be anything you want.

In the cell below, import the data and display it (first and last five rows).

[22]:
```
young_f1_champs = pd.read_clipboard()
young_f1_champs
```

[22]:

	Driver	Age	Season
1	Germany Sebastian Vettel	23 years, 134 days	2010
2	United Kingdom Lewis Hamilton	23 years, 300 days	2008
3	Spain Fernando Alonso	24 years, 58 days	2005
4	Netherlands Max Verstappen	24 years, 73 days	2021
5	Brazil Emerson Fittipaldi	25 years, 273 days	1972
6	Germany Michael Schumacher	25 years, 314 days	1994
7	Austria Niki Lauda	26 years, 197 days	1975
8	Canada Jacques Villeneuve	26 years, 200 days	1997
9	United Kingdom Jim Clark	27 years, 188 days	1963
10	Finland Kimi Räikkönen	28 years, 4 days	2007

1.3 3. Load the RMS titanic data and export a subset of columns

Load the titanic data, make a new `DataFrame` of the fare paid and the survival columns, and export it as a `.csv` file.

[41]:
```
titanic = pd.read_csv("data/titanic.csv")
titanic_wanted = titanic[['Fare', 'Survived']]
titanic_df = pd.DataFrame(titanic_wanted)
titanic_df.to_csv('titanic_df.csv')
```

Import your new `.csv` file into a new `DataFrame` and show it (first and last five rows).

[42]:
```
titanic_cols = pd.read_csv("titanic_df.csv")
titanic_cols
```

[42]:

	Unnamed: 0	Fare	Survived
0	0	7.2500	0
1	1	71.2833	1
2	2	7.9250	1
3	3	53.1000	1
4	4	8.0500	0
..	…	…	…
886	886	13.0000	0
887	887	30.0000	1
888	888	23.4500	0
889	889	30.0000	1
890	890	7.7500	0

[891 rows x 3 columns]

1.4 4. Fetch specific rows of data of the titanic data

Fetch all the second class passengers of the titanic data and put them in a new `DataFrame` and show it.

[55]:
```
pass_class = titanic['Pclass']
second_class_df = pd.DataFrame(pass_class[pass_class==2])
second_class_df
```

[55]:

	Pclass
9	2
15	2
17	2
20	2
21	2
..	…
866	2
874	2
880	2
883	2
886	2

[184 rows x 1 columns]

Fetch all the first and third class passengers, put them in a new `DataFrame`, and show it.

[58]:
```
first_third_class = pd.DataFrame(pass_class[pass_class!=2])
first_third_class
```

[58]:
```
     Pclass
0         3
1         1
2         3
3         1
4         3
..      …
885       3
887       1
888       3
889       1
890       3

[707 rows x 1 columns]
```

1.5 5. Plot some Titanic data

First, import `matplotlib`

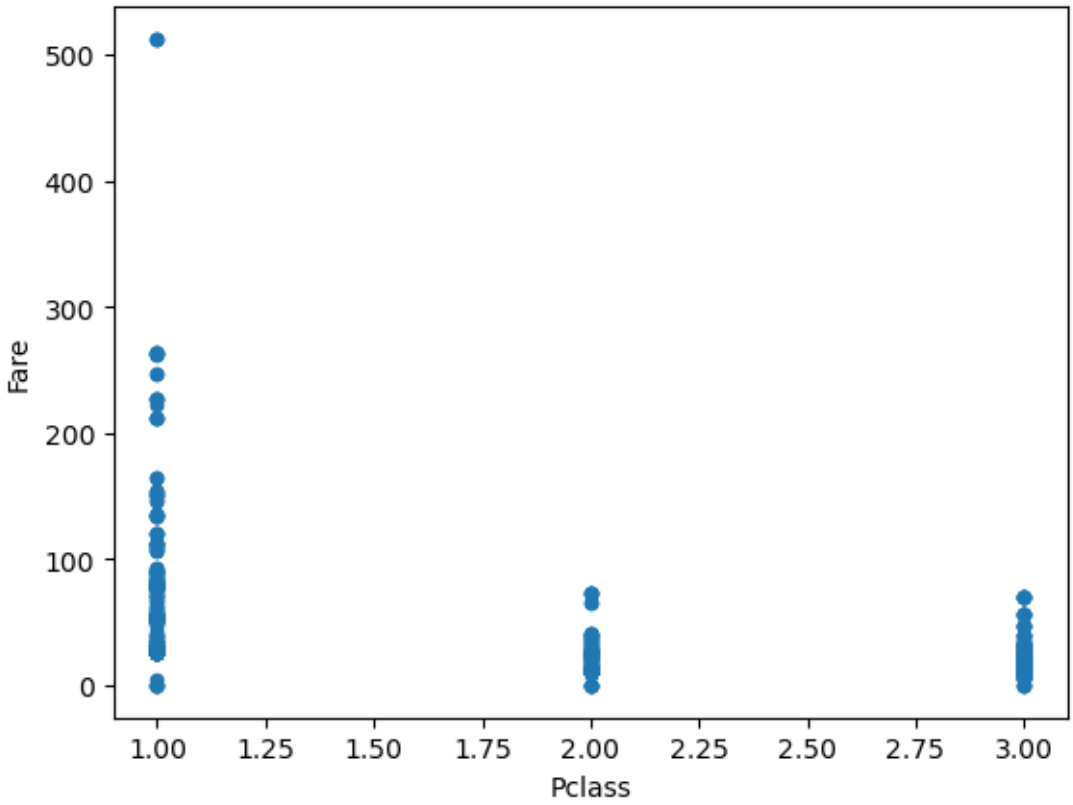
[59]:
```
import matplotlib as plt
```

1.5.1 5.a - Scatter plot

Make a scatter plot of fare vs. cabin class (seems like these should be perfectly related).

[69]:
```
titanic.plot.scatter(x = "Pclass", y = "Fare")
```

[69]:
```
<AxesSubplot:xlabel='Pclass', ylabel='Fare'>
```

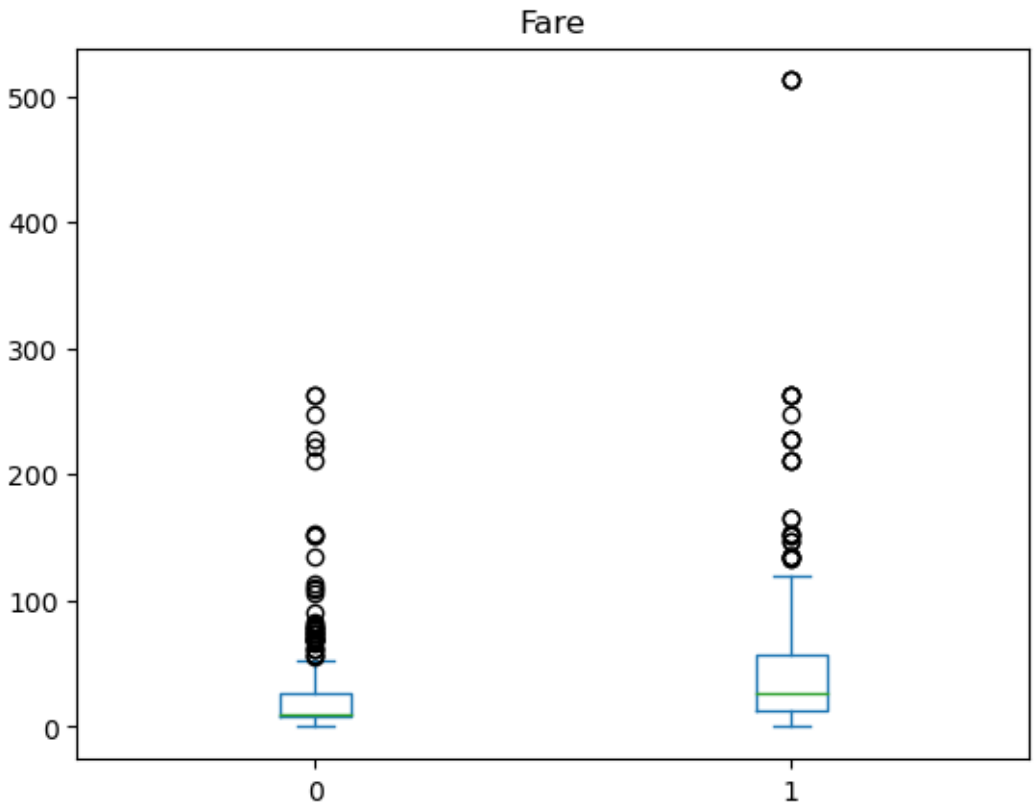


1.5.2 5.b - Distribution plot (challenging!)

Plot the distributions of fare paid for survivors and deceased in a way that makes for a good visual comparison.

[83]:
```
titanic.plot.box(column = "Fare", by = "Survived")
```

[83]:
```
Fare    AxesSubplot(0.125,0.11;0.775x0.77)
dtype: object
```



1.6 6. Calculate new columns

1.6.1 6.a - Compute total number of relatives

Create a new column in your titanic `DataFrame` quantifying the total number of relatives on board (siblings + parents – the number of siblings are in `SibSp` and the number of parents are in `Parch`).

[92]:
```
titanic['total_relatives'] = titanic['SibSp'] + titanic['Parch']
titanic
```

[92]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3
```

```
                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                            Montvila, Rev. Juozas    male  27.0      0
887                     Graham, Miss. Margaret Edith  female  19.0      0
888          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                            Behr, Mr. Karl Howell    male  26.0      0
890                              Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked  total_relatives  \
0        0         A/5 21171   7.2500   NaN        S                1
1        0          PC 17599  71.2833   C85        C                1
2        0  STON/O2. 3101282   7.9250   NaN        S                0
3        0            113803  53.1000  C123        S                1
4        0            373450   8.0500   NaN        S                0
..     …               …      …   …      …              …
886      0            211536  13.0000   NaN        S                0
887      0            112053  30.0000   B42        S                0
888      2        W./C. 6607  23.4500   NaN        S                3
889      0            111369  30.0000  C148        C                0
890      0            370376   7.7500   NaN        Q                0

     relatives?
0         False
1         False
2         False
3         False
4         False
..          …
886       False
887       False
888        True
889       False
890       False

[891 rows x 14 columns]
```

1.6.2 6.b - Did a person have any relatives on board?

Add another column – a Boolean column – indicating whether each person had any relatives on board.

[98]:
```
titanic['any_relatives'] = titanic['total_relatives'] > 1
titanic
```

[98]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	total_relatives	any_relatives
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	1	False
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1	0	PC 17599	71.2833	C85	C	1	False
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	0	False
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1	False
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0	False
..	…	…	…	…	…	…	…	…	…	…	…	…	…	…
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	0	False
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	0	False
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	3	True
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	0	False
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	0	False

```
4            False
..             …
886          False
887          False
888           True
889          False
890          False

[891 rows x 14 columns]
```

1.7 7. Computing descriptive statistics

1.7.1 7.a - Compute a mean for a column

Compute the proportion of survivors of the RMS Titanic. **Hint**: the coding of `Survival` as 0 or 1 really works to our advantage here: the proportion of survivors in any group is easily computed using a common statistical function. The 7.a section header should also give you a big clue!

[99]:
```
titanic['Survived'].mean()
```

[99]:
```
0.3838383838383838
```

1.7.2 7.a - Compute a mean for a subset of data

Compute the proportion of survivors for the females on the RMS Titanic (you can do this in one go, or two steps, using an intermediate object containing just the female data).

[104]:
```
female = titanic[titanic['Sex'] == 'female']
female['Survived'].mean()
```

[104]:
```
0.7420382165605095
```

1.7.3 7.b - Compute statistics by group

Compute the proportion of female vs. male survivors of the RMS Titanic.

[106]:
```
male = titanic[titanic['Sex'] == 'male']
female['Survived'].mean()/male['Survived'].mean()
```

[106]:
```
3.928037164728569
```

Now compute the proportion of female vs. male survivors of the RMS Titanic, *along with the* ***s****tandard* ***e****rror of the* ***m****ean*. The **bold** type should give you a hint about the name of the method to compute the standard error. To do this, you'll need to combine the `groupby()` and `agg()` methods!

[114]:
```
sex_diff = titanic[['Survived', 'Sex']].groupby('Sex').mean()
sex_diff
```

[114]:
```
        Survived
Sex
female  0.742038
male    0.188908
```

[116]:
```
sex_dict = {
    'Survived':['mean','std','sem']
}
sex_diff.agg(sex_dict)
```

[116]:
```
      Survived
mean  0.465473
std   0.391122
sem   0.276565
```

What does this tell you about gender roles when the RMS Titanic was sunk?

There was difference between who survived, which makes sense given that women and children were most likely saved first.

Compute the proportion of survivors by cabin class and their standard error.

[126]:
```
first = titanic[titanic['Pclass']==1]
print(first['Survived'].mean())
print(first['Survived'].sem())
```

```
0.6296296296296297
0.03293377139415192
```

[127]:
```
second = titanic[titanic['Pclass']==2]
print(second['Survived'].mean())
print(secondt['Survived'].sem())
```

```
0.47282608695652173
0.03293377139415192
```

[128]:
```
third = titanic[titanic['Pclass']==3]
print(third['Survived'].mean())
print(third['Survived'].sem())
```

```
0.24236252545824846
0.019358219881041493
```

What does this tell you about socio-economic status when the RMS Titanic was sunk?

The higher the passenger class the more likely they were to survive.