ECE 5984- Homework 2

Question 1.

Excel file is imported to working directory and passed into pandas dataframe as seen in Figure 1.

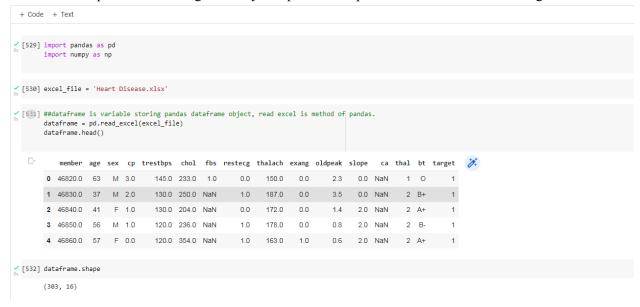


Figure 1

In Figure 2.describe() function is used to display the descriptive statistics of the dataframe with non-numeric values included.

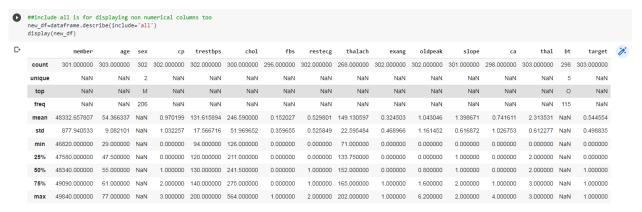


Figure 2

In order to find cardinality, nunique() function is called and each columns' number of unique value is displayed as shown in Figure 3. Then these values are passed into an array called cardinality as shown.

```
[534] #finding cardinality with nunique fnc, which returns the number of non-unique values in the columns
     dataframe.nunique(axis=0, dropna=False)
     member
                 302
                  41
     age
     ср
                   5
     trestbps
                  50
     chol
                 153
     fbs
     restecg
     thalach
                  88
     exang
     oldpeak
     slope
     thal
     target
     dtype: int64
[535] cardinality = dataframe.nunique(axis=0, dropna=False).values
[536] ##these values are stored in an array
    cardinality
     array([302, 41, 3, 5, 50, 153, 3, 4, 88, 3, 41, 4, 6, 4, 6, 2])
```

Figure 3

In Figure 4, a new, empty dataframe is created so all the staticsc will be added here to create final output. Then, with a for loop array elements are matched with the column names of df and this information is kept in data_to_append and it is appended to dataframe resulting in adding the first row to df, which indicates cardinality of each column.



Figure 4

Second row of df will be the mean values of each column. This information was already in new_df data frame that can be observed in Figure 5.

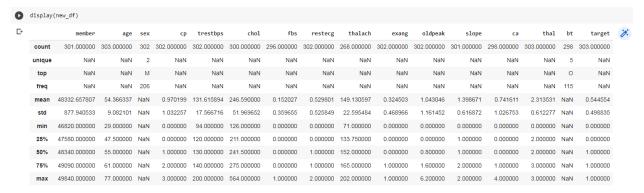


Figure 5

Therefore the row 'mean' in new_df is copied to variable first as seen in Figure 6 and in Figure 7 it is added to the output dataframe df.

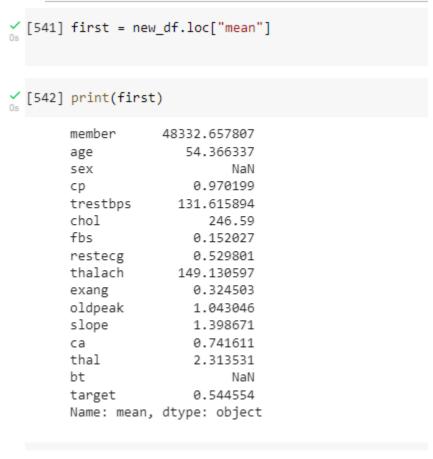


Figure 6

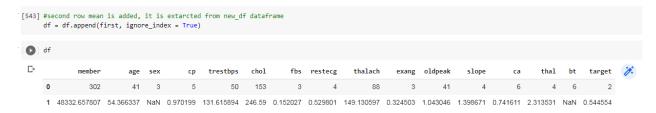


Figure 7

For adding the next row which is the median of each row a loop is used. It iterates over the column names of dataframe(initial dataframe with excel file is passed to) and if column name is not 'sex' or 'bt', since they have no median, data_to_append holds the median value for the specific column under the same column name with dataframe. Hence it can be later appended to df, the output dataframe. Variable i holds the index in this loop. This is displayed in Figure 8 and Figure 9. One can observe in Figure 9, median value for columns 'sex' and 'bt' is NaN since data_to_append has no columns of 'sex' and 'bt' so nothing is assigned to them.

```
data_to_append = {}
i=0
for col in dataframe.columns:
    if col != 'sex' and col != 'bt':
        data_to_append[dataframe.columns[i]] = dataframe[col].median()
    else:
        pass
    i=i+1
df = df.append(data_to_append, ignore_index = True)
```

name, member, cengen, 202, acype, 110aco+

Figure 8

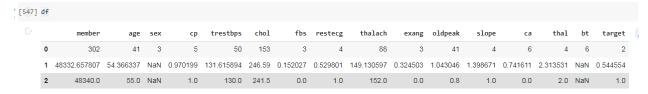


Figure 9

```
df.iloc[2,:]

    member

                 48340.0
    age
                    55.0
    sex
                     NaN
                     1.0
    ср
    trestbps
                   130.0
    chol
                   241.5
    fbs
                     0.0
                     1.0
    restecg
    thalach
                   152.0
                     0.0
    exang
    oldpeak
                     0.8
    slope
                     1.0
    ca
                     0.0
    thal
                     2.0
    bt
                     NaN
    target
                     1.0
    Name: 2, dtype: object
```

Figure 10: df.iloc[2,:] indicates 2 indexed row which is median row

```
√ [551] dataframe.iloc[:,0]

        0
               46820.0
        1
               46830.0
        2
               46840.0
        3
               46850.0
               46860.0
        298
               49800.0
        299
               49810.0
        300
               49820.0
        301
               49830.0
               49840.0
        302
        Name: member, Length: 303, dtype: float64
```

Figure 11: dataframe.iloc[:,0] indicates 0 indexed column which is member column

In Figure 12, number of data that are the same with median value is found. The loop iterates over all columns of the dataframe and if that value is equal to any value in df's second row(median row) it is added to total and total is stored in variable count. Again this value is passed to the corresponding column and stored in data_to_append. Then it is appended to output data frame df.

. ..

```
for i in range(16):
    count = (dataframe.iloc[:,i] == df.iloc[2,i]).sum()
    data_to_append[dataframe.columns[i]]=count
    df = df.append(data_to_append, ignore_index = True)
```

Figure 12: loop for finding n at median

)] df																	
	member	age	sex	ср	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	bt	target	
0	302	41	3	5	50	153	3	4	88	3	41	4	6	4	6	2	
1	48332.657807	54.366337	NaN	0.970199	131.615894	246.59	0.152027	0.529801	149.130597	0.324503	1.043046	1.398671	0.741611	2.313531	NaN	0.544554	
2	48340.0	55.0	NaN	1.0	130.0	241.5	0.0	1.0	152.0	0.0	0.8	1.0	0.0	2.0	NaN	1.0	
3	1	8	0	50	36	0	251	152	7	204	13	139	170	166	0	165	

Figure 13: df with n at median added

```
{'age': 8,
        'bt': 0,
        'ca': 170,
        'chol': 0,
        'cp': 50,
        'exang': 204,
        'fbs': 251,
        'member': 1,
        'oldpeak': 13,
        'restecg': 152,
        'sex': 0,
        'slope': 139,
        'target': 165,
        'thal': 166,
        'thalach': 7,
        'trestbps': 36}
```

Figure 14: data_to_append

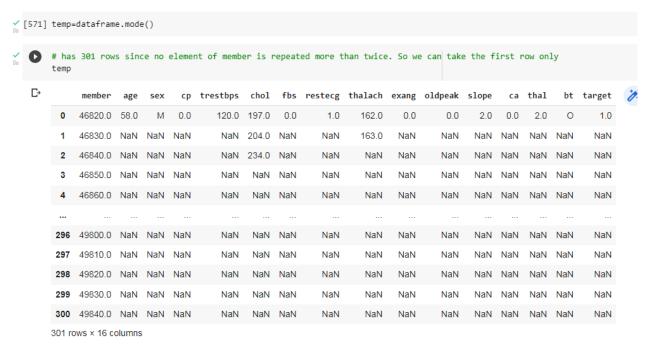


Figure 15: mode of dataframe

In Figure 15, mode of dataframe is displayed. The reason for multiple rows is that the member column has multiple mode values since no value of the 'member' is repeated more than once. Hence we can get the first row of temp and add that to df. This can be observed in Figure 16.

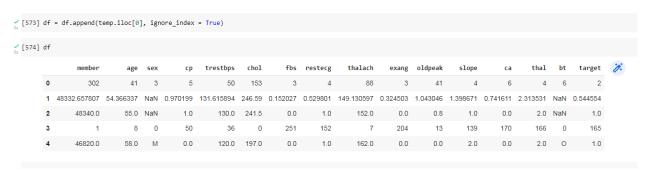


Figure 16: Updated df

In Figure 17, n at mode is found the same way as n at median is found in Figure 12.

```
575 data_to_append = {}
        for i in range(16):
          count = (dataframe.iloc[:,i] == df.iloc[4,i]).sum()
        data_to_append[dataframe.columns[i]]=count
df = df.append(data_to_append, ignore_index = True)

√ [576] df

                                                                               fbs
                                                                                                 thalach
                                                                                                                    oldpeak
                                                                                                                                                       thal
                                                       trestbps
                                                                                    restecg
                     302
                                 41
                                                                                                                               1.398671 0.741611 2.313531
         1 48332.657807 54.366337 NaN 0.970199 131.615894
                                                                 246.59 0.152027
                                                                                    0.529801 149.130597 0.324503 1.043046
                                                                                                                                                            NaN
                                                                                                                                                                  0.544554
                                55.0 NaN
                                                           130.0
                                                                                                    152.0
                                   8
                                                              36
                                                                               251
                                                                                         152
                                                                                                               204
                                                                                                                           13
                                                                                                                                    139
                                                                                                                                                        166
                  46820.0
                                58.0 M
                                                 0.0
                                                           120.0
                                                                   197.0
                                                                               0.0
                                                                                         1.0
                                                                                                    162.0
                                                                                                               0.0
                                                                                                                          0.0
                                                                                                                                    2.0
                                                                                                                                              0.0
                                                                                                                                                        2.0
                                                                                                                                                               0
                                                                                                                                                                        1.0
```

Figure 17: n at mode

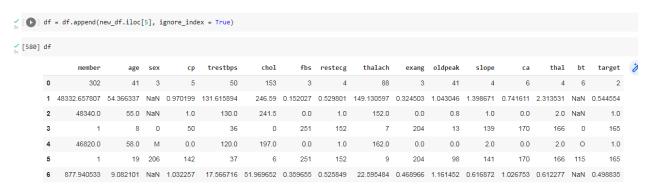


Figure 18: std dev is copied from new df(descriptive statistics dataframe)

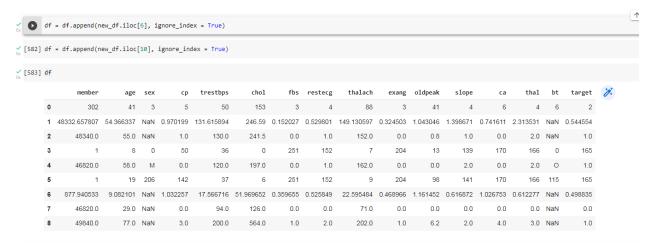


Figure 19: min and max are copied from new df(row 7 is min, row 8 is max)

```
[584] data_to_append = {}
       for i in range(len(dataframe.columns)):
   count = (dataframe.iloc[:,i] == 0).sum()
   data_to_append[dataframe.columns[i]]=count
       df = df.append(data_to_append, ignore_index = True)
✓ [585] df
                                                                 chol
                                                                                restecg
                                                                                            thalach
                                                                                                              oldpeak
                                                                                                                          slope
                                                                                                                                              thal
                                                                                                       exang
                              41
                                              5
                                                                                                                                       6
        0
                   302
                                                                  153
                                                                             3
                                                                                                 88
                                                                                                           3
                                                                                                                   41
                                                        50
        1 48332.657807 54.366337 NaN 0.970199 131.615894
                                                                246.59 0.152027 0.529801
                                                                                         149 130597 0 324503
                                                                                                             1.043046
                                                                                                                       1.398671 0.741611 2.313531
                             58.0
                                  M
                                             0.0
                                                                197.0
                                                                           0.0
                                                                                    1.0
                                                                                              162.0
                                                                                                                   0.0
                                                                                                                                     0.0
                46820.0
                                                      120.0
                                                                                                         0.0
                                                                                                                            2.0
                                                                                                                                               2.0
                                                                                                                                                              1.0
                              19 206
                                             142
                                                        37
                                                                   6
                                                                           251
                                                                                    152
                                                                                                         204
                                                                                                                   98
                                                                                                                            141
                                                                                                                                     170
                                                                                                                                              166
                                                                                                                                                             165
             877.940533
                        9.082101
                                  NaN 1.032257
                                                  17.566716 51.969652 0.359655 0.525849
                                                                                         49840.0
                             77.0 NaN
                                                      200.0
                                                                564.0
                                                                           1.0
                                                                                    2.0
                                                                                              202.0
                                                                                                         1.0
                                                                                                                   6.2
                                                                                                                           2.0
                                                                                                                                     4.0
                                                                                                                                               3.0 NaN
                                                                                                                                                             1.0
                                             3.0
                                    0
                                                                           251
                                                                                    146
                                                                                                 0
                                                                                                         204
                                                                                                                                     170
                                                                                                                                                             138
                                             142
                                                         0
                                                                   0
```

Figure 20: Number of zeros is count

```
for i in range(len(dataframe.columns)):
    count = (dataframe.iloc[:,i].isnull()).sum()
    data_to_append[dataframe.columns[i]]=count
    df = df.append(data_to_append, ignore_index = True)
```

Figure 21: Number of null entries is count

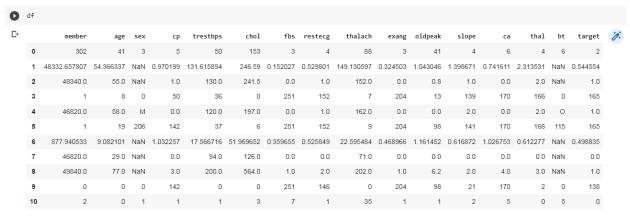


Figure 22: df with all rows are added

Figure 23: stat column is added to df

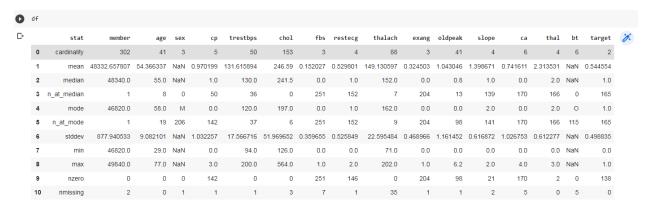


Figure 24: Final output data frame

.Question 2.



Figure 25: Dataframe is pushed to excel file

Α	В	C	D	Е	F	G	Н	1	J	K	L	M	N	0	Р	Q	R
	stat	member	age	sex	ср	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	bt	target
0	cardinality	302	41	3	5	50	153	3	4	88	3	41	4	6	4	6	2
1	mean	48332.66	54.36634		0.970199	131.6159	246.59	0.152027	0.529801	149.1306	0.324503	1.043046	1.398671	0.741611	2.313531		0.544554
2	median	48340	55		1	130	241.5	0	1	152	0	8.0	1	0	2		1
3	n_at_media	1	8	0	50	36	0	251	152	7	204	13	139	170	166	0	165
4	mode	46820	58 N	И	0	120	197	0	1	162	0	0	2	0	2 ()	1
5	n_at_mode	1	19	206	142	37	6	251	152	9	204	98	141	170	166	115	165
6	stddev	877.9405	9.082101		1.032257	17.56672	51.96965	0.359655	0.525849	22.59548	0.468966	1.161452	0.616872	1.026753	0.612277		0.498835
7	min	46820	29		0	94	126	0	0	71	0	0	0	0	0		0
8	max	49840	77		3	200	564	1	2	202	1	6.2	2	4	3		1
9	nzero	0	0	0	142	0	0	251	146	0	204	98	21	170	2	0	138
10	nmissing	2	0	1	1	1	3	7	1	35	1	1	2	5	0	5	0

Figure 25: Excel file

Question 3.

a.&b.

Member: Numeric and ID. Because it is the identifier of each data instance. 2 values are missing,

Age: Feature. It is numeric. No missing values.

Sex: Feature. It is binary since it can be either male or female. 1 missing value.

Cp: Feature. It is categorical.1 missing value.

Trestbps: Feature. Numeric. Continuous. 1 missing value.

Chol: Numeric. Feature. Continuous. 3 missing values.

Fbs: Feature. Binary. Because it can take 1 or 0 only. 7 missing values.

Restecg: Feature. Ordinal. 0 missing values.

Thalach: Feature. Numerical. 35 missing values.

Exang: Feature. Binary. 1 missing.

Oldpeak: Feature. Numeric. Continuous. 1 missing value.

Slope: Feature. Categorical. 2 missing value.

Ca: Feature. Numerical. 5 missing and 5 invalid values (invalid value is 4 because the range is supposed

to be 0-3)

Thal: Feature. Categorical. Only 118 values are valid according to the data sheet because it says the values in the column must be 3,6 and 7 but there are only 0,1,2 and 3 values.

Bt: Categorical. Feature. 5 missing values. Target: Target. Binary. No missing values.

c.

Members: Can be discarded since it is not a feature but id.

Sex: Replace with most frequently seen value since it is categorical.

Cp: Delete row or use smote. No replacing because this value is crucial for the target and it is categorical.

Trestbps: Replace with mean. Thalach: Replace with mean.

Exang: Smote or replace with most frequent value

Oldpeak: Replace with mean. Slope: Smote or delete row. Ca: Replace with mode. Thal: Drop feature.

Bt: Delete row, since its value cannot be predicted. Or smote.

I think the 'thal' column should be removed completely since it has too many invalid values.

Question 4.

Covariance matrix:

	<u>member</u>	age	<u>trestbps</u>	<u>chol</u>	<u>thalach</u>	<u>oldpeak</u>	<u>ca</u>	target
<u>member</u>	768218.8							
age	1480.577	82.21233						
<u>trestbps</u>	1697.479	44.51458	307.5677					
<u>chol</u>	846.1829	102.62	110.5826	2691.842				
<u>thalach</u>	-8146.83	-73.0569	-15.3469	16.40282	508.6509			
<u>oldpeak</u>	307.2909	2.237494	3.943852	3.533738	-8.95472	1.344505		
<u>ca</u>	344.0951	2.557886	1.816236	3.522033	-4.19026	0.261877	1.042273	
target_	-377.021	-1.01797	-1.27155	-2.0686	4.649978	-0.25133	-0.19916	0.248015

Figure 26: Covariance matrix

Correlation:

	member	age	trestbps	<u>chol</u>	<u>thalach</u>	oldpeak	ca	target
member	1							
age	0.185808	1						
trestbps	0.110312	0.279508	1					
<u>chol</u>	0.018508	0.217346	0.121418	1				
<u>thalach</u>	-0.40954	-0.36278	-0.03798	0.013848	1			
<u>oldpeak</u>	0.300974	0.212657	0.193315	0.058719	-0.34325	1		
<u>ca</u>	0.383924	0.276326	0.101285	0.066333	-0.17823	0.221042	1	
target	-0.86329	-0.22544	-0.14555	-0.08	0.413508	-0.43539	-0.39172	1

Figure 27: Correlation matrix

Question 5.

3 predictions most highly correlated with target are; member (but it's not a predictor so it will be ignored). Oldpeak, thalach, ca are the most correlated ones with target with values -0.43539, 0.413508 and -0.39172 respectively.

3 most highly correlated predictors are; (member is ignored again), age and thalac with -0.36278, oldpeak and thalac with -0.34325 and, age and trestbps with 0.279508.