CS60050 : Machine Learning

Report

ASSIGNMENT - 2 [E]

Prepared by:
Group No. 51
Shristi Singh (19CS10057)
Mayank Kumar (19CS30029)

October 20, 2021

I. Analysis of the dataset

Shape of dataset: (583, 11)

Basic details about dataset:

The dataset has 11 attributes that decide the target, i.e. if the person is liver patient (0) or not (1).

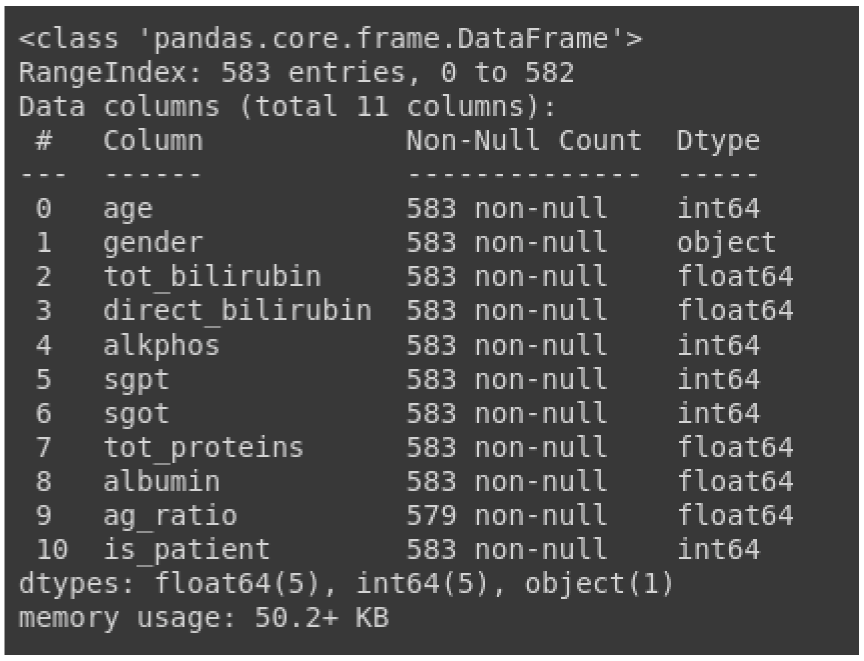
- Age (age)
- Gender (gender)
- Total Bilirubin (tot_bilirubin)
- Direct Bilirubin (direct_bilirubin)
- Alkaline Phosphatase (alkphos)
- Alanine Aminotransferase (sgpt)
- Aspartate Aminotransferase (sgot)
- Total Proteins (tot_proteins)
- Albumin (albumin)
- Albumin to Globulin Ratio (ag_ratio)
- Target field (is_patient)

A change in the dataset:

The dataset initially given has wrong column names. This can be proved by the fact that some columns have values in the range which is not possible for a human being. In the attribute information given at kaggle, the order of those attributes are the real column names for the dataset. So, we need to rename and assign proper names to each column.

Overview of the dataset:

The target class label is an integer, and 4 values are missing in the ag_ratio column. We have got a object data type column 'gender'.

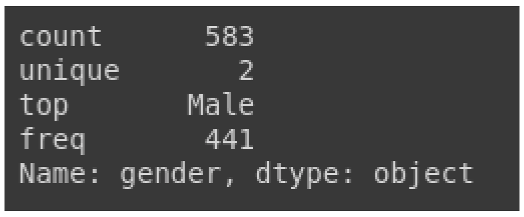


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   age               583 non-null    int64
 1   gender            583 non-null    object
 2   tot_bilirubin     583 non-null    float64
 3   direct_bilirubin  583 non-null    float64
 4   alkphos           583 non-null    int64
 5   sgpt              583 non-null    int64
 6   sgot              583 non-null    int64
 7   tot_proteins      583 non-null    float64
 8   albumin           583 non-null    float64
 9   ag_ratio          579 non-null    float64
 10  is_patient        583 non-null    int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

Description of attributes:

1. Age (age): Age of the person
2. Gender (gender): Gender - Male or Female
3. Total Bilirubin (tot_bilirubin): Bilirubin is a yellowish pigment made during the normal breakdown of red blood cells. It passes through the liver and is eventually excreted out of the body. Higher than normal levels of bilirubin may indicate different types of liver or bile duct problems.
   Normal bilirubin levels fall in 0.1 to 1.2 mg/dL. Anything above 1.2 mg/dL is usually considered high.
4. Direct Bilirubin (direct_bilirubin): Bilirubin bound to the glucuronic acid is called direct or conjugated bilirubin. Bilirubin not bound to the glucuronic acid is called indirect or unconjugated bilirubin. tot_bilirubin = direct_bilirubin + indirect_bilirubin
   Normal level for direct bilirubin is less than 0.4 mg/dL.
5. Alkaline Phosphatase (alkphos): ALP test measures the amount of alkaline phosphatase enzyme circulating in the bloodstream. ALP helps break down proteins in the body and exists in different forms, depending on where it originates. Abnormal levels of ALP in blood most often indicate malnutrition, kidney cancer tumors, intestinal issues, a pancreas problem, or a serious infection. This test is important as in most cases, it reports the absence of liver disease. The normal range for serum ALP level is 20 to 140 IU/L.
6. Alanine Aminotransferase (sgpt): The ALT or SGPT (Serum Glutamic-Pyruvic Transaminase) test helps to find out if a disease, drug or injury has damaged the liver. If the liver is damaged, more ALT is released into the bloodstream and its level rises. The normal range is 7 to 55 units per liter. A very high ALT levels can be caused by acute viral hepatitis, an overdose of drugs or liver cancer.
7. Aspartate Aminotransferase (sgot): AST or SGOT (Serum Glutamic-Oxaloacetic Transaminase) is an enzyme made by the liver to convert food into energy. A normal ALT test result can range from 7 to 55 units per liter. High levels of these enzymes can be a sign that the liver is injured or irritated, and the enzymes are leaking out of the liver cells. A very high AST levels can be caused by acute viral hepatitis, damage to the liver from toxic substances or a blockage in blood flow to the liver.
8. Total Proteins (tot_proteins): The total protein test measures the total amount of two classes of proteins found in the fluid portion of your blood - albumin and globulin. Albumin helps prevent fluid from leaking out of blood vessels. Globulins are an important part of the immune system. The normal range is 6.0 to 8.3 grams per deciliter (g/dL) or 60 to 83 g/L. High blood protein may be a symptom of underlying medical conditions, including dehydration, infections like hepatitis C or cancers like multiple myeloma. This test helps diagnose health conditions like kidney disease, liver disease, malnutrition, etc.
9. Albumin (albumin): Albumin is a protein that is produced in the liver and then enters the bloodstream where it is carried to other parts of the body. Albumin's biological functions are to keep fluid from leaking out of the blood and to carry substances like hormones, enzymes, and vitamins in the body. The typical value for serum albumin in blood is 3.5 to 5.4 g/dL. Low albumin levels can indicate the presence of liver disease.
10. Albumin to Globulin Ratio (ag_ratio): The A/G Ratio test compares the concentrations of albumin and globulin in the blood. $ag\_ratio = \frac{Albumin}{Globulin} = \frac{Albumin}{Total\ Proteins - Albumin}$. An imbalance in the ratio of albumin to globulin may signify ongoing inflammation, liver problems, or in rare cases immunodeficiency. In general, an albumin/globulin ratio between 1.1 and 2.5 is considered normal.

Gender (gender) column: There are two types of values in the class, and Male has more count.



```
count      583
unique       2
top       Male
freq       441
Name: gender, dtype: object
```

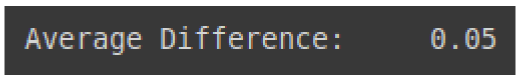
We will encode the gender class to change its type from object to int. We assign gender as 1 if it is Male and 0 in the other case, i.e, Female.

Albumin to Globulin Ratio (ag_ratio) column:

There are four rows whose ag_ratio value is missing.

We know that $ag\_ratio = \frac{Albumin}{Globulin} = \frac{Albumin}{Total\ Proteins - Albumin}$

This formula can be verified for the dataset by printing the given ag_ratio value and the computed value side by side, and checking the average difference.

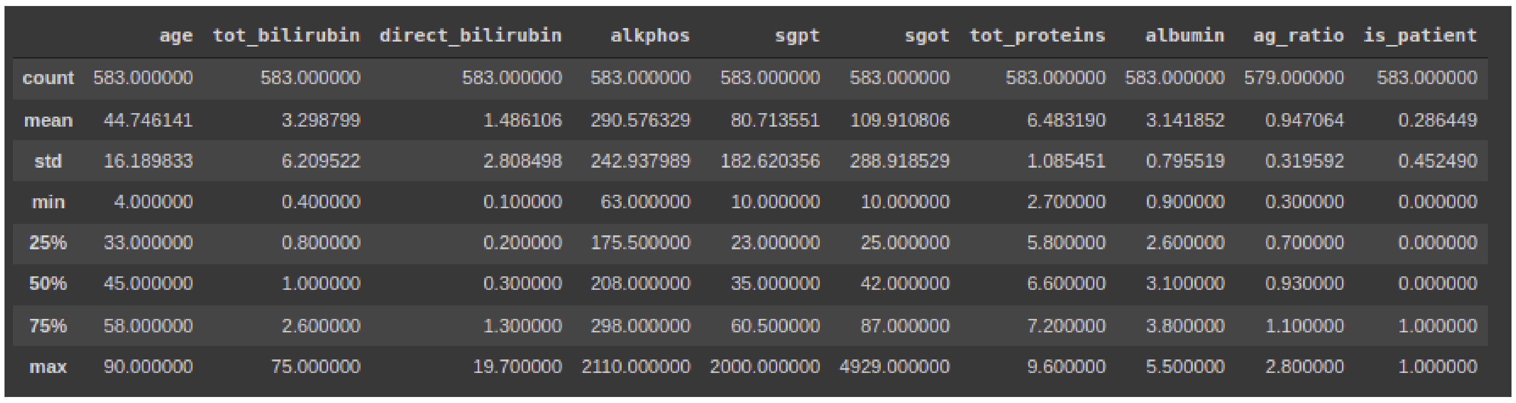


```
Average Difference:     0.05
```

Here, the average difference is very low, and thus, it verifies the formula.

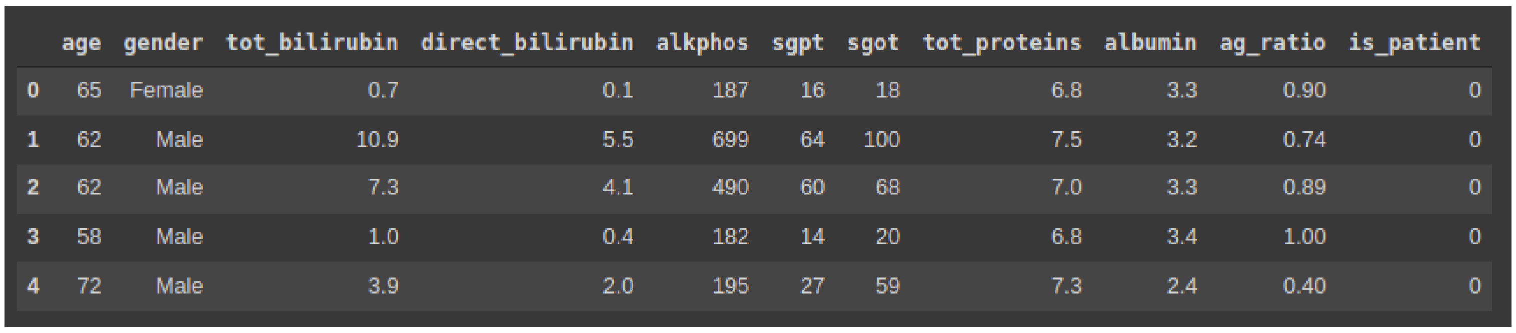
So, for the missing rows, we fill them with their theoretically computed values.

Descriptive Statistics of the dataset:



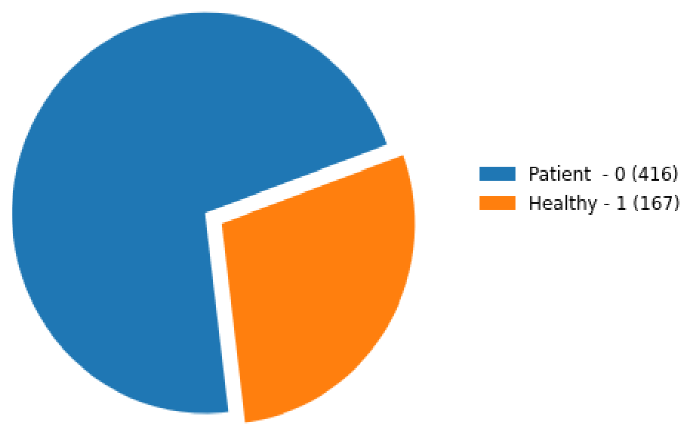
	age	tot_bilirubin	direct_bilirubin	alkphos	sgpt	sgot	tot_proteins	albumin	ag_ratio	is_patient
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	579.000000	583.000000
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	0.947064	0.286449
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	0.319592	0.452490
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	0.300000	0.000000
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	0.700000	0.000000
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	0.930000	0.000000
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	1.100000	1.000000
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	2.800000	1.000000

First 5 rows in dataset:



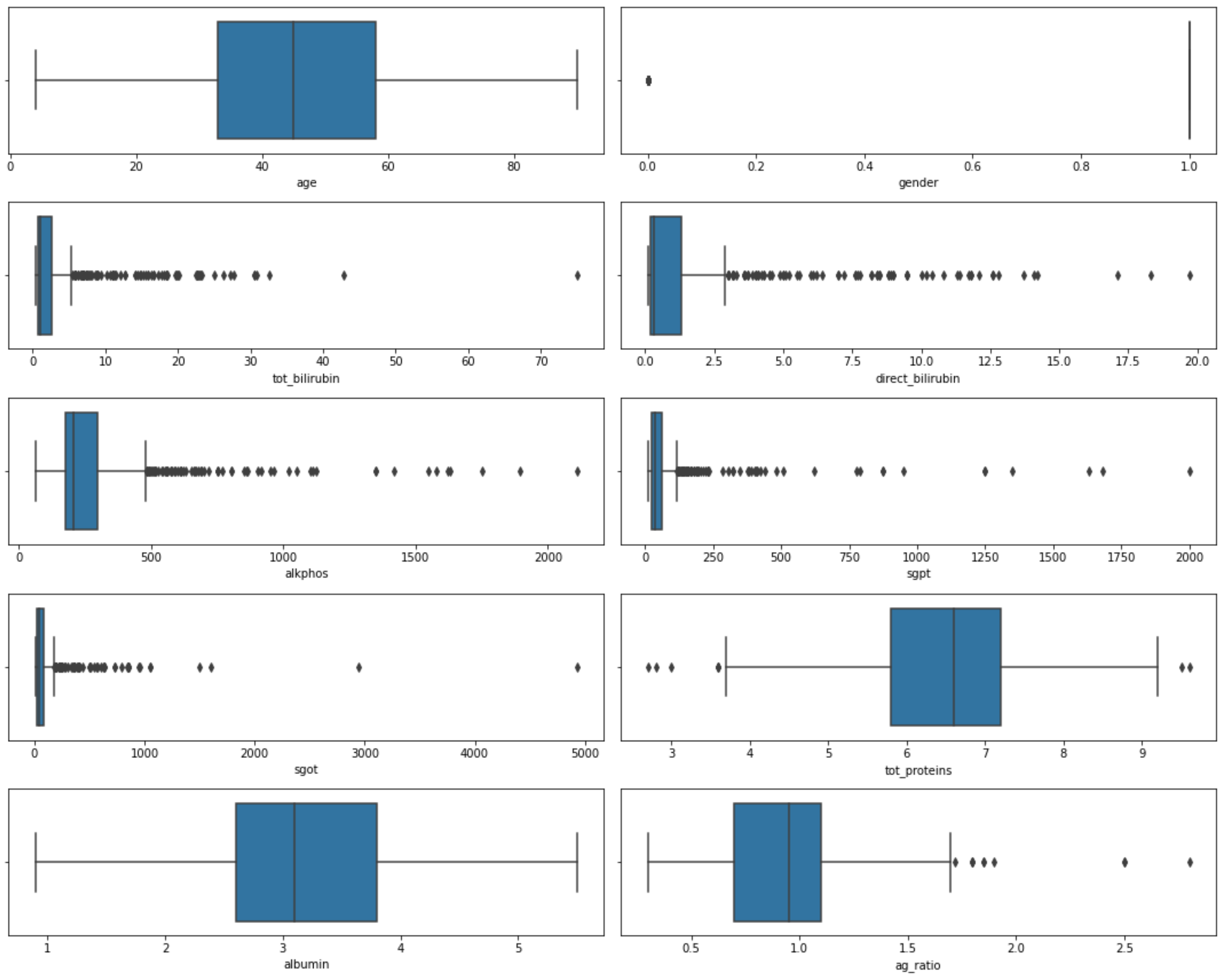
	age	gender	tot_bilirubin	direct_bilirubin	alkphos	sgpt	sgot	tot_proteins	albumin	ag_ratio	is_patient
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	0
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	0
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	0
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	0
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	0

Count for Target Values (Class Labels)



Here, we can see that most of the rows have is_patient as 0, i.e, most of them are liver patients.

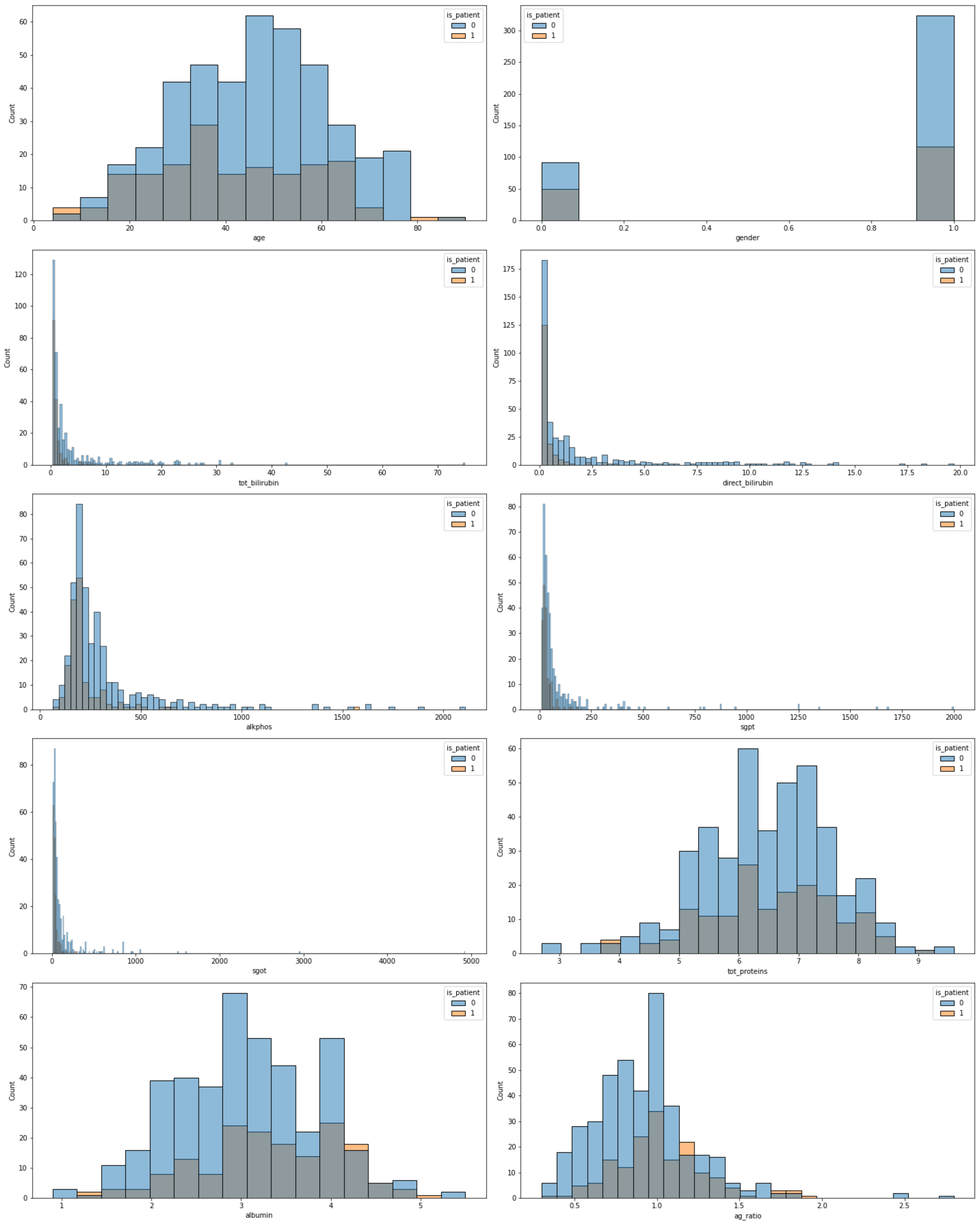
Distribution of attribute values (univariate):



The boxplot displays the distribution of data for an attribute based on a five number summary - minimum, first quartile, median, third quartile and maximum. It helps in analyzing if data is symmetrical or skewed, or how they are grouped.

From the box plots, an important observation here is that there are a lot of outlier points in most of the columns, so we would have to get rid of them so that the model does not go towards overfitting. These outliers cannot be removed directly as an extreme value might give us crucial information to decide if the person is a liver patient. Since the target value 0 has much more frequency than 1, we might need to take this into account while checking for outliers.

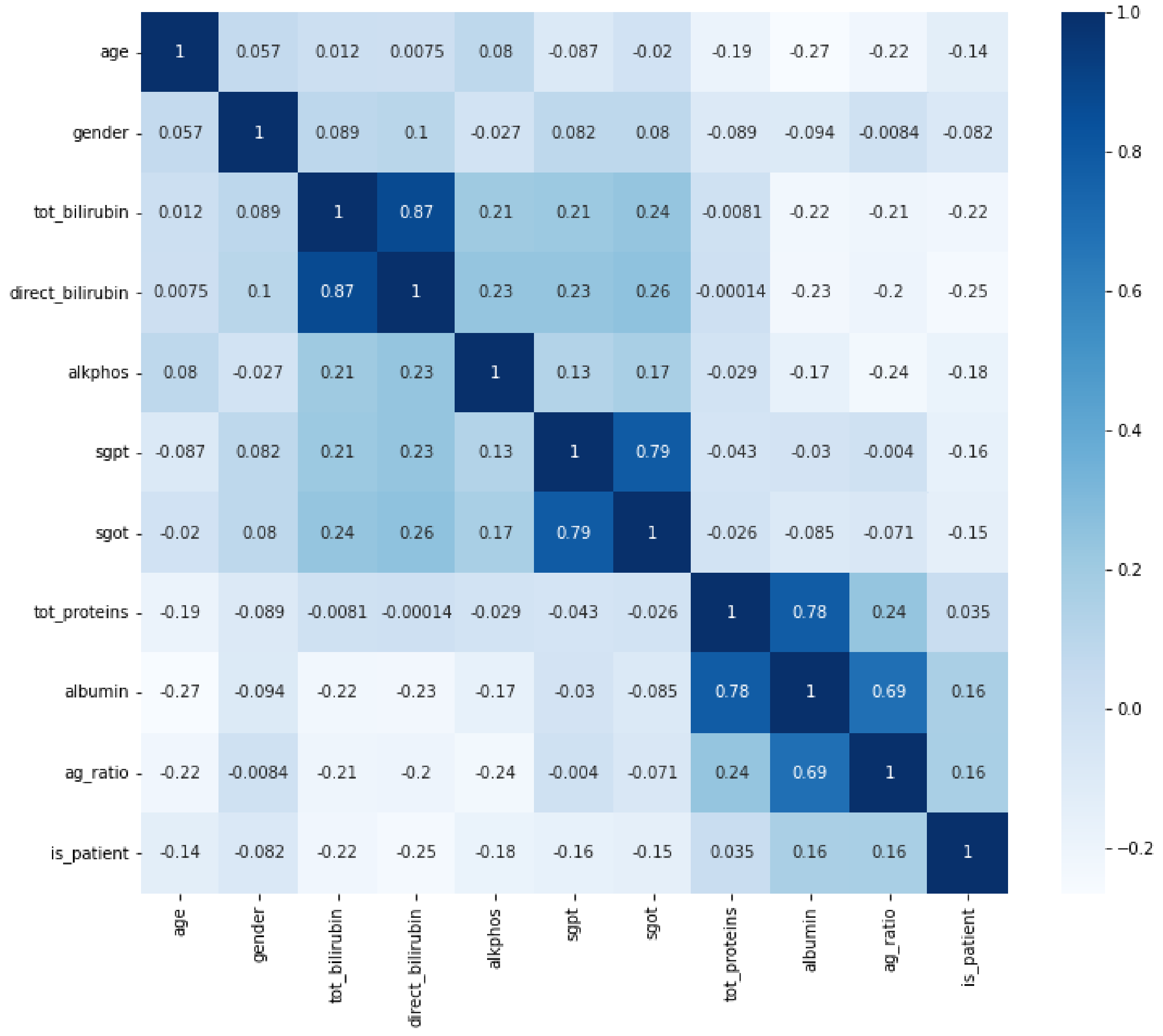
<u>Distribution of attribute values (bivariate):</u>



These bivariate plots describe the distribution of data in a detailed manner and we can see that for all the attributes, the probability of being a liver patient is high if its value is beyond the normal range. Also, most of the males (1) are liver patients, and the patients are more common in people at higher age. We can see that in all the attributes, people having the respective value in normal range tend to be healthy while people having abnormal value have higher chance of being a liver patient.

Heatmap (Multi-Collinearity of Attributes):

Heatmap is used to better visualize the magnitude of a phenomenon in a dataset and assist by directing towards areas that matter the most through colour-codings.



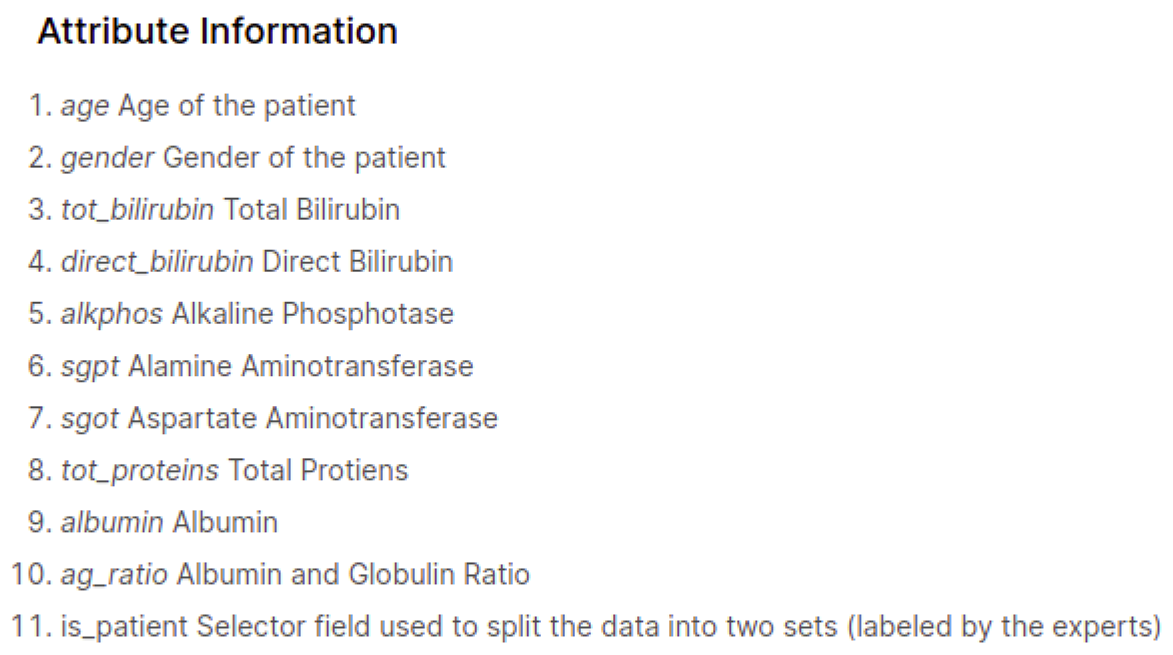
	age	gender	tot_bilirubin	direct_bilirubin	alkphos	sgpt	sgot	tot_proteins	albumin	ag_ratio	is_patient
age	1	0.057	0.012	0.0075	0.08	-0.087	-0.02	-0.19	-0.27	-0.22	-0.14
gender	0.057	1	0.089	0.1	-0.027	0.082	0.08	-0.089	-0.094	-0.0084	-0.082
tot_bilirubin	0.012	0.089	1	0.87	0.21	0.21	0.24	-0.0081	-0.22	-0.21	-0.22
direct_bilirubin	0.0075	0.1	0.87	1	0.23	0.23	0.26	-0.00014	-0.23	-0.2	-0.25
alkphos	0.08	-0.027	0.21	0.23	1	0.13	0.17	-0.029	-0.17	-0.24	-0.18
sgpt	-0.087	0.082	0.21	0.23	0.13	1	0.79	-0.043	-0.03	-0.004	-0.16
sgot	-0.02	0.08	0.24	0.26	0.17	0.79	1	-0.026	-0.085	-0.071	-0.15
tot_proteins	-0.19	-0.089	-0.0081	-0.00014	-0.029	-0.043	-0.026	1	0.78	0.24	0.035
albumin	-0.27	-0.094	-0.22	-0.23	-0.17	-0.03	-0.085	0.78	1	0.69	0.16
ag_ratio	-0.22	-0.0084	-0.21	-0.2	-0.24	-0.004	-0.071	0.24	0.69	1	0.16
is_patient	-0.14	-0.082	-0.22	-0.25	-0.18	-0.16	-0.15	0.035	0.16	0.16	1

Among the attributes, it can be seen that there is a very strong correlation between tot_bilirubin & direct_bilirubin, sgpt & sgot, tot_proteins & albumin, and albumin & ag_ratio. This is as we would have expected according to the description of attributes.

II. Procedure

Pre-processing:

As already mentioned, at first, we change the labels of each column according to the description of attribute information provided in Kaggle.



Attribute Information

1. *age* Age of the patient
2. *gender* Gender of the patient
3. *tot_bilirubin* Total Bilirubin
4. *direct_bilirubin* Direct Bilirubin
5. *alkphos* Alkaline Phosphotase
6. *sgpt* Alamine Aminotransferase
7. *sgot* Aspartate Aminotransferase
8. *tot_proteins* Total Protiens
9. *albumin* Albumin
10. *ag_ratio* Albumin and Globulin Ratio
11. is_patient Selector field used to split the data into two sets (labeled by the experts)

Then, for the gender column, we have encoded it as 1 if the person is Male and 0 for Female to convert them into categorical data.

Handling missing data: We found four rows which have their value missing in the ag_ratio column. We have filled the missing values with their theoretically computed value according to its formula.

Attribute Selection: For this, we will add two new columns - indirect bilirubin and globulin. These attributes are already defined and are equally important to determine if a person is a liver patient. They would give better results if dealt separately. So, we will also drop tot_bilirubin and tot_proteins attributes as they are already being considered and can be computed through a linear combination of other attributes.

KMeans Class (Implementation):

KMeans clustering is an unsupervised algorithm that makes inferences using only input vectors without referring to the labelled outcomes. The objective of kmeans is to group similar data points together and discover underlying patterns. These groups are called clusters, and using kmeans, we classify a data item into a certain cluster. The 'means' in the name kmeans refers to finding the centroid.

This algorithm initially selects k data points randomly as centroids if the centroids are not passed through parameters. If all the K centroids are passed in its parameters, it simply uses them for further computations.
For the rest of the dataset, each of the rows is assigned to a cluster whose centroid is closest to it.
Then, we re-calculate the centroid and repeat the assignment process.
This is continued until either the position of centroids stabilize and their values do not change, or the defined number of maximum iterations has been achieved.
There can be some residual difference of very low order in the old and current position of centroids, and even when the convergence is reached, the two positions might not be equal. To avoid this, instead of equality, we

will check if the difference is less than equals a very small value, which is defaulted as 10^{-6}, so that we halt the process when it converges.

Upon convergence of the centroids, we store the label of clusters assigned to each row of the dataset. We also compute and check for the training loss in this label assignment. To make predictions on a list of data items, we iterate through the list, and for each item, we assign the label of the closest cluster centroid from it.

To use the kmeans classifier model, the attributes and target is separated into X and y. A number 'k' is taken as input from the user, which is the number of clusters to be used for kmeans algorithm. Then, we run our input through the model and get their output labels.

Evaluating clustering performance:

KMeans is a clustering algorithm, and validating its performance is a bit tricky as compared to supervised learning models. For an ideal clustering, the intra cluster distance is minimal while the inter cluster distance is maximal. There are two types of measures to assess the performance

- Extrinsic Measures: Requires ground truth labels. Ex- Adjusted Rand Index, Adjusted Mutual Information, Homogeneity, Normalized Mutual Information, etc.
- Intrinsic Measures: Doesn't require ground truth labels. Ex- Silhouette Coefficient, Calinski-Harabasz Index, etc.

Hopkins Statistic: Before checking for the values of different metrics, it is important to check the cluster tendency of the given dataset and find if the dataset contains meaningful clusters. This can be done using Hopkins Statistic, which is a statistics hypothesis test which measures the probability that a given dataset is generated by a uniform data distribution and tests the spatial randomness of the data. We will use pyclustertend package to get the Hopkins test value. The data is highly clustered if it tends to 1.

The clustering performance is evaluated using the following metrics. We have taken k=2 for evaluating this.

- Adjusted Rand Index: It measures the similarity of data points present in the clusters. Random label assignments have an adjusted Rand index score close to 0.0 for any value of n_clusters and n_samples. $ARI = \frac{RI - Expected\ RI}{max(RI) - Expected\ RI},\ where\ RI = \frac{No. of\ agreeing\ pairs}{Total\ no. of\ pairs}.$
  sklearn.metrics.adjusted_rand_score is used to measure this metric.
- Adjusted Mutual Info Score: It is an adjustment of the Mutual Information score to account for chance by adopting a hypergeometric model of randomness. Its value is 1 if the two partitions are identical. $AMI(U,\ V) = \frac{MI(U,V) - E\{MI(U,V)\}}{max\{H(U),H(V)\} - E\{MI(U,V)\}},\ where\ MI\ is\ the\ mutual\ information\ and\ H\ represents\ entropy.$
  sklearn.metrics.cluster.adjusted_mutual_info_score is used to measure this metric.
- Homogeneity Score: It checks if a cluster contains only samples belonging to a single class. The score is bounded between 0 and 1, with higher values denoting a perfect labelling.
  sklearn.metrics.cluster.homogeneity_score is used to measure this metric.
- Normalized Mutual Info Score: It normalizes the mutual info score to scale between 0 and 1, i.e, no mutual information and perfect correlation. Unlike ARI and AMI, this metric is not adjusted for chance.
  sklearn.metrics.cluster.normalized_mutual_info_score is used to measure this metric.
- Fowlkes-Mallows Score: It is the geometric mean of precision and recall. It ranges from 0 to 1, where the higher value denotes better similarity between two clusters. If TP is true positive, FP is false positive and FN is false negative, then Fowlkes-Mallows Index, $FMI = \frac{TP}{\sqrt{(TP+FP)\times(TP+FN)}}.$
  sklearn.metrics.cluster.fowlkes_mallows_score is used to measure this metric.

- Silhouette Coefficient: It gives a measure for similarity of a point to its own cluster compared to the other clusters. Its score is bounded between -1 and +1, where the higher score indicates that clusters are dense and well separated. Its formula is given by: $s = \frac{b-a}{max(a, b)}$.
  sklearn.metrics.silhouette_score is used to measure this metric.
- Calinski-Harabasz Index: CH score is defined as the ratio between the within-cluster dispersion and the between-cluster dispersion. The higher the Index, the better the performance. This is given by formula: $CH\ index = \frac{tr(B_k)}{tr(W_k)} \times \frac{n_E - k}{k-1}$, where tr($B_k$) is the trace of the between-cluster dispersion matrix and tr(W_k) is the trace of the within-cluster dispersion matrix.
  sklearn.metrics.calinski_harabasz_score is used to measure this metric.

Determining the best suitable K:

Elbow Curve Method: K-means algorithm works such that the total within-cluster variation is minimal. To check for such variations, the Within-cluster Sum of Squared Errors (WSS) is computed for different values of k, and that k is chosen after which the WSS falls suddenly (elbow).

Silhouette Method: The silhouette coefficient gives a measure for similarity of a point to its own cluster compared to the other clusters. Its range is -1 to +1, the higher value indicating that the data points belong to correct clusters.

Test-A (Initializing K cluster centroids with random points):

1. K random points are selected from the dataset.
2. The remaining data is splitted into two parts in the ratio 80:20. Here, the larger part is the training dataset, while the smaller one is the test dataset.
3. KMeans algorithm is applied on the training data with the selected K points as cluster centroids.
4. Then, the test data is labelled using the stable centroids found from the previous step.
5. Adjusted Rand Index (ARI) and Fowlkes-Mallows Score (FMS) metrics are used to evaluate the clustering performance on test dataset.
6. For each set of K random points selected in step 1, steps 2 to 5 are repeated 50 times, and the average of ARI and FMS metrics is reported.
7. Steps 1 to 6 are considered as one iteration and these steps are repeated 50 times.
8. The dispersion of the average metric is reported.

Heuristic to initialize K cluster centroids to minimize the observed variation:

The average metric for each iteration in the previous part varied a lot. This shows that the kmeans algorithm is sensitive to the initialization of the centroids, and a poor initialization of centroids might result in poor clustering. To minimize this, instead of random initialization, we will use **kmeans++** to initialize the centroids.

KMeans++: This algorithm starts with selecting the first cluster centroid from the dataset randomly. Then, for each point, its distance from the closest previously chosen centroid is computed. We choose the next centroid in a way that ensures maximum distance of the point from the nearest centroid. This step is repeated until we get K centroids.

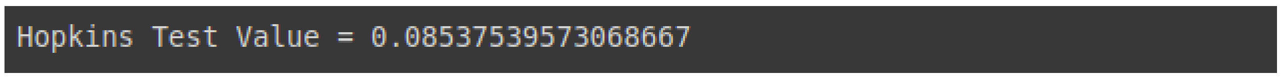
This approach by kmeans++ algorithm also ensures that the time taken to converge cluster centroids will be much lesser, as the centroids are already far from each other and would be expected to be in different clusters. Also, due to the same reason, a better cluster could be formed which would make the clustering performance even better.

We use the get_kmeanspp_centroids() function to implement this heuristic and return the centroids and their respective index in the dataset. We would need the indexes so that we do not include the centroids in our training or test data. Now that we have described the process of getting K centroids, we proceed with those steps in the Test-A mentioned earlier to observe the change in variations. Do note that we will be using the kmeans++ heuristic (instead of random initialization) in step 1.

III. Results and Analysis

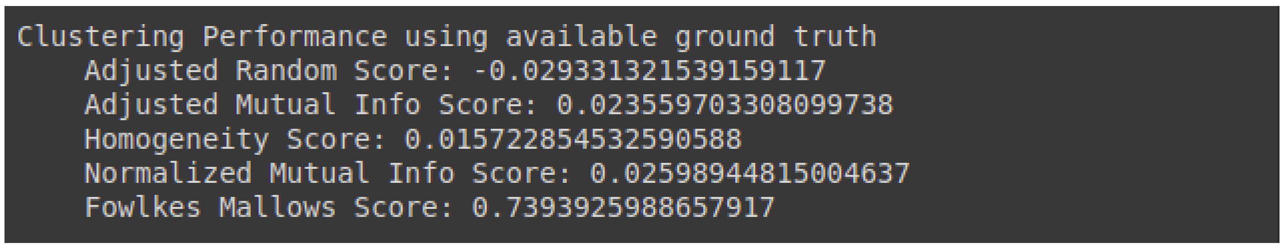
Clustering Performance:

Hopkins Test
The result of Hopkins Test from pyclustertend package is:



```
Hopkins Test Value = 0.08537539573068667
```

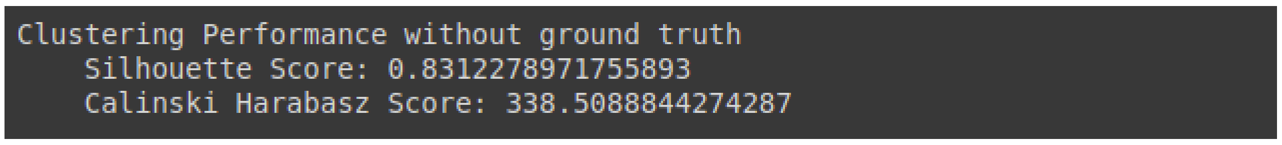
Clustering Performance Using Available Ground Truth



```
Clustering Performance using available ground truth
    Adjusted Random Score: -0.029331321539159117
    Adjusted Mutual Info Score: 0.023559703308099738
    Homogeneity Score: 0.015722854532590588
    Normalized Mutual Info Score: 0.02598944815004637
    Fowlkes Mallows Score: 0.7393925988657917
```

- For our results, we have got a low, negative ARI, and the reason might be that the metric would evaluate properly for large, equal-sized clusters. On random assignment, the metric assumes that the target classes are equiprobable, but our dataset has most of the rows corresponding to 0 (Patient).
- The Adjusted Mutual Info score is used for unbalanced clusters but the clusters should be small enough. Although the value is positive for AMI, it is low, which might be because of the size of the clusters formed. The given dataset has large cluster for 0 (Patient).
- We have seen that Homogeneity score checks for the belongingness of a sample to a single class. If we look into the dataset and its description, it is evident that the value for each attribute is very different for the two genders - male and female. This might have affected the evaluation of this score, resulting in a low value.
- The normalized mutual information score is low, and it was already expected as we have already mentioned that NMI is not adjusted for chance. So, even if NMI achieves its maximum value of 1, it is not a suitable clustering comparison measure.
- From the formula of the Fowlkes-Mallows Index, it is evident that no property is assumed about the cluster structure and the score is evaluated just on the basis of precision and recall. Thus, it proves to be significantly advantageous than the other evaluation methods. We can see that the Fowlkes-Mallows Score comes out to be a decent value of 0.739.

Clustering Performance Without Ground Truth

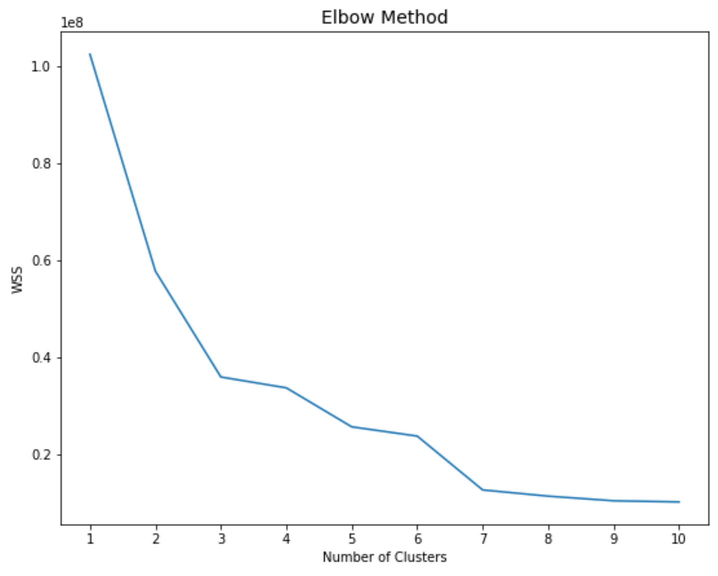


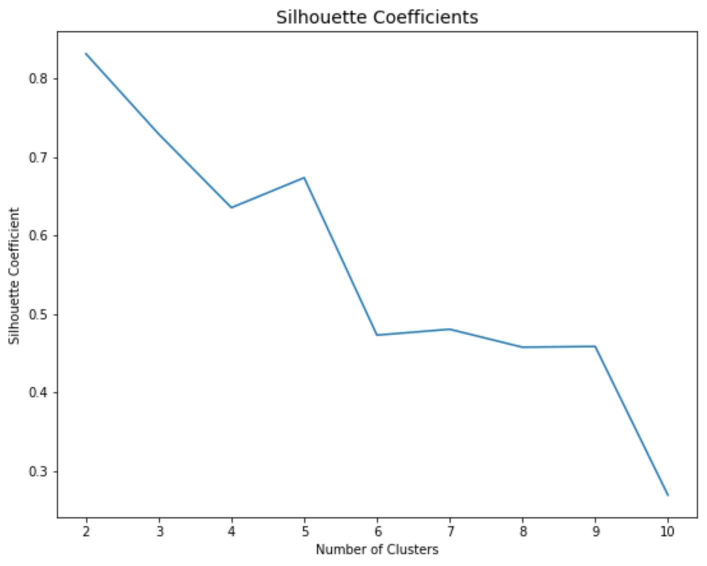
```
Clustering Performance without ground truth
    Silhouette Score: 0.8312278971755893
    Calinski Harabasz Score: 338.5088844274287
```

- In this case, the value of 0.831 for Silhouette coefficient and 338.51 for CH Index asserts the division of data points into fairly dense and well-separated clusters.

In general, it is observed that the scores for metrics in presence of the ground truth are very low, and the common reason behind this might be the value from Hopkins test which tends to zero. In the case of clustering performance without the ground truth, although the values are high, the clusters might not have formed in a desired way and the high score would then be an inaccurate interpretation of the evaluation.

Determining Most Suitable K:

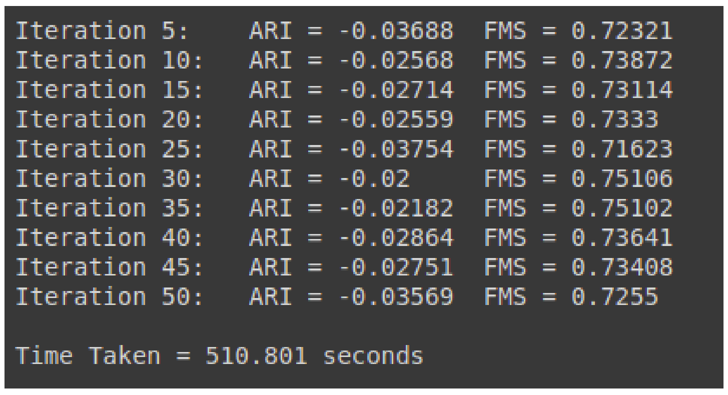




As expected from the Elbow Method, we can see that the plot looks like an arm with its elbow at 3, although the elbow is not very sharp. This is just by visualizing the plot, and we cannot directly say that the optimal K is 3. It is just an indication that the optimal K should be somewhere around 3.
For the Silhouette Coefficients plot, we can see that the global maxima is at k = 2. We can also see peak points at 4, 7 and 9, but these are definitely less than the maximum point at 2. Also, 7 and 9 are not around 3, the elbow point from the Elbow Curve method.
So, the ***most suitable number of clusters (K) = 2***.

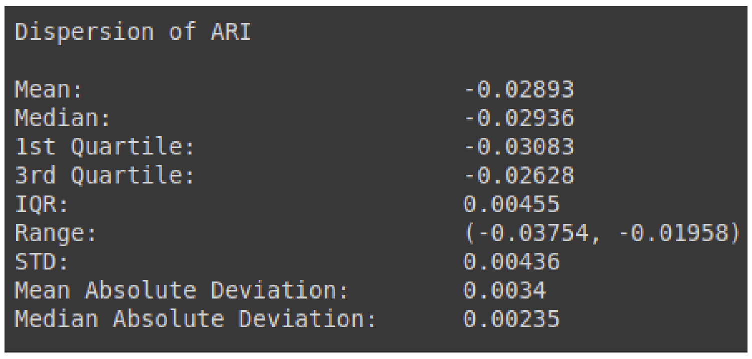
Test-A with random initialization:

Output for metrics Adjusted Rand Index (ARI), Normalized Mutual Information and Fowlkes Mallows Score (FMS) for 50 iterations. Do note that the metrics reported for each iteration are average for initialized centroids over 50 different 80:20 splits.

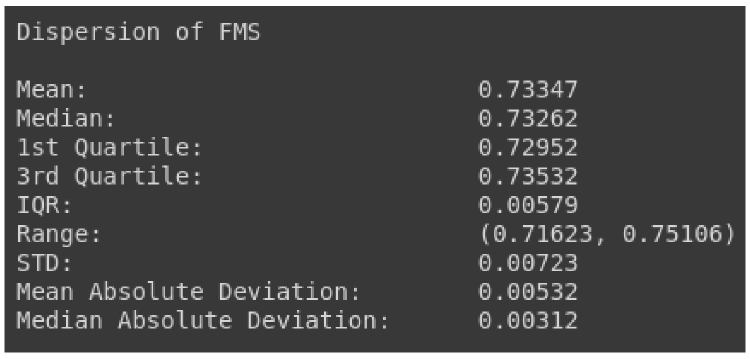


```
Iteration 5:    ARI = -0.03688  FMS = 0.72321
Iteration 10:   ARI = -0.02568  FMS = 0.73872
Iteration 15:   ARI = -0.02714  FMS = 0.73114
Iteration 20:   ARI = -0.02559  FMS = 0.7333
Iteration 25:   ARI = -0.03754  FMS = 0.71623
Iteration 30:   ARI = -0.02     FMS = 0.75106
Iteration 35:   ARI = -0.02182  FMS = 0.75102
Iteration 40:   ARI = -0.02864  FMS = 0.73641
Iteration 45:   ARI = -0.02751  FMS = 0.73408
Iteration 50:   ARI = -0.03569  FMS = 0.7255

Time Taken = 510.801 seconds
```

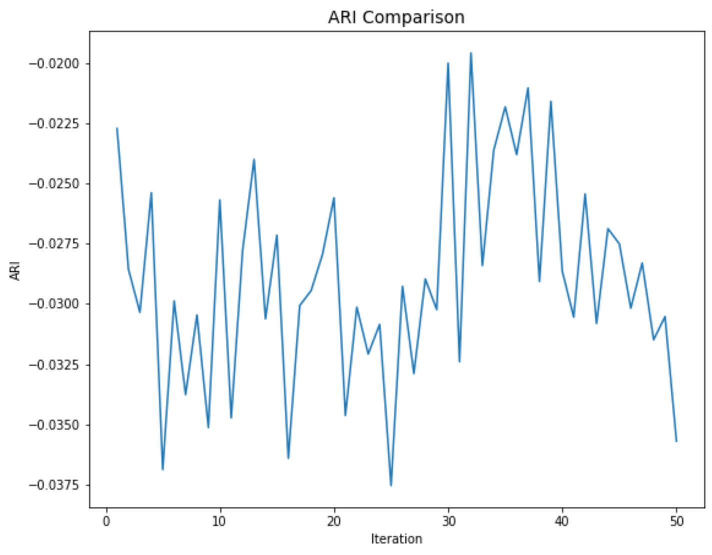
Dispersion of the metrics for 50 iterations

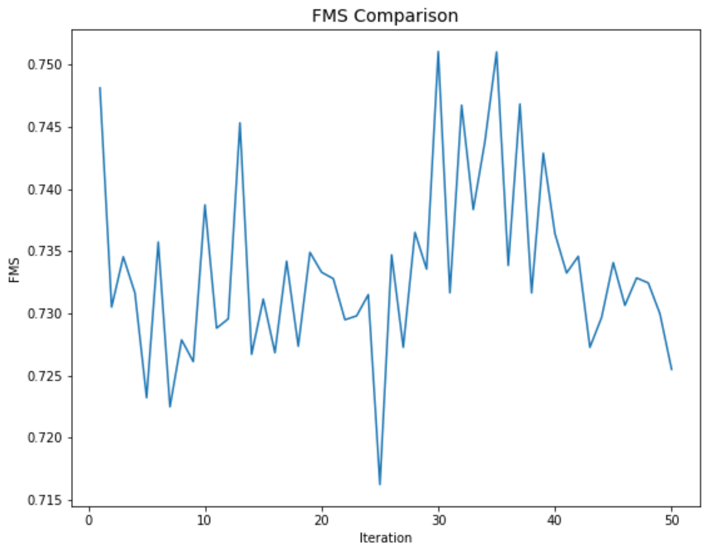


```
Dispersion of ARI

Mean:                           -0.02893
Median:                         -0.02936
1st Quartile:                   -0.03083
3rd Quartile:                   -0.02628
IQR:                            0.00455
Range:                          (-0.03754, -0.01958)
STD:                            0.00436
Mean Absolute Deviation:        0.0034
Median Absolute Deviation:      0.00235
```



```
Dispersion of FMS

Mean:                           0.73347
Median:                         0.73262
1st Quartile:                   0.72952
3rd Quartile:                   0.73532
IQR:                            0.00579
Range:                          (0.71623, 0.75106)
STD:                            0.00723
Mean Absolute Deviation:        0.00532
Median Absolute Deviation:      0.00312
```





Here, both Adjusted Rand Index and Fowlkes-Mallows Score varies randomly for the fifty iterations. From the dispersion reported above, we can see the different deviations in both metrics, and it asserts that the KMeans algorithm is sensitive to initial cluster centroids. To make the model better and more stable, we need it to give similar results for different iterations. This is done by using KMeans++ for selecting the initial K centroids.

Another point to be noted from the output is that it was executed in 510.801 seconds. The main reason for taking a long time is that in each of the 50 iterations, we take 50 different 80:20 splits, and this makes the loop go for 2500 times. Also, as the centroids are initialized randomly, it takes more time for their position to get stabilized.

Test-A with the heuristic (kmeans++):

Output for metrics Adjusted Rand Index (ARI), Normalized Mutual Information and Fowlkes Mallows Score (FMS) for 50 iterations. Do note that the metrics reported for each iteration are average for initialized centroids over 50 different 80:20 splits.

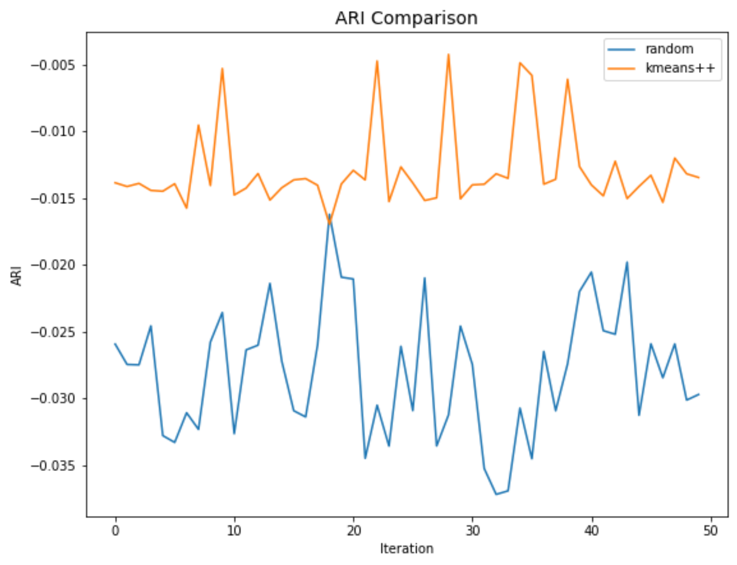
```
Iteration 5:    ARI = -0.00394  FMS = 0.77113
Iteration 10:   ARI = -0.011    FMS = 0.75627
Iteration 15:   ARI = -0.01307  FMS = 0.7592
Iteration 20:   ARI = -0.01849  FMS = 0.75794
Iteration 25:   ARI = -0.01423  FMS = 0.75787
Iteration 30:   ARI = -0.015    FMS = 0.75404
Iteration 35:   ARI = -0.01402  FMS = 0.76272
Iteration 40:   ARI = -0.01852  FMS = 0.75751
Iteration 45:   ARI = -0.00843  FMS = 0.76619
Iteration 50:   ARI = -0.01339  FMS = 0.75534

Time Taken for KMeans++ = 193.357 seconds
```

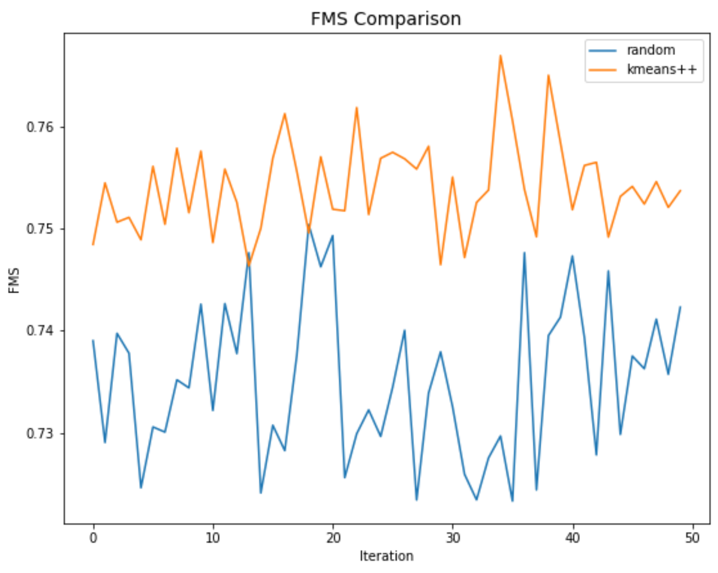
Dispersion of the metrics for 50 iterations

```
Dispersion of ARI for KMeans++

Mean:                           -0.0129
Median:                         -0.01392
1st Quartile:                   -0.01435
3rd Quartile:                   -0.01318
IQR:                            0.00117
Range:                          (-0.01696, -0.00425)
STD:                            0.00306
Mean Absolute Deviation:        0.00207
Median Absolute Deviation:      0.00073
```

```
Dispersion of FMS for KMeans++

Mean:                           0.75416
Median:                         0.75385
1st Quartile:                   0.75128
3rd Quartile:                   0.75692
IQR:                            0.00563
Range:                          (0.74637, 0.76702)
STD:                            0.00444
Mean Absolute Deviation:        0.00354
Median Absolute Deviation:      0.00307
```

Comparison between outputs of Test-A with random initialization and the heuristic kmeans++





Upon using the heuristic kmeans++ to select the initial K cluster centroids, we can see that the deviations for each deviation parameter has decreased.
For ARI, Interquartile Range decreased by 74.28% (0.00455 to 0.00117), Standard Deviation decreased by 29.82% (0.00436 to 0.00306), Mean Absolute Deviation decreased by 39.12% (0.0034 to 0.00207) and Median Absolute Deviation decreased by 68.94% (0.00235 to 0.00073). Similarly, all of these parameters were decreased for FMS as well, with Interquartile Range going down by 2.84%, Standard Deviation by 38.59%, Mean Absolute Deviation by 33.46% and Median Absolute Deviation 1.6%.

These decrement in all the deviation parameters shows that the kmeans++ approach was successful in minimizing the observed variations from random initialization. Although the decrease in deviation looks too small numerically, the percentage decrease has significant figures. Even then, the reason for small change might be that the values for different metrics were already very small.

Along with minimized variations, we can see that the range of the metrics for kmeans++ is more than that of random initialization. The orange line in the plots represents reported metrics for kmeans++, which is clearly above the blue line (random initialization). We know that for both Adjusted Rand Index and Fowlkes Mallows Score, higher value represents better performance. This verifies the already mentioned point, i.e, the clustering performance will get better upon using kmeans++.

Also, from the output shown above for kmeans++, we can see that the time taken is 193.357 seconds, which is much lesser than 510.801 seconds in the previous case. This is clearly because of the approach used in kmeans++, which chooses a centroid far from the previously chosen centroids, and the chances of initial centroids to be in different clusters increase.

Note: Steps to run the notebook is in the README.md file.

IV. References

https://www.kaggle.com/jeevannagaraj/indian-liver-patient-dataset
https://moodlecse.iitkgp.ac.in/moodle/pluginfile.php/32112/mod_resource/content/1/US_Learning.pdf
https://dl.matlabyar.com/siavash/ML/Book/Ethem%20Alpaydin-Introduction%20to%20Machine%20Learning-The%20MIT%20Press%20(2014).pdf
https://cs.nju.edu.cn/zlj/Course/DM_15_Lecture/Lecture_5.pdf
https://jmlr.csail.mit.edu/papers/volume17/15-627/15-627