CAP4612 - Homework 2

- All the code you need to do this homework I have provided. You might have to do a little modification that is all. Do not overthink it.

- ***Important your results might be different than mine due to the samples that you are using. I provide you with numbers and figures are just to guide you.***

- If I ask you to make a comment on some of your calculations, there is no right or wrong answer that I'm looking for. I want to see how you are interpreting and understanding the material.

0. On the top of your python script file put the following information:

```
# Name:   YOUR NAME
# ID:     YOUR PANTHER ID

# CERTIFICATION: I understand FIU's academic policies, and I certify that this work is my
#                own and that none of it is the work of any other person.
```

1. Load the Cancer_Data.csv data into a dataframe named cancer_data and show the first 5 entries of the data frame. Here's an example of what I'm looking for:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	ri
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.0	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	NaN	386.1	0.14250	NaN	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.1	1297.0	0.10030	0.13280	0.1980	0.10430	...	

Here is information on the features of the dataset:

- 1) ID number
- 2) Diagnosis (M = malignant, B = benign)
- 3-32)
- Ten real-valued features are computed for each cell nucleus:
- a) radius (mean of distances from center to points on the perimeter)
- b) texture (standard deviation of gray-scale values)
- c) perimeter
- d) area
- e) smoothness (local variation in radius lengths)
- f) compactness (perimeter^2 / area - 1.0)
- g) concavity (severity of concave portions of the contour)
- h) concave points (number of concave portions of the contour)
- i) symmetry
- j) fractal dimension ("coastline approximation" - 1)

2. Print out the following information of the dataframe in step 1.

Hint: There is a simple function call on the dataframe that does this.

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symme
count	5.690000e+02	565.000000	564.000000	565.000000	565.000000	568.000000	564.000000	556.000000	558.000000	5
mean	3.037183e+07	14.123581	19.286436	92.012319	655.822655	0.096316	0.103982	0.089747	0.049576	
std	1.250206e+08	3.515269	4.313091	24.219501	352.848451	0.014037	0.052455	0.079679	0.038692	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	
25%	8.692180e+05	11.700000	16.167500	75.210000	420.300000	0.086290	0.064315	0.029930	0.020692	
50%	9.060240e+05	13.340000	18.835000	86.340000	551.100000	0.095865	0.092350	0.061880	0.033950	
75%	8.813129e+06	15.780000	21.802500	104.100000	788.500000	0.105300	0.130400	0.131950	0.074122	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	

8 rows × 31 columns

3. Write the code that prints out the following information of the dataframe in step 1.
Replace #row with number of rows and #col with the number of columns in the dataframe

```
There are #row rows and #col columns
```

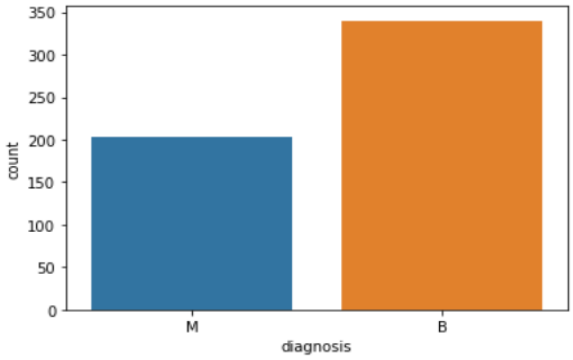
4. Print out the features (columns) of the dataframe

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

5. Use seaborn and mathplot to plot the following diagram.

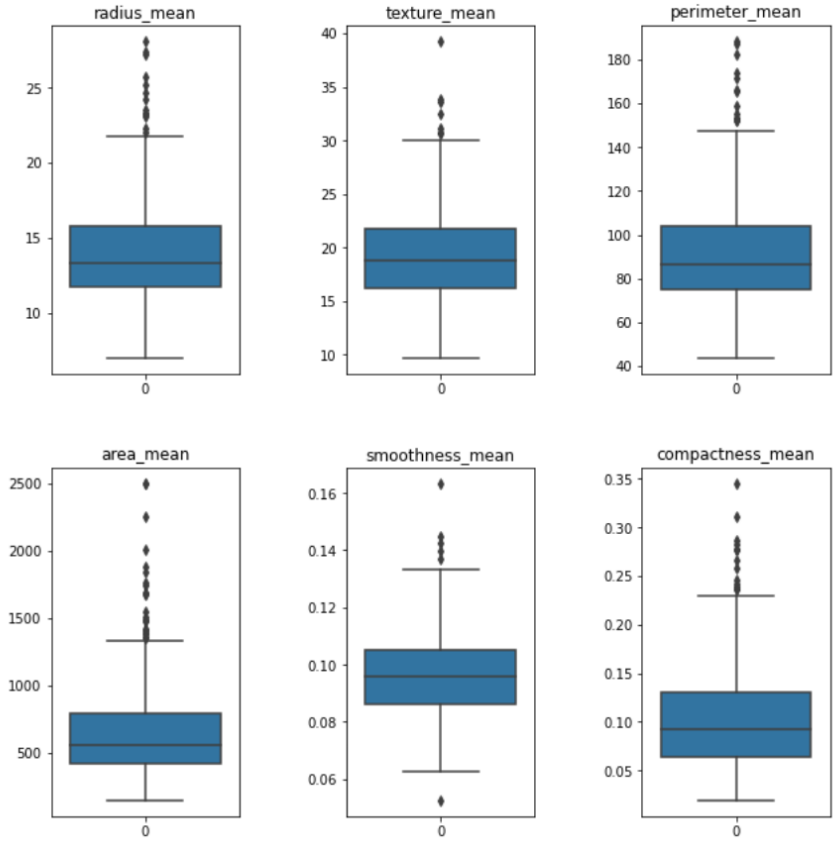
If you get any warnings use this code to suppressed them:

```
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

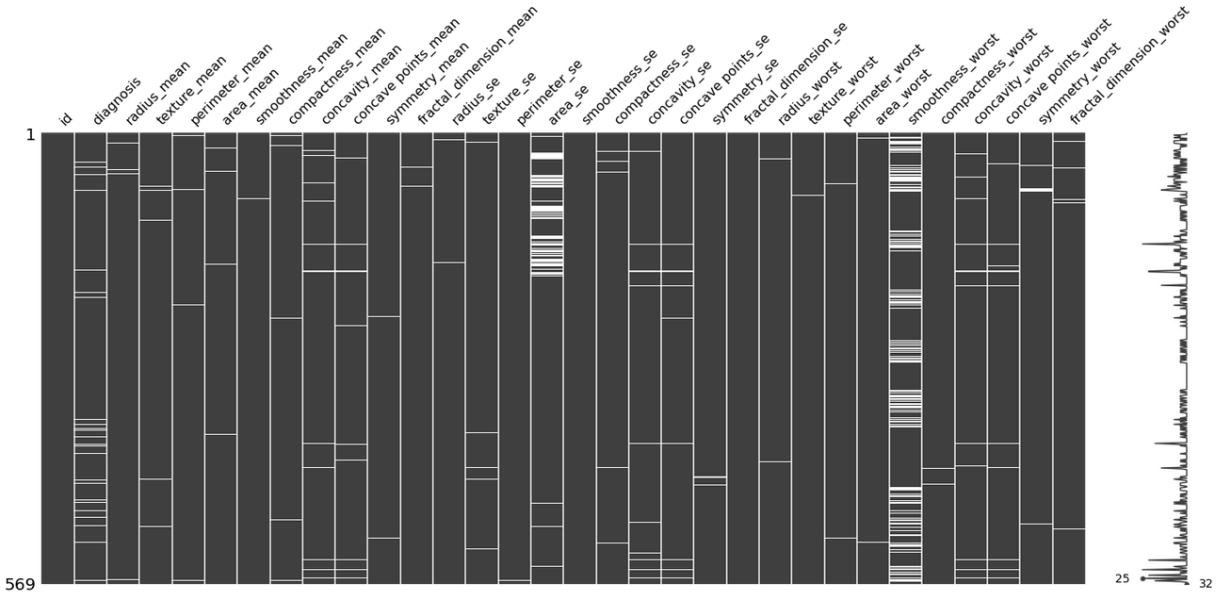


6. Use seaborn, mathplot, subplot and boxplot to plot the following diagram. Notice the number of row and columns in the diagram.

Boxplot are for the following features: radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_mean and compactness_mean.



7. Plot the following diagram using the dataframe from step 1.



8. Write the code that prints out all the features name that have missing data:

```
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'compactness_se',
 'concavity_se',
 'concave points_se',
 'symmetry_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst',
 'symmetry_worst',
 'fractal_dimension_worst']
```

9. Output the dtypes of the dataframe:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                543 non-null    object
 2   radius_mean              565 non-null    float64
 3   texture_mean             564 non-null    float64
 4   perimeter_mean           565 non-null    float64
 5   area_mean                565 non-null    float64
 6   smoothness_mean          568 non-null    float64
 7   compactness_mean         564 non-null    float64
 8   concavity_mean           556 non-null    float64
 9   concave points_mean      558 non-null    float64
 10  symmetry_mean            567 non-null    float64
 11  fractal_dimension_mean   567 non-null    float64
 12  radius_se                567 non-null    float64
 13  texture_se               564 non-null    float64
 14  perimeter_se             568 non-null    float64
 15  area_se                  501 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           564 non-null    float64
 18  concavity_se             557 non-null    float64
 19  concave points_se        559 non-null    float64
 20  symmetry_se              565 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             567 non-null    float64
 23  texture_worst            568 non-null    float64
 24  perimeter_worst          567 non-null    float64
 25  area_worst               567 non-null    float64
 26  smoothness_worst         434 non-null    float64
 27  compactness_worst        567 non-null    float64
 28  concavity_worst          556 non-null    float64
 29  concave points_worst     557 non-null    float64
 30  symmetry_worst           563 non-null    float64
 31  fractal_dimension_worst  562 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

10. Write the code that outputs all the percentage of missing data for the features listed in step 8.

```
diagnosis : 26 (4.569%)
radius_mean : 4 (0.703%)
texture_mean : 5 (0.879%)
perimeter_mean : 4 (0.703%)
area_mean : 4 (0.703%)
smoothness_mean : 1 (0.176%)
compactness_mean : 5 (0.879%)
concavity_mean : 13 (2.285%)
concave points_mean : 11 (1.933%)
symmetry_mean : 2 (0.351%)
fractal_dimension_mean : 2 (0.351%)
radius_se : 2 (0.351%)
texture_se : 5 (0.879%)
perimeter_se : 1 (0.176%)
area_se : 68 (11.951%)
compactness_se : 5 (0.879%)
concavity_se : 12 (2.109%)
concave points_se : 10 (1.757%)
symmetry_se : 4 (0.703%)
radius_worst : 2 (0.351%)
texture_worst : 1 (0.176%)
perimeter_worst : 2 (0.351%)
area_worst : 2 (0.351%)
smoothness_worst : 135 (23.726%)
compactness_worst : 2 (0.351%)
concavity_worst : 13 (2.285%)
concave points_worst : 12 (2.109%)
symmetry_worst : 6 (1.054%)
fractal_dimension_worst : 7 (1.23%)
```

11. Write the code that drops features that have more than 22% of their data missing from the cancer_data dataframe. Create a new dataframe named **cancer_data_cleaned** that has all the features that have less than 22% of their data missing**.** Show the percentage of missing data of the cancer_data_cleaned dataframe.

Important: You are to automate this process by defining a method and calling it.

```
diagnosis : 26 (4.569%)
radius_mean : 4 (0.703%)
texture_mean : 5 (0.879%)
perimeter_mean : 4 (0.703%)
area_mean : 4 (0.703%)
smoothness_mean : 1 (0.176%)
compactness_mean : 5 (0.879%)
concavity_mean : 13 (2.285%)
concave points_mean : 11 (1.933%)
symmetry_mean : 2 (0.351%)
fractal_dimension_mean : 2 (0.351%)
radius_se : 2 (0.351%)
texture_se : 5 (0.879%)
perimeter_se : 1 (0.176%)
area_se : 68 (11.951%)
compactness_se : 5 (0.879%)
concavity_se : 12 (2.109%)
concave points_se : 10 (1.757%)
symmetry_se : 4 (0.703%)
radius_worst : 2 (0.351%)
texture_worst : 1 (0.176%)
perimeter_worst : 2 (0.351%)
area_worst : 2 (0.351%)
compactness_worst : 2 (0.351%)
concavity_worst : 13 (2.285%)
concave points_worst : 12 (2.109%)
symmetry_worst : 6 (1.054%)
fractal_dimension_worst : 7 (1.23%)
```

11. Drop all rows with a threshold of 1, print out the information shown below on the **cancer_data_cleaned** dataframe**.**

Hint: Look at the drop_row function in the notes. Your numbers maybe a little different

```
Samples Before Removal : 569
Samples After Removal : 413
```

12. Output the dtypes of the **cancer_data_cleaned** dataframe:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       413 non-null    int64
 1   diagnosis                413 non-null    object
 2   radius_mean              413 non-null    float64
 3   texture_mean             413 non-null    float64
 4   perimeter_mean           413 non-null    float64
 5   area_mean                413 non-null    float64
 6   smoothness_mean          413 non-null    float64
 7   compactness_mean         413 non-null    float64
 8   concavity_mean           413 non-null    float64
 9   concave points_mean      413 non-null    float64
 10  symmetry_mean            413 non-null    float64
 11  fractal_dimension_mean   413 non-null    float64
 12  radius_se                413 non-null    float64
 13  texture_se               413 non-null    float64
 14  perimeter_se             413 non-null    float64
 15  area_se                  413 non-null    float64
 16  smoothness_se            413 non-null    float64
 17  compactness_se           413 non-null    float64
 18  concavity_se             413 non-null    float64
 19  concave points_se        413 non-null    float64
 20  symmetry_se              413 non-null    float64
 21  fractal_dimension_se     413 non-null    float64
 22  radius_worst             413 non-null    float64
 23  texture_worst            413 non-null    float64
 24  perimeter_worst          413 non-null    float64
 25  area_worst               413 non-null    float64
 26  compactness_worst        413 non-null    float64
 27  concavity_worst          413 non-null    float64
 28  concave points_worst     413 non-null    float64
 29  symmetry_worst           413 non-null    float64
 30  fractal_dimension_worst  413 non-null    float64
 31  missing_count            413 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 106.5+ KB
```

13. Encode all object type features in **cancer_data_cleaned** dataframe the using the LabelEncoder(). Output the first five rows of **cancer_data_cleaned .**

See that the diagnosis is encoded.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	r
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	
5	843786	1	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	...	
7	84458202	1	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	...	

5 rows × 32 columns

14. Output the dtypes of the cancer_data_cleaned dataframe:

See the diagnosis type changed

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       413 non-null    int64
 1   diagnosis                413 non-null    int64
 2   radius_mean              413 non-null    float64
 3   texture_mean             413 non-null    float64
 4   perimeter_mean           413 non-null    float64
 5   area_mean                413 non-null    float64
 6   smoothness_mean          413 non-null    float64
 7   compactness_mean         413 non-null    float64
 8   concavity_mean           413 non-null    float64
 9   concave points_mean      413 non-null    float64
 10  symmetry_mean            413 non-null    float64
 11  fractal_dimension_mean   413 non-null    float64
 12  radius_se                413 non-null    float64
 13  texture_se               413 non-null    float64
 14  perimeter_se             413 non-null    float64
 15  area_se                  413 non-null    float64
 16  smoothness_se            413 non-null    float64
 17  compactness_se           413 non-null    float64
 18  concavity_se             413 non-null    float64
 19  concave points_se        413 non-null    float64
 20  symmetry_se              413 non-null    float64
 21  fractal_dimension_se     413 non-null    float64
 22  radius_worst             413 non-null    float64
 23  texture_worst            413 non-null    float64
 24  perimeter_worst          413 non-null    float64
 25  area_worst               413 non-null    float64
 26  compactness_worst        413 non-null    float64
 27  concavity_worst          413 non-null    float64
 28  concave points_worst     413 non-null    float64
 29  symmetry_worst           413 non-null    float64
 30  fractal_dimension_worst  413 non-null    float64
 31  missing_count            413 non-null    float64
dtypes: float64(30), int64(2)
memory usage: 106.5 KB
```

15. Write a function that automates the logistic regression with test size from 0.1 – 0.9. It returns a dataframe named report_dataframe.

def automation_lr(x, y):

….

return report_dataframe

x-> feature data to use for the train_test_split(……)
y -> feature data to use for the train_test_split(……)

in the train_test_split(…) use the shuffle = True option

normalized_score = test_score / train_score

Example of the report_dataframe that is returned

	Test_Size	Train_Accurary_Score	Test_Accurary_Score	Normalized_Score
0	0.1	0.973046	0.928571	0.954294
1	0.2	0.966667	0.927711	0.959701
2	0.3	0.965398	0.959677	0.994074
3	0.4	0.963563	0.939759	0.975296
4	0.5	0.956311	0.956522	1.000221
5	0.6	0.987879	0.923387	0.934717
6	0.7	0.991870	0.944828	0.952572
7	0.8	1.000000	0.939577	0.939577
8	0.9	1.000000	0.895161	0.895161

16. Call automation_lr(x, y) with x being features 2-31 and y being feature 1 of the cancer_data_cleaned dataframe. Look at step 14 to see where I am getting the feature numbers. Then output the report_dataframe after the function call.

	Test_Size	Train_Accurary_Score	Test_Accurary_Score	Normalized_Score
0	0.1	0.973046	0.928571	0.954294
1	0.2	0.966667	0.927711	0.959701
2	0.3	0.965398	0.959677	0.994074
3	0.4	0.963563	0.939759	0.975296
4	0.5	0.956311	0.956522	1.000221
5	0.6	0.987879	0.923387	0.934717
6	0.7	0.991870	0.944828	0.952572
7	0.8	1.000000	0.939577	0.939577
8	0.9	1.000000	0.895161	0.895161

17. Pick a test size that you think is the best then re-estimate the Logistic Regression with x being features 2-31 and y being feature 1. Output the following:

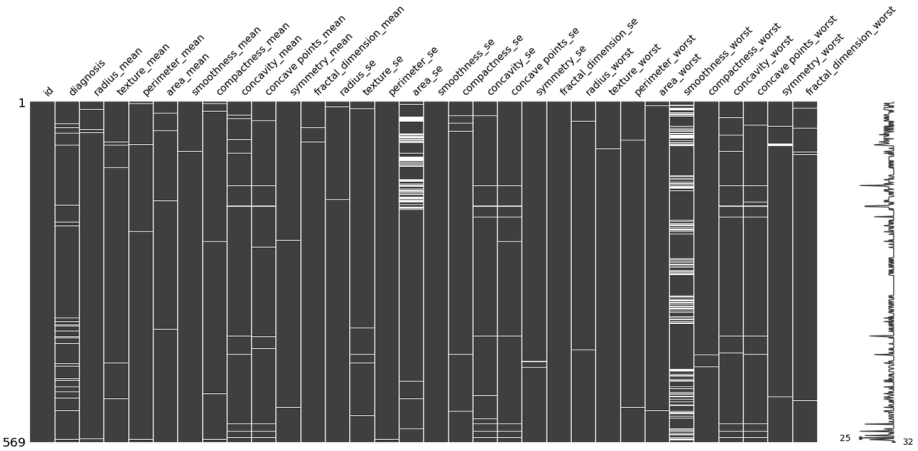
```
Train Accurary Score =  0.9818181818181818
Test Accurary Score =  0.9314516129032258
```

Your numbers maybe different…..

18. Take the first 10 rows of the cancer_data_cleaned features 2-31 and then use your model from step 17 to predict the diagnosis on these rows.

19. Write 2-3 sentences about how you feel about your estimated model's predictions.

20. We will now be working with K-Nearest Meighbors. Make a copy of the cancer_data from step1. Name your copy **knn_data**. Output the following diagram using the knn_data



21. Drop all features on the knn_data with more than 22% data. Create a new dataframe named **knn_data_cleaned** that contains the feature with less than 22% data missing. This is like step 10. Display dtypes knn_data_cleaned dataframe.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	..
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	..
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	..
2	84300903	M	19.69	21.25	130.0	1203.0	0.10960	0.15990	0.1974	0.12790	..
3	84348301	M	11.42	20.38	NaN	386.1	0.14250	NaN	0.2414	0.10520	..
4	84358402	M	20.29	14.34	135.1	1297.0	0.10030	0.13280	0.1980	0.10430	..

22. Drop all rows with a threshold of 1, print out the information shown below on the knn_data_cleaned dataframe**.**

Hint: Look at the drop_row function in the notes. Your numbers maybe a little different

```
Samples Before Removal : 569
Samples After Removal : 320
```

23. Repeat steps 12-19 using the knn_data_cleaned dataframe.

24. Put the data file and your python script into a folder and zip the folder. Upload the zipped file to Canvas using the assignment link. You are done ☺