Clustering Project using HCV dataset

Dataset: https://archive.ics.uci.edu/ml/datasets/HCV+data
Source : UC Irvine Machine Learning Repository

Main objective of the analysis :

The model we build will be focused on **clustering** the dataset, where we try to **find groups or clusters within the dataset**, which helps us to identify inherent patterns, to check any **possible correlation** between different **attributes**, and if at all, any **grouping** can be made among related factors.

Brief description of the data set

Dataset : https://archive.ics.uci.edu/ml/datasets/HCV+data

Problem Type : Clustering

The data set contains laboratory values of blood donors and Hepatitis C patients and demographic values like age.

No target variable(label) is relevant here, as it is UNSUPERVISED LEARNING(Clustering)

Features :

```
['Unnamed: 0',
'Category',
'Age',
'Sex',
'ALB',
'ALP',
'ALT',
'AST',
'BIL',
'CHE',
'CHOL',
'CREA',
'GGT',
'PROT']
```

The data has 615 **rows** and 14 **columns**.

Brief summary of data exploration

Most of the columns are of type = ‘float64’

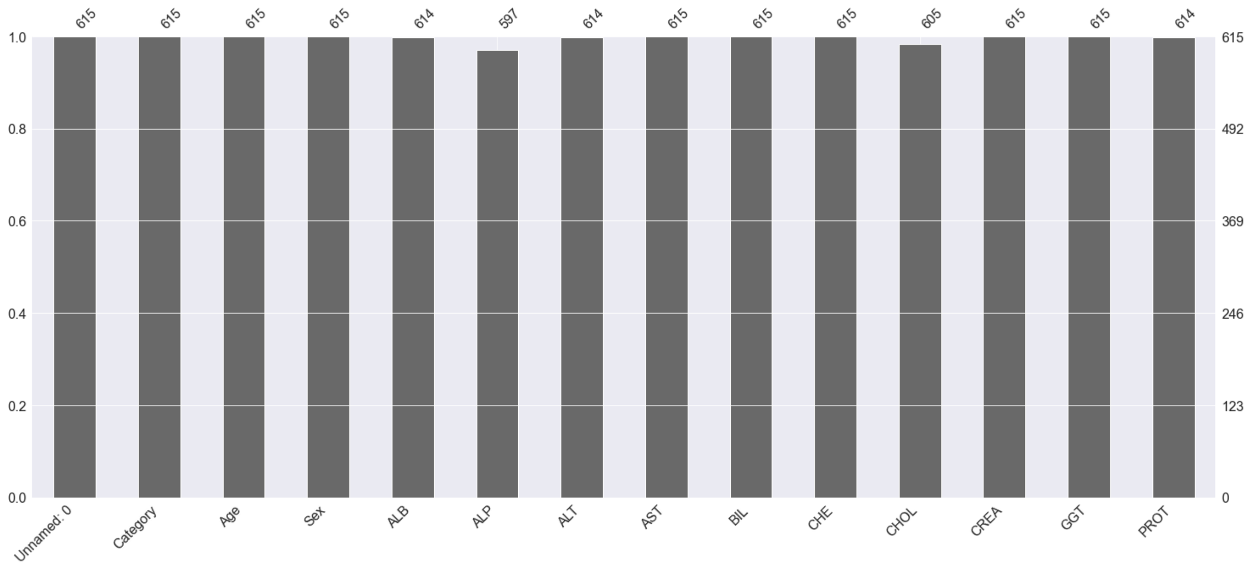
data.dtypes

```
Unnamed: 0      int64
Category       object
Age             int64
Sex            object
ALB           float64
ALP           float64
ALT           float64
AST           float64
BIL           float64
CHE           float64
CHOL          float64
CREA          float64
GGT           float64
PROT          float64
```

1. There are **missing values** in the dataset.
   The graph is a result of code:

   msno.bar(data)

   <AxesSubplot:>



The missing values(NaN) are dropped and checked for any NaN values in the dataset.

```
data.dropna(inplace=True)
data.isna().any()
          Unnamed: 0    False
          Category      False
          Age           False
          Sex           False
          ALB           False
          ALP           False
          ALT           False
          AST           False
          BIL           False
          CHE           False
          CHOL          False
          CREA          False
          GGT           False
          PROT          False
dtype: bool
```

In [120]:

2. We check for the **unique values and their counts** for the **object data types** in columns ‘Sex’ and “Category” that tell us about 5 types of blood donors and two types of genders, male and female present amongst the donors.

   ```
   data.Category.value_counts()

   0=Blood Donor            526
   3=Cirrhosis              24
   1=Hepatitis              20
   2=Fibrosis               12
   0s=suspect Blood Donor      7
   Name: Category, dtype: int64


   data.Sex.value_counts().sort_index()

   f    226
   m    363
   Name: Sex, dtype: int64
   ```

3. I have found **correlations** between numerical data columns, i.e, all columns barring ‘Category” and “sex” and found the columns that have maximum correlation amongst each other.
4. **Skewness** is found out and all the columns with skewness > 0.75 are listed.

   ```
   CREA    14.955189
   BIL      8.089304
   ALT      6.815926
   GGT      5.936910
   AST      5.246583
   ALP      4.756845
   dtype: float64
   ```
5. **Scaling** is done using StandardScaler from sklearn.preprocessing.

Variations of Unsupervised Learning models :

4 variations of the unsupervised model.

For my training of the unsupervised model, I have used **2** different **clustering techniques**, **KMeans** and **AgglomerativeClustering,** both imported from **sklearn.cluster**

However, for **each** one of them, I have **changed hyperparameters 2 times, in essence using 4** variations of the unsupervised model.

1. km = KMeans(n_clusters=**5**, random_state=42)

2. km = KMeans(n_clusters=**8**, random_state=42)

3. ag = AgglomerativeClustering(n_clusters=**5**, linkage='**ward**', compute_full_tree=True)

4. ag = AgglomerativeClustering(n_clusters=**8**, linkage='**single**', compute_full_tree=True)

A paragraph explaining the model that best suits the main objective(s) of this analysis:

I consider **model 1:**

km = KMeans(n_clusters=**5**, random_state=42)

as a final model that best fits the needs in terms of my dataset, as the **“Category” column has 5 different values** and I have chosen to make **5 different clusters**, which seems to be pretty reasonable.

K-Means clustering is the most widely used model for clustering for general datasets and so for common usage, it comes in pretty handy, so I recommend this KMeans model.

KMeans Clustering model is pretty good when dealing with datasets without outliers and do not tend to overfit. Hence, I decided to go for this model.

As there is no right or wrong answer in Clustering, any fit is a good fit, as long as it serves to form different clusters, as every new cluster reveals new information and follows new patterns of grouping together similar data.

Summary Key Findings and Insights

I have copied here the data frames that are results of a comparative study of different models , put together side by side. It may be noted that although **cluster assignment is arbitrary**, the respective primary cluster numbers for the different categories in “Category” may not be identical to each other, and also may not be the same for both K-means and agglomerative clustering.Though the cluster numbers are not identical, the **clusters are very consistent within a single ‘Category” type**.

```
(data[['Category','agglom_W','kmeans_5']]
 .groupby(['Category','agglom_W'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

		number
Category	**agglom_W**	
0=Blood Donor	**0**	6
	1	240
	2	280
0s=suspect Blood Donor	**0**	4
	2	1
	4	2

1=Hepatitis	**0**	4
	1	11
	2	3
	4	2
2=Fibrosis	**0**	4
	1	8
3=Cirrhosis	**0**	16
	1	1
	3	3
	4	4

In [140]:

```
(data[['Category','agglom_W','kmeans_8']]
.groupby(['Category','agglom_W'])
.size()
.to_frame()
.rename(columns={0:'number'}))
```

Out[140]:

		number
Category	**agglom_W**	
0=Blood Donor	**0**	6
	1	240

	2	280
0s=suspect Blood Donor	0	4
	2	1
	4	2
1=Hepatitis	0	4
	1	11
	2	3
	4	2
2=Fibrosis	0	4
	1	8
3=Cirrhosis	0	16
	1	1
	3	3
	4	4

In [141]:

```
(data[['Category','agglom_s','kmeans_2']]
.groupby(['Category','agglom_s'])
.size()
.to_frame()
.rename(columns={0:'number'}))
```

Out[141]:

		number

Category	agglom_s	
0=Blood Donor	0	526
0s=suspect Blood Donor	0	4
	2	1
	5	1
	6	1
1=Hepatitis	0	20
2=Fibrosis	0	12
3=Cirrhosis	0	17
	1	3
	3	2
	4	1
	7	1

In [142]:

```
(data[['Category','agglom_s','kmeans_8']]
.groupby(['Category','agglom_s'])
.size()
.to_frame()
.rename(columns={0:'number'}))
```

Out[142]:

		number
Category	agglom_s	
0=Blood Donor	0	526

0s=suspect Blood Donor	**0**	4
	2	1
	5	1
	6	1
1=Hepatitis	**0**	20
2=Fibrosis	**0**	12
3=Cirrhosis	**0**	17
	1	3
	3	2
	4	1
	7	1

In [ ]:

```

```

In [ ]:

```

```

Suggestions for next steps in analyzing this data

Given the **versatile nature of clustering**, there is a great room for improvement in the next iteration of this study/analysis of the HCV+data.
Other clustering algorithms like **Mean Shift** or **DBSCAN** might also be used to study this dataset.
Furthermore, we can alter the **hyperparameters** like number of clusters and type of linkage can be changed for Agglomerative Clustering and KMeans Clustering, as the case maybe.