Cal Colistra Github link: https://github.com/CalColistra/IDS/tree/main/Project4

Introduction to Data Science

Project 4

3/16/2022

Purpose:

The purpose of this project is to find relationships between various fields in a dataset about adults. This dataset shows information about adults such as their age, workclass, education, marital status, occupation, sex, capital gain, capital loss, and income. In the second problem of this project I worked with a different dataset about nutrition. Similarly, the purpose of this part of the project is to find relationships between different food items, their weight in grams, saturated fat, and cholesterol.

Methodology:

To easily evaluate the data in the adults dataset, I made various data frames, contingency tables and bar graphs that were based on the contingency tables. I also looked at what changes when I sort the data to show only those who are over 40. Lastly, I identified outliers by standard scaling the age field and I noticed that there were 60 rows of those who are at least 80 years old. For the nutrition dataset I again made dataframes for easy evaluation. I created a new variable, saturated fat per gram, by dividing the values in the saturated fat column by the values in the weight in grams column. I also standard scaled the new saturated fat per gram column and found that there were only outliers on the high end of the scale and not on the low end. Lastly, I plotted the scaled saturated fat per gram column in a density plot to show the distribution.

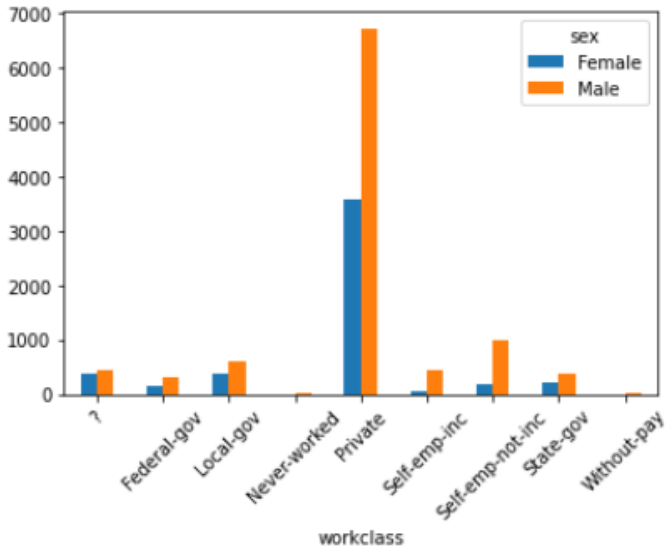
Summary:

Problem 1 Python: https://github.com/CalColistra/IDS/blob/main/Project4/problem1/problem1_python.ipynb

Create a contingency table of workclass and sex, and plot it.

```
#Cross tabulation of workclass and sex:
p_crosstab = pd.crosstab(adults.workclass, adults.sex)
print(p_crosstab)
#plot it:
barplot = p_crosstab.plot.bar(rot=45)
```

```
sex               Female  Male
workclass
?                    377   452
Federal-gov          149   305
Local-gov            377   592
Never-worked           1     4
Private             3574  6707
Self-emp-inc          54   444
Self-emp-not-inc     178   992
State-gov            201   385
Without-pay            1     4
```

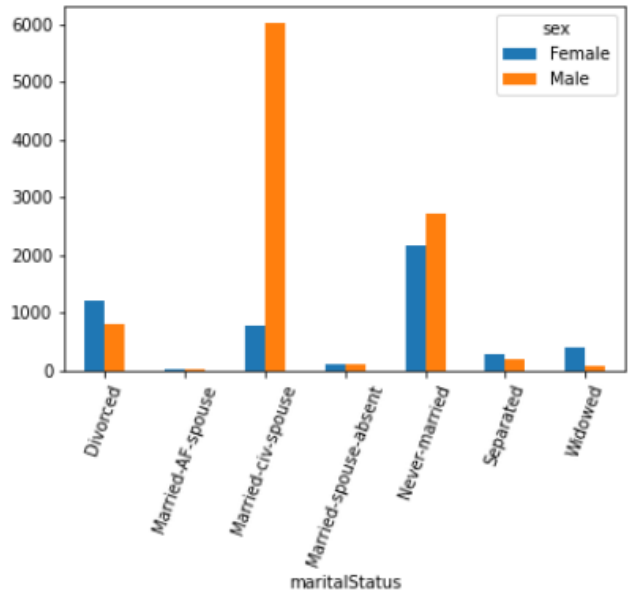


The contingency table between work class and sex shows how the workclass of the majority of people in the dataset is private and, of those who have a private workclass, there is almost double the amount of males than there are females.

Create a contingency table of sex and marital status, and plot it.

```
[6]  #Cross tabulation of sex and marital status:
     p_crosstab = pd.crosstab(adults.maritalStatus, adults.sex)
     print(p_crosstab)
     #plot it:
     barplot = p_crosstab.plot.bar(rot=70)
```

sex	Female	Male
maritalStatus		
Divorced	1219	795
Married-AF-spouse	7	4
Married-civ-spouse	761	6010
Married-spouse-absent	95	104
Never-married	2160	2717
Separated	290	182
Widowed	380	73



The contingency table between sex and marital status shows how most of the dataset is either married-civ-spouse or never married. For married-civ-spouse, there are much more males than females and for never married there is a more even ratio between males and females because both males and females fall in the range: 2100-3000.

Create a new data frame, adultOver40, for those whose age > 40.

```
[8] #copy original df into new one:
    adultOver40 = adults.copy()
    #drop all rows where age is under 41:
    adultOver40.drop(adultOver40[adultOver40['age'] < 41].index, inplace = True)
    #print:
    adultOver40
```

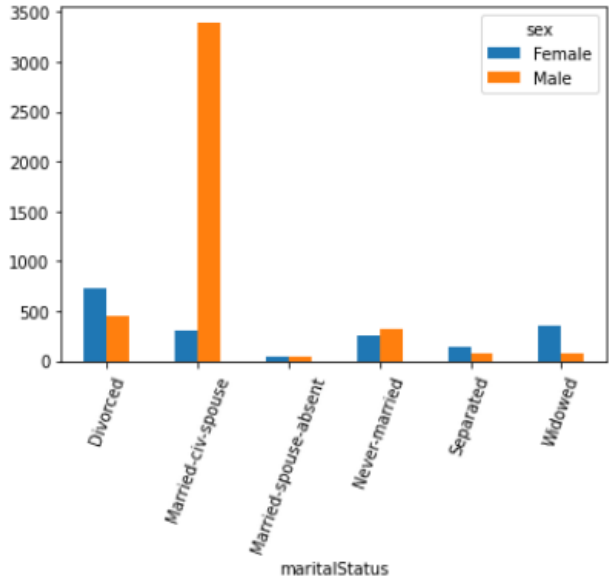
	age	workclass	education	maritalStatus	occupation	sex	capitalGain	capitalLoss	income
0	50	Self-emp-not-inc	13	Married-civ-spouse	Exec-managerial	Male	0	0	<=50K
2	49	Private	5	Married-spouse-absent	Other-service	Female	0	0	<=50K
3	52	Self-emp-not-inc	9	Married-civ-spouse	Exec-managerial	Male	0	0	>50K
7	43	Private	7	Married-civ-spouse	Transport-moving	Male	0	2042	<=50K
8	54	?	10	Married-civ-spouse	?	Male	0	0	>50K
...	...	...	...	...	...	...	...	...	...
14787	45	Local-gov	12	Divorced	Prof-specialty	Female	0	0	<=50K
14790	65	Self-emp-not-inc	15	Never-married	Prof-specialty	Male	1086	0	<=50K
14791	43	State-gov	10	Divorced	Adm-clerical	Female	0	0	<=50K
14792	43	Self-emp-not-inc	10	Married-civ-spouse	Craft-repair	Male	0	0	<=50K
14795	58	Private	9	Widowed	Adm-clerical	Female	0	0	<=50K

6144 rows × 9 columns

Recreate the contingency table of sex and marital status for adultOver40.

```
[9] #Cross tabulation of sex and marital status:
    p_crosstab = pd.crosstab(adultOver40.maritalStatus, adultOver40.sex)
    print(p_crosstab)
    #plot it:
    barplot = p_crosstab.plot.bar(rot=70)
```

```
sex                    Female  Male
maritalStatus
Divorced                  720   448
Married-civ-spouse        305  3383
Married-spouse-absent      38    47
Never-married             256   315
Separated                 133    74
Widowed                   354    71
```



When I removed those who are under 41 years old, I noticed that there are much less who fall under the never married category. The category that has the most females is divorced and the category with the most males is married-civ-spouse.

Determine whether any outliers exist for the education field.

```
[ ] #import & initialize the standard scaler:
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    #scale "education" column and place values in new column:
    adults['education_scaled'] = scaler.fit_transform(adults[["education"]])
    #check if mean = 0 after standardization:
    m1 = adults['education_scaled'].mean()
    print("The new shifted mean: %6.6f" % (m1))
```

```
The new shifted mean: -0.000000
```

```
[ ] #make new df with outliers for education:
    education_outliers = adults.query('(education_scaled > 3) | (education_scaled < -3)')
    #print how many outliers are there:
    print("number of outliers in education =", len(education_outliers))
```

```
number of outliers in education = 113
```

Standardize (zero mean) the age variable, and identify how many outliers there are. What is the most extreme outlier?

```
[ ] #scale "age" column and place values in new column:
    adults['age_scaled'] = scaler.fit_transform(adults[["age"]])
    #check if mean = 0 after standardization:
    m1 = adults['age_scaled'].mean()
    print("The new shifted mean: %6.6f" % (m1))
    #make new df with outliers for age:
    age_outliers = adults.query('(age_scaled > 3) | (age_scaled < -3)')
    #print how many outliers there are:
    print("Number of outliers in 'age' column = ", len(age_outliers))
```

```
The new shifted mean: 0.000000
Number of outliers in 'age' column =  60
```

```
[ ] #show the most extreme outliers:
    age_outliers.sort_values(by='age_scaled', ascending = False)
```

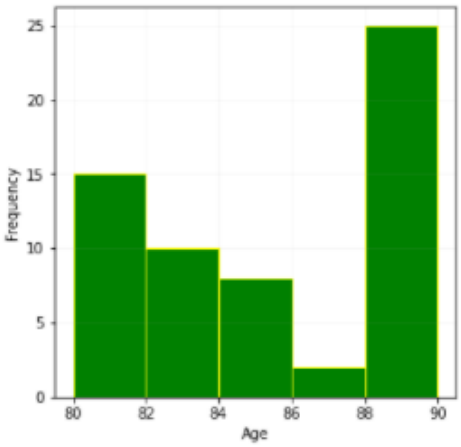
Those of age 90 are tied for the most extreme outlier because they have the highest standard deviation of 3.75.

Age anomaly? Select only records with age at least 80, and construct a histogram of age. Explain what you see.

```
[ ] #make a new df with only those of age over 79:
    over_79 = adults.query('age > 79')
    print("There are", len(over_79), "rows with age of at least 80")
```

```
There are 60 rows with age of at least 80
```

```
#plot a histogram of rows with age of at least 80:
import matplotlib.pyplot as plt
plt.figure(figsize=(5, 5))
plt.hist(over_79['age'], bins=5, color="green", edgecolor="yellow")
plt.xlabel('Age')
plt.ylabel('Frequency')
# alpha is the % of transparency
plt.grid(axis='y', alpha=0.1)
plt.grid(axis='x', alpha=0.1)
plt.show()
```



Of the 60 outliers, most (25) of them are in range [88, 90] and only about 2 are in range [86,88].

Problem 1 R: https://github.com/CalColistra/IDS/blob/main/Project4/problem1/problem1_r.ipynb

Problem 2 Python: https://github.com/CalColistra/IDS/blob/main/Project4/problem2/problem2_python.ipynb

Sort the data set by saturated_ fat and produce a listing of the five food items highest in saturated fat. Comment on the validity of comparing food items of different sizes.

```
nutrition.sort_values(by='saturatedFat', ascending =False).head(5)
```

	foodItem	weightInGrams	saturatedFat	cholesterol
378	CHEESECAKE 1 CAKE	1110.0	119.9	2053
535	ICE CREAM; VANLLA; RICH 16% FT1/2 GAL	1188.0	118.3	703
458	YELLOWCAKE W/ CHOCFRSTNG;COMML1 CAKE	1108.0	92.0	609
581	CREME PIE 1 PIE	910.0	90.1	46
890	LARD 1 CUP	205.0	80.4	195

Comparing food items of different sizes means that the larger items will most likely have larger amounts of saturatedFat and cholesterol. It is important to keep this in mind because if a small item is compared to a large item there will be a big difference in the results.

Create a new variable, saturated_ fat_ per_gram, by dividing the amount of saturated fat by the weight in grams. Sort the data set by saturated_ fat_ per_gram and produce a listing of the five food items highest in saturated fat per gram. Which food has the most saturated fat per gram?

```
[ ] #make a new column and put the saturated_fat_per_gram into it:
    nutrition['saturated_fat_per_gram'] = nutrition.apply(
        lambda row: row['saturatedFat']/row['weightInGrams'],
        axis = 1)
    #print the first five in descending order:
    nutrition.sort_values(by='saturated_fat_per_gram', ascending =False).head(5)
```

	foodItem	weightInGrams	saturatedFat	cholesterol	saturated_fat_per_gram
908	BUTTER; SALTED 1 TBSP	14.0	7.1	31	0.507143
909	BUTTER; UNSALTED 1 TBSP	14.0	7.1	31	0.507143
710	BUTTER; UNSALTED 1/2 CUP	113.0	57.1	247	0.505310
709	BUTTER; SALTED 1/2 CUP	113.0	57.1	247	0.505310
913	BUTTER; UNSALTED 1 PAT	5.0	2.5	11	0.500000

Butter; Salted, 1 TBSP has the most saturated fat per gram

Standardize (zero mean) the field saturated_ fat_ per_gram, and produce a listing of all the food items that are outliers at the high end of the scale. How many food items are outliers at the low end of the scale?

```
[ ] #import & initialize the standard scaler:
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    #scale "saturated_fat_per_gram" column and put values into new column:
    nutrition[["saturated_fat_per_gram_scaled"]] = scaler.fit_transform(nutrition[["saturated_fat_per_gram"]])
```

```
[ ] #check if mean = 0 after standardization:
    m1 = nutrition['saturated_fat_per_gram_scaled'].mean()
    print("The new shifted mean: %6.6f" % (m1))
```

```
The new shifted mean: 0.000000
```

```
#make & print new df with high end outliers for saturated_fat_per_gram:
nutrition_outliers = nutrition.query('(saturated_fat_per_gram_scaled > 3)')
nutrition_outliers.sort_values(by='saturated_fat_per_gram_scaled', ascending =False)
```

	foodItem	weightInGrams	saturatedFat	cholesterol	saturated_fat_per_gram	saturated_fat_per_gram_scaled
908	BUTTER; SALTED 1 TBSP	14.00	7.1	31	0.507143	7.110475
909	BUTTER; UNSALTED 1 TBSP	14.00	7.1	31	0.507143	7.110475
709	BUTTER; SALTED 1/2 CUP	113.00	57.1	247	0.505310	7.082741
710	BUTTER; UNSALTED 1/2 CUP	113.00	57.1	247	0.505310	7.082741
912	BUTTER; SALTED 1 PAT	5.00	2.5	11	0.500000	7.002408
913	BUTTER; UNSALTED 1 PAT	5.00	2.5	11	0.500000	7.002408

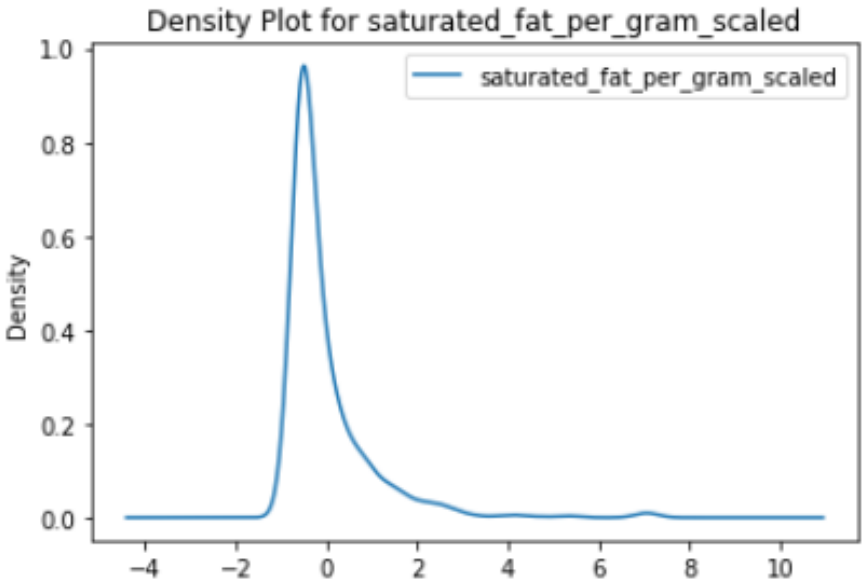
```
[ ] #make & print new df with low end outliers for saturated_fat_per_gram:
    nutrition_outliers2 = nutrition.query('(saturated_fat_per_gram_scaled < -3)')
    nutrition_outliers2.sort_values(by='saturated_fat_per_gram_scaled', ascending =True)
    #seems like there are no outliers below -3 standard deviations
```

foodItem	weightInGrams	saturatedFat	cholesterol	saturated_fat_per_gram	saturated_fat_per_gram_scaled

There are no outliers below -3 standard deviations.

Create a density plot for saturated_ fat_ per_gram.

```
[ ]  #make a new df with just "saturated_fat_per_gram_scaled" :
     sat = [nutrition["saturated_fat_per_gram_scaled"]]
     headers = ["saturated_fat_per_gram_scaled"]
     sat2 = pd.concat(sat, axis=1, keys=headers)
     #make a density plot for new df:
     import matplotlib.pyplot as plt
     sat2.plot.density()
     plt.title('Density Plot for saturated_fat_per_gram_scaled')
     plt.show()
```



Problem 2 R: https://github.com/CalColistra/IDS/blob/main/Project4/problem2/problem2_r.ipynb

Conclusion:

Depending on how I sorted or calculated the datasets I noticed that I can gain insight on many different aspects of the datasets. For example, when I created contingency tables for sex and marital status in the dataset about adults, my results were different than when I removed the data for all people under 40. This demonstrates how I could do the same for any age group or even a specific age if I want to gain more insight on how it affects the other parts of the dataset. Another example is when I identified outliers in the age variable. I was also able to plot a histogram of those who are at least 80 years old and this showed how the outliers were distributed. I noticed that a majority of those who are over 80 fell into one category, the 88-90 age threshold. In the nutrition dataset I was able to sort to see the food items with the highest saturated fat. Doing this made me notice that some items have different sizes than others which is important to keep in mind when comparing foods of different sizes. Next I made a new column called saturated fat per gram which consisted of the saturated fat divided by the weight in grams. This is a useful tool because there are plenty of other calculations that may allow me to gain deeper insight of other variables in other datasets.

References:

https://realpython.com/pandas-sort-python/#:~:text=To%20sort%20the%20DataFrame%20based,not%20modify%20the%20original%20DataFrame.

https://www.adamsmith.haus/python/answers/how-to-create-a-pandas-dataframe-from-columns-in-other-dataframes-in-python

https://stackoverflow.com/questions/44327341/how-to-print-certain-rows-of-a-dataframe-with-a-condition

https://www.programmingr.com/examples/r-dataframe/sort-r-data-frame/