Introduction for challenge problem solution

1. Collect and clean data

Get data in csv format and save as “bikes.csv”. Add it to the Pycharm project by using: “pd.read_csv()”. Due to the 1st to the 6th row are useless for our following operations, add “header = 7” to the pd.read operation, which means the program starts read the sheet from the 7th row.

Drop columns ”gender_Feamle”, ”Wearing Helmet_No”, ”Passenger on the Bike_No”, “Riding on the sidewalk_No”, and for the remaining columns, replace empty cells with 0 in order for the convenience of judgement.

```
# pretreatment
# bikes = pd.read_csv('C:\\Users\\Think\\Desktop\\6200\\challenge Q\\bikes.csv', header=7)
bikes = pd.read_csv('bikes.csv', header=7)
df0 = bikes.drop(columns=["Female", "No", "No.1", "No.2"])
df1 = df0.fillna(0)
df1.columns = ['datetime', 'gender', 'had_helmet', 'had_passenger', 'on_sidewalk']
```

In the following operation, in order to get a form like the example, we set columns like the examples shown. In datetime column, a loop is set to give all the items a datetime. Specifically, If the cell in datetime row is empty (now it is filled with 0), we fill it with the former date time. If it is not empty, we keep the value in this cell and meanwhile save the value for the use of next empty cell. All the value are added with a date: “2010-9-24”.

In gender row, we replace “X” with “Male”, “0” with “Female”, and in other row, replace “X” with “Yes”, and “0”with “No”. Here we finish the question of problem 1.

```python
# problem 1
a = '7:00'
m = 0
for i in df1['datetime']:
    if i == 0:
        df1.iloc[m, 0] = "2010-9-24 " + a
    else:
        a = i
        df1.iloc[m, 0] = "2010-9-24 " + i
    m += 1
df1["gender"].replace("X", 'Male', inplace=True)
df1["gender"].replace(0, 'Female', inplace=True)
df1.replace(0, "No", inplace=True)
df1.replace("X", "Yes", inplace=True)
df1["datetime"] = pd.to_datetime(df1["datetime"])
print(df1)
df1.to_csv('out.csv')
```

	A	B	C	D	E	F	G
1		datetime	gender	had_helme	had_passe	on_sidewalk	
2	0	2010/9/24 7:00	Male	No	No	No	
3	1	2010/9/24 7:00	Male	No	No	No	
4	2	2010/9/24 7:00	Male	No	No	No	
5	3	2010/9/24 7:00	Male	No	No	No	
6	4	2010/9/24 7:00	Male	No	No	No	
7	5	2010/9/24 7:15	Male	Yes	No	No	
8	6	2010/9/24 7:15	Female	Yes	No	No	
9	7	2010/9/24 7:15	Male	Yes	No	No	
10	8	2010/9/24 7:15	Female	Yes	No	Yes	
11	9	2010/9/24 7:15	Male	Yes	No	No	
12	10	2010/9/24 7:15	Male	Yes	No	No	
13	11	2010/9/24 7:15	Male	Yes	No	No	
14	12	2010/9/24 7:15	Female	Yes	No	No	
15	13	2010/9/24 7:30	Male	No	No	No	
16	14	2010/9/24 7:30	Male	Yes	No	No	
17	15	2010/9/24 7:30	Male	No	No	No	
18	16	2010/9/24 7:30	Female	Yes	No	No	
19	17	2010/9/24 7:30	Male	No	No	No	
20	18	2010/9/24 7:30	Female	Yes	No	No	
21	19	2010/9/24 7:30	Male	No	No	Yes	
22	20	2010/9/24 7:30	Female	No	No	No	
23	21	2010/9/24 7:30	Male	No	No	No	
24	22	2010/9/24 7:45	Male	Yes	No	No	
25	23	2010/9/24 7:45	Female	Yes	No	No	
26	24	2010/9/24 7:45	Female	Yes	No	No	

2. Gather summary statistics

Using statement: ".describe()", we get the summary of the statistics.

```
# problem 2
df2 = df1.describe(include="all")
print(df2)
df2.to_csv("out2.csv")
```

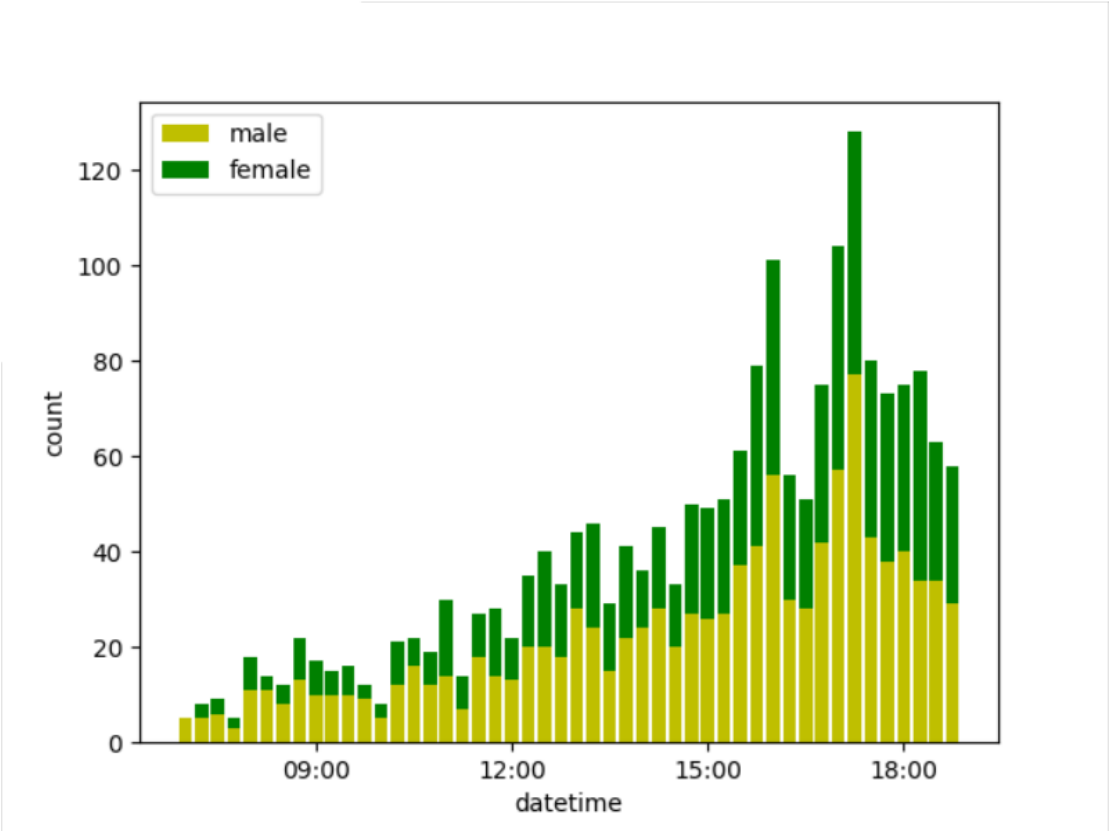
	A	B	C	D	E	F
1		datetime	gender	had_helme	had_passe	on_sidewalk
2	count	1958	1958	1958	1958	1958
3	unique	48	2	2	2	2
4	top	2010/9/24 17:15	Male	No	No	No
5	freq	128	1097	1007	1953	1885
6	first	2010/9/24 7:00				
7	last	2010/9/24 18:45				
8						

3. Visualization

To display the data of gender by date, we can abstract the gender data by using ".groupby()" and get the data of "datetime" and "gender". Then, abstract columns "datetime" as index, and "Male", "Female" as two series. Draw plot according to this and put Male number Histogram at the bottom of Female number Histogram. To adjust the scale of axis properly, we can set four point as "09:00"…"18:00", from the 8th data to the 48th data, with distance between two data of 12.

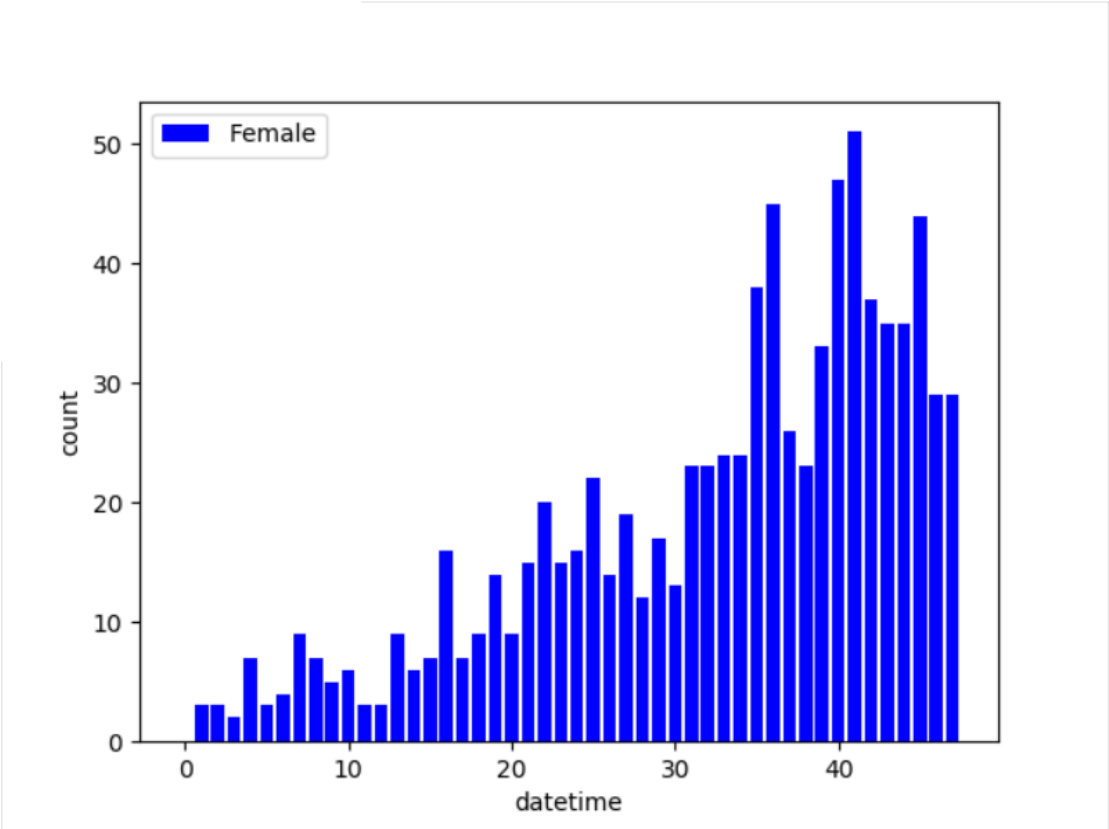
```
# problem 3
df3 = df1.groupby(by=["datetime", "gender"])
df4 = df3.size().unstack()
df4 = df4.fillna(0)
print(df4)
datetime_list = df4.index
Male_number = df4["Male"]
Female_number = df4["Female"]
plt.bar(range(len(Male_number)), Male_number, label='male', fc='y')
plt.bar(range(len(Female_number)), Female_number, bottom=Male_number, label='female', tick_label=datetime_list, fc='g')
plt.legend()
plt.xlabel("datetime")
plt.ylabel("count")
plt.xticks(np.arange(8, 48, 12), ("09:00", "12:00", "15:00", "18:00"))
plt.show()
```

And plot is shown as above:



4. Distribution Fitting

To fitting the distribution of female number in each time, we get the pdf of each time by letting the female number of each time divided by the total number. Draw the plot of the Female as following:



By observing we can know that the distribution is discrete, bounded and non-negative. Calculate useful statistics as following:

```
Mean = 0
i = 0
for index in range(len(Female_pdf)):
    Mean = Mean + new_index[i] * Female_pdf[i]
    i += 1
print("Mean= ", Mean)

Mean_2 = 0
i = 0
for index in range(len(Female_pdf)):
    Mean_2 = Mean_2 + (new_index[i]**2) * Female_pdf[i]
    i += 1
print("Mean_2= ", Mean_2)
```

```
Variance = 0
for index in range(len(new_index)):
    Variance = Variance + np.square(new_index[j] - Mean) * Female_pdf[j]
    j += 1
Variance = np.sum(Female_variance)
print("Var =", Variance)
print("new_index[0]=", new_index[0])
```

```
Std = np.sqrt(Variance)
Cov = Std / Mean
print("cov =", Cov)
Lexis = Variance / Mean
print("Lexis =", Lexis)


Ske = 0
k = 0
for index in range(len(Female_pdf)):
    Ske = Ske + pow(new_index[k] - Mean, 3)/48
    k += 1
print(Ske)
Skewness = Ske / pow(Variance, 1.5)
print("Skewness =", Skewness)
```

And get result:

```
Name: Female, dtype: float64
Mean= 0.702477739063105
Var = 1.0
new_index[0]= 0.020833333333333332
cov = 1.4235326536235877
Lexis = 1.4235326536235877
-0.05507907240607825
Skewness = -0.05507907240607825
```

Due to skewness < 0, the data is skewed to the left. Thus we can assume the data is beta distributed. Due to the range of beta is [0,1], we convert the range of index into [0,1] by letting them divided by 48.

```
new_index = np.arange(1.00, 49.00)
counter = 0
for index in range(len(new_index)):
    new_index[counter] = new_index[counter] / 48
    counter += 1
print(new_index)
```

Using moment estimate method, we get the value of the parameters of beta distribution.

```
def f(a1):
    x = float(a1[0])
    y = float(a1[1])
    return [
        x/(x+y),
        (x * y)/(np.square(x + y) * (x + y + 1))+x**2/((x+y)**2)
    ]


a0 = [Mean, Mean_2]
result = fsolve(f, a0)
print(result)
# print(result[0], result[1])
```

And get result:

```
[5.03583692e-19 1.55016595e+00]
```

Then we continue chi-square test:

```
def b(x):
    float(x)
    return (1/beta(result[0], result[1]))*(x**(result[0]-1))*((1-x)**(result[1]-1))


i = 10
chi = 0
while i <= 40:
    chi = chi + (Female_number[i]**2)/(861*b(new_index[i]))
    i += 1
chi = chi - 861
print("chi teat result =", chi)

if chi > 65:
    print("the chi-test is not passed")
else:
    print("the chi-test is passed")
```

And the result:

```
chi teat result = 5.032862758755806e+19
the chi-test is not passed

Process finished with exit code 0
```

We can see the test is not passed, so the fitting to beta distribution is failed.

Maybe another distribution should be tested.