Heart Data Analysis(Unsupervised)

- **Objective of the analysis**

  The objective of this analysis is to be able to define a model to predict if a patient can have an heart attack based on some heart parameters

- **Description of the data set**

  For this project I'm going to use a Data Set representing Heart Attack classification.
  Each case of heart attack has some parameters:
  1. age - age in years
  2. sex - sex (1 = male; 0 = female)
  3. cp - chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 0 = asymptomatic)
  4. trestbps - resting blood pressure (in mm Hg on admission to the hospital)
  5. chol - serum cholesterol in mg/dl
  6. fbs - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
  7. restecg - resting electrocardiographic results (1 = normal; 2 = having ST-T wave abnormality; 0 = hypertrophy)
  8. thalach - maximum heart rate achieved
  9. exang - exercise induced angina (1 = yes; 0 = no)
  10. oldpeak - ST depression induced by exercise relative to rest
  11. slope - the slope of the peak exercise ST segment (2 = upsloping; 1 = flat; 0 = downsloping)
  12. ca - number of major vessels (0-3) colored by fluoroscopy
  13. thal - 2 = normal; 1 = fixed defect; 3 = reversible defect
  14. num - the predicted attribute - diagnosis of heart disease (angiographic disease status)

  (Value 0 = < diameter narrowing; Value 1 = > 50% diameter narrowing)

- ## Data Describe

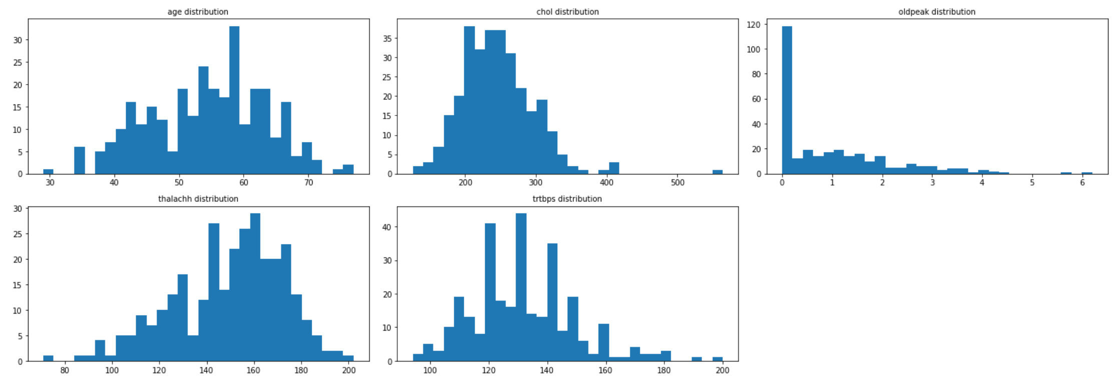
```
#Numerical Data
Heart_Data.describe()
```

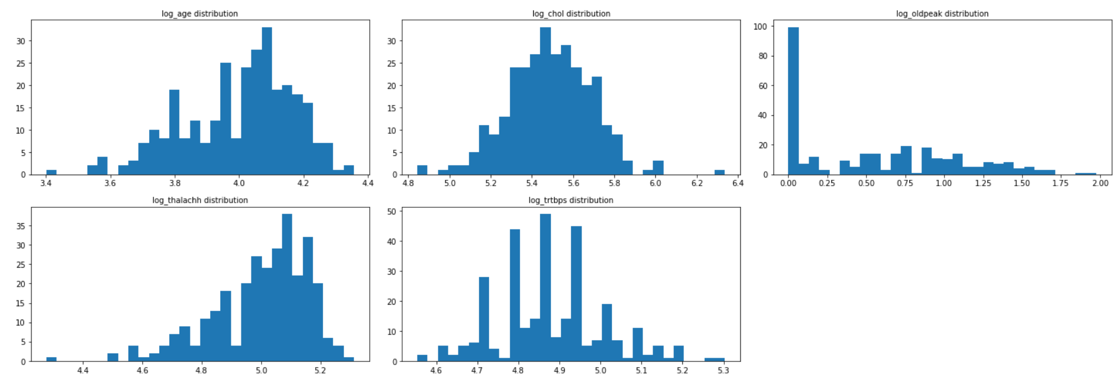
	age	sex	trtbps	chol	fbs	thalachh	exng	oldpeak	caa	output	restecg_hypetrophic	restecg_normal	restecg_abnormal	cp_asymptomatic
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	131.623762	246.264026	0.148515	149.646865	0.326733	1.039604	0.729373	0.544554	0.485149	0.501650	0.013201	0.471947
std	9.082101	0.466011	17.538143	51.830751	0.356198	22.905161	0.469794	1.161075	1.022606	0.498835	0.500606	0.500824	0.114325	0.500038
min	29.000000	0.000000	94.000000	126.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	120.000000	211.000000	0.000000	133.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	55.000000	1.000000	130.000000	240.000000	0.000000	153.000000	0.000000	0.800000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
75%	61.000000	1.000000	140.000000	274.500000	0.000000	166.000000	1.000000	1.600000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000
max	77.000000	1.000000	200.000000	564.000000	1.000000	202.000000	1.000000	6.200000	4.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- ## Numerical Data

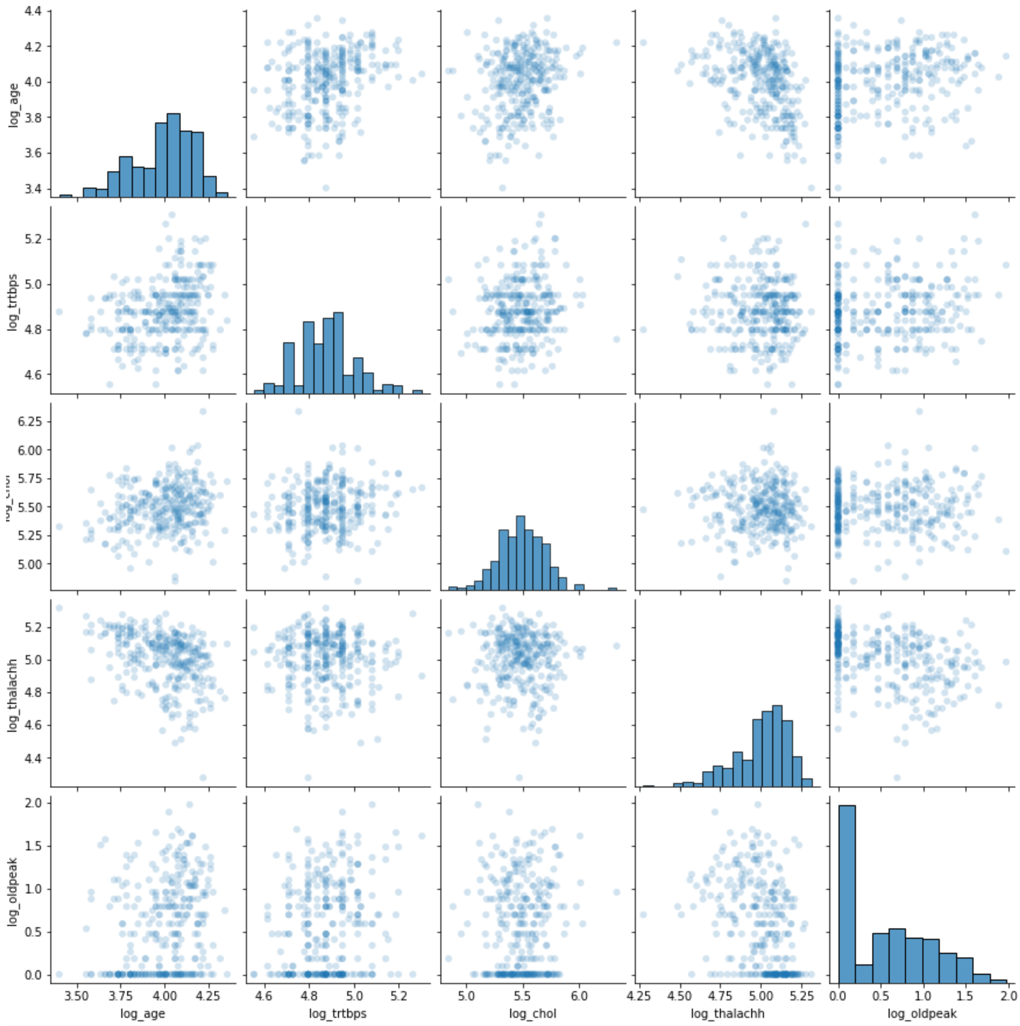
Numerical Data Distribution.

```
Heart_Data_num=Heart_Data.drop(columns=['slp','thall','restecg','cp','sex','output','caa','exng','fbs'])
hist_loop(data=Heart_Data_num,
          rows=3,
          cols=3,
          figsize=(20,10))
```





The pair plot doesn't show any strong correlation between all numerical data



Train models

- Fit a **K-means clustering** model with two clusters and
- Fit 2 **Agglomerative clustering** models with two clusters (ward-link and complete-link clustering)
- Compare the results to those obtained by K-means with regards to wine color by reporting the number of red and white observations in each cluster for both K-means and agglomerative clustering.
- Visualize the **dendrogram** produced by agglomerative clustering

Kmeans

```python
#Kmeans
km = KMeans(n_clusters=2, random_state=42)
km = km.fit(data[feature_columns])

data['kmeans'] = km.predict(data[feature_columns])
(data[['output','kmeans']]
 .groupby(['kmeans','output'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

✓ 0.1s

kmeans	output	number
0	0	21
	1	128
1	0	117
	1	37

Agglomerative Clustering

```python
#Agglomerative Clustering
for linkage in ['complete', 'ward']:
    ag = AgglomerativeClustering(n_clusters=2, linkage=linkage, compute_full_tree=True)
    ag = ag.fit(data[feature_columns])
    data[str('agglom_'+linkage)] = ag.fit_predict(data[feature_columns])
```

[37] ✓ 0.7s

```python
(data[['output','agglom_ward']]
 .groupby(['output','agglom_ward'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

[38] ✓ 0.6s

output	agglom_ward	number
0	0	93
	1	45
1	0	43
	1	122

Agglomerative Vs Kmeans Comparison

```
# Comparing AgglomerativeClustering with KMeans
(data[['output','agglom_complete', 'agglom_ward','kmeans']]
 .groupby(['output', 'agglom_complete', 'agglom_ward', 'kmeans'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

✓ 0.1s

				number
output	agglom_complete	agglom_ward	kmeans	
0	0	0	0	3
			1	90
		1	0	18
			1	26
	1	1	1	1
1	0	0	0	15
			1	28
		1	0	112
			1	9
	1	1	0	1

Dendrogram

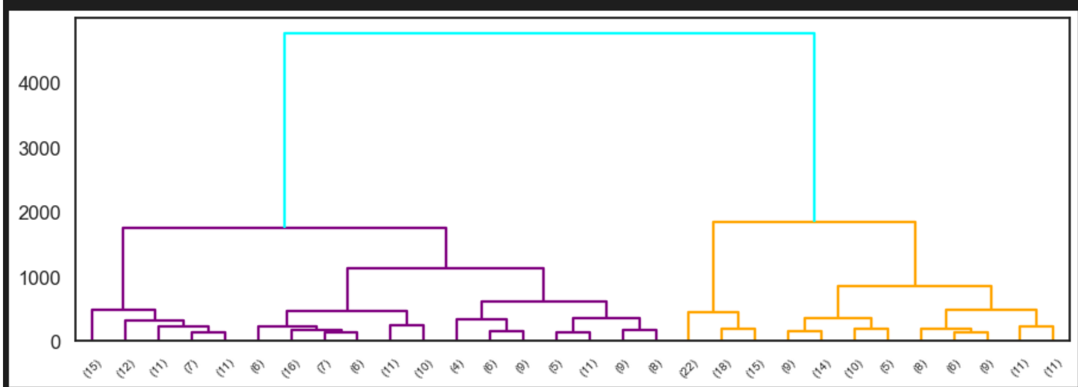
```
Z = hierarchy.linkage(ag.children_, method='ward')

fig, ax = plt.subplots(figsize=(15,5))


hierarchy.set_link_color_palette(['purple', 'orange'])

den = hierarchy.dendrogram(Z, orientation='top',
                           p=30, truncate_mode='lastp',
                           show_leaf_counts=True, ax=ax,
                           above_threshold_color='cyan')
```

✓ 0.5s



Conclusions

Comparing the results seems that i have better results with **Kmeans** and A**gglomerative Ward Clustering** even if my dataset is very small , maybe i should have used a larger dataset to have a very significative clustering.