Context

Cirrhosis is a late stage of scarring (fibrosis) of the liver caused by many forms of liver diseases and conditions, such as hepatitis and chronic alcoholism. The following data contains the information collected from the Mayo Clinic trial in primary biliary cirrhosis (PBC) of the liver conducted between 1974 and 1984. A description of the clinical background for the trial and the covariates recorded here is in Chapter 0, especially Section 0.2 of Fleming and Harrington, Counting Processes and Survival Analysis, Wiley, 1991. A more extended discussion can be found in Dickson, et al., Hepatology 10:1-7 (1989) and in Markus, et al., N Eng J of Med 320:1709-13 (1989).

A total of 424 PBC patients, referred to Mayo Clinic during that ten-year interval, met eligibility criteria for the randomized placebo-controlled trial of the drug D-penicillamine. The first 312 cases in the dataset participated in the randomized trial and contain largely complete data. The additional 112 cases did not participate in the clinical trial but consented to have basic measurements recorded and to be followed for survival. Six of those cases were lost to follow-up shortly after diagnosis, so the data here are on an additional 106 cases as well as the 312 randomized participants.

Attribute Information:

ID: unique identifier

N\_Days: number of days between registration and the earlier of death, transplantation, or study analysis time in July 1986

Status: status of the patient C (censored), CL (censored due to liver tx), or D (death)

D (Death): Indicates that the patient has deceased.

C (Censored): Refers to patients who are still alive at the time of analysis but have been censored (meaning their survival time is not fully observed).

CL (Censored due to liver transplantation): Denotes patients who are alive at the time of analysis due to having undergone liver transplantation

Drug: type of drug D-penicillamine or placebo

Age: age in [days]

Sex: M (male) or F (female)

Ascites: presence of ascites N (No) or Y (Yes)

Hepatomegaly: presence of hepatomegaly N (No) or Y (Yes)

Spiders: presence of spiders N (No) or Y (Yes)

Edema: presence of edema N (no edema and no diuretic therapy for edema), S (edema present without diuretics, or edema resolved by diuretics), or Y (edema despite diuretic therapy)

Bilirubin: serum bilirubin in [mg/dl]

Cholesterol: serum cholesterol in [mg/dl]

Albumin: albumin in [gm/dl]

Copper: urine copper in [ug/day]

Alk\_Phos: alkaline phosphatase in [U/liter]

SGOT: SGOT in [U/ml]

Triglycerides: triglicerides in [mg/dl]

Platelets: platelets per cubic [ml/1000]

Prothrombin: prothrombin time in seconds [s]

Stage: histologic stage of disease (1, 2, 3, or 4)

**Importing Necessary Libraries**

# Ignore warnings

import warnings

warnings.filterwarnings('ignore')

​

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import pickle

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import IsolationForest

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.model\_selection import cross\_val\_score

​

​

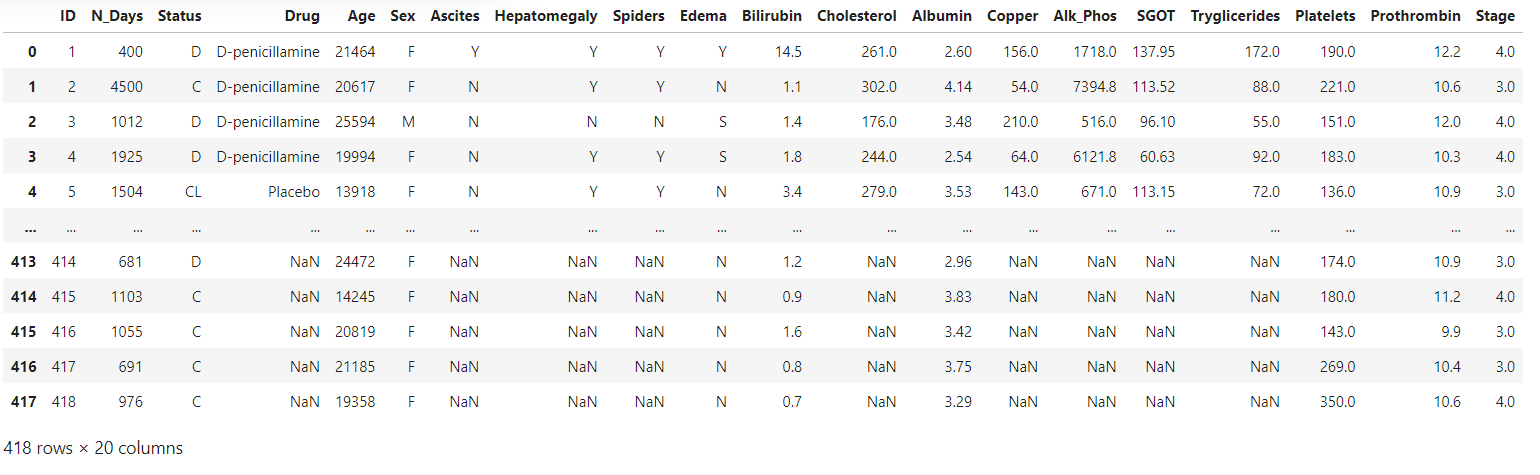
%matplotlib inline

**Loading the Dataset**

df=pd.read\_csv('cirrhosis.csv')

**Initial Data Analysis**

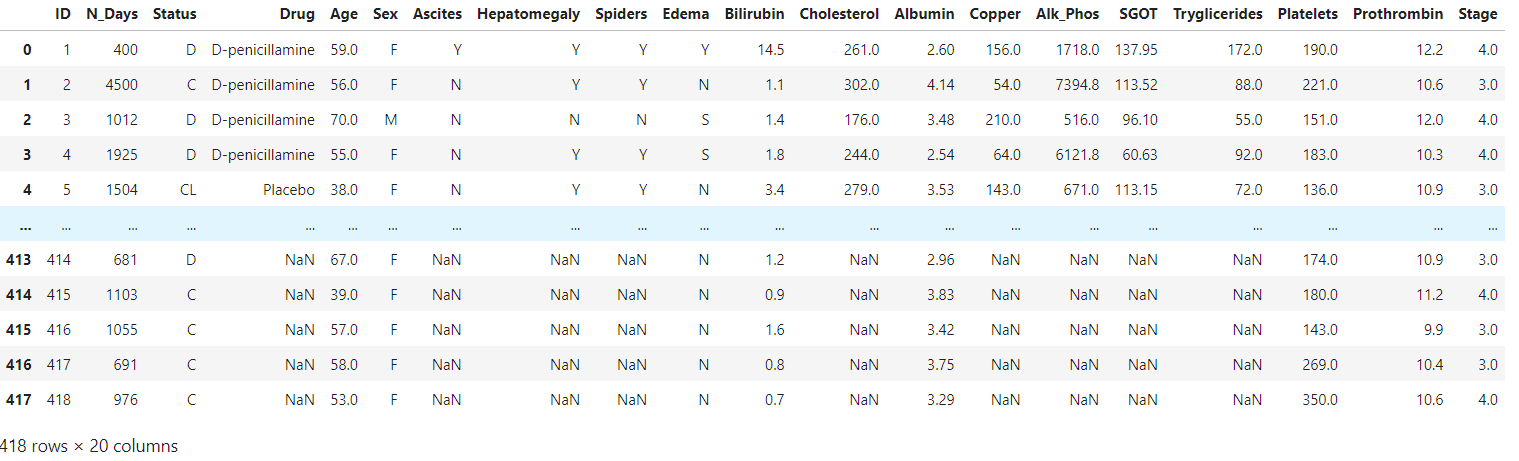
df



# Convert age from days to years

df['Age'] = (df['Age'] / 365.25).round()

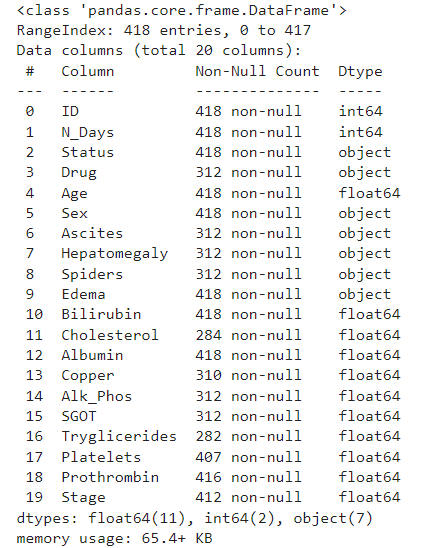
df



df.shape

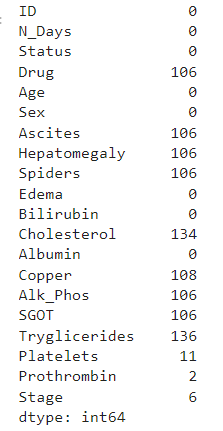
(418, 20)

df.info()



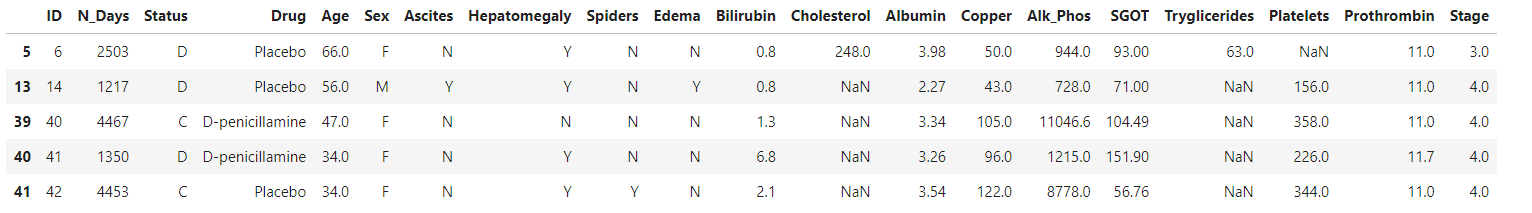
**Data Cleaning**

df.isnull().sum()



# Extracting rows with missing values

df[df['Drug'].isnull() | df['Ascites'].isnull() | df['Hepatomegaly'].isnull() | df['Spiders'].isnull() | df['Cholesterol'].isnull() | df['Copper'].isnull() | df['Alk\_Phos'].isnull() | df['SGOT'].isnull() | df['Tryglicerides'].isnull() | df['Platelets'].isnull() | df['Prothrombin'].isnull() | df['Stage'].isnull()].head()

# Removing rows with missing values

df = df.dropna(subset=['Drug', 'Ascites', 'Hepatomegaly', 'Spiders', 'Cholesterol', 'Copper', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin', 'Stage'])

df.isnull().sum()



# Finding duplicate rows (keeping all instances)

duplicate\_rows = df[df.duplicated(keep=False)]

# Displaying the records

duplicate\_rows



# Displaying the number of duplicate rows

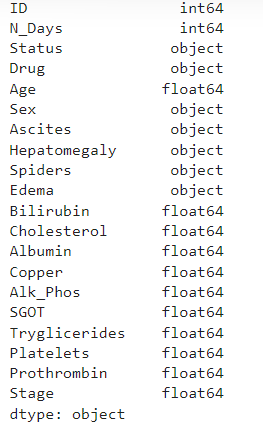
print(f"The dataset contains {df.duplicated().sum()} duplicate rows that need to be removed.")

The dataset contains 0 duplicate rows that need to be removed.

df

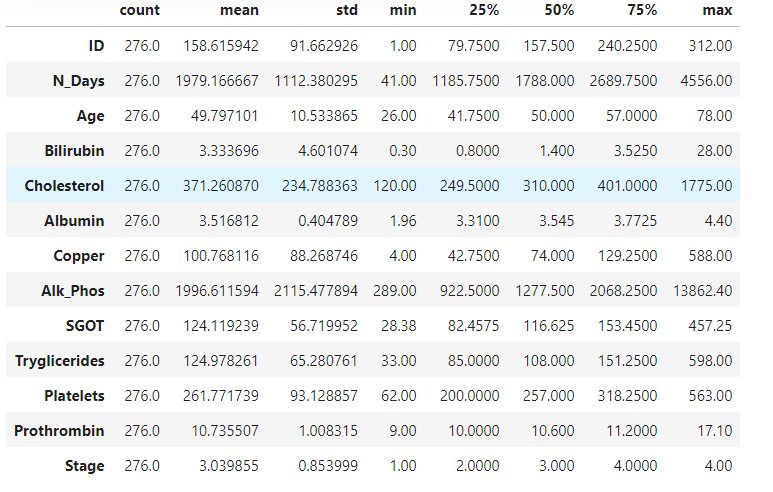


df.dtypes



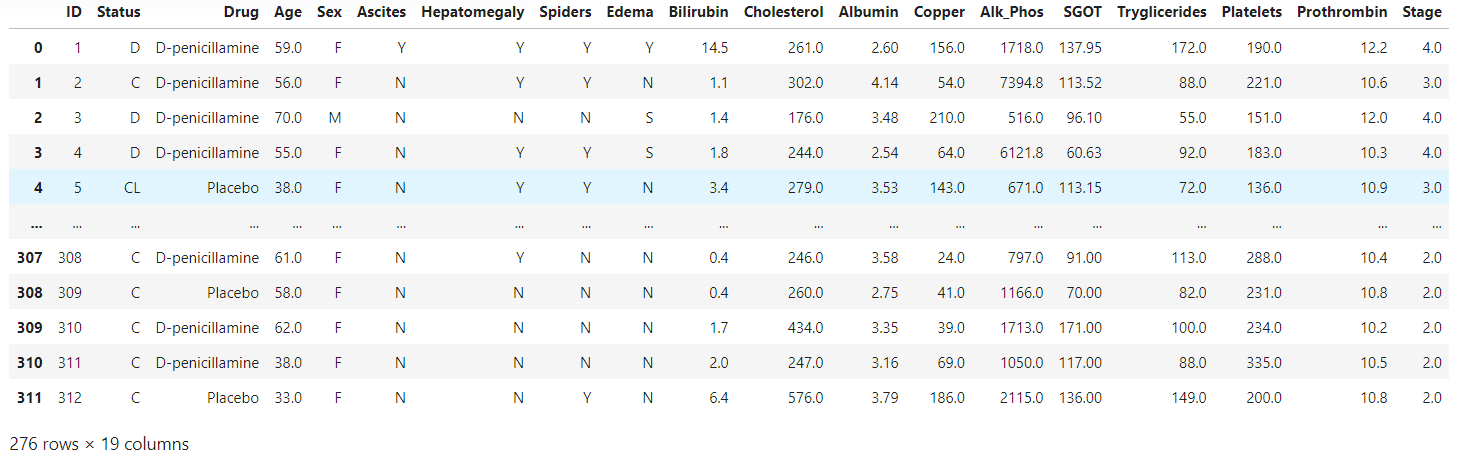
# Summary statistics for numerical variables

df.describe().T

#drop N\_Days column

df.drop('N\_Days', axis=1, inplace=True)

df



**Exploratory Data Analysis (EDA)**

**Descriptive Statistics**

# Numerical Variable Analysis

numerical\_vars = ['Age', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk\_Phos',

'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']

​

# Summary statistics

print("Summary Statistics:")

df[numerical\_vars].describe()



**Data Visualization**

# Histograms for numerical variables

plt.figure(figsize=(14, 10))

for i, var in enumerate(numerical\_vars, 1):

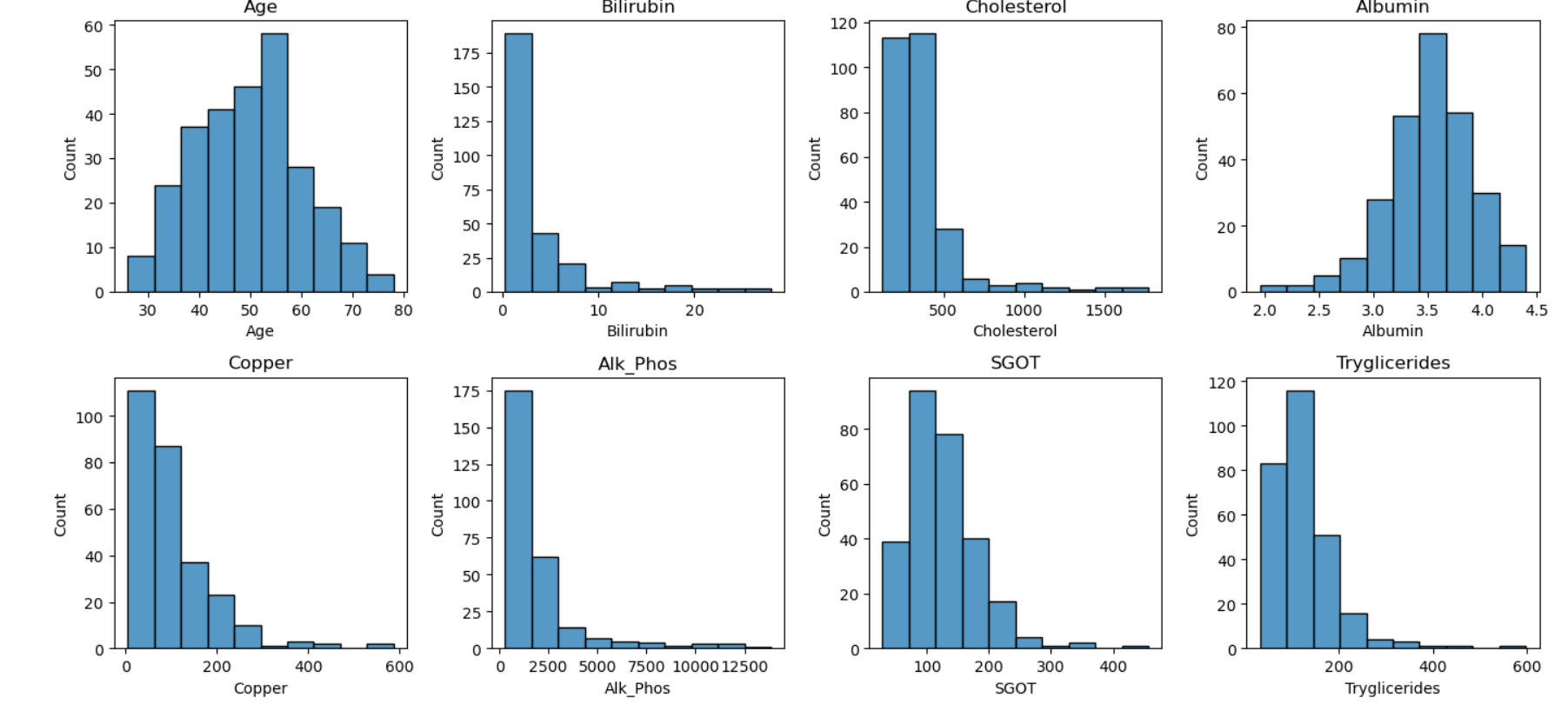
plt.subplot(3, 4, i)

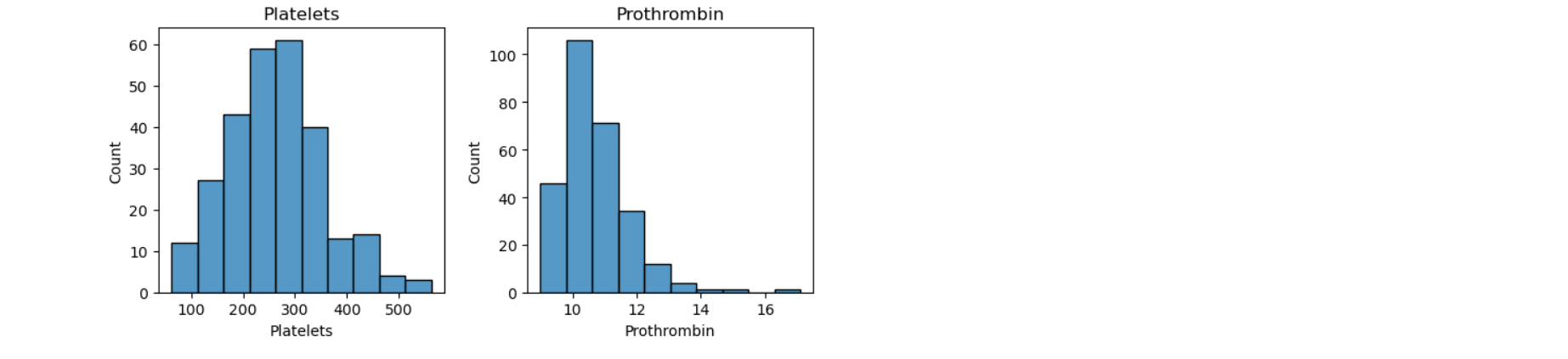
sns.histplot(df[var], bins=10)

plt.title(var)

plt.tight\_layout()

plt.show()





Bin width = (max value - min value) / number of bins

# Calculate correlation matrix

corr\_matrix = df[numerical\_vars].corr()

​

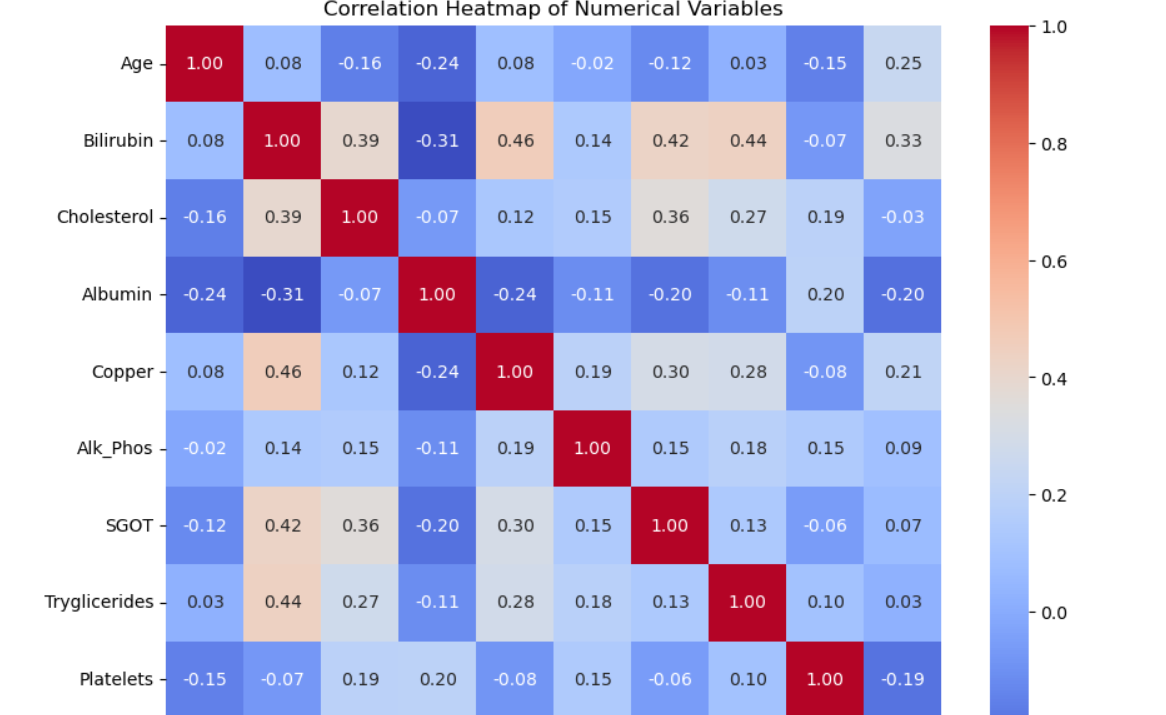
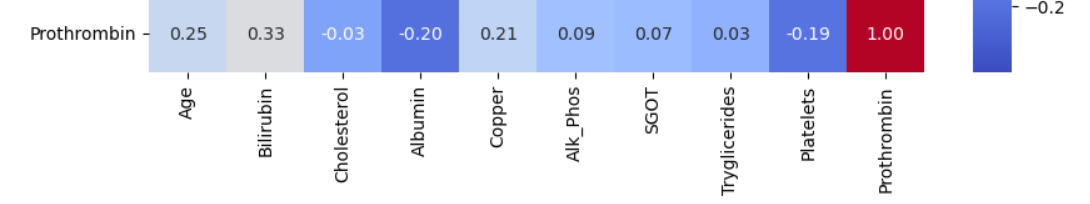
# Plot correlation heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot\_kws={"size": 10})

plt.title('Correlation Heatmap of Numerical Variables')

plt.show()

Color Coding: Darker colors represent stronger correlations.

Lighter colors indicate weaker correlations.

Positive correlations (when one variable increases, the other tends to increase) are usually shown in warm colors like red or orange.

Negative correlations (when one variable increases, the other tends to decrease) are usually shown in cool colors like blue or green.

# Group by 'Stage' and compute mean for all blood tests

means\_by\_stage = df.groupby('Stage').agg({

'Bilirubin': 'mean',

'Cholesterol': 'mean',

'Albumin': 'mean',

'Copper': 'mean',

'Alk\_Phos': 'mean',

'SGOT': 'mean',

'Tryglicerides': 'mean',

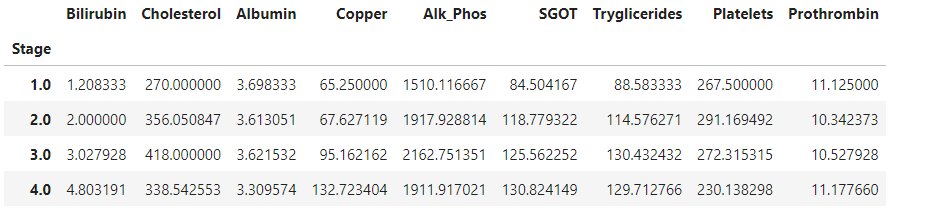
'Platelets': 'mean',

'Prothrombin': 'mean'

})

​

means\_by\_stage



# List of categorical variables

categorical\_vars = ['Status', 'Drug', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Stage']

​

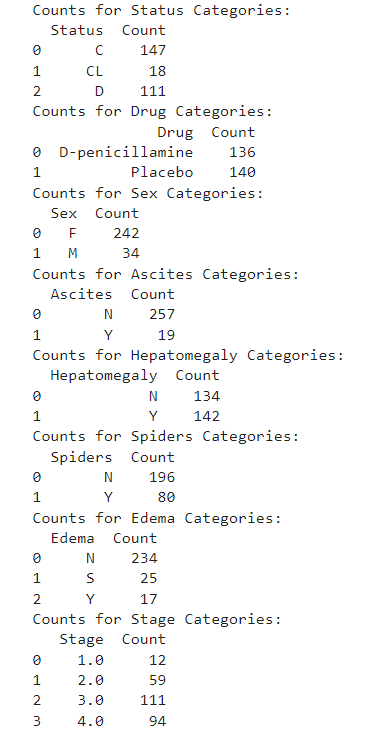
# Group and count each categories of Status, Drugs, Sex, Symptoms, Stage

for var in categorical\_vars:

counts = df.groupby(var).size().reset\_index(name='Count')

print(f"Counts for {var} Categories:")

print(counts)



# Set up the plot grid

fig, axes = plt.subplots(len(categorical\_vars), 1, figsize=(10, len(categorical\_vars)\*5))

​

# Iterate through each categorical variable

for i, var in enumerate(categorical\_vars):

# Group and count each subcategory

counts = df.groupby(var).size().reset\_index(name='Count')

# Plot the bar graph

sns.barplot(x='Count', y=var, data=counts, ax=axes[i])

# Set title and labels

axes[i].set\_title(f"Counts for {var}")

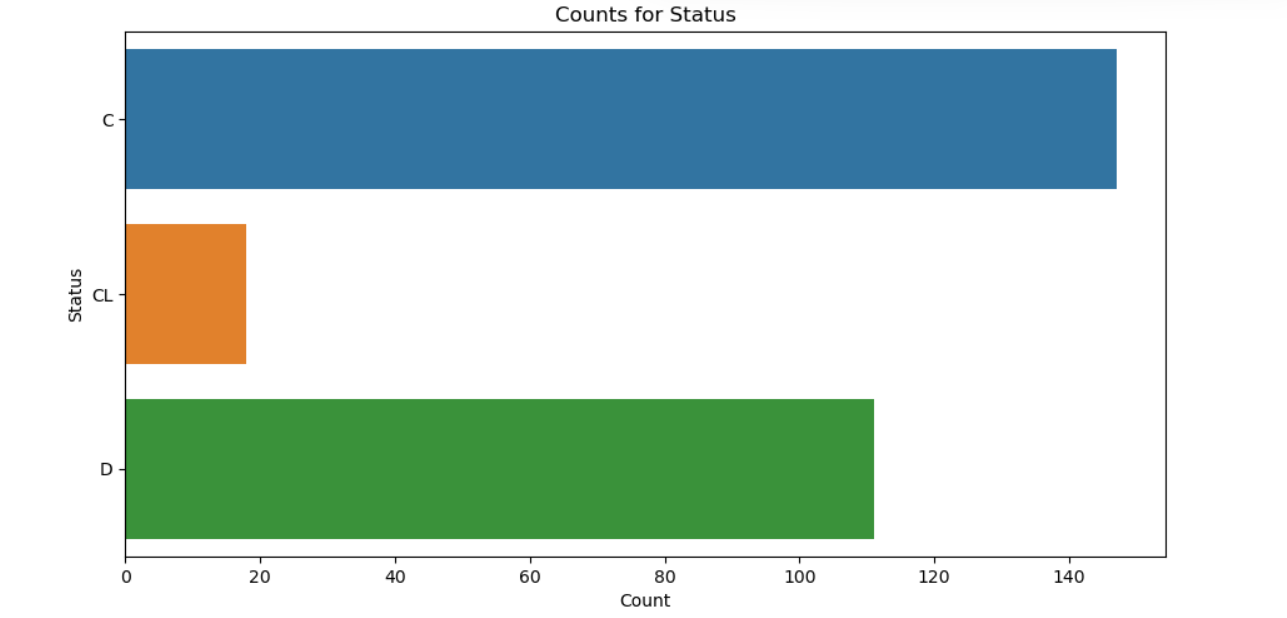
axes[i].set\_ylabel(var)

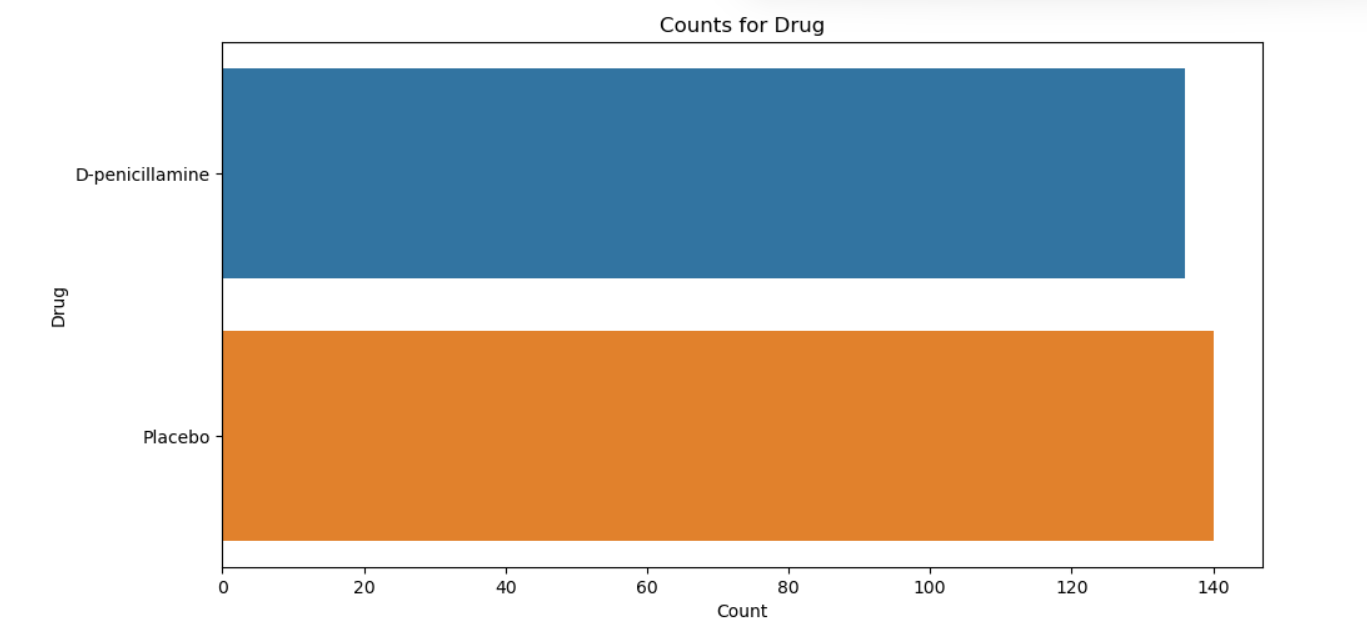
axes[i].set\_xlabel("Count")

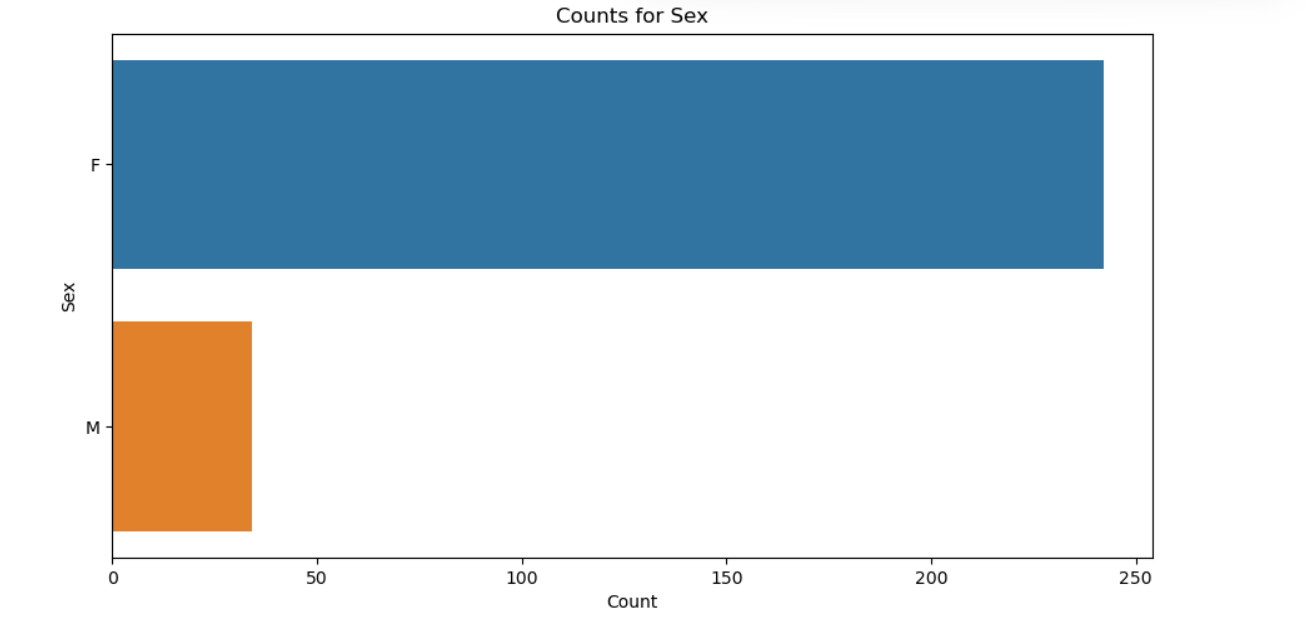
​

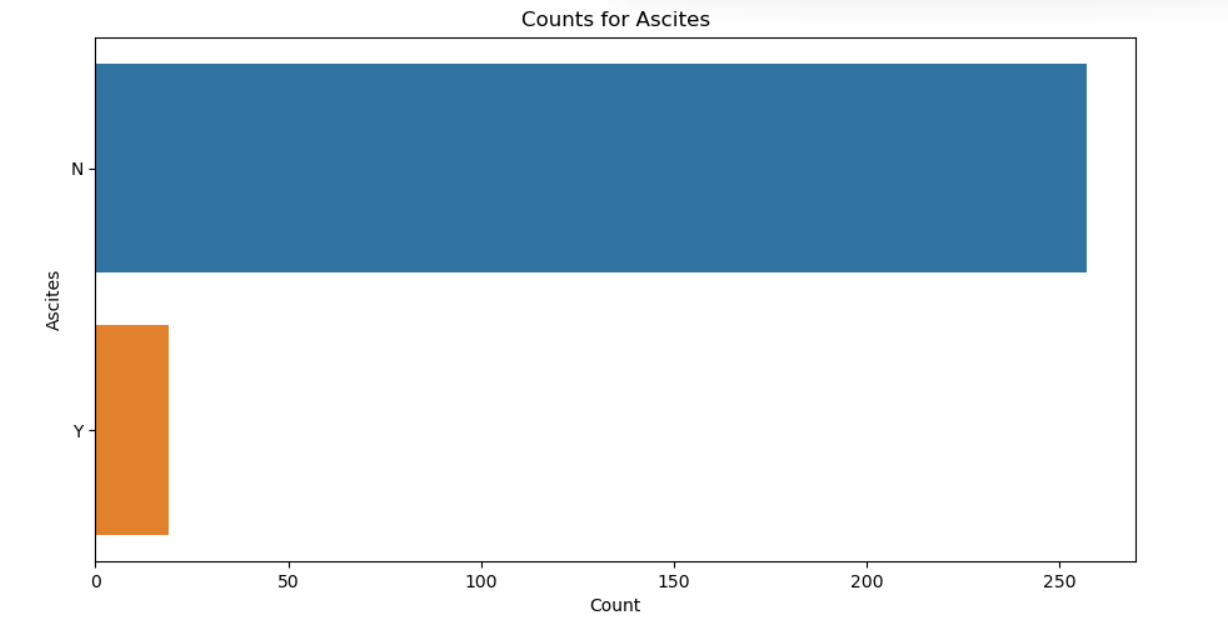
plt.tight\_layout()

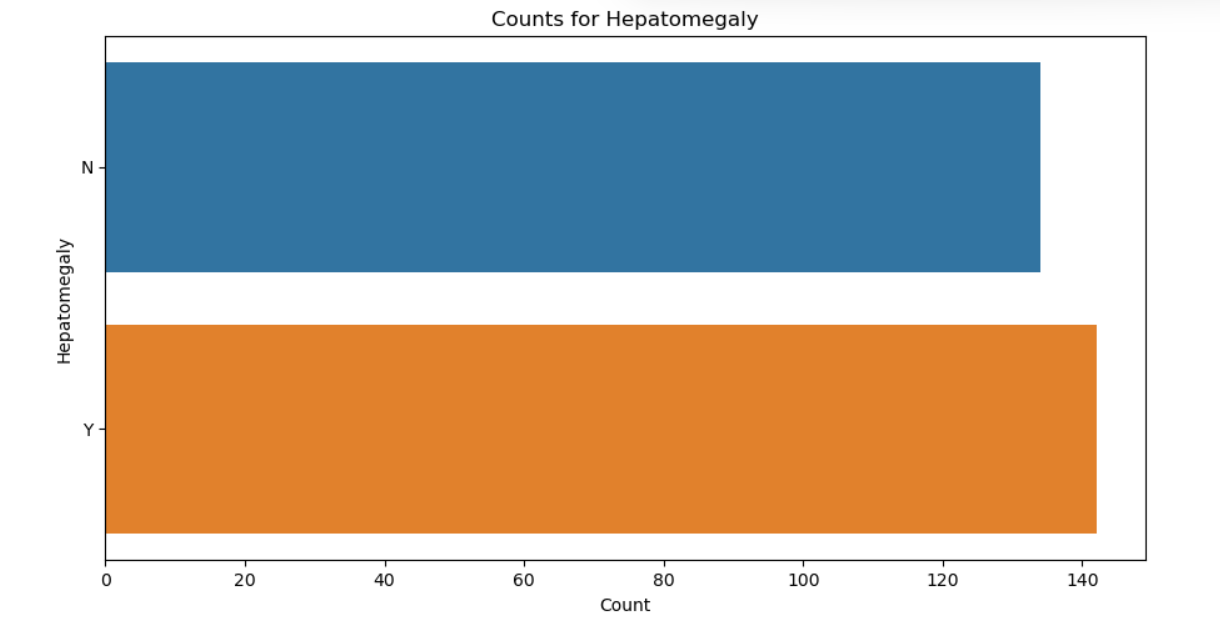
plt.show()

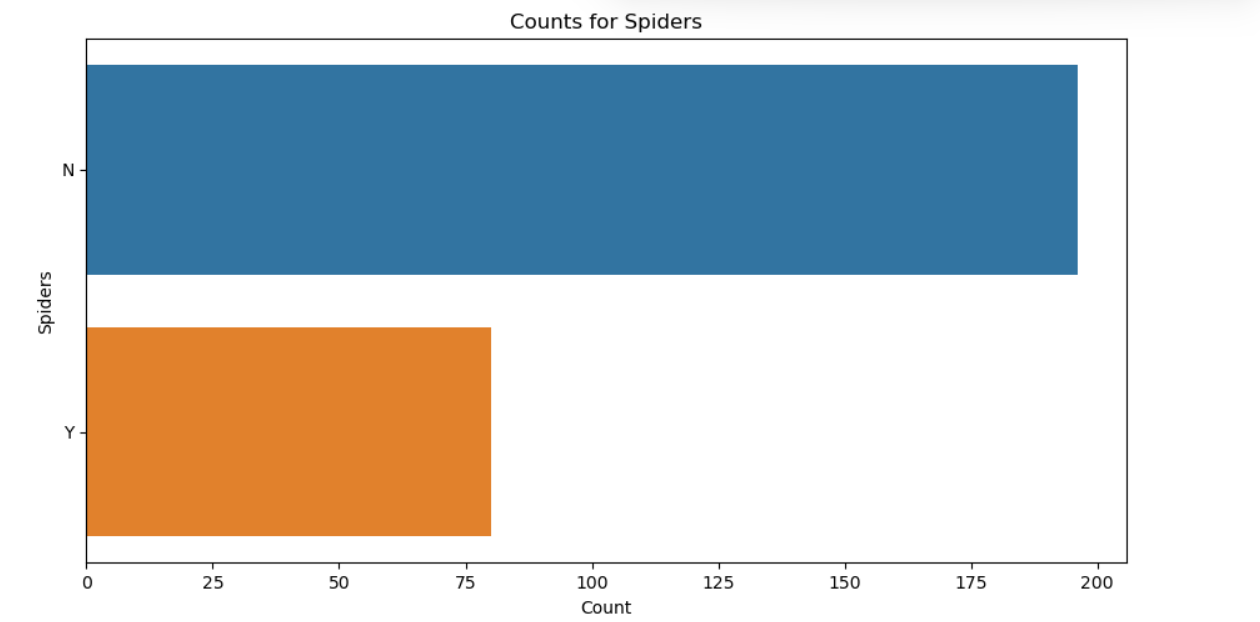


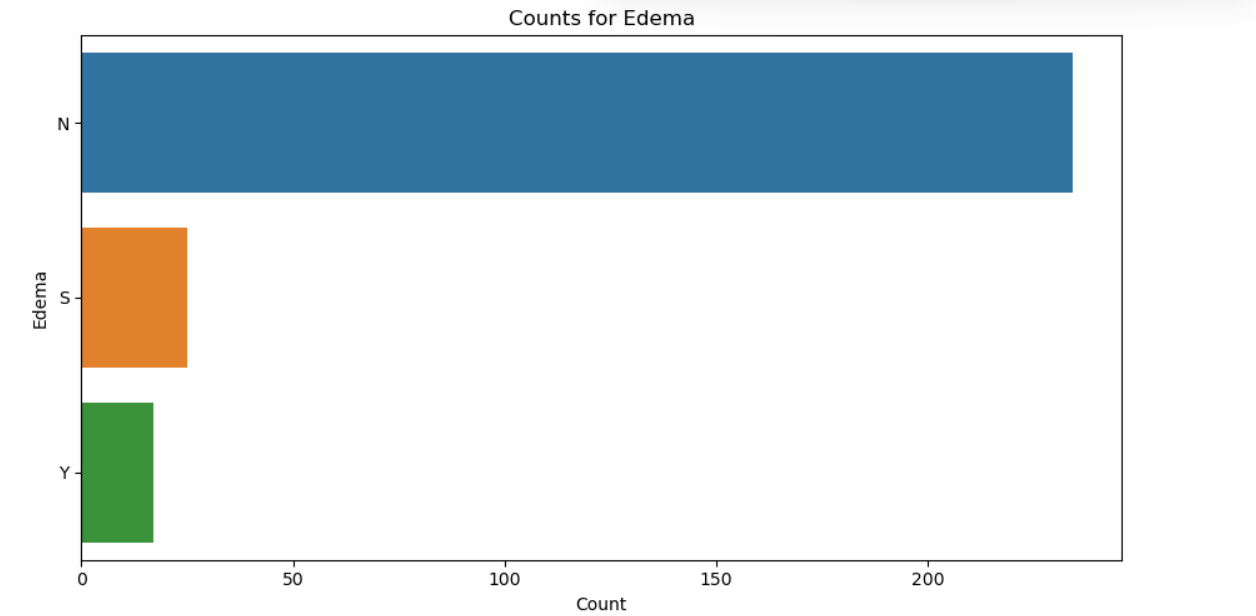


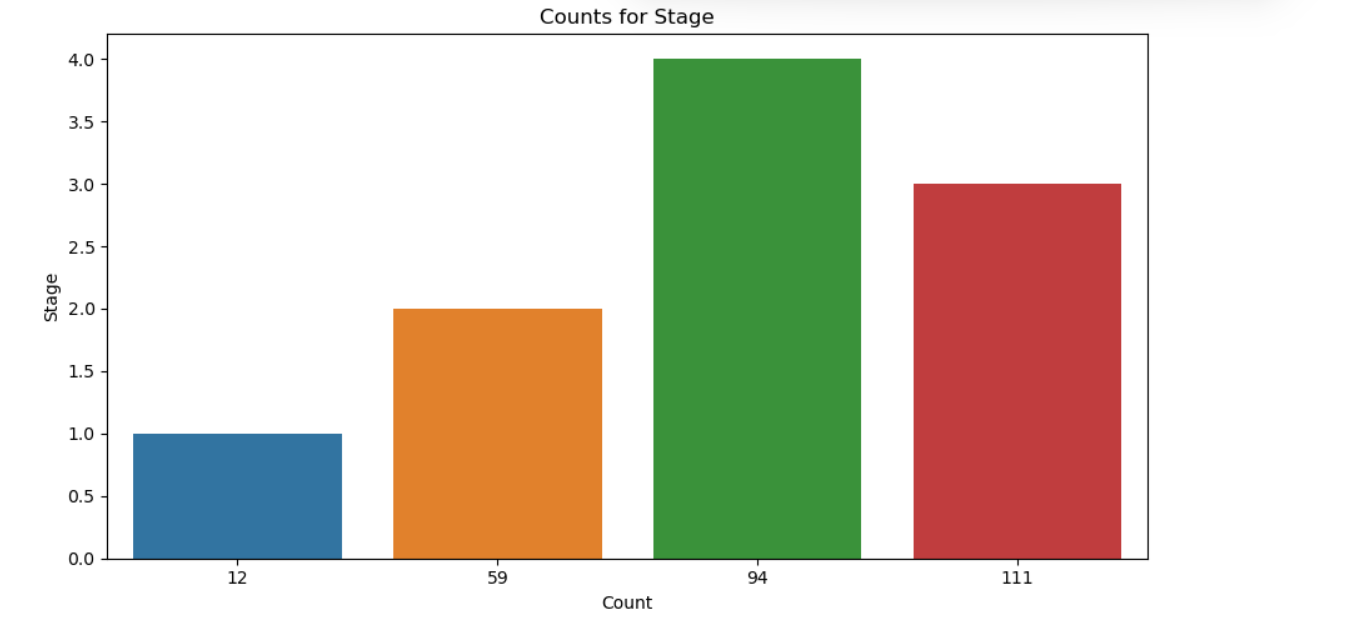












# List of categorical variables

categorical\_vars = ['Status', 'Drug', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema']

​

# Group and count each categories of Status, Drugs, Sex, Symptoms for each Stage

for var in categorical\_vars:

print(f"Counts for {var} Categories by Stage:")

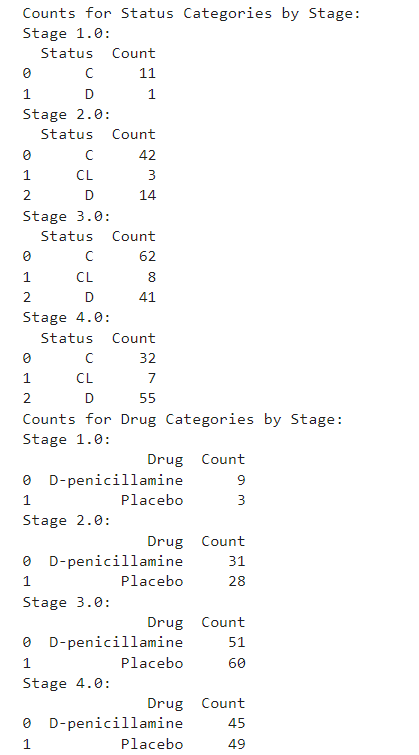
for stage in sorted(df['Stage'].unique()):

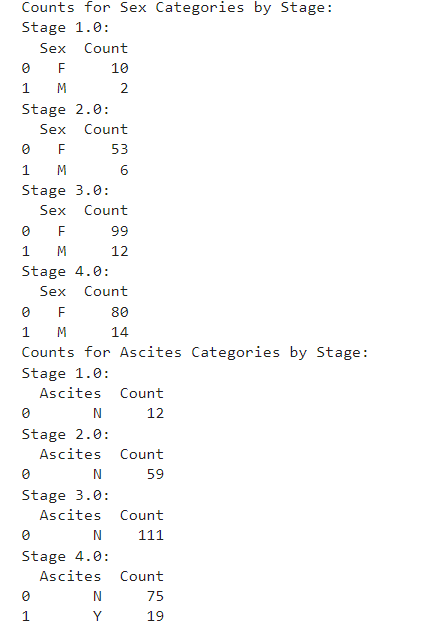
stage\_data = df[df['Stage'] == stage]

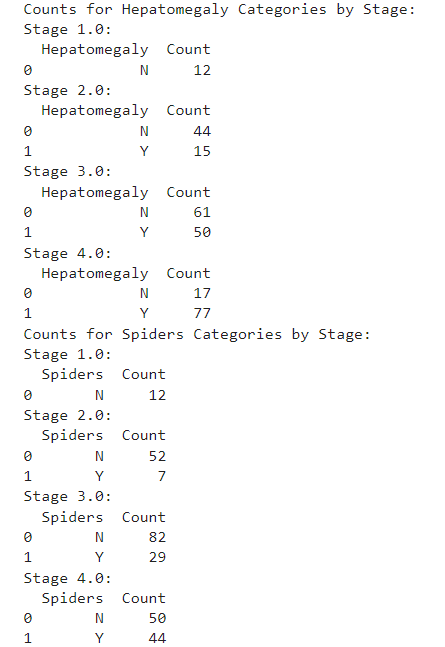
counts = stage\_data.groupby(var).size().reset\_index(name='Count')

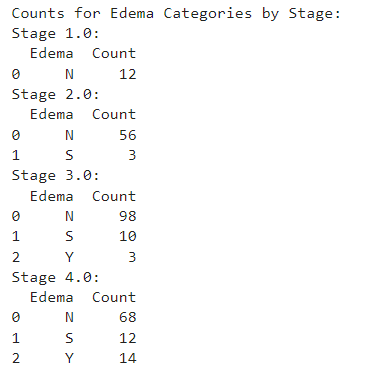
print(f"Stage {stage}:")

print(counts)









# Set up the plot grid

fig, axes = plt.subplots(len(categorical\_vars), len(df['Stage'].unique()), figsize=(15, 10), sharey='row')

​

# Iterate through each categorical variable

for i, var in enumerate(categorical\_vars):

# Iterate through each stage

for j, stage in enumerate(sorted(df['Stage'].unique())):

# Group and count each subcategory for the current stage

stage\_data = df[df['Stage'] == stage]

counts = stage\_data.groupby(var).size().reset\_index(name='Count')

# Plot the bar graph

sns.barplot(x='Count', y=var, data=counts, ax=axes[i, j])

# Set title and labels

axes[i, j].set\_title(f"Stage {stage}")

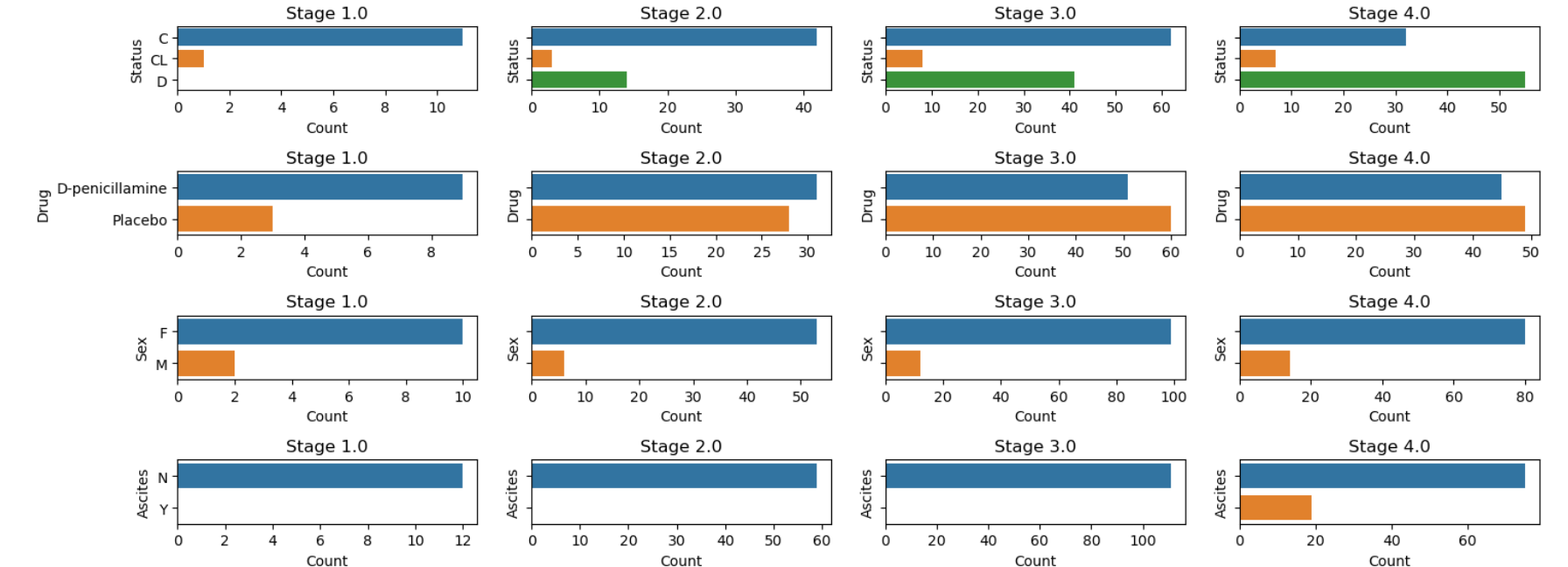
axes[i, j].set\_ylabel(var)

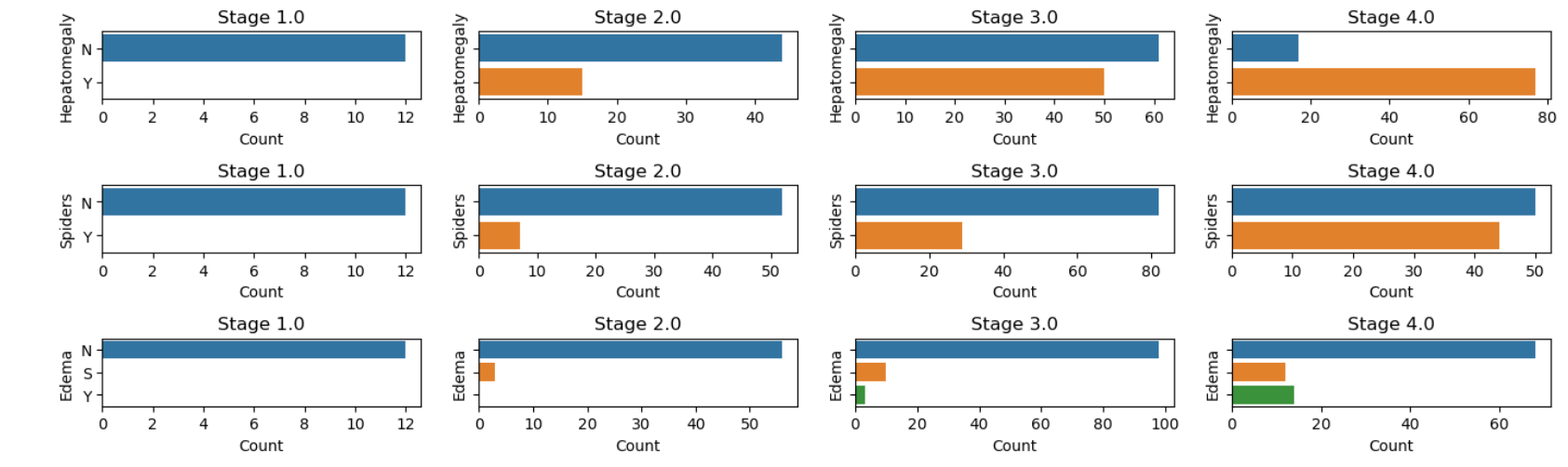
axes[i, j].set\_xlabel("Count")

​

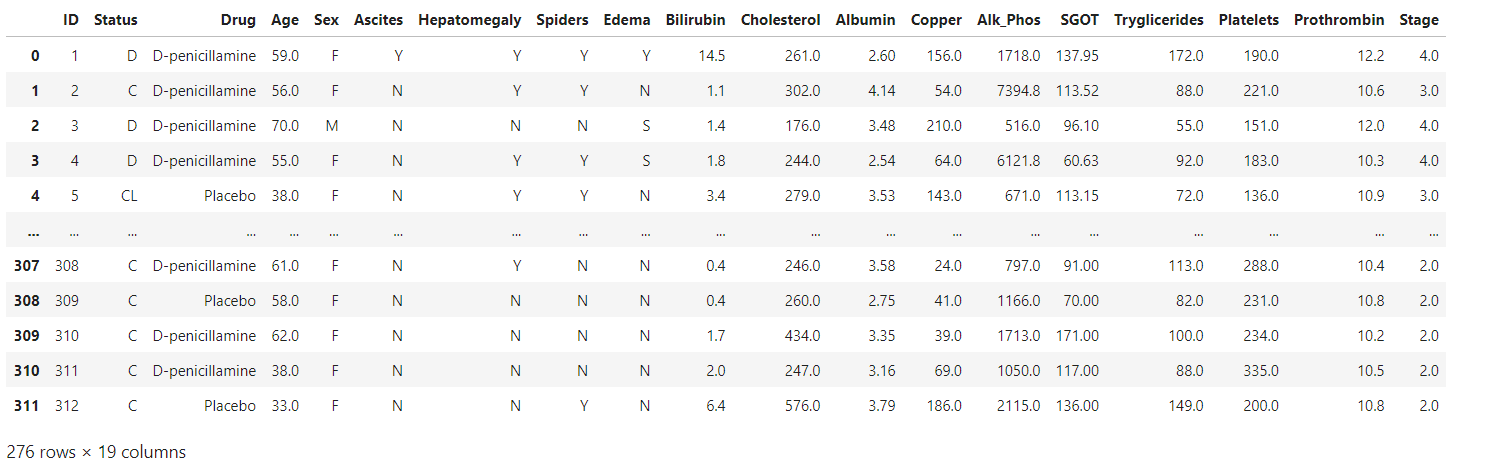
plt.tight\_layout()

plt.show()





df



**Feature Engineering**

Feature Engineering helps to make a Patients centric dataset.

data = pd.DataFrame()

​

# Status encoding

# Encode 'Status' into a single new column

status\_mapping = {'D': 1, 'C': 2, 'CL': 3}

data['Status\_Encoded'] = df['Status'].map(status\_mapping)

​

# Copy other columns from the original dataframe

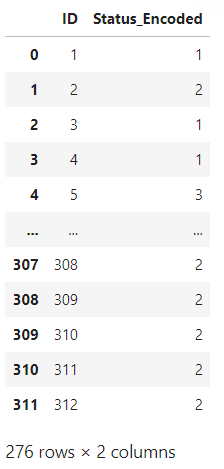
data['ID'] = df['ID']

​

# Rearrange the columns

data = data[['ID', 'Status\_Encoded']]

data



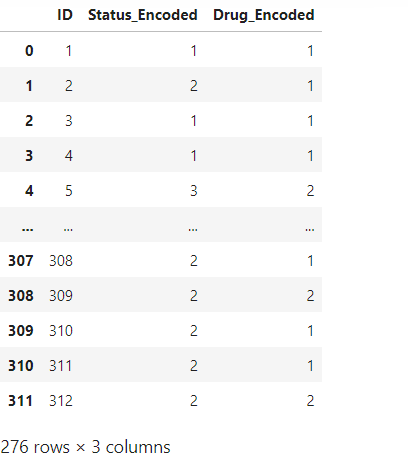
# Drugs encoding

# Encode 'Drug' into a single new column

drug\_mapping = {'D-penicillamine': 1, 'Placebo': 2}

data['Drug\_Encoded'] = df['Drug'].map(drug\_mapping)

data



# Define age groups and their corresponding numerical representations

age\_groups = {

'0-30': 1,

'31-40': 2,

'41-50': 3,

'51-60': 4,

'61-70': 5,

'71+': 6

}

​

# Encode 'Age' into a single new column based on age groups

def encode\_age(age):

if age <= 30:

return age\_groups['0-30']

elif age <= 40:

return age\_groups['31-40']

elif age <= 50:

return age\_groups['41-50']

elif age <= 60:

return age\_groups['51-60']

elif age <= 70:

return age\_groups['61-70']

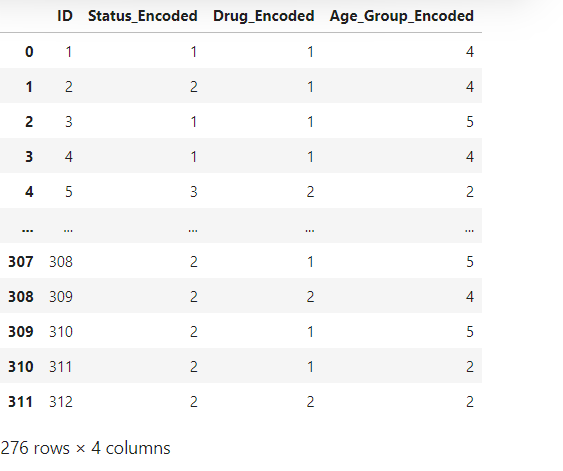
else:

return age\_groups['71+']

​

data['Age\_Group\_Encoded'] = df['Age'].apply(encode\_age)

data



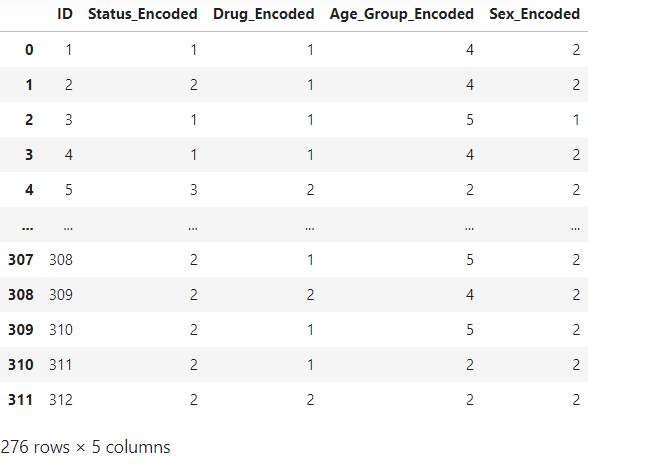
# Sex encoding

# Encode 'Sex' into a single new column

sex\_mapping = {'M': 1, 'F': 2}

data['Sex\_Encoded'] = df['Sex'].map(sex\_mapping)

data



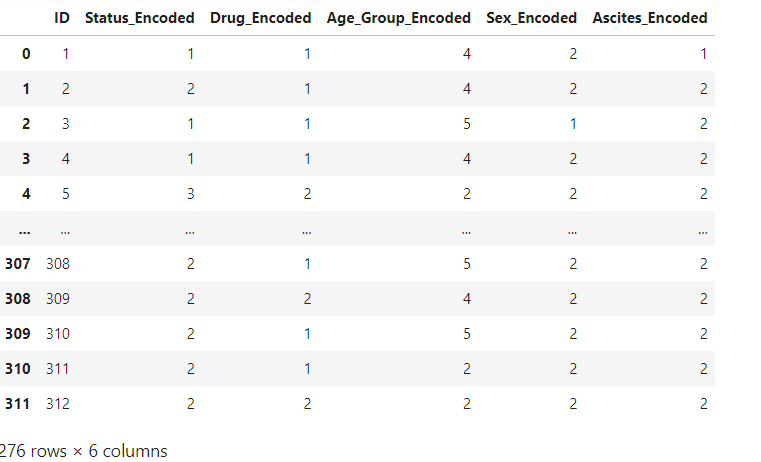
# Ascites encoding

# Encode 'Ascites' into a single new column

ascites\_mapping = {'Y': 1, 'N': 2}

data['Ascites\_Encoded'] = df['Ascites'].map(ascites\_mapping)

data



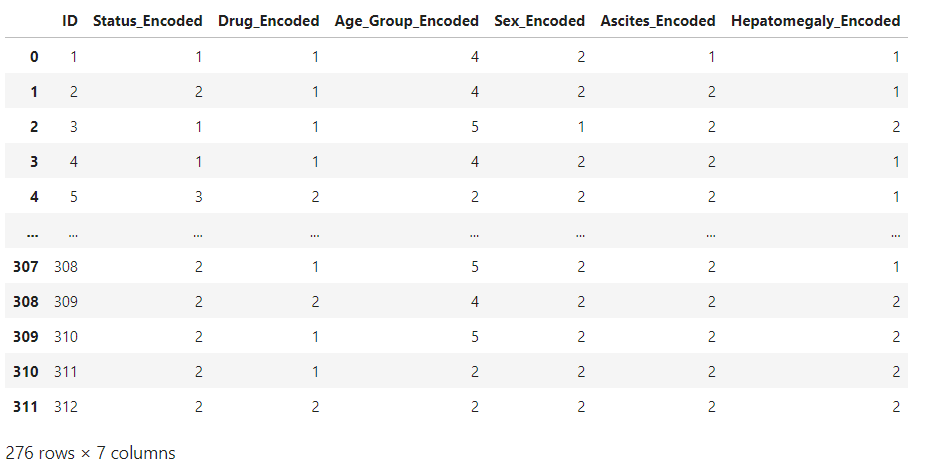
# Hepatomegaly encoding

# Encode 'Hepatomegaly' into a single new column

Hepatomegaly\_mapping = {'Y': 1, 'N': 2}

data['Hepatomegaly\_Encoded'] = df['Hepatomegaly'].map(Hepatomegaly\_mapping)

data



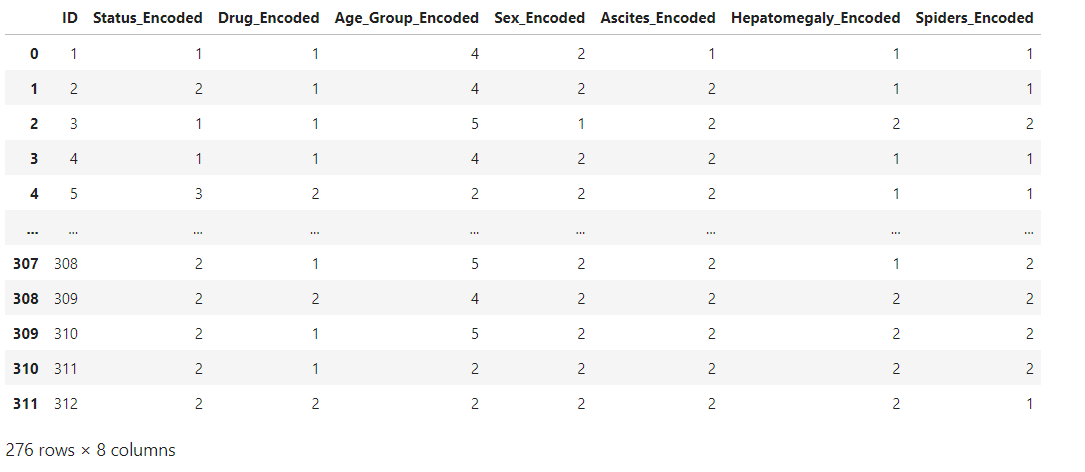
# Spiders encoding

# Encode 'Spiders' into a single new column

Spiders\_mapping = {'Y': 1, 'N': 2}

data['Spiders\_Encoded'] = df['Spiders'].map(Spiders\_mapping)

data



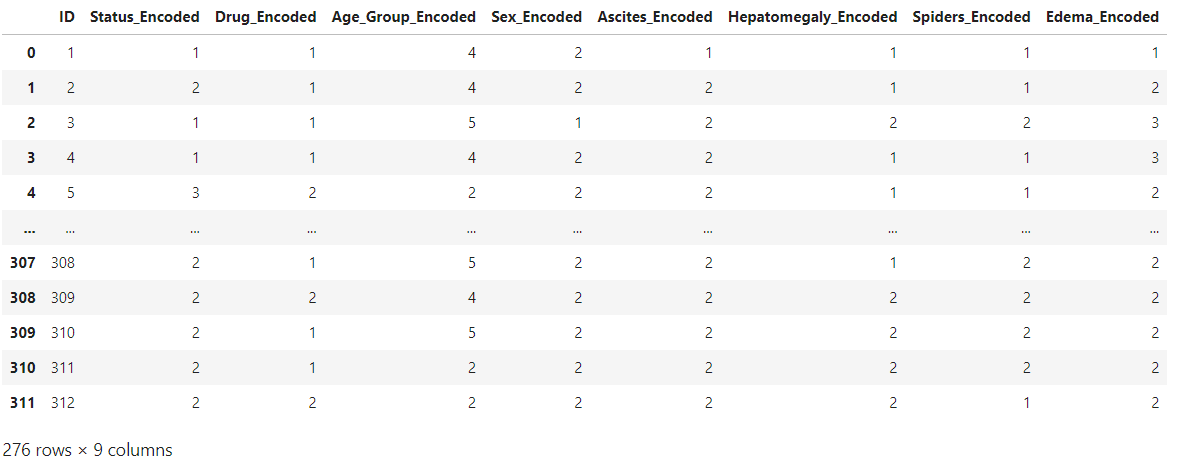
# Edema encoding

# Encode 'Edema' into a single new column

Edema\_mapping = {'Y': 1, 'N': 2, 'S': 3}

data['Edema\_Encoded'] = df['Edema'].map(Edema\_mapping)

data



# Define the threshold for Bilirubin

threshold = 1.2

​

# Encode 'Bilirubin' into a single new column based on the threshold

def encode\_bilirubin(bilirubin):

if bilirubin > threshold:

return 1

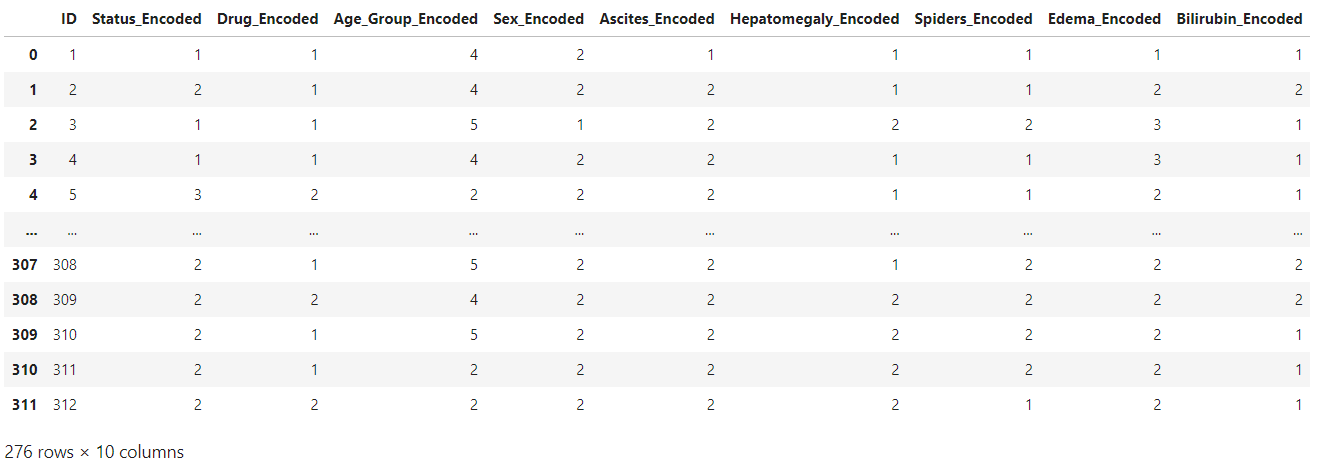
else:

return 2

​

data['Bilirubin\_Encoded'] = df['Bilirubin'].apply(encode\_bilirubin)

data



# Define the threshold for Cholesterol

threshold = 200

​

# Encode 'Cholesterol' into a single new column based on the threshold

def encode\_cholesterol(cholesterol):

if cholesterol >= threshold:

return 2

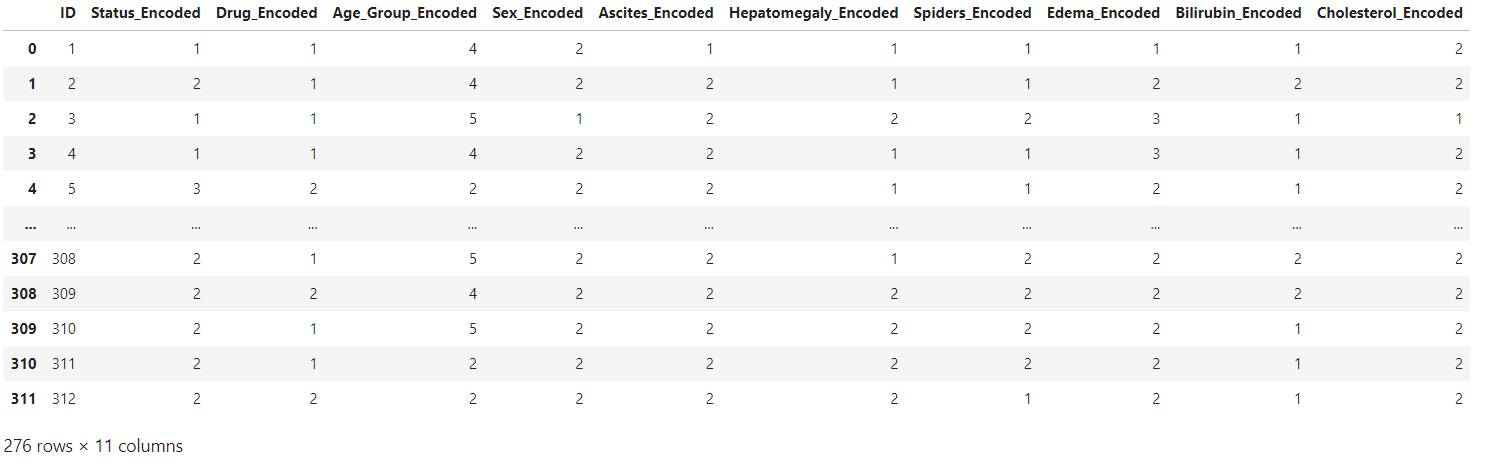
else:

return 1

​

data['Cholesterol\_Encoded'] = df['Cholesterol'].apply(encode\_cholesterol)

data



# Define the threshold for Albumin

threshold = 3.5

​

# Encode 'Albumin' into a single new column based on the threshold

def encode\_albumin(albumin):

if albumin >= threshold:

return 2

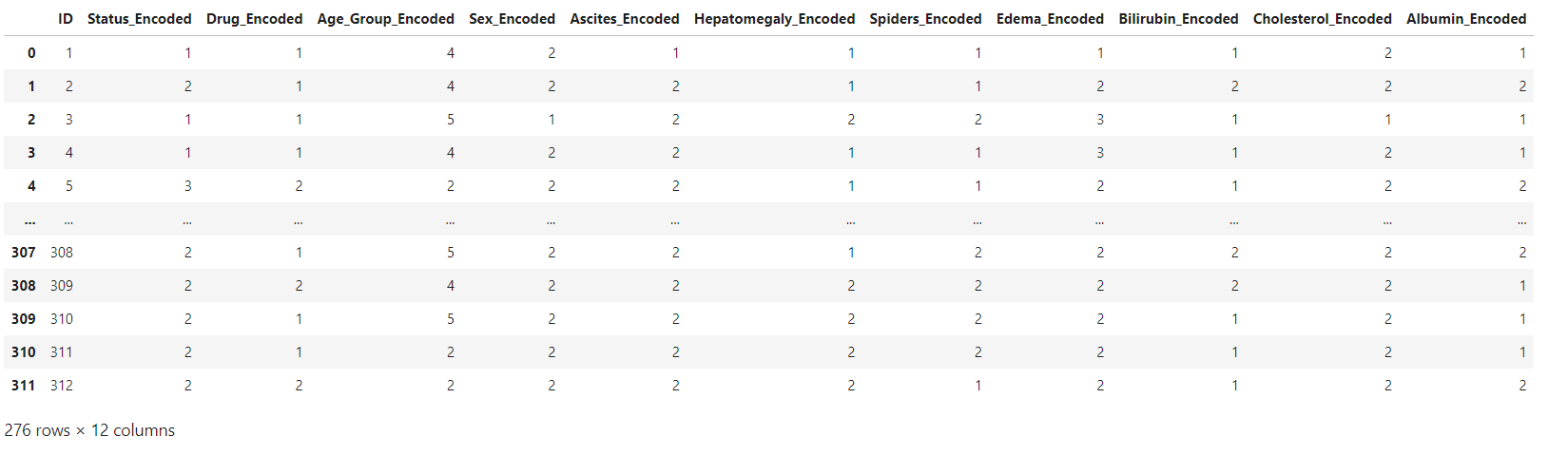
else:

return 1

​

data['Albumin\_Encoded'] = df['Albumin'].apply(encode\_albumin)

data



# Encode 'Alk\_Phos' into a single new column based on gender-specific thresholds

def encode\_alk\_phos(row):

if row['Sex'] == 'F':

if row['Alk\_Phos'] > 104:

return 1

else:

return 2

elif row['Sex'] == 'M':

if row['Alk\_Phos'] > 120:

return 1

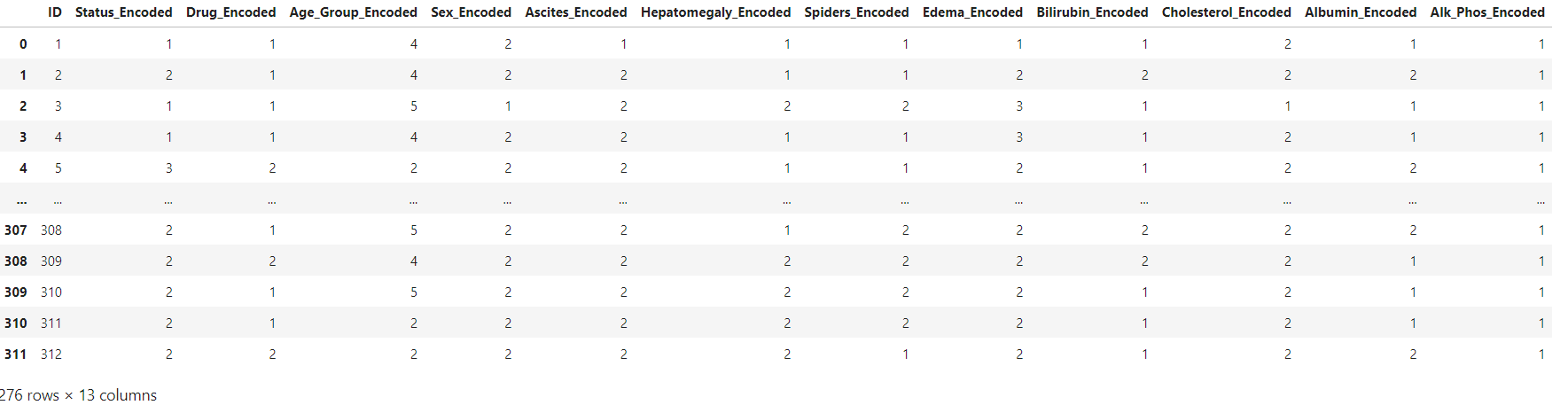
else:

return 2

​

data['Alk\_Phos\_Encoded'] = df.apply(encode\_alk\_phos, axis=1)

data



# Define the threshold for SGOT

threshold = 40

​

# Encode 'SGOT' into a single new column based on the threshold

def encode\_sgot(sgot):

if sgot > threshold:

return 1

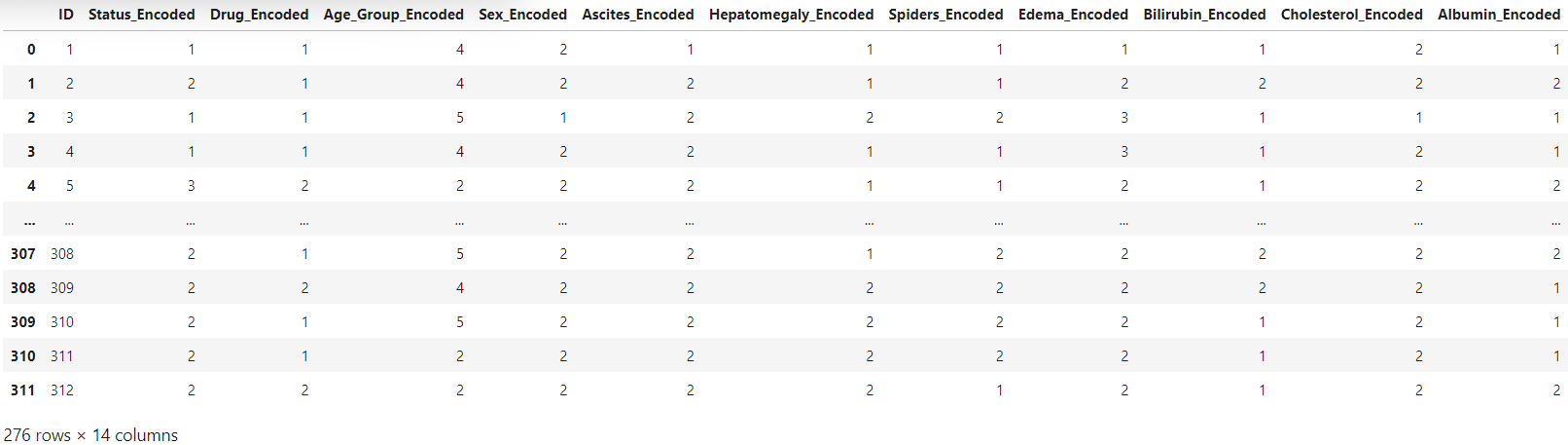
else:

return 2

​

data['SGOT\_Encoded'] = df['SGOT'].apply(encode\_sgot)

data

# Define the threshold for Tryglicerides

threshold = 150

​

# Encode 'Tryglicerides' into a single new column based on the threshold

def encode\_triglycerides(triglycerides):

if triglycerides < threshold:

return 1

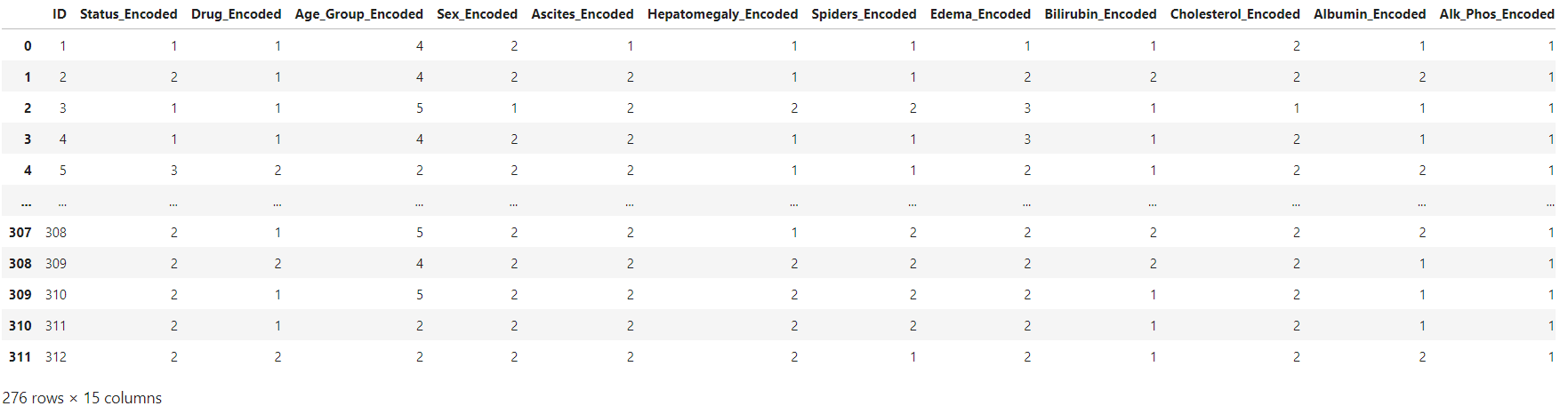
else:

return 2

​

data['Triglycerides\_Encoded'] = df['Tryglicerides'].apply(encode\_triglycerides)

data

# Define the threshold for Platelets

threshold = 150000

​

# Encode 'Platelets' into a single new column based on the threshold

def encode\_platelets(platelets):

if platelets < threshold:

return 1

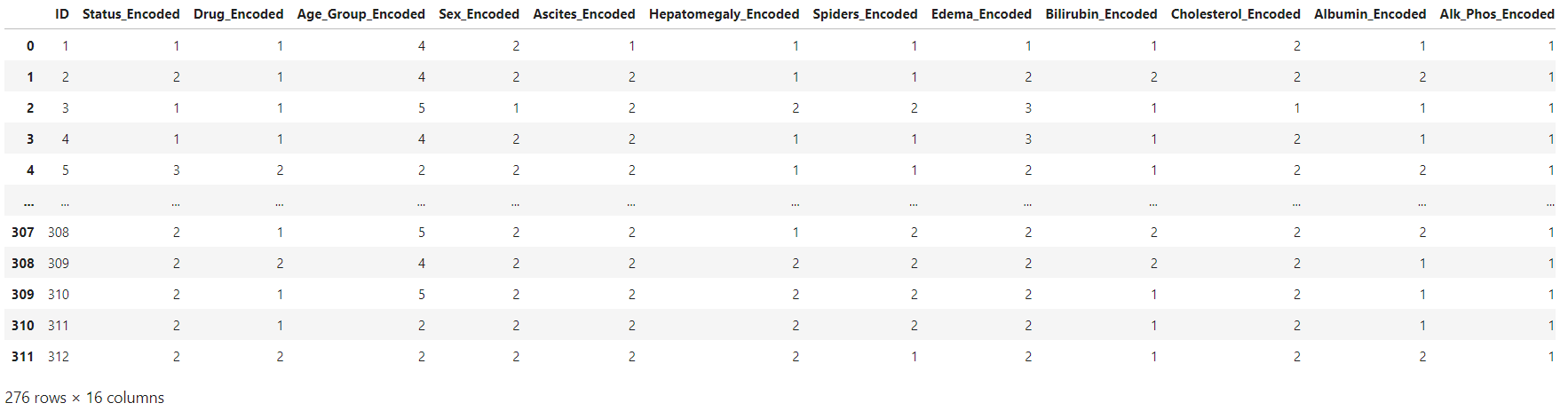
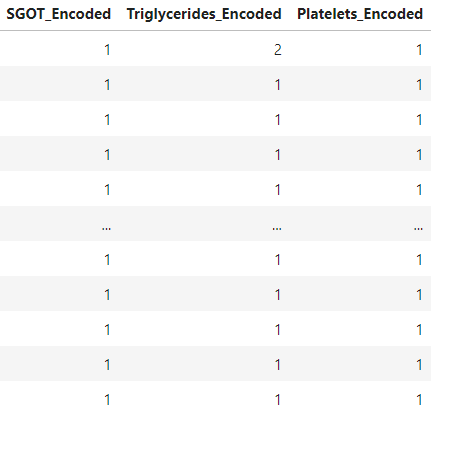
else:

return 2

​

data['Platelets\_Encoded'] = df['Platelets'].apply(encode\_platelets)

data

# Define the threshold for Prothrombin

threshold = 13

​

# Encode 'Prothrombin' into a single new column based on the threshold

def encode\_prothrombin(prothrombin):

if prothrombin > threshold:

return 1

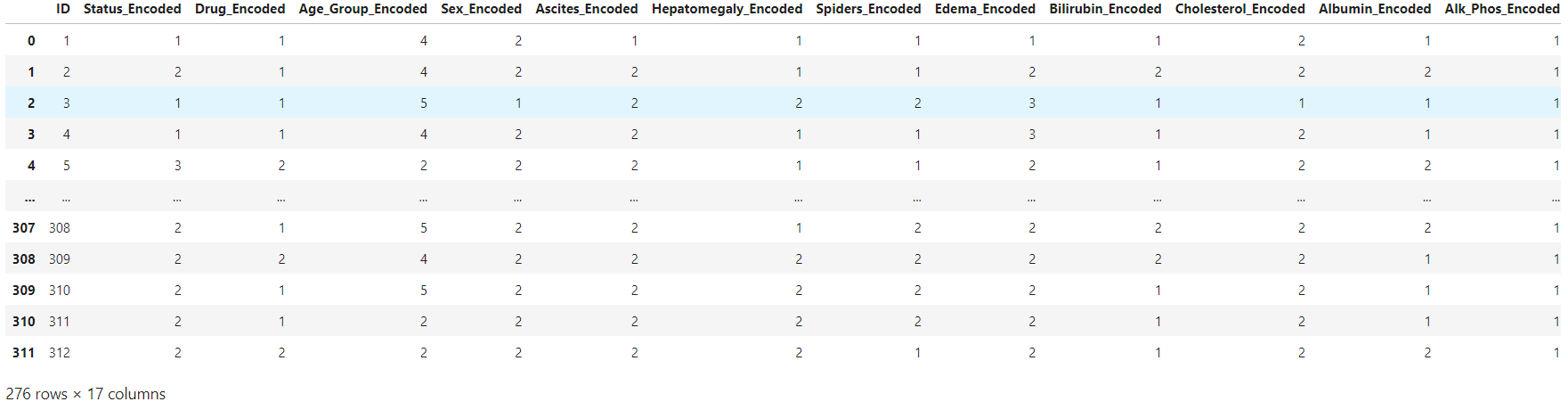
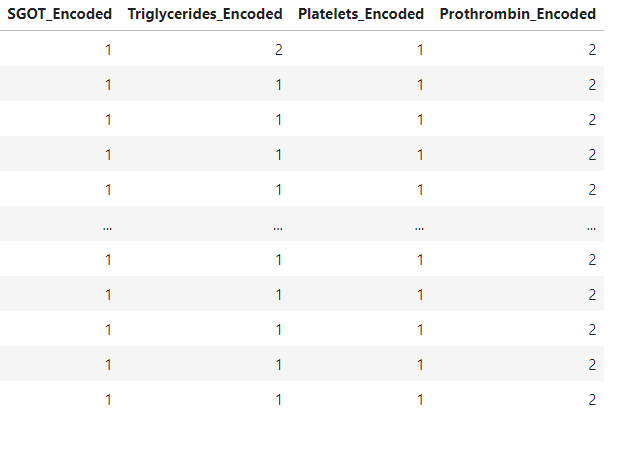
else:

return 2

​

data['Prothrombin\_Encoded'] = df['Prothrombin'].apply(encode\_prothrombin)

data

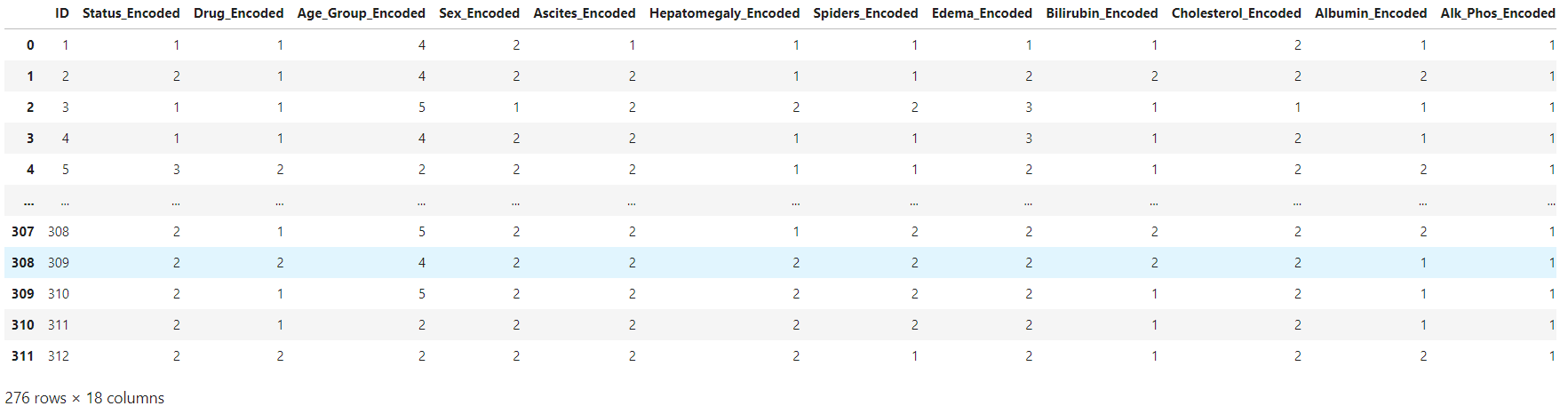
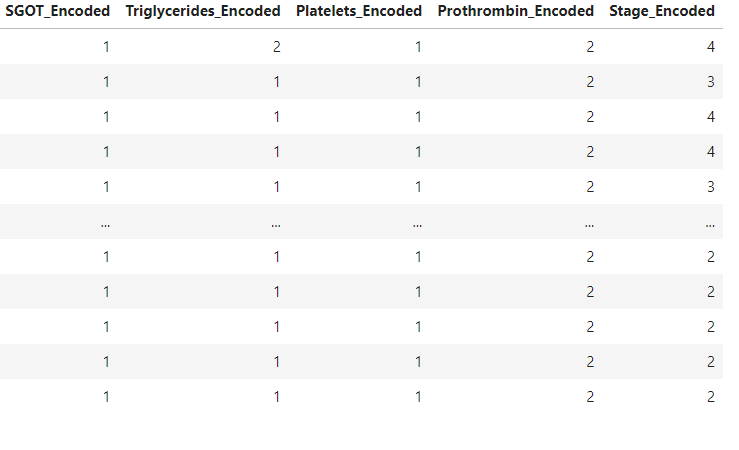
 

# Encode 'Stage' into a single new column

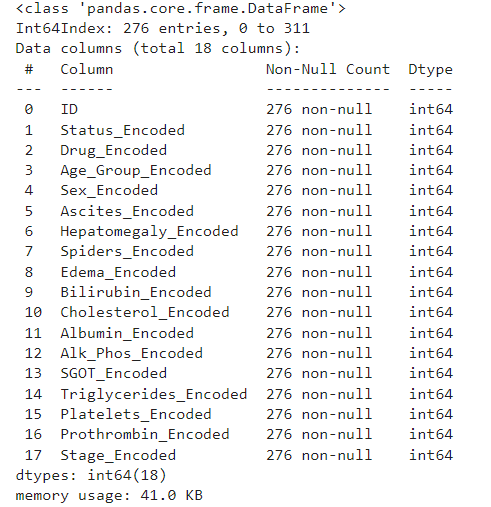
stage\_mapping = {1.0: 1, 2.0: 2, 3.0: 3, 4.0: 4}

data['Stage\_Encoded'] = df['Stage'].map(stage\_mapping)

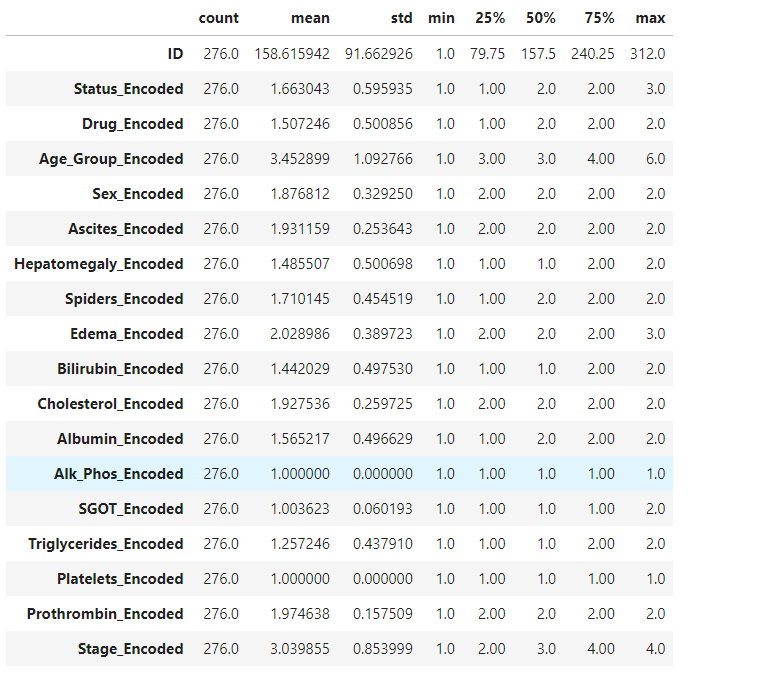
data

data.info()



data.describe().T



Now the Feature Engineering process is completed. Variety of new features focussing on the patients have been created which will give a deeper understanding of their symptoms and blood tests.

Now the next step is to train disease prediction, disease's stage prediction and drug prediction model using Logistic Regression and Random Forest Classifier.

**Disease Prediction, Disease's Stage Prediction and Drug Prediction Model using Logistic Regression and Random Forest Classifier**

Prediction Models:

Logistic Regression:

Interpretability: Logistic regression provides a straightforward interpretation. The coefficients of the model indicate the impact of each feature on the probability of cirrhosis occurrence.

Binary Classification: It’s well-suited for binary classification tasks.

Linear Decision Boundary: Assumes a linear relationship between features and the log-odds of cirrhosis.

Working Principle:

Logistic Regression models the log-odds (logit) of the probability of the positive class.

The logistic function (sigmoid) maps the log-odds to a probability value between 0 and 1.

The decision boundary is a linear combination of features.

The model learns the coefficients (weights) for each feature during training.

Given features (X\_1, X\_2, ..., X\_n), the logistic regression model predicts the probability of the target variable Y being 1 (positive class) as:

P(Y = 1 | X\_1, X\_2, ..., X\_n) = 1 / (1 + e^(-(w\_0 + w\_1 X\_1 + w\_2 X\_2 + ... + w\_n X\_n)))

here:

Y is the binary target variable (1 for positive class, 0 for negative class).

(w\_0, w\_1, ..., w\_n) are the coefficients (weights) learned by the model.

Random Forest Classifier:

Ensemble Method: Random Forest combines multiple decision trees, reducing overfitting and improving generalization.

Non-Linearity: Unlike logistic regression, Random Forest can capture non-linear relationships between features and outcomes.

Feature Importance: Provides feature importance scores, helping identify which features contribute most to predictions.

Robustness: Handles missing data and outliers better than some other models.

Working Principle:

Random Forest builds a forest of decision trees (usually hundreds or thousands).

Each tree is trained on a random subset of the data (samples) and a random subset of features.

During prediction, each tree votes, and the majority vote determines the final class.

The ensemble reduces overfitting and improves generalization.

The final prediction is based on the majority vote of individual decision trees.

Each tree contributes to the final prediction by considering different subsets of data and features.

# Splitting the dataset into training and testing sets

X = data.drop(columns=['ID', 'Status\_Encoded', 'Drug\_Encoded', 'Stage\_Encoded'])

y\_disease = data['Status\_Encoded']

y\_stage = data['Stage\_Encoded']

y\_drug = data['Drug\_Encoded']

​

X\_train, X\_test, y\_train\_disease, y\_test\_disease, y\_train\_stage, y\_test\_stage, y\_train\_drug, y\_test\_drug = train\_test\_split(X, y\_disease, y\_stage, y\_drug, test\_size=0.2, random\_state=42)

# Disease Prediction Models

logistic\_regression\_model = LogisticRegression()

random\_forest\_model = RandomForestClassifier()

​

logistic\_regression\_model.fit(X\_train, y\_train\_disease)

random\_forest\_model.fit(X\_train, y\_train\_disease)

​

# Disease Stage Prediction Models

logistic\_regression\_stage\_model = LogisticRegression(multi\_class='multinomial', solver='lbfgs')

random\_forest\_stage\_model = RandomForestClassifier()

​

logistic\_regression\_stage\_model.fit(X\_train, y\_train\_stage)

random\_forest\_stage\_model.fit(X\_train, y\_train\_stage)

​

# Drug Prediction Models

logistic\_regression\_drug\_model = LogisticRegression()

random\_forest\_drug\_model = RandomForestClassifier()

​

logistic\_regression\_drug\_model.fit(X\_train, y\_train\_drug)

random\_forest\_drug\_model.fit(X\_train, y\_train\_drug)

**Models Evaluation**

# Disease Prediction Evaluation

disease\_predictions\_lr = logistic\_regression\_model.predict(X\_test)

disease\_predictions\_rf = random\_forest\_model.predict(X\_test)

​

# Disease Stage Prediction Evaluation

stage\_predictions\_lr = logistic\_regression\_stage\_model.predict(X\_test)

stage\_predictions\_rf = random\_forest\_stage\_model.predict(X\_test)

​

# Drug Prediction Evaluation

drug\_predictions\_lr = logistic\_regression\_drug\_model.predict(X\_test)

drug\_predictions\_rf = random\_forest\_drug\_model.predict(X\_test)

disease\_predictions\_lr

array([2, 2, 1, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1, 2, 1, 2,

2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1,

2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 2, 1], dtype=int64)

disease\_predictions\_rf

array([2, 2, 1, 1, 2, 2, 1, 3, 1, 2, 3, 2, 2, 2, 2, 1, 2, 2, 1, 2, 1, 2,

2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1,

2, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1], dtype=int64)

stage\_predictions\_lr

array([2, 4, 4, 3, 3, 3, 4, 3, 2, 2, 3, 4, 3, 3, 2, 4, 3, 3, 4, 3, 4, 2,

4, 3, 3, 3, 3, 4, 4, 2, 3, 4, 4, 4, 3, 4, 3, 4, 3, 3, 3, 4, 4, 4,

3, 2, 3, 3, 4, 4, 3, 3, 4, 4, 3, 3], dtype=int64)

stage\_predictions\_rf

array([2, 4, 4, 3, 4, 3, 4, 3, 3, 2, 3, 4, 3, 3, 2, 4, 3, 3, 4, 3, 4, 2,

4, 3, 2, 3, 2, 4, 4, 2, 2, 4, 4, 3, 3, 4, 2, 4, 3, 3, 3, 4, 4, 4,

3, 2, 3, 3, 4, 4, 3, 3, 3, 4, 2, 4], dtype=int64)

drug\_predictions\_lr

array([1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 2, 2, 1,

1, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 1, 2,

2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 2], dtype=int64)

drug\_predictions\_rf

array([1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 1,

1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2,

2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 1, 2], dtype=int64)

Compare y\_test\_disease, y\_test\_stage and y\_test\_drug with [disease\_predictions\_lr, disease\_predictions\_rf], [stage\_predictions\_lr, stage\_predictions\_rf] and [drug\_predictions\_lr and drug\_predictions\_rf] respectively. We will be able to see the difference between the actual testing sets and the predicted testing sets.

True Positive (TP): Instances correctly predicted as positive.

True Negative (TN): Instances correctly predicted as negative.

False Positive (FP): Instances incorrectly predicted as positive (Type I error).

False Negative (FN): Instances incorrectly predicted as negative (Type II error).

Precision (Positive Predictive Value): TP / Support(TP + FP)

Recall (Sensitivity): TP / Support(TP + FN)

F1-Score: 2 \* Precision \* Recall / (Precision + Recall)

Support: The number of occurrences of each class.

Accuracy: No. of correctly classified samples(TP + TN) / Total Samples(TP + TN + FP + FN)

Macro average: ∑ Metric / N

Weighted average: ∑ Metric \* Support / ∑ Support

where, N is the number of classes.

Metric is the metric (precision, recall, or F1-score) for the ith class.

Support is the support (number of instances) of the ith class.

# Performance Metrics

print("Disease Prediction Metrics:")

print("Logistic Regression:")

print(classification\_report(y\_test\_disease, disease\_predictions\_lr))

print("Random Forest:")

print(classification\_report(y\_test\_disease, disease\_predictions\_rf))

​

print("\nDisease Stage Prediction Metrics:")

print("Logistic Regression:")

print(classification\_report(y\_test\_stage, stage\_predictions\_lr))

print("Random Forest:")

print(classification\_report(y\_test\_stage, stage\_predictions\_rf))

​

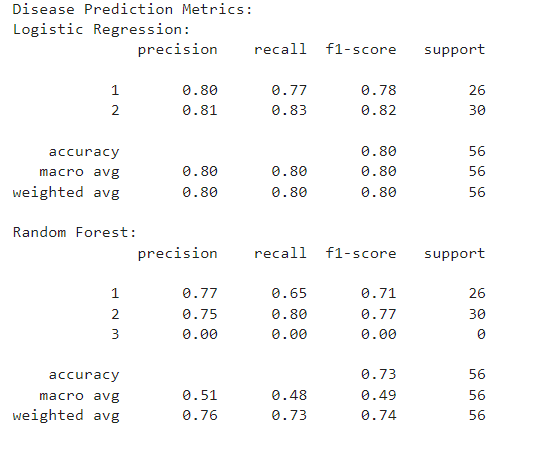
print("\nDrug Prediction Metrics:")

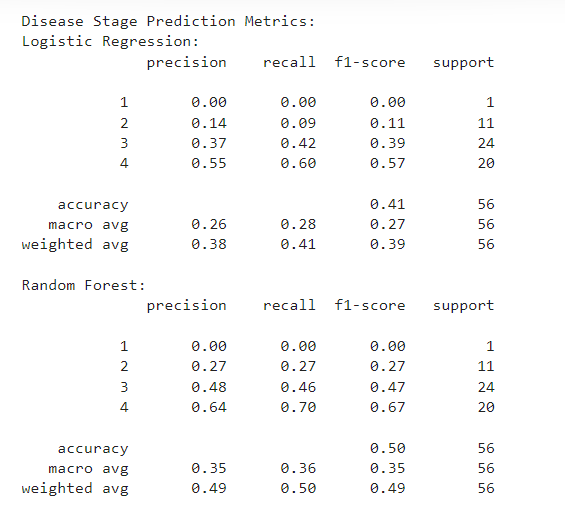
print("Logistic Regression:")

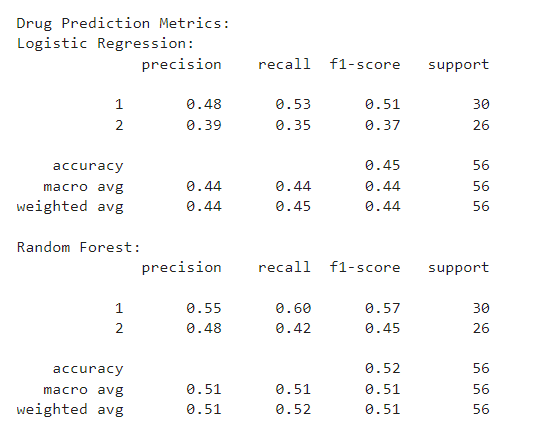
print(classification\_report(y\_test\_drug, drug\_predictions\_lr))

print("Random Forest:")

print(classification\_report(y\_test\_drug, drug\_predictions\_rf))







Disease Prediction Metrics Comparision:

Logistic Regression generally outperforms Random Forest in all metrics.

Random Forest shows poorer performance especially in precision, recall, and F1-score for Class 3, likely due to the imbalance or lack of data for that class.

Logistic Regression exhibits better overall performance with higher accuracy, precision, recall, and F1-score.

Disease Stage Prediction Metrics Comparision:

Random Forest generally outperforms Logistic Regression in most metrics.

Random Forest has better overall performance with higher accuracy, precision, recall, and F1-score.

Both models struggle to predict Class 1, possibly due to insufficient data or inherent complexity in distinguishing this class from others.

Drug Prediction Metrics Comparision:

Random Forest outperforms Logistic Regression in most metrics, including accuracy, precision, recall, and F1-score.

Random Forest generally provides better predictions for both classes of drugs compared to Logistic Regression.

Overall, Random Forest demonstrates better performance in drug prediction compared to Logistic Regression.

(2x2)

Actual C1 Actual C2

Predicted C1[TP, FN],

Predicted C2[FP, TP]

(3x3)

Actual C1 Actual C2 Actual C3

Predicted C1[TP, FP, FP],

Predicted C2[FN, TP, FP],

Predicted C3[FN, FN, TP]

(4x4)

Here's how to calculate each value of the 4x4 confusion matrix:

The value at position (i, j) in the confusion matrix represents the number of instances that belong to class i but are classified as class j.

Each value is counted based on the model's predictions and the actual ground truth labels.

# Confusion Matrix

conf\_matrix\_lr\_disease = confusion\_matrix(y\_test\_disease, disease\_predictions\_lr)

conf\_matrix\_rf\_disease = confusion\_matrix(y\_test\_disease, disease\_predictions\_rf)

​

conf\_matrix\_lr\_stage = confusion\_matrix(y\_test\_stage, stage\_predictions\_lr)

conf\_matrix\_rf\_stage = confusion\_matrix(y\_test\_stage, stage\_predictions\_rf)

​

conf\_matrix\_lr\_drug = confusion\_matrix(y\_test\_drug, drug\_predictions\_lr)

conf\_matrix\_rf\_drug = confusion\_matrix(y\_test\_drug, drug\_predictions\_rf)

conf\_matrix\_lr\_disease

array([[20, 6],

[ 5, 25]], dtype=int64)

conf\_matrix\_rf\_disease

array([[17, 8, 1],

[ 5, 24, 1],

[ 0, 0, 0]], dtype=int64)

conf\_matrix\_lr\_stage

array([[ 0, 0, 1, 0],

[ 0, 1, 8, 2],

[ 0, 6, 10, 8],

[ 0, 0, 8, 12]], dtype=int64)

conf\_matrix\_rf\_stage

array([[ 0, 0, 1, 0],

[ 0, 3, 6, 2],

[ 0, 7, 11, 6],

[ 0, 1, 5, 14]], dtype=int64)

conf\_matrix\_lr\_drug

array([[16, 14],

[17, 9]], dtype=int64)

conf\_matrix\_rf\_drug

array([[18, 12],

[15, 11]], dtype=int64)

Cross-Validation

Data Splitting: We divide the available dataset into multiple folds or subsets.

Training and Validation: For each fold:

We use one fold as a validation set (to evaluate the model).

The remaining folds serve as a training set (to train the model).

Repetition: We repeat this process multiple times, each time using a different fold as the validation set.

Averaging Results: Finally, we average the results from each validation step to produce a more robust estimate of the model’s performance by computing overall Evaluation metrics.

# Cross-Validation

cv\_scores\_lr\_disease = cross\_val\_score(logistic\_regression\_model, X, y\_disease, cv=5)

cv\_scores\_rf\_disease = cross\_val\_score(random\_forest\_model, X, y\_disease, cv=5)

​

cv\_scores\_lr\_stage = cross\_val\_score(logistic\_regression\_stage\_model, X, y\_stage, cv=5)

cv\_scores\_rf\_stage = cross\_val\_score(random\_forest\_stage\_model, X, y\_stage, cv=5)

​

cv\_scores\_lr\_drug = cross\_val\_score(logistic\_regression\_drug\_model, X, y\_drug, cv=5)

cv\_scores\_rf\_drug = cross\_val\_score(random\_forest\_drug\_model, X, y\_drug, cv=5)

cv\_scores\_lr\_disease

array([0.73214286, 0.70909091, 0.76363636, 0.65454545, 0.67272727])

cv\_scores\_rf\_disease

array([0.73214286, 0.61818182, 0.78181818, 0.50909091, 0.58181818])

cv\_scores\_lr\_stage

array([0.55357143, 0.61818182, 0.45454545, 0.50909091, 0.45454545])

cv\_scores\_rf\_stage

array([0.46428571, 0.47272727, 0.41818182, 0.47272727, 0.45454545])

cv\_scores\_lr\_drug

array([0.51785714, 0.43636364, 0.50909091, 0.63636364, 0.54545455])

cv\_scores\_rf\_drug

array([0.375 , 0.54545455, 0.50909091, 0.47272727, 0.58181818])

Now for assessing models' stability and generalizability we will calculate mean and variance of obtained validation scores.

# Calculate mean and variance for logistic regression models for disease prediction

mean\_cv\_score\_lr\_disease = np.mean(cv\_scores\_lr\_disease)

var\_cv\_score\_lr\_disease = np.var(cv\_scores\_lr\_disease)

​

# Calculate mean and variance for random forest models for disease prediction

mean\_cv\_score\_rf\_disease = np.mean(cv\_scores\_rf\_disease)

var\_cv\_score\_rf\_disease = np.var(cv\_scores\_rf\_disease)

# Calculate mean and variance for logistic regression models for stage prediction

mean\_cv\_score\_lr\_stage = np.mean(cv\_scores\_lr\_stage)

var\_cv\_score\_lr\_stage = np.var(cv\_scores\_lr\_stage)

​

# Calculate mean and variance for random forest models for stage prediction

mean\_cv\_score\_rf\_stage = np.mean(cv\_scores\_rf\_stage)

var\_cv\_score\_rf\_stage = np.var(cv\_scores\_rf\_stage)

# Calculate mean and variance for logistic regression models for drug prediction

mean\_cv\_score\_lr\_drug = np.mean(cv\_scores\_lr\_drug)

var\_cv\_score\_lr\_drug = np.var(cv\_scores\_lr\_drug)

​

# Calculate mean and variance for random forest models for drug prediction

mean\_cv\_score\_rf\_drug = np.mean(cv\_scores\_rf\_drug)

var\_cv\_score\_rf\_drug = np.var(cv\_scores\_rf\_drug)

# Print out the mean and variance for each model

print("Logistic Regression Model for Disease - Mean:", mean\_cv\_score\_lr\_disease, "Variance:", var\_cv\_score\_lr\_disease)

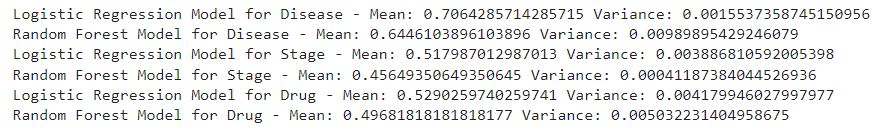
print("Random Forest Model for Disease - Mean:", mean\_cv\_score\_rf\_disease, "Variance:", var\_cv\_score\_rf\_disease)

print("Logistic Regression Model for Stage - Mean:", mean\_cv\_score\_lr\_stage, "Variance:", var\_cv\_score\_lr\_stage)

print("Random Forest Model for Stage - Mean:", mean\_cv\_score\_rf\_stage, "Variance:", var\_cv\_score\_rf\_stage)

print("Logistic Regression Model for Drug - Mean:", mean\_cv\_score\_lr\_drug, "Variance:", var\_cv\_score\_lr\_drug)

print("Random Forest Model for Drug - Mean:", mean\_cv\_score\_rf\_drug, "Variance:", var\_cv\_score\_rf\_drug)

Disease Prediction Models:

The Logistic Regression model has a higher mean accuracy and lower variance compared to the Random Forest model. This suggests that the Logistic Regression model is more stable and generalizable for disease prediction.

Stage Prediction Models:

The Logistic Regression model has a higher mean accuracy and the Random Forest model exhibits better stability and generalizability with lower variance. Therefore, if stability and consistency in performance are prioritized, the Random Forest model may be preferred for stage prediction.

Drug Prediction Models:

Both models have similar mean accuracies, but the Logistic Regression model has slightly lower variance. However, the difference in variance is relatively small, indicating similar stability and generalizability for drug prediction between the two models.

**Cirrhosis Disease Prediction, Disease's Stage Prediction & Drug Prediction Machine**

Patients can entry the following information: Age, Sex, Symptoms in Y/N and Blood Tests levels. The Machine will predict that if the patient has cirrhosis disease or not. If the patient has cirrhosis disease, it will also predict the stage of cirrhosis and will also recommend a drug for curing the disease by using the machine learning models: Logistic Regression & Random Forest.

# Define a mapping from encoded drug labels to drug names

drug\_mapping = {

1: 'D-penicillamine',

2: 'Placebo',

​

}

​

# Function to predict cirrhosis disease and stage

def predict\_cirrhosis(patient\_data):

# Preprocess the patient data

patient\_df = pd.DataFrame(patient\_data, index=[0])

patient\_df['Age\_Group\_Encoded'] = patient\_df['Age'].apply(encode\_age)

patient\_df['Sex\_Encoded'] = patient\_df['Sex'].map(sex\_mapping)

patient\_df['Ascites\_Encoded'] = patient\_df['Ascites'].map(ascites\_mapping)

patient\_df['Hepatomegaly\_Encoded'] = patient\_df['Hepatomegaly'].map(Hepatomegaly\_mapping)

patient\_df['Spiders\_Encoded'] = patient\_df['Spiders'].map(Spiders\_mapping)

patient\_df['Edema\_Encoded'] = patient\_df['Edema'].map(Edema\_mapping)

patient\_df['Bilirubin\_Encoded'] = patient\_df['Bilirubin'].apply(encode\_bilirubin)

patient\_df['Cholesterol\_Encoded'] = patient\_df['Cholesterol'].apply(encode\_cholesterol)

patient\_df['Albumin\_Encoded'] = patient\_df['Albumin'].apply(encode\_albumin)

patient\_df['Alk\_Phos\_Encoded'] = patient\_df.apply(encode\_alk\_phos, axis=1)

patient\_df['SGOT\_Encoded'] = patient\_df['SGOT'].apply(encode\_sgot)

patient\_df['Triglycerides\_Encoded'] = patient\_df['Tryglicerides'].apply(encode\_triglycerides)

patient\_df['Platelets\_Encoded'] = patient\_df['Platelets'].apply(encode\_platelets)

patient\_df['Prothrombin\_Encoded'] = patient\_df['Prothrombin'].apply(encode\_prothrombin)

# Predict disease

disease\_prediction\_lr = logistic\_regression\_model.predict(patient\_df.drop(columns=['Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol', 'Albumin', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']))

disease\_prediction\_rf = random\_forest\_model.predict(patient\_df.drop(columns=['Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol', 'Albumin', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']))

# If disease is predicted

if disease\_prediction\_lr[0] == 1 or disease\_prediction\_rf[0] == 1:

# Predict stage

stage\_prediction\_lr = logistic\_regression\_stage\_model.predict(patient\_df.drop(columns=['Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol', 'Albumin', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']))

stage\_prediction\_rf = random\_forest\_stage\_model.predict(patient\_df.drop(columns=['Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol', 'Albumin', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']))

# Predict drug

drug\_prediction\_lr = logistic\_regression\_drug\_model.predict(patient\_df.drop(columns=['Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol', 'Albumin', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']))

drug\_prediction\_rf = random\_forest\_drug\_model.predict(patient\_df.drop(columns=['Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema', 'Bilirubin', 'Cholesterol', 'Albumin', 'Alk\_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']))

# Map drug predictions to drug names

drug\_prediction\_lr\_name = drug\_mapping.get(drug\_prediction\_lr[0], 'Unknown Drug')

drug\_prediction\_rf\_name = drug\_mapping.get(drug\_prediction\_rf[0], 'Unknown Drug')

# Return predictions

return {

'disease\_prediction\_lr': disease\_prediction\_lr[0],

'disease\_prediction\_rf': disease\_prediction\_rf[0],

'stage\_prediction\_lr': stage\_prediction\_lr[0],

'stage\_prediction\_rf': stage\_prediction\_rf[0],

'drug\_prediction\_lr': drug\_prediction\_lr\_name,

'drug\_prediction\_rf': drug\_prediction\_rf\_name

}

else:

return "No cirrhosis disease predicted"

​

# User inputs

patient\_age = float(input("Enter patient's age (in years): "))

patient\_sex = input("Enter patient's sex (M/F): ")

patient\_ascites = input("Does the patient have Ascites? (Y/N): ")

patient\_hepatomegaly = input("Does the patient have Hepatomegaly? (Y/N): ")

patient\_spiders = input("Does the patient have Spiders? (Y/N): ")

patient\_edema = input("Does the patient have Edema? (Y/N): ")

patient\_bilirubin = float(input("Enter patient's Bilirubin (mg/dL) level: "))

patient\_cholesterol = float(input("Enter patient's Cholesterol (mg/dL) level: "))

patient\_albumin = float(input("Enter patient's Albumin (g/dL) level: "))

patient\_alk\_phos = float(input("Enter patient's Alk\_Phos (U/Liter) level: "))

patient\_sgot = float(input("Enter patient's SGOT (U/mL) level: "))

patient\_triglycerides = float(input("Enter patient's Tryglicerides (mg/dL) level: "))

patient\_platelets = float(input("Enter patient's Platelets (per cubic millimeter/1000) level: "))

patient\_prothrombin = float(input("Enter patient's Prothrombin (in seconds (s)) level: "))

​

# Predict disease and stage

patient\_data = {

'Age': patient\_age,

'Sex': patient\_sex,

'Ascites': patient\_ascites,

'Hepatomegaly': patient\_hepatomegaly,

'Spiders': patient\_spiders,

'Edema': patient\_edema,

'Bilirubin': patient\_bilirubin,

'Cholesterol': patient\_cholesterol,

'Albumin': patient\_albumin,

'Alk\_Phos': patient\_alk\_phos,

'SGOT': patient\_sgot,

'Tryglicerides': patient\_triglycerides,

'Platelets': patient\_platelets,

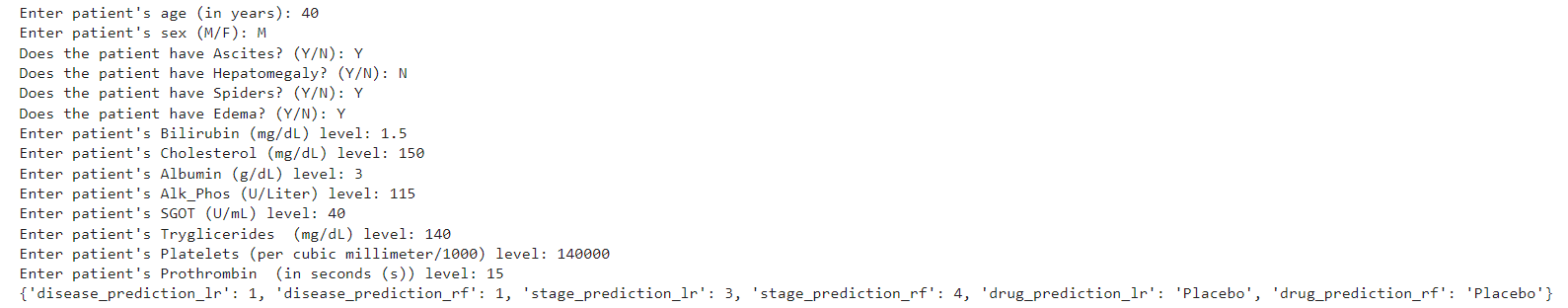
'Prothrombin': patient\_prothrombin

}

​

prediction\_result = predict\_cirrhosis(patient\_data)

print(prediction\_result)



**Literature Survey**

1. **Title of paper: An Integrated Architecture for Prediction of Heart Disease from the Medical Database**

**Name of the Authors: Gunasekar Thangarasu, Kayalvizhi Subramanian and P. D. D. Dominic**

**Year of Publication: 2018**

**Summary Points:**

**The paper discusses the design of a new framework for predicting heart disease using a combined technique of Hesitant Fuzzy based Decision tree Algorithm and Genetic algorithm. The study aimed to help identify methods and procedures for predicting heart disease to assist medical practitioners in an efficient way.**

**The proposed algorithms, Hesitant Fuzzy based Decision tree Algorithm and Genetic algorithm, were capable of tolerating a certain level of noise data, but when the health conditions of people are more complex, the noise data can sometimes lead to misleading results.**

**The study acknowledges the complexity of human health which covers genetic and environmental factors. The proposed work encourages progression and provides hope for application in identifying different illnesses such as diabetic disease and hypertension, among others.**

**The architecture that was developed provides insight into decision-making in identifying patient risk prediction using both structured and unstructured data. The design strategies that encouraged progression included the evolution of a new tool.**

**In conclusion, this paper provides valuable insight into the design of a new framework for predicting heart disease using a combined technique of Hesitant Fuzzy based Decision tree Algorithm and Genetic algorithm. While the study has its limitations, its future work provides hope for the development of tools and design strategies that encourage progression.**

1. **Title of paper: Big Data Analysis for Prediction of Coronary Artery Disease**

**Name of the Authors: Prerna Jain, Amandeep Kaur**

**Year of Publication: 2018**

**Summary Points:**

**The provided PDF contains information on various case studies using big data tools in healthcare. The following algorithms and techniques are discussed in the PDF:**

**1. K-means clustering algorithm for the analysis and data archival of HIV/AIDS disease data using Big data tool Mongo DB.**

**2. Genetic algorithm for the analysis and data archival of mental health data using Big data tool Mongo DB.**

**3. Machine learning, virtual screening, map reduce, and Mahout for the design and identification of drug for breast cancer.**

**The authors mention the use of Big data tools and techniques for managing, extracting, storing, and analyzing large amounts of healthcare data. The methodology used for these case studies consists of three core models: data input, analytic model, and decision support tools that improve business value and increase healthcare data management.**

**Advantages of using Big data tools and techniques in healthcare include an improved way of treatment, cost reduction, and time-saving. The big data tools and techniques help in managing, extracting, and storing large amounts of data, which have become increasingly difficult to analyze due to the rapid increase in patients, diseases, doctors, and hospitals.**

**Future scope includes the usage of Big data tools and techniques in various sectors like healthcare, public sector, retail, manufacture, and personal location data.**

1. **Title of paper: Diabetes Disease Prediction using Machine Learning on Big Data of Healthcare**

**Name of the Authors: Ayman Mir, Sudhir N. Dhage**

**Year of Publication: 2021**

**Summary Points:**

**The PDF file describes a research study on the use of machine learning algorithms for the classification of the Diabetes disease. The WEKA tool, a machine learning and data mining toolkit, is used to evaluate different machine learning techniques on real-time data. Four machine learning algorithms were used; Naive Bayes, SVM, Random Forest, and Simple CART. The study utilized the Pima Indians Diabetes Database as input for the classifiers implemented using various algorithms. The dataset contains 768 instances, including 9 attributes and two target classes that are tested positive and tested negative.**

**The proposed methodology involved preprocessing the input dataset using WEKA tool, training and testing the dataset using the four machine learning algorithms, and gathering results. The methodology flowchart is depicted in Figure 2 of the PDF. The best performing algorithm can be determined based on the experimental results for building a classification model to predict the diabetes disease.**

**The research used the WEKA tool to perform performance evaluation and comparison of various machine learning techniques conveniently on real-time data. The limitations of the methodology were not discussed in the given pages of the PDF. The future scope of the study includes testing the proposed methodology on different datasets and comparing its performance with other existing methods.**

1. **Title of paper: Prediction of Chronic Disease by Machine Learning**

**Name of the Authors: Anandajayam.P, Aravindkumar.S, Arun.P, Ajith.A**

**Year of Publication: 2021**

**Summary Points:**

**Based on the PDF, the following points can be summarized:**

**Algorithms used: Support Vector Machine (SVM), Recurrent Neural Network (RNN), Decision Tree, and Naive Bayes were used as algorithms for the analysis.**

**Methodology/Architecture used: The SVM method used the Poly method where a circular or irregular shape was drawn over the x and y axis, and features were extracted using the given formula. The RNN method had loops that used information from previous passes in a network. The Decision Tree method involved apportioning the informational index into subsets, pruning, and tree selection. The Naive Bayes classifier required a small amount of training data to estimate.**

**Advantages and disadvantages of methodology/architecture: SVM is good for classification tasks, but the training time is slower. RNN is useful for modeling sequences of data, but it can suffer from vanishing gradient problems. Decision Trees are easy to understand and interpret, but they can overfit and lead to unstable results. Naive Bayes classifiers are simple and require less training data, but assumptions of independence can be unrealistic.**

**Future scope: The PDF does not mention specific future scope related to the analysis methods.**

**Remarks: The PDF provides detailed descriptions and formulas for each analysis method along with visual representations and performance metrics. It is a helpful source for gaining an understanding of the algorithms and their applications.**

1. **Title of paper: Prediction of Diabetes Using Machine Learning Algorithms in Healthcare**

**Name of the Authors: Muhammad Azeem Sarwar, Nasir Kamal, Wajeeha Hamid, Munam Ali Shah**

**Year of Publication: 2018**

**Summary Points:**

**The important points discussed in the paper as follows:**

**Algorithms Used: The paper discusses the application of various machine learning algorithms in predictive analytics for healthcare. Specifically, the Support Vector Machines (SVM) algorithm was used for the prediction of Hydrocephalus, prediction of diabetes types, complications, and treatments.**

**Methodology/Architecture Used: The methodology used in the paper involved obtaining a large dataset of healthcare, preprocessing the data, and applying various machine learning algorithms on the dataset. Hadoop MapReduce was also used for processing the large dataset, and results were distributed over different servers based on the geographical locations.**

**Advantages & Disadvantages of Methodology/Architecture: The use of machine learning algorithms in predictive analytics for healthcare can help in disease prediction, better decision making, and patient treatment. However, the methodology may require a large amount of data and computational resources, and the accuracy and performance of the applied algorithms may vary based on the nature of the dataset.**

**Future Scope: The paper provides a comprehensive survey of the literature on big data analytics and machine learning for healthcare and presents the potential of predictive analytics in disease prediction, better decision making, and patient treatment. The authors suggest further research can be done in exploring other machine learning techniques and optimal architecture for processing large data sets.**

**Remarks: The paper seems to provide a detailed overview of the predictive analytics in healthcare, including the relevant literature and the application of various machine learning algorithms. However, it does not cover all aspects, and further research may be required for a more comprehensive understanding of the field.**

1. **Title of paper: Disease Prediction Using Machine Learning Over Big Data**

**Name of the Authors: Shraddha Subhash Shirsath, Prof. Shubhangi Patil**

**Year of Publication: 2018**

**Summary Points:**

**The PDF discusses a proposed healthcare big data system that relies on the Naive Bayes algorithm for disease prediction. The system architecture involves using the Naive Bayes algorithm to clarify the hospital data which is then stored. The CNN algorithm is used for extracting text characteristics for selecting characteristics automatically from a large pool of data. Both structured and unstructured hospital data are used for the CNN-MDRP algorithm, which ultimately helps to improve disease prediction accuracy. The Naive Bayes algorithm converts the dataset into a frequency table and creates a likelihood table by finding probabilities. This algorithm is useful for a large amount of data sets but performs poorly when numerical data is involved in the training set.**

**The system requires a configuration that includes JDK 1.8, Mongo DB database, and Apache Tomcat server. There is a lot of missing data due to human error, so data imputations are performed to fill structured data. Data integration is used for processing data to improve its quality.**

**The advantages of the proposed system include its ability to predict diseases accurately over a large volume of data from a hospital. The methodology used allows for the automatic selection of characteristics from a large volume of data. However, the disadvantage of the Naive Bayes algorithm is its poor performance with numerical data in the training set.**

**In the future, the proposed system could be enhanced with the addition of more machine learning algorithms to improve disease prediction accuracy. Additionally, the system could be used in other domains, such as finance, weather forecasting, and more.**

1. **Title of paper: Big Data Analytics for Prediction Modelling in Healthcare Databases**

**Name of the Authors: Ritu Chauhan, Eiad Yafi**

**Year of Publication: 2021**

**Summary Points:**

**Algorithms used: The PDF mentions several algorithms for data analysis and mining such as data clustering, SPAM (Spatial Algorithm for Mining Grid Data), fluorescence navigation with indocyanine green, and more.**

**Methodology/architecture used: The paper discusses varied data analytics approaches used in healthcare in the past. In section III, the methodology for detecting patterns among the big databases is discussed using data clustering, machine learning, and other methods. The article also discusses the use of Big Data Analytics for ICT monitoring and development.**

**Advantages & Disadvantages: The PDF does not provide a detailed discussion on the advantages and disadvantages of the methodologies used, but it highlights that Big Data Analytics can identify and manage high-risk and high-cost patients, making it useful for quality improvement initiatives.**

**Future Scope: The article suggests that Big Data Analytics is an opportunity for healthcare research, despite challenges such as data privacy concerns, data integration, and data quality. The paper concludes that big data can become a cornerstone for quality improvement, research, and cost optimization in healthcare.**

1. **Title of paper: Big Data Analytics in Healthcare**

**Name of the Authors: Guorong Chen, Mohaiminul Islam**

**Year of Publication: 2019**

**Summary Points:**

**The PDF file provides a comprehensive overview of the benefits and significance of big data analytics in the healthcare industry. It starts by discussing the need for healthcare organizations to prioritize improving patient care and enhancing access to healthcare services. It notes that while patients and their families spend a lot of money on healthcare services, many healthcare providers do not offer high-quality care, which can be harmful to patients. The PDF emphasizes the importance of innovative technologies in this regard.**

**The PDF then explains how modern technology, such as robots, digital health records, and predictive machines, is transforming the medical sector, making it more efficient, and improving patient outcomes. Big data analytics has emerged as a powerful tool in this regard. Using big data analytics, healthcare providers can collect, process, analyze, and visualize large amounts of data to uncover patterns, insights, and new treatments for various health conditions.**

**The PDF describes the three main steps of the big data analytics process, which are data collection, processing, and analysis. It notes that healthcare organizations can use various data sources and tools to process, visualize, and analyze data and extract meaningful insights into patient health. The PDF mentions that one of the significant advantages of big data analytics is its ability to uncover unknown contextual patterns and insights by examining large datasets.**

**Lastly, the PDF provides examples of healthcare applications and tools that are currently available for use in the industry, such as Electronic Health Records (EHRs), Real-Time Alerting systems, Predictive Analytics, and tools that help reduce fraud and enhance security. The PDF highlights the benefits of EHRs in particular, explaining how they enable patients to save all of their data electronically, making it easier to transfer it to doctors or other healthcare professionals.**

1. **Title of paper: Performance Analysis of Supervised Machine Learning Algorithms on Medical Dataset**

**Name of the Authors: Amit Juyal, Chetan Pandey, Janmejay Pant, Ankur Dumka, Vikas Tomar**

**Year of Publication: 2020**

**Summary Points:**

**This PDF is about the performance analysis of supervised machine learning algorithms on a medical dataset. The study uses an ensemble learning method to predict heart problems and compares the performance of various evaluating parameters such as F-measure, Recall, ROC, precision, and accuracy. The study uses various classifiers, such as Decision Tree (DT), Naïve Bayes (NB), Support Vector Machine (SVM), Random Forest (RF) algorithms for predicting heart problems. The result showed that by combining two ML algorithms, DT with NB, 81.1% accuracy was achieved. Simultaneously, models like the Support Vector machine (SVM), Decision tree, Naïve Bayes, Random Forest models were also trained and tested individually.**

**The paper describes the importance of medical diagnosis using machine learning algorithms, and how machine learning is used in various areas, such as biometric recognition, handwriting recognition, and medical diagnosis. The study concludes that the use of ensemble learning methods in machine learning can help to improve the prediction of heart-related problems in patients, thus helping doctors to diagnose the diseases accurately and start treatment in time.**

**Regarding the methodology/architecture used, the study uses the supervised learning method to predict heart disease. This method is suitable for classifying input data based on labeled examples for each class. The advantages of supervised learning are that it produces accurate predictions and is easy to implement. However, one major disadvantage of supervised learning is that it requires a large amount of labeled data for training the model.**

**The future scope of the study lies in the further improvement of the accuracy of the predictions using machine learning algorithms, which can help in the accurate diagnosis of heart-related problems.**

1. **Title of paper: Analytical Approach towards Prediction of Diseases Using Machine Learning Algorithms**

**Name of the Authors: Ayushi Grover, Anukriti Kalani, Sanjay Kumar Dubey**

**Year of Publication: 2020**

**Summary Points:**

**Algorithms used: Various machine learning algorithms were evaluated for their predictive capability in healthcare domain by utilizing big data analytics and data mining techniques. Some of the algorithms mentioned are Convolutional neural network, Support Vector Machines, Random Forest, K-Nearest Neighbors, and Naïve Bayes.**

**Methodology/architecture used: The methodology used involved abstraction and examination of data using Electronic Health Record datasets followed by application of machine learning algorithms to predict diseases such as heart disease. A model was predicted which employed the EHR data and ML algorithms were applied to this data to predict heart disease.**

**Advantages and disadvantages of methodology/architecture: The advantages of the methodology are the potential to aid in the early and accurate prediction of diseases, which could decrease the rate of fatality. The results generated from machine learning algorithms are compared based on precision and effective time taken for analysis of patient data in the healthcare domain. The disadvantages of the methodology were not mentioned.**

**Future scope: It is suggested that the utilization of wearable technology can enable human-cloud integration in the next generation of the healthcare domain.**

**Remarks: Evaluating the potential of machine learning algorithms by employing big data analytics and data mining techniques for the prediction of diseases in the healthcare domain. The evaluation is based on precision and effective time taken for analysis of patient data.**

1. **Title of paper: Predictive Analytics Model Based on Multiclass Classification for Asthma Severity by Using Random Forest Algorithm**

**Name of the Authors: Wasif Akbar, Sehrish Saleem, Wei-Ping Wu, Arslan Javed, Muhammad Faheem, Muhammad Asim Saleem**

**Year of Publication: 2020**

**Summary Points:**

**The PDF discuss the use of big data analytics in the healthcare community. Various models and techniques have been presented to quantify and interpret changes in human activities for healthcare applications. The proposed PAM-RF method predicts the severity level of asthma disease using statistical classification techniques and random forest machine learning algorithms. The dataset used for prediction contains 16000 patient records with different respiratory diseases.**

**Algorithms used include regular pattern mining, cluster analysis, prediction, deep learning applications, and machine learning algorithms such as Random Forest. The methodology used involves statistical classification techniques and the construction of a prediction model with data. The dataset is collected from multiple hospitals and loaded into an RDD.**

**Advantages of the methodology used include high accuracy and fault tolerance in processing large datasets in a scalable manner. Disadvantages may include the need for high computational power and large amounts of data for effective implementation.**

**Future scope of this research includes the potential for real-time analysis of medical big data using Spark Streaming and Apache Kafka in the healthcare industry.**

1. **Title of paper: PREDICTION OF PROBABILITY OF DISEASE BASED ON SYMPTOMS USING MACHINE LEARNING ALGORITHM**

**Name of the Authors: Harini D K, Natesh M**

**Year of Publication: 2018**

**Summary Points:**

**the PDF discusses a proposed system for disease risk prediction in the healthcare field. The proposed system combines both structured and unstructured data from medical records using machine learning algorithms like latent factor model and CNN-MDRP. The system also consults with hospital experts to extract useful features for structured data and selects features automatically using a CNN algorithm for unstructured text data. The proposed CNN-MDRP algorithm provides higher accuracy in disease prediction by leveraging both structured and unstructured data of patients.**

**The methodology proposed in the PDF aims to improve the accuracy of risk classification in disease prediction using machine learning algorithms and data from medical records. The methodology involves collecting data from studies, patient examination, and clinical trial information to train the machine learning models. The system architecture comprises a convolutional neural network-based multimodal disease risk prediction algorithm that combines structured and unstructured data from the hospital.**

**The advantages of the proposed system include higher accuracy in disease prediction through the use of structured and unstructured data, and the proposed CNN-MDRP algorithm. However, the disadvantages of the system are not explicitly mentioned in the provided pages.**

**The future scope of the system involves improving the accuracy of the machine learning models used and expanding the data sources to other hospitals for better validation.**

1. **Title of paper: Prediction and Analysis of Heart Disease Using Data Mining Techniques**

**Name of the Authors: Anusha N.B, Chaitra K, Chandana H.M, Kiran G, Swathi D.V**

**Year of Publication: 2020**

**Summary Points:**

**Based on the PDF, here are the summarized important points:**

**Algorithms used: K-Nearest Neighbor Algorithm and Principle Component Analysis Algorithm were used for data mining in the diagnosis of heart disease.**

**Methodology/Architecture used: The proposed work includes the collection of raw data, training data using Inception model, testing of data, and determining accuracy. The system design involves software testing, which can be done manually or using automated tools.**

**Advantages & disadvantages of methodology/architecture: The use of K-Nearest Neighbor Algorithm and Principle Component Analysis Algorithm showed an improved accuracy in the diagnosis of heart disease. However, the limitations of manual testing and the need for hundreds of nodes for stress testing were noted.**

**Future scope: The study can be expanded to include other data mining techniques for the diagnosis of various diseases. Additionally, automation can be further developed to improve the efficiency of software testing.**

1. **Title of paper: Role of IoT and Bigdata Analytics in Healthcare for Disease Prediction**

**Name of the Authors: Yasmeen Shaikh, V. K. Parvati, S. R. Biradar**

**Year of Publication: 2020**

**Summary Points:**

**The PDF file includes various articles that focus on big data analytics in the healthcare industry. The first page provides an introduction to the document, highlighting the importance of working together to prevent chronic diseases and make medical facilities more easily available. Various literature reviews conducted by different authors, discussing big data in healthcare, and the benefits of integrating information from different sources in real-time.**

**Discussion on the role of big data analytics in healthcare. Highlights the importance of integrating healthcare data and the ability to use big data analytics to improve patient care and reduce costs.**

**The challenges of deploying big data analytical technologies in healthcare. The authors discuss the importance of data privacy and the potential risks associated with handling patient data.**

**Discusses the importance of proper training and methodology utilization for big data analytical techniques. Emphasizes the need for a proper framework for storage of the large volume of data collected in healthcare.**

**Talks about the benefits of big data analytics in healthcare, which include improved patient care, reduced costs, increased satisfaction, and timely prediction of diseases. Electronic health records are critical to achieving advanced patient care.**

1. **Title of paper: Performance Evaluation of Supervised Machine Learning Algorithms in Prediction of Heart Disease**

**Name of the Authors: P. Sujatha, K. Mahalakshmi**

**Year of Publication: 2020**

**Summary Points:**

**Based on the information provided in the PDF, the following are the important points:**

**Algorithms used: The PDF describes multiple supervised machine learning algorithms that were implemented on heart disease dataset from Kaggle website using the Python 3.7 tool. The heart disease dataset contains 303 records and includes both positive and negative observations. The PDF does not mention specific algorithms used beyond the general reference to "popular supervised machine learning algorithms", but does evaluate and compare performance using metrics such as accuracy, precision, F1-score, and AUC.**

**Methodology/Architecture used: The methodology used was supervised machine learning with a focus on preprocessing, feature selection, and training/testing the classifiers. Feature selection was done with Boruta feature selection technique, which selects the most significant attributes from the dataset with respect to an outcome variable and eliminates unrelated and redundant attributes. The PDF contains a detailed description of the features selected, including their description and meaning. The classifiers were split into 70% training and 30% testing, and confusion matrix was visualized to estimate the performance of the algorithms.**

**Advantages & disadvantages of methodology/architecture: The PDF does not mention specific advantages or disadvantages of the methodology used, but does note that inconsistent and incomplete data can significantly influence the performance of machine learning algorithms.**

**Future scope: The PDF does mention areas for improvement such as more complete and consistent data, and the improvement of accuracy by the selection of more significant features.**

1. **Title of paper: Heart Disease Prediction using Machine Learning**

**Name of the Authors: S.Nandhini, Monojit Debnath, Anurag Sharma, Pushkar**

**Year of Publication: 2018**

**Summary Points:**

**This PDF file discusses a system for real-time prediction of heart disease that can be used by patients with coronary disease. The system is able to monitor and predict heart disease using machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM), Logistic Regression, and Decision Trees. The diagnosis system is based on the heart disease dataset instance.**

**The system is very inexpensive as it uses a pulse sensor and sends data to a mobile device via an Arduino suite microcontroller. The methodology/architecture includes an end-to-end privacy mechanism that only allows the patient or doctor to view the patient's details with the approval of the patient, ensuring privacy protection.**

**The advantages of this system include its real-time prediction capabilities and ability to monitor heart disease. Additionally, the system is inexpensive and includes a privacy mechanism for protecting patient privacy. The disadvantages of the methodology and architecture used are not discussed in this PDF file.**

**Future scope of the system includes further research and development to improve its accuracy and effectiveness as well as the inclusivity of more dataset instances for improved prediction accuracy.**