

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.impute import SimpleImputer
from sklearn.cluster import KMeans
from sklearn import model_selection
from sklearn import metrics
from sklearn.preprocessing import scale
from sklearn import datasets, metrics, cluster
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
```

```
In [2]: #Load the data set
```

```
In [3]: df=pd.read_csv("cirrhosis.csv")
```

```
In [4]: #Data Exploration
```

```
In [5]: df.head() #checks if the dataset has been loaded
```

```
Out[5]:
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Eder
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	
2	3	1012	D	D-penicillamine	25594	M	N	N	N	
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	
4	5	1504	CL	Placebo	13918	F	N	Y	Y	

```
In [6]: df.info() #displays information about the data set
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   ID                    418 non-null   int64  
 1   N_Days                418 non-null   int64  
 2   Status                418 non-null   object  
 3   Drug                  312 non-null   object  
 4   Age                   418 non-null   int64  
 5   Sex                   418 non-null   object  
 6   Ascites               312 non-null   object  
 7   Hepatomegaly          312 non-null   object  
 8   Spiders               312 non-null   object  
 9   Edema                 418 non-null   object  
10  Bilirubin             418 non-null   float64 
11  Cholesterol            284 non-null   float64 
12  Albumin               418 non-null   float64 
13  Copper                310 non-null   float64 
14  Alk_Phos              312 non-null   float64 
15  SGOT                  312 non-null   float64 
16  Tryglicerides         282 non-null   float64 
17  Platelets             407 non-null   float64 
18  Prothrombin           416 non-null   float64 
19  Stage                 412 non-null   float64 
dtypes: float64(10), int64(3), object(7)
memory usage: 65.4+ KB

```

In [7]: `df.describe()` *#Basic statistics*

Out[7]:

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	
count	418.000000	418.000000	418.000000	418.000000	284.000000	418.000000	310
mean	209.500000	1917.782297	18533.351675	3.220813	369.510563	3.497440	97
std	120.810458	1104.672992	3815.845055	4.407506	231.944545	0.424972	85
min	1.000000	41.000000	9598.000000	0.300000	120.000000	1.960000	4
25%	105.250000	1092.750000	15644.500000	0.800000	249.500000	3.242500	41
50%	209.500000	1730.000000	18628.000000	1.400000	309.500000	3.530000	73
75%	313.750000	2613.500000	21272.500000	3.400000	400.000000	3.770000	123
max	418.000000	4795.000000	28650.000000	28.000000	1775.000000	4.640000	588

In [8]: *#Data Cleaning*

In [9]: `df.isnull().sum()` *#Checks and counts missing values*

```
Out[9]: ID          0
        N_Days      0
        Status      0
        Drug        106
        Age         0
        Sex         0
        Ascites     106
        Hepatomegaly 106
        Spiders     106
        Edema       0
        Bilirubin   0
        Cholesterol 134
        Albumin     0
        Copper      108
        Alk_Phos    106
        SGOT        106
        Tryglicerides 136
        Platelets   11
        Prothrombin 2
        Stage       6
        dtype: int64
```

```
In [10]: df.dropna() #drops missing values
```

```
Out[10]:
```

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	
2	3	1012	D	D-penicillamine	25594	M	N	N	N	
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	
4	5	1504	CL	Placebo	13918	F	N	Y	Y	
...
307	308	1153	C	D-penicillamine	22347	F	N	Y	N	
308	309	994	C	Placebo	21294	F	N	N	N	
309	310	939	C	D-penicillamine	22767	F	N	N	N	
310	311	839	C	D-penicillamine	13879	F	N	N	N	
311	312	788	C	Placebo	12109	F	N	N	Y	

276 rows × 20 columns



```
In [11]: df.duplicated() #checks for duplicated entries
```

```
Out[11]: 0      False
         1      False
         2      False
         3      False
         4      False
         ...
        413     False
        414     False
        415     False
        416     False
        417     False
        Length: 418, dtype: bool
```

```
In [12]: # Creating treatment groups/categories
```

```
In [13]: df["Drug"] = df["Drug"].astype(str).str.lower()
df["treatment_group"] = np.where(
    df["Drug"].str.contains("penicill"), "Treatment",
    np.where(df["Drug"].str.contains("placebo"), "Placebo", np.nan)
)
```

```
In [14]: #create an outcome of treatment variable
```

```
In [15]: df["Outcome"] = np.where(df["Status"]== "D",1,0)
```

```
In [16]: #Convert Age from days to years
```

```
In [17]: df["Age_years"] = df["Age"]/365
```

```
In [18]: #Make sure all values in the variables are in numeric
```

```
In [19]: for var in ["Bilirubin", "Albumin", "Age_years"]:
df[var] = pd.to_numeric(df[var], errors = "coerce")
```

```
In [20]: #Visualisations
```

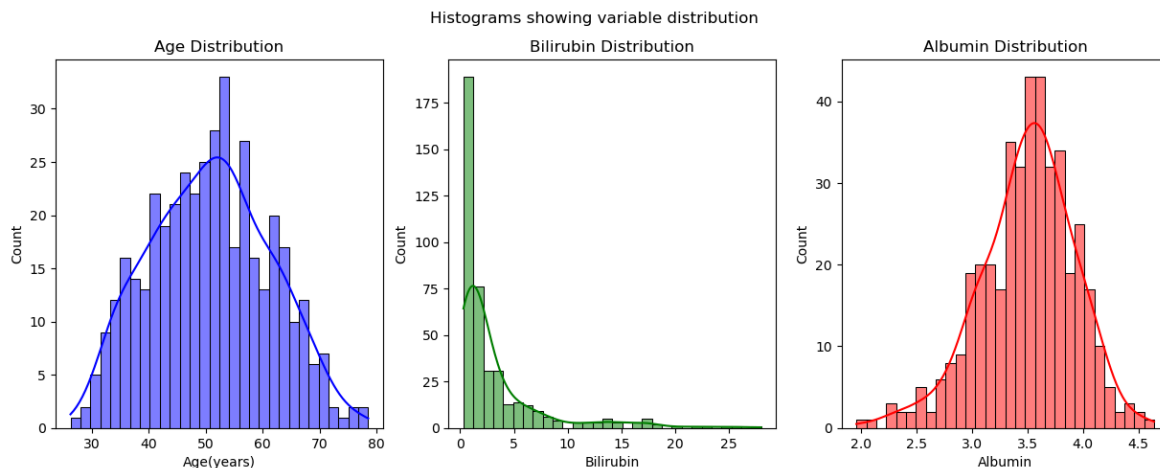
```
In [21]: #Histograms
```

```
In [22]: fig, axes = plt.subplots(1,3,figsize = (15,5))
sns.histplot(df["Age_years"], bins=30, kde=True, ax=axes[0], color="blue")
axes[0].set_title("Age Distribution")
axes[0].set_xlabel("Age(years)")

sns.histplot(df["Bilirubin"], bins=30, kde=True, ax=axes[1], color="green")
axes[1].set_title("Bilirubin Distribution")
axes[1].set_xlabel("Bilirubin")

sns.histplot(df["Albumin"], bins=30, kde=True, ax=axes[2], color="red")
axes[2].set_title("Albumin Distribution")
axes[2].set_xlabel("Albumin")

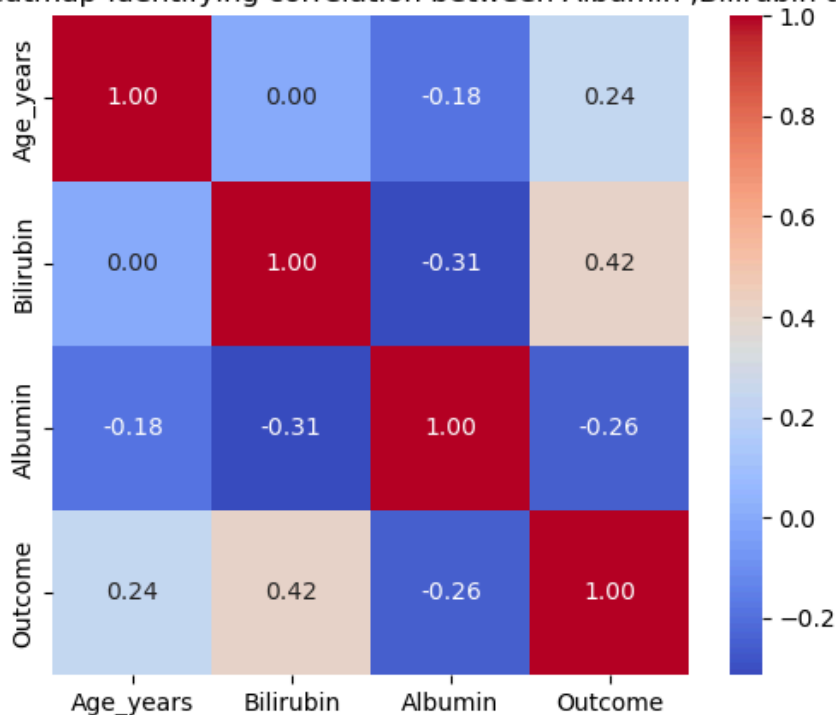
plt.suptitle("Histograms showing variable distribution")
plt.show()
```



In [23]: *#Correlation Heatmap*

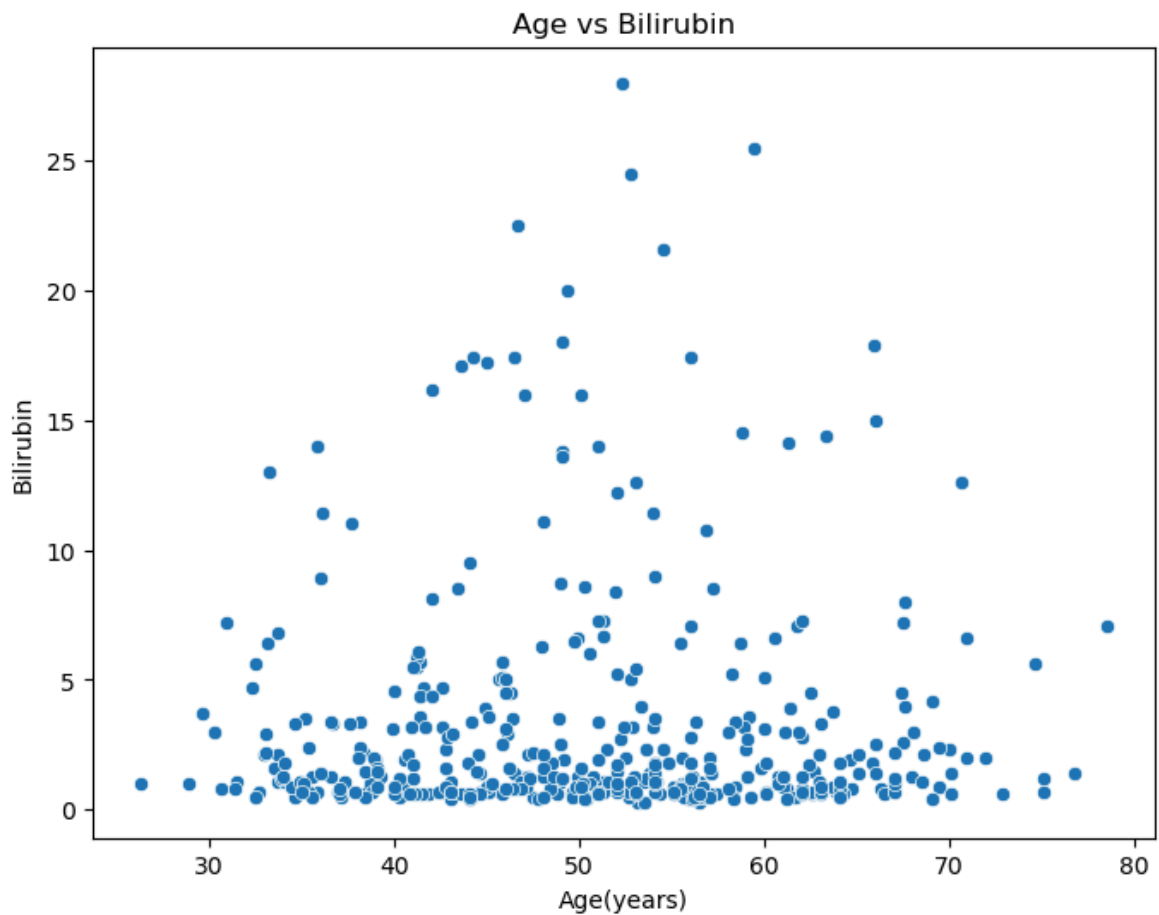
```
In [24]: plt.figure(figsize = (6,5))
corr = df[["Age_years", "Bilirubin", "Albumin","Outcome"]].corr()
sns.heatmap(corr,annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap-Identifying correlation between Albumin ,Bilirubi
plt.show()
```

Correlation Heatmap-Identifying correlation between Albumin ,Bilirubin and Age



In [25]: *#Scatter plot*

```
In [26]: plt.figure(figsize =(8,6))
sns.scatterplot(data=df, x="Age_years", y="Bilirubin")
plt.title("Age vs Bilirubin")
plt.xlabel("Age(years)")
plt.ylabel("Bilirubin")
plt.show()
```



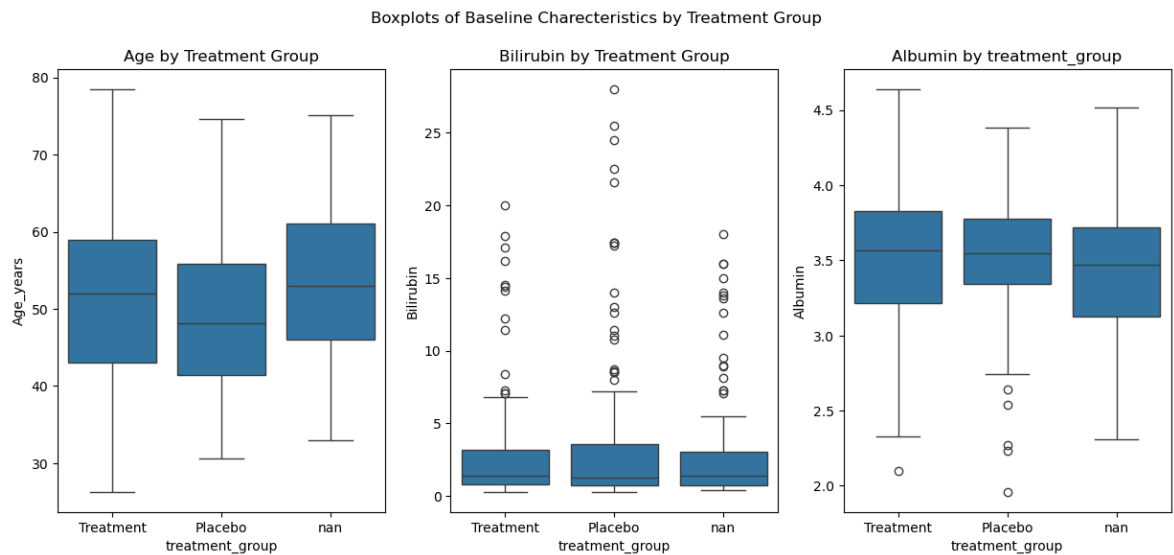
In [27]: *#Boxplot by treatment group*

```
In [28]: fig, axes = plt.subplots(1, 3, figsize=(15, 6))
sns.boxplot(data=df, x= "treatment_group", y="Age_years", ax=axes[0])
axes[0].set_title("Age by Treatment Group")

sns.boxplot(data=df, x= "treatment_group", y="Bilirubin", ax=axes[1])
axes[1].set_title("Bilirubin by Treatment Group")

sns.boxplot(data=df, x="treatment_group", y="Albumin", ax=axes[2])
axes[2].set_title("Albumin by treatment_group")

plt.suptitle("Boxplots of Baseline Charecteristics by Treatment Group")
plt.show()
```



In [29]: `#Modelling`

In [30]: `# SUPERVISED MODEL-Logistic Regression`

In [31]: `# Separate features and Target Variables`
`x= df[["Age_years"]]`
`y= df["Outcome"]`

In [32]: `print("Age_years") #Verify Segmentation`
`print("Outcome")`

Age_years
Outcome

In [33]: `#Split dataset into training and testing sets`
`x_train, x_test, y_train, y_test =model_selection.train_test_split(`
`x,y,test_size = 0.3`
`)`

In [34]: `# Inspect training and test data`
`print(x_train)`
`print(y_train)`

```

      Age_years
385  54.038356
244  45.802740
84   47.249315
0    58.805479
147  30.884932
..    ...
191  52.756164
49   53.545205
409  39.027397
199  32.254795
216  36.104110

```

```
[292 rows x 1 columns]
```

```

385    0
244    0
84     1
0      1
147    1
..
191    0
49     1
409    0
199    0
216    1

```

```
Name: Outcome, Length: 292, dtype: int32
```

```

In [35]: #Scale the Age Variable
Scaler = StandardScaler()
x_train_scaled =Scaler.fit_transform(x_train)
x_test_scaled  =Scaler.transform(x_test)

```

```

In [36]: lm = LogisticRegression()
lm.fit(x_train, y_train)
predicted = lm.predict(x_test)

```

```

In [37]: #Print the classification Report
print(metrics.classification_report(y_test , predicted))
print(metrics.confusion_matrix(y_test, predicted))

```

	precision	recall	f1-score	support
0	0.64	0.92	0.75	76
1	0.62	0.20	0.30	50
accuracy			0.63	126
macro avg	0.63	0.56	0.53	126
weighted avg	0.63	0.63	0.57	126

```

[[70  6]
 [40 10]]

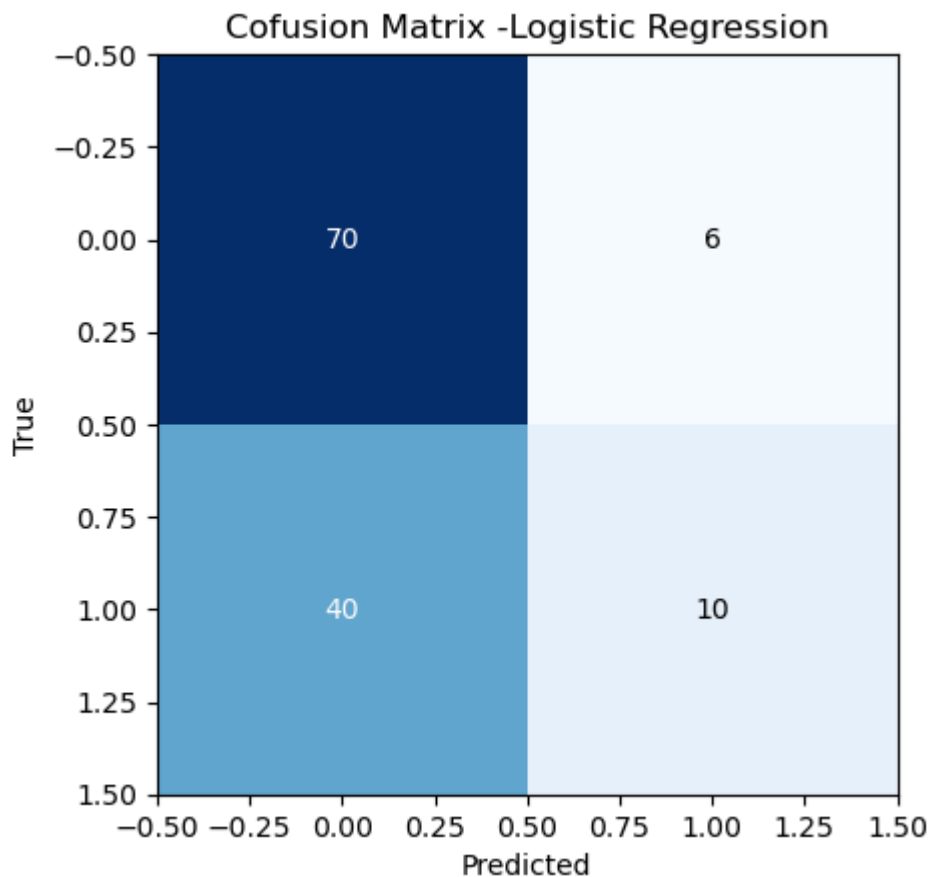
```

```

In [38]: # Plot the confusion Matrix
cm = confusion_matrix(y_test, predicted)
plt.imshow(cm, cmap= "Blues")
plt.title("Cofusion Matrix -Logistic Regression")
plt.xlabel("Predicted"); plt.ylabel("True")
for (i, j), v in np.ndenumerate(cm):
    plt.text(j, i, str(v), ha= "center" ,va="center", color= "white" if v >

```

```
cm.max()/2 else "black")
plt.show()
```



```
In [39]: # UNSUPERVISED MODEL- K-MEANS
```

```
In [40]: #Inspect the data for keys
df.keys()
```

```
Out[40]: Index(['ID', 'N_Days', 'Status', 'Drug', 'Age', 'Sex', 'Ascites',
               'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol',
               'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets',
               'Prothrombin', 'Stage', 'treatment_group', 'Outcome', 'Age_years'],
              dtype='object')
```

```
In [41]: #Scale the features
df_scaled = df[["Age_years", "Bilirubin", "Albumin"]]
scaler = StandardScaler()
scaled = scaler.fit_transform(df[["Age_years", "Bilirubin", "Albumin"]])
df_scaled = pd.DataFrame(scaled,
                          columns=["Age_years", "Bilirubin", "Albumin"])
```

```
In [42]: df_scaled.head()
```

Out[42]:

	Age_years	Bilirubin	Albumin
0	0.768941	2.562152	-2.114296
1	0.546706	-0.481759	1.513818
2	1.852567	-0.413611	-0.041088
3	0.383244	-0.322748	-2.255651
4	-1.210972	0.040704	0.076708

In [43]: *#Explore the Dataset Dimensions*
 n_samples, n_features = df.shape
 print("Number of samples:", n_samples)
 print("Number of features:", n_features)

Number of samples: 418

Number of features: 23

In [44]: X = df_scaled *#Check for best of K*

In [45]:

```

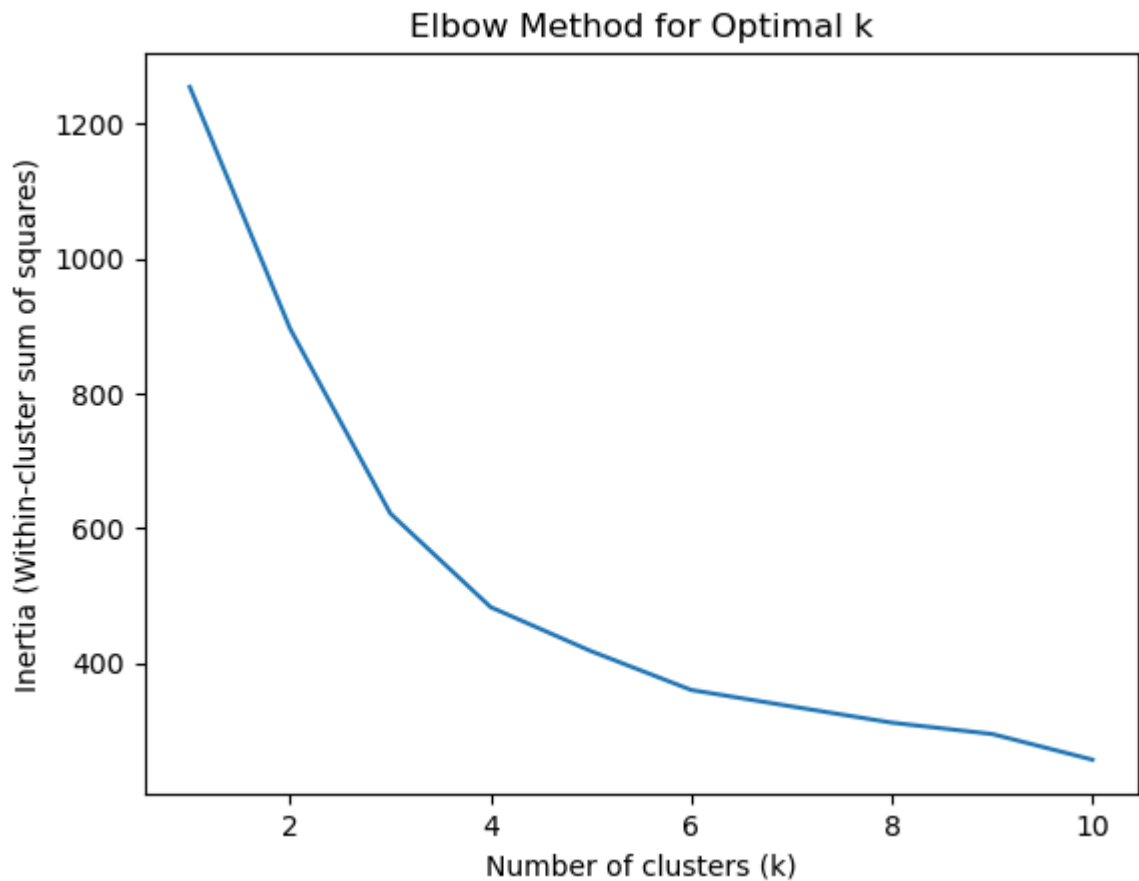
inertia = []
K = range(1,11)

for k in K:
    kmeans = KMeans(n_clusters=k,
random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

plt.plot(K, inertia)
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia (Within-cluster sum of squares)")
plt.title("Elbow Method for Optimal k")
plt.show()

```

[illegible]

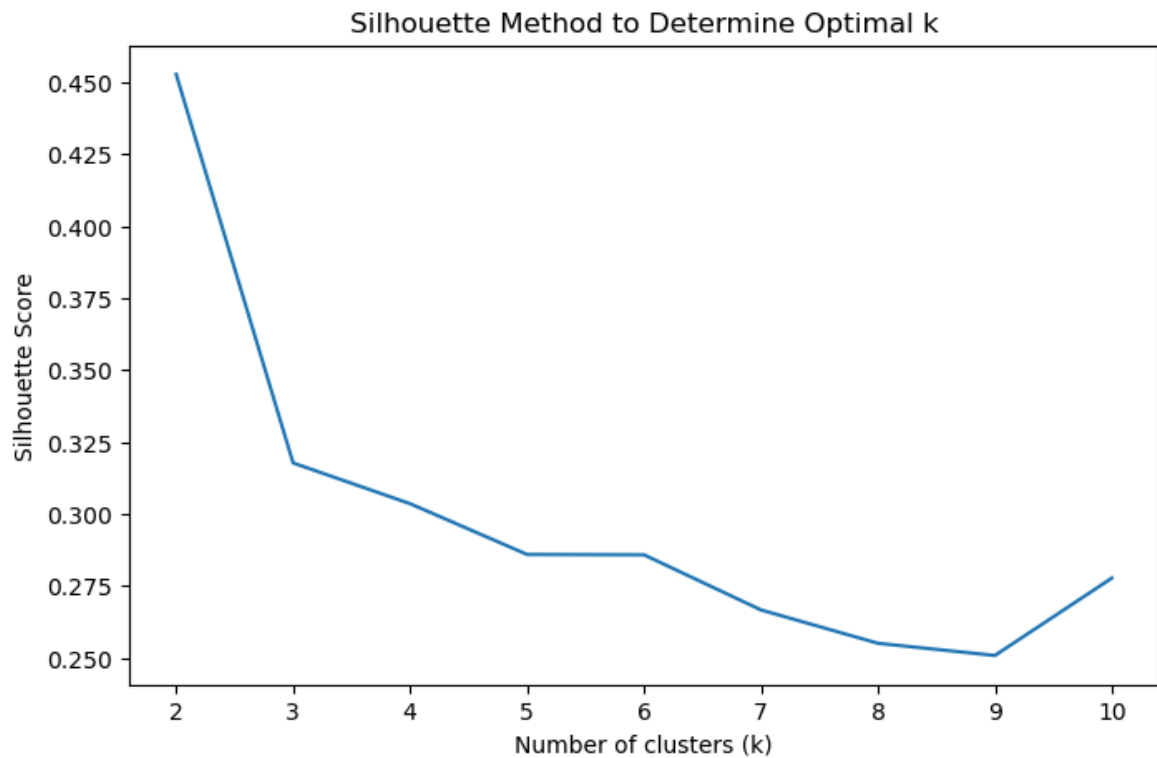


```
In [46]: X =df_scaled
silhouette_scores = []

for k in range(2,11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X)
    score = silhouette_score(X, labels)
    silhouette_scores.append(score)

plt.figure(figsize=(8,5))
plt.plot(range(2,11), silhouette_scores)
plt.xlabel("Number of clusters (k)")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Method to Determine Optimal k")
plt.show()
```

[illegible]



```
In [47]: print(kmeans.get_params())
```

```
{'algorithm': 'lloyd', 'copy_x': True, 'init': 'k-means++', 'max_iter': 300, 'n_clusters': 10, 'n_init': 'auto', 'random_state': 42, 'tol': 0.0001, 'verbose': 0}
```

```
In [48]: #Evaluate clustering performance
```

```
print("Silhouette Coefficient:", metrics.silhouette_score(df_scaled, kmeans.labels_))
print("Calinski-Harabasz Coefficient:", metrics.calinski_harabasz_score(df_scaled,
```

```
Silhouette Coefficient: 0.2777031157237317
```

```
Calinski-Harabasz Coefficient: 176.08081910099304
```

```
In [49]: #Visualize the clusters
```

```
besk_k = 3
```

```
final_kmeans = KMeans(n_clusters=besk_k, random_state=42)
```

```
final_kmeans.fit(X)
```

```
df["Cluster"] = final_kmeans.labels_
```

```
print("Cluster centers:\n", final_kmeans.cluster_centers_)
```

```
print("\nCluster counts:\n", df["Cluster"].value_counts())
```

```
Cluster centers:
```

```
[[-0.68322819 -0.2599841  0.49798214]
```

```
[ 0.04932069  2.17088504 -1.28683949]
```

```
[ 0.89711668 -0.3260552  -0.26594365]]
```

```
Cluster counts:
```

```
Cluster
```

```
0    211
```

```
2    158
```

```
1     49
```

```
Name: count, dtype: int64
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

```
In [50]: pca = PCA(n_components=2)
pca_result = pca.fit_transform(df_scaled)

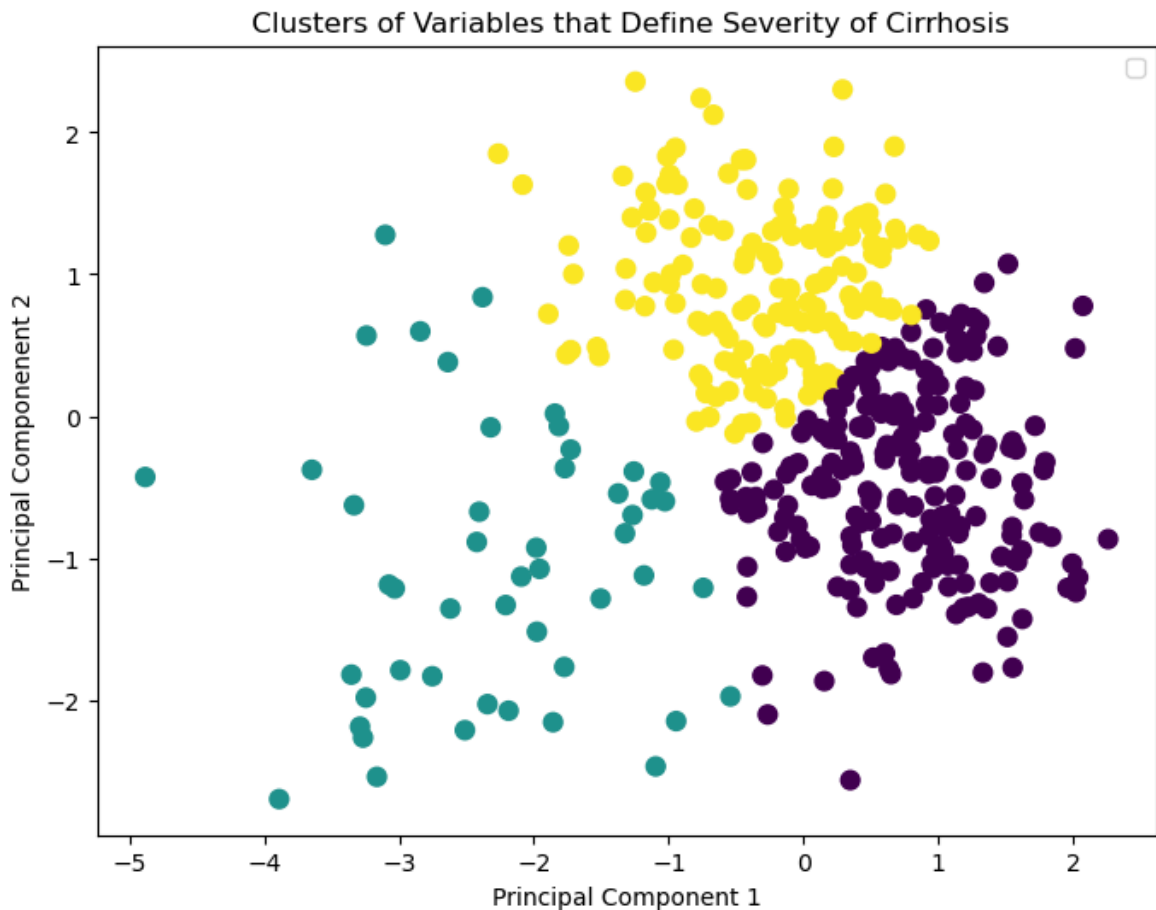
df["PCA1"] =pca_result[:, 0]
df["PCA2"] =pca_result[:, 1]
```

```
In [51]: print("Original data shape:", df_scaled.shape)
print("Transformed 2D shape:", pca_result.shape)
```

```
Original data shape: (418, 3)
Transformed 2D shape: (418, 2)
```

```
In [52]: plt.figure(figsize=(8,6))
plt.scatter(df["PCA1"], df["PCA2"], c=df["Cluster"], cmap= "viridis", s=60)
plt.title(" Clusters of Variables that Define Severity of Cirrhosis")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.show()
```

```
C:\Users\Owner\AppData\Local\Temp\ipykernel_26516\4041042170.py:6: UserWarning: No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```



```
In [53]: #Evaluate the clusters of 2D Data
print("Silhouette Coefficient:", metrics.silhouette_score(pca_result, final_kmean
print("Calinski-Harabasz Coefficient:", metrics.calinski_harabasz_score(pca_resul
print("Homogeneity score;", metrics.homogeneity_score(df["Outcome"], df["Cluster"]
print("Completeness score:", metrics.completeness_score(df["Outcome"], df["Cluste
```

Silhouette Coefficient: 0.4054910298762692

Calinski-Harabasz Coefficient: 353.0027228933699

Homogeneity score; 0.13237060235496753

Completeness score: 0.09151394227094996

```
In [54]: #visualising Cluster charecteristics
df.groupby("Cluster")[["Age_years", "Bilirubin", "Albumin"]].mean()
```

```
Out[54]:
```

	Age_years	Bilirubin	Albumin
Cluster			
0	43.642135	2.076303	3.708815
1	51.291306	12.777551	2.951224
2	60.143870	1.785443	3.384557

```
In [ ]:
```