HW4

December 5, 2022

[1]:
```
#In this cell, we import all the info needed - pandas, numpy, matplotlib for
 ↪data visualization, and we import our data - diabetes.csv
%matplotlib inline
import pandas as pd
import numpy as np
db_df = pd.read_csv("diabetes.csv")
print(db_df)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...  ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

[2]:
```
#from cells 2 to 9, I am checking if all the data is in the correct format for␣
 ↪later use
#(I need them all to be float types as they are all numerical values
#(integer type would cause inaccurate data as it would round up or down)
#I was lucky as this data file was pretty good. after checking it seems that␣
 ↪they are all float types, so it did not need to be altered
db_df['Pregnancies'].describe()
```

[2]:
```
count    768.000000
mean       3.845052
std        3.369578
min        0.000000
25%        1.000000
50%        3.000000
75%        6.000000
max       17.000000
Name: Pregnancies, dtype: float64
```

[3]:
```
db_df['Glucose'].describe()
```

[3]:
```
count    768.000000
mean     120.894531
std       31.972618
min        0.000000
25%       99.000000
50%      117.000000
75%      140.250000
max      199.000000
Name: Glucose, dtype: float64
```

[4]:
```
db_df['SkinThickness'].describe()
```

[4]:
```
count    768.000000
mean      20.536458
std       15.952218
min        0.000000
25%        0.000000
50%       23.000000
75%       32.000000
max       99.000000
Name: SkinThickness, dtype: float64
```

[5]:
```
db_df['Insulin'].describe()
```

[5]:
```
count    768.000000
mean      79.799479
std      115.244002
```

```
min        0.000000
25%        0.000000
50%       30.500000
75%      127.250000
max      846.000000
Name: Insulin, dtype: float64
```

[6]:
```
db_df['BMI'].describe()
```

[6]:
```
count    768.000000
mean      31.992578
std        7.884160
min        0.000000
25%       27.300000
50%       32.000000
75%       36.600000
max       67.100000
Name: BMI, dtype: float64
```

[7]:
```
db_df['DiabetesPedigreeFunction'].describe()
```

[7]:
```
count    768.000000
mean       0.471876
std        0.331329
min        0.078000
25%        0.243750
50%        0.372500
75%        0.626250
max        2.420000
Name: DiabetesPedigreeFunction, dtype: float64
```

[8]:
```
db_df['Age'].describe()
```

[8]:
```
count    768.000000
mean      33.240885
std       11.760232
min       21.000000
25%       24.000000
50%       29.000000
75%       41.000000
max       81.000000
Name: Age, dtype: float64
```

[9]:
```
db_df['Outcome'].describe()
```

[9]:
```
count    768.000000
mean       0.348958
```

```
std         0.476951
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
Name: Outcome, dtype: float64
```

[10]:
```
#since there is a large amount of data, Jupyterhub isnt printing all 700+ rows, so I can't see if any Nan values might exist.
#this code is meant to check if any exist, so I can deal with them and change them to 0 if they exist
#since 'false' was printed, that means no Nan values exist. We have finished cleaning up the data at this point.
db_df.isnull().values.any()
```

[10]:
```
False
```

[11]:
```
# QUESTION 1 - IS THE AVERAGE BMI OF THOSE WITH DIABETES LOWER OR HIGHER?
#Average BMI of those who have diabetes vs. those who do not
# splitting dataframe by group - those with and those without diabetes
# now we have the data split so we can compare and anaylze

db_group = db_df.groupby(db_df.Outcome)
db_df_pos = db_group.get_group(1)
db_df_neg = db_group.get_group(0)
print(db_df_pos)
print(db_df_neg)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
2              8      183             64              0        0  23.3
4              0      137             40             35      168  43.1
6              3       78             50             32       88  31.0
8              2      197             70             45      543  30.5
..           ...      ...            ...            ...      ...   ...
755            1      128             88             39      110  36.5
757            0      123             72              0        0  36.3
759            6      190             92              0        0  35.5
761            9      170             74             31        0  44.0
766            1      126             60              0        0  30.1

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
2                       0.672   32        1
4                       2.288   33        1
6                       0.248   26        1
```

```
8                         0.158   53        1
..                          …   …      …
755                       1.057   37        1
757                       0.258   52        1
759                       0.278   66        1
761                       0.403   43        1
766                       0.349   47        1

[268 rows x 9 columns]
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
1              1       85             66             29        0  26.6
3              1       89             66             23       94  28.1
5              5      116             74              0        0  25.6
7             10      115              0              0        0  35.3
10             4      110             92              0        0  37.6
..           …      …              …              …       …   …
762            9       89             62              0        0  22.5
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
1                       0.351   31        0
3                       0.167   21        0
5                       0.201   30        0
7                       0.134   29        0
10                      0.191   30        0
..                        …   …      …
762                     0.142   33        0
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
767                     0.315   23        0

[500 rows x 9 columns]
```

[12]:
```
# Now we will find the mean BMI of non-diabetic and diabetic individuals, and␣
 ↪compare them.
# the code should give the user sufficient explanatition of the data
#we find that Average bmi is about 4.8 units greater in diabetics
meanBMI_dbpos = db_df_pos["BMI"].mean()
meanBMI_dbneg = db_df_neg["BMI"].mean()
print('Average BMI in diabetic individuals: about', round(meanBMI_dbpos,2))
print('Average BMI in non-diabetic individuals: about', round(meanBMI_dbneg,2))
if meanBMI_dbpos > meanBMI_dbneg:
```

```
    difference1 = meanBMI_dbpos - meanBMI_dbneg #difference will show distance␣
 ↪between values
    print('On average, BMI is about', round(difference1,2), 'units greater in␣
 ↪individuals with diabetes')
elif meanBMI_dbneg > meanBMI_dbpos:
    difference2 =  meanBMI_dbneg - meanBMI_dbpos
    print('On average, BMI is about', round(difference2,2),'units greater in␣
 ↪individuals without diabetes')
elif meanBMI_dbneg == meanBMI_dbpos:
    print('Average BMI in diabetic and non-diabetic individuals is the same')
```

```
Average BMI in diabetic individuals: about 35.14
Average BMI in non-diabetic individuals: about 30.3
On average, BMI is about 4.84 units greater in individuals with diabetes
```

[13]:
```
# QUESTION 2 - What percent of our data are diagnosed with diabetes?
# we are to assume our data was collected randomly - so our of our over 700␣
 ↪individuals, how many were diabietic on average?
#Maybe this data can reflect a larger percentage of the population's amount of␣
 ↪diabetic people

percent_db = (len(db_df_pos.index)/len(db_df.index))*100 # percentage is found␣
 ↪as 100 x (amount of diabetics/ total amount of people in dataset)
print(round(percent_db, 2),'%',' of this dataset of people are diabetic')

#we find that less than 35% of the population in our data are diabetic. Awesome!
 ↪
```

```
34.9 %  of this dataset of people are diabetic
```

[14]:
```
#TES -I searched up how to find the mean of your dataset and was seeing if it␣
 ↪would work
mean_dbpos = db_df_pos.mean(axis=0)
mean_dbneg = db_df_neg.mean(axis=0)
print(mean_dbpos)
print(mean_dbneg)
print(mean_dbneg.describe())
# It does!
```

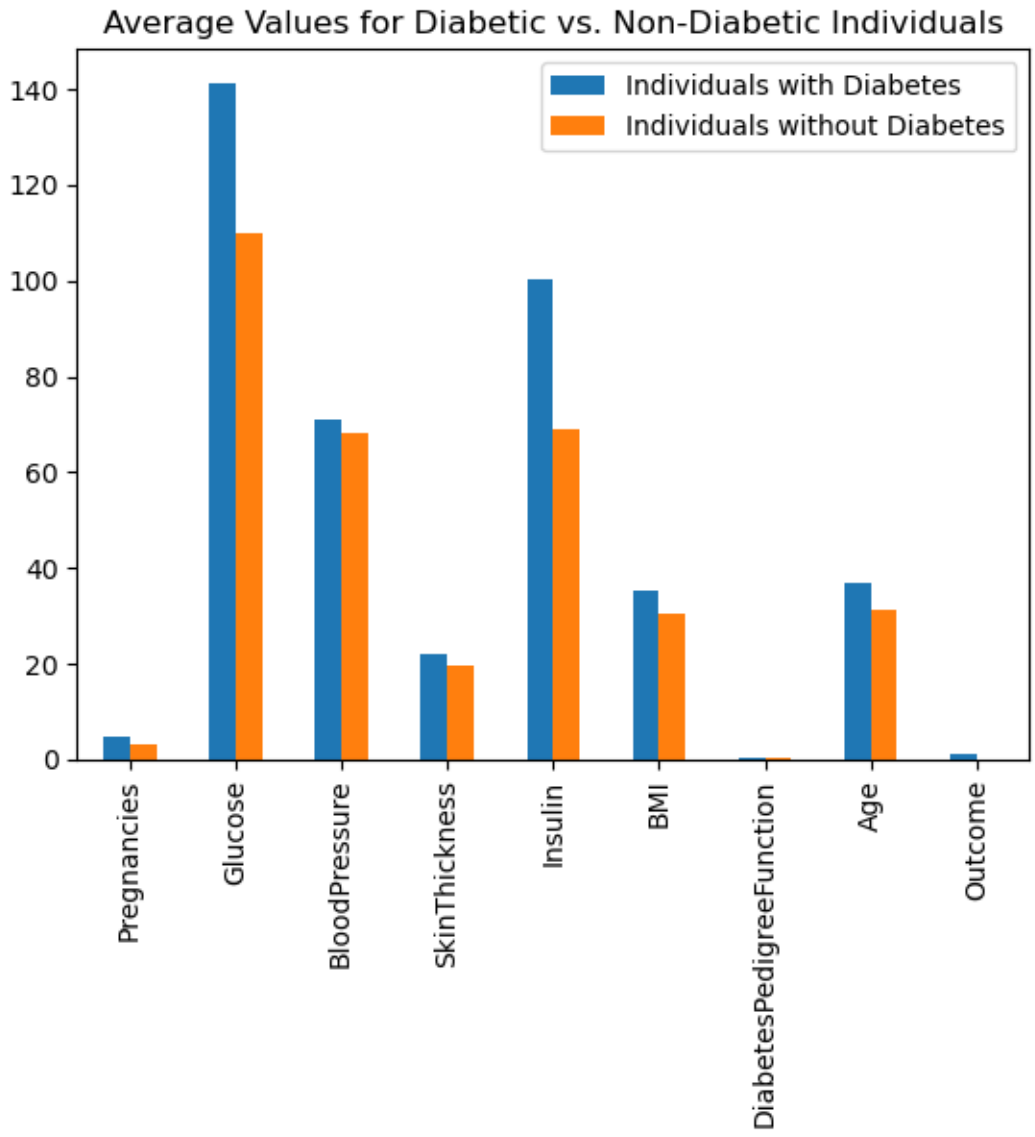
```
Pregnancies                 4.865672
Glucose                   141.257463
BloodPressure              70.824627
SkinThickness              22.164179
Insulin                   100.335821
BMI                        35.142537
DiabetesPedigreeFunction    0.550500
Age                        37.067164
```

```
Outcome                         1.000000
dtype: float64
Pregnancies                     3.298000
Glucose                       109.980000
BloodPressure                  68.184000
SkinThickness                  19.664000
Insulin                        68.792000
BMI                            30.304200
DiabetesPedigreeFunction        0.429734
Age                            31.190000
Outcome                         0.000000
dtype: float64
count      9.000000
mean      36.871326
std       37.964524
min        0.000000
25%        3.298000
50%       30.304200
75%       68.184000
max      109.980000
dtype: float64
```

[15]:
```
#QUESTION 3 - COMPARING AVERAGES OF ALL DATA BETWEEN DIABETICS AND NON-DIABETICS
# Here we want a side-by-side plot to compare the average values between both␣
 ↪types of individuals
#having this visual comparision is helpful to understand the data better and␣
 ↪what is more common
#to do this, we need to take the mean of each column of the diabetic and␣
 ↪non-diabetic datasets (which was tested above, but is still redone here:
mean_dbpos = db_df_pos.mean(axis=0)
mean_dbneg = db_df_neg.mean(axis=0)
# now that we took the mean, we find that the datasets are a 'series' type, so␣
 ↪we cannot use the 'groupby' function
# instead we need to combine the series (so both datasets can be on the same␣
 ↪bargraph), and then we can use the 'plot.bar' function

# here I name my datasets
diabetic = pd.Series(mean_dbpos,name='Individuals with Diabetes')
nondiabetic = pd.Series(mean_dbneg,name='Individuals without Diabetes')
#here I merge them
all_dta = pd.merge(diabetic, nondiabetic, left_index = True, right_index = True)
#here we print the bargraph!
all_dta.plot.bar(rot=90, title="Average Values for Diabetic vs. Non-Diabetic␣
 ↪Individuals");

#we find that on average, almost all categories are higher in value for␣
 ↪diabetics than for non-diabetics. Interesting!
```


Average Values for Diabetic vs. Non-Diabetic Individuals
Individuals with Diabetes
Individuals without Diabetes
140
120
100
80
60
40
20
0
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome

[ ]: