LIVER CIRRHOSIS STAGE PREDICTION

By Machine Learning

Moe Htet Min
Unified Mentor

Table of Contents

Predicting Liver Cirrhosis Stage from Patient Data

GitHub: https://github.com/Moehtetmin28/Liver-Cirrhosis-Stage-Detection

Objective

This project aims to create an automated system which forecasts liver cirrhosis severity using Mayo Clinic obtained patient information. Diverse features about patient demographics together with clinical findings appear in the dataset.

Dataset Description

A Mayo Clinic study conducted from 1974 to 1984 serves as the source for the collected data about primary biliary cirrhosis (PBC) of the liver. The database includes these following columns:

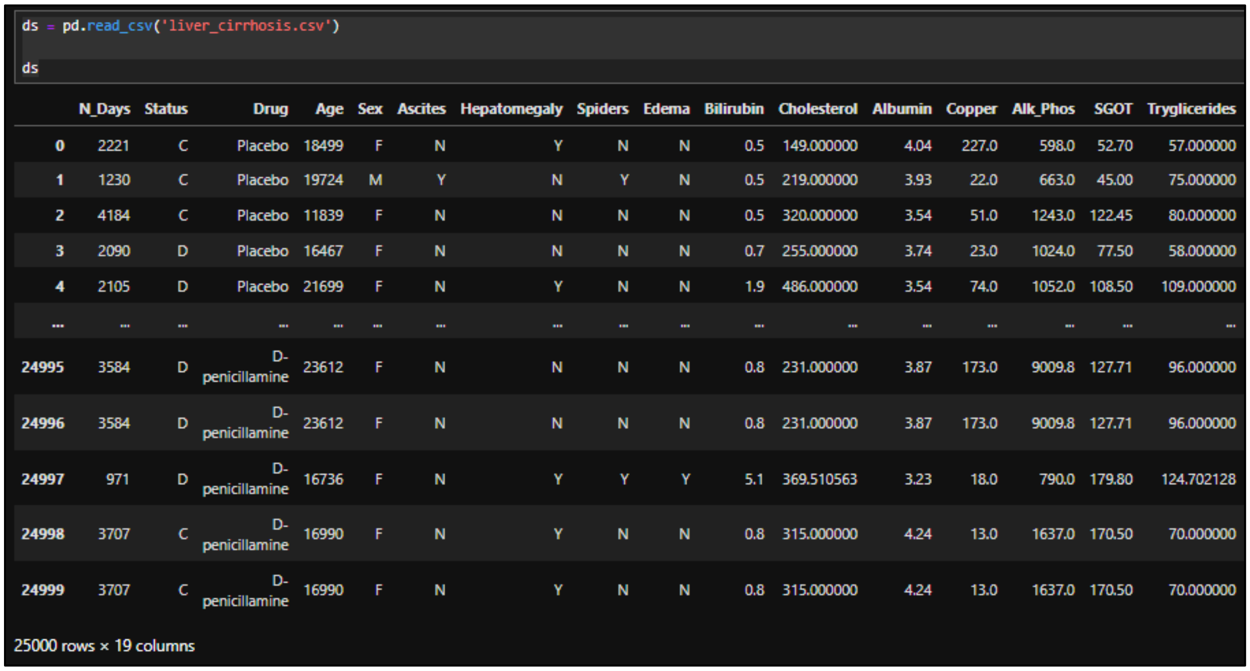
- **N_Days:** Number of days between registration and the earlier of death, transplantation, or study analysis time in 1986.
- **Status:** Status of the patient (C: censored, CL: censored due to liver transplantation, D: death).
- **Drug:** Type of drug (D-penicillamine or placebo).
- **Age:** Age in days.
- **Sex:** Sex of the patient (M: male, F: female).
- **Ascites:** Presence of ascites (N: No, Y: Yes).
- **Hepatomegaly:** Presence of hepatomegaly (N: No, Y: Yes).
- **Spiders:** Presence of spiders (N: No, Y: Yes).
- **Edema:** Presence of edema (N: no edema and no diuretic therapy for edema, S: edema present without diuretics, 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] (a liver enzyme).
- **Triglycerides:** Triglycerides in [mg/dl].
- **Platelets:** Platelets per cubic [ml/1000].

- **Prothrombin:** Prothrombin time in seconds [s].
- **Stage:** Histologic stage of disease (1, 2, or 3).

Methodology

1. Loading the Dataset

Pandas DataFrame serves as my initial point for importing the dataset.

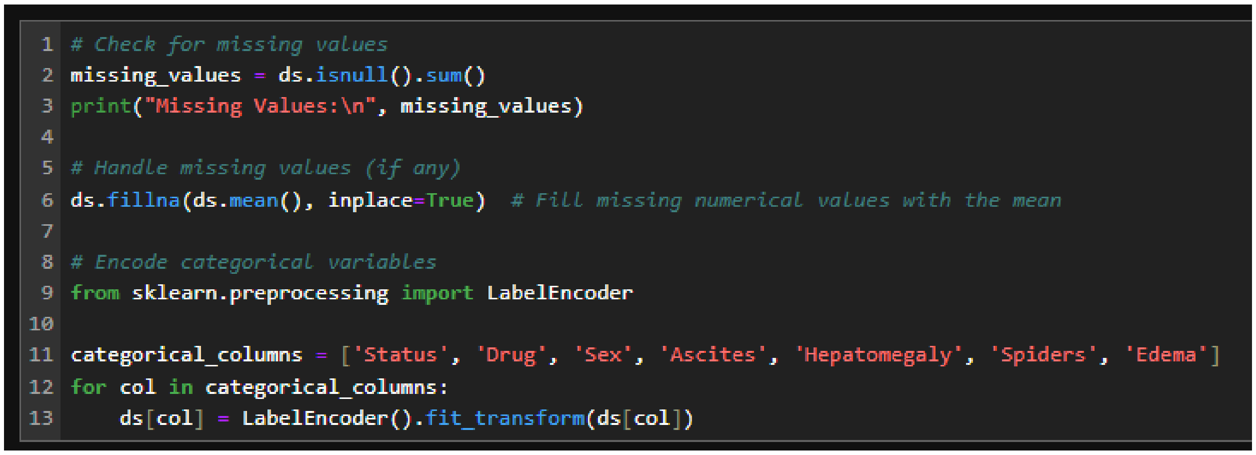


```
ds = pd.read_csv('liver_cirrhosis.csv')

ds
```

```
       N_Days  Status          Drug    Age  Sex  Ascites  Hepatomegaly  Spiders  Edema  Bilirubin  Cholesterol  Albumin  Copper  Alk_Phos    SGOT  Tryglicerides
0        2221       C       Placebo  18499    F        N             Y        N      N        0.5   149.000000     4.04   227.0     598.0   52.70      57.000000
1        1230       C       Placebo  19724    M        Y             N        Y      N        0.5   219.000000     3.93    22.0     663.0   45.00      75.000000
2        4184       C       Placebo  11839    F        N             N        N      N        0.5   320.000000     3.54    51.0    1243.0  122.45      80.000000
3        2090       D       Placebo  16467    F        N             N        N      N        0.7   255.000000     3.74    23.0    1024.0   77.50      58.000000
4        2105       D       Placebo  21699    F        N             Y        N      N        1.9   486.000000     3.54    74.0    1052.0  108.50     109.000000
...       ...     ...           ...    ...  ...      ...           ...      ...    ...        ...          ...      ...     ...       ...     ...            ...
24995    3584       D            D-  23612    F        N             N        N      N        0.8   231.000000     3.87   173.0    9009.8  127.71      96.000000
                       penicillamine
24996    3584       D            D-  23612    F        N             N        N      N        0.8   231.000000     3.87   173.0    9009.8  127.71      96.000000
                       penicillamine
24997     971       D            D-  16736    F        N             Y        Y      Y        5.1   369.510563     3.23    18.0     790.0  179.80     124.702128
                       penicillamine
24998    3707       C            D-  16990    F        N             Y        N      N        0.8   315.000000     4.24    13.0    1637.0  170.50      70.000000
                       penicillamine
24999    3707       C            D-  16990    F        N             Y        N      N        0.8   315.000000     4.24    13.0    1637.0  170.50      70.000000
                       penicillamine

25000 rows × 19 columns
```

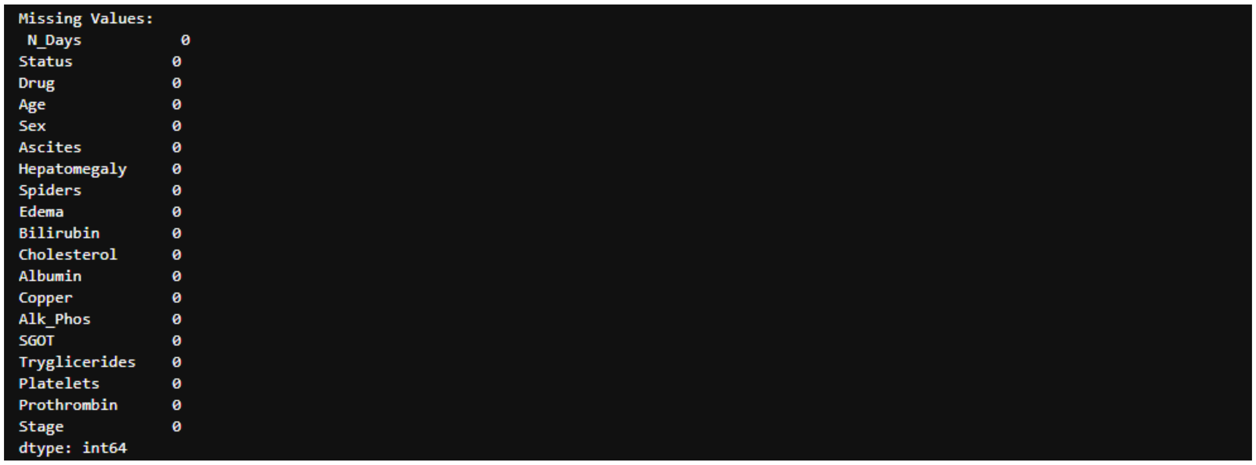
2. Data Preprocessing

The dataset cleaning process involves checking for missing data as well as variable normalization and categorical variable encoding.



```python
# Check for missing values
missing_values = ds.isnull().sum()
print("Missing Values:\n", missing_values)

# Handle missing values (if any)
ds.fillna(ds.mean(), inplace=True)  # Fill missing numerical values with the mean

# Encode categorical variables
from sklearn.preprocessing import LabelEncoder

categorical_columns = ['Status', 'Drug', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema']
for col in categorical_columns:
    ds[col] = LabelEncoder().fit_transform(ds[col])
```

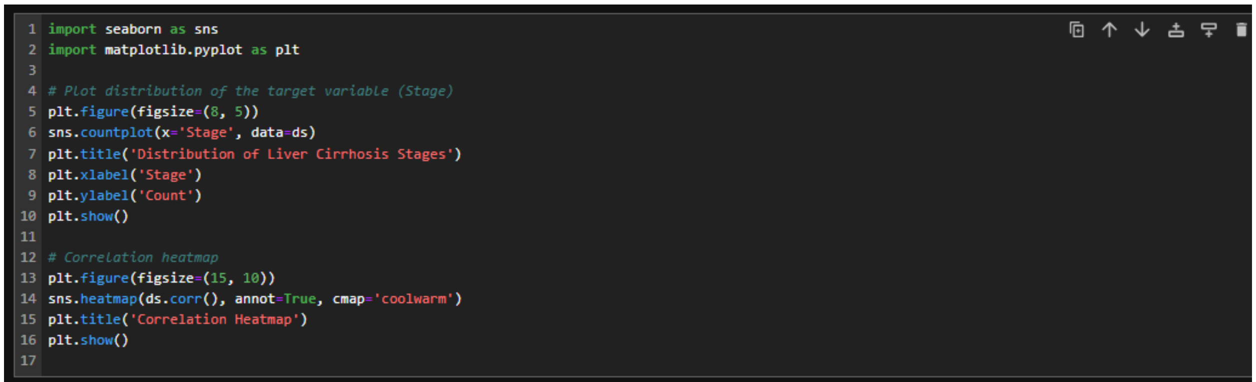
Output:



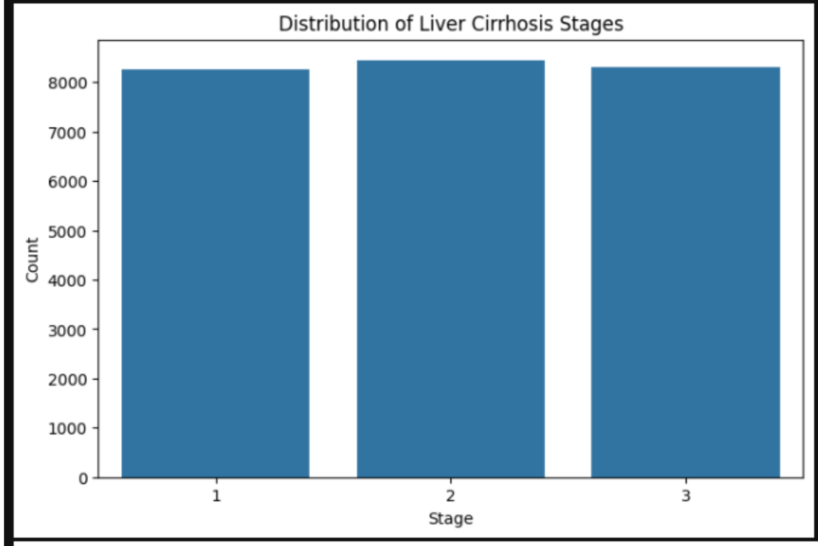
```
Missing Values:
 N_Days          0
Status          0
Drug            0
Age             0
Sex             0
Ascites         0
Hepatomegaly    0
Spiders         0
Edema           0
Bilirubin       0
Cholesterol     0
Albumin         0
Copper          0
Alk_Phos        0
SGOT            0
Tryglicerides   0
Platelets       0
Prothrombin     0
Stage           0
dtype: int64
```

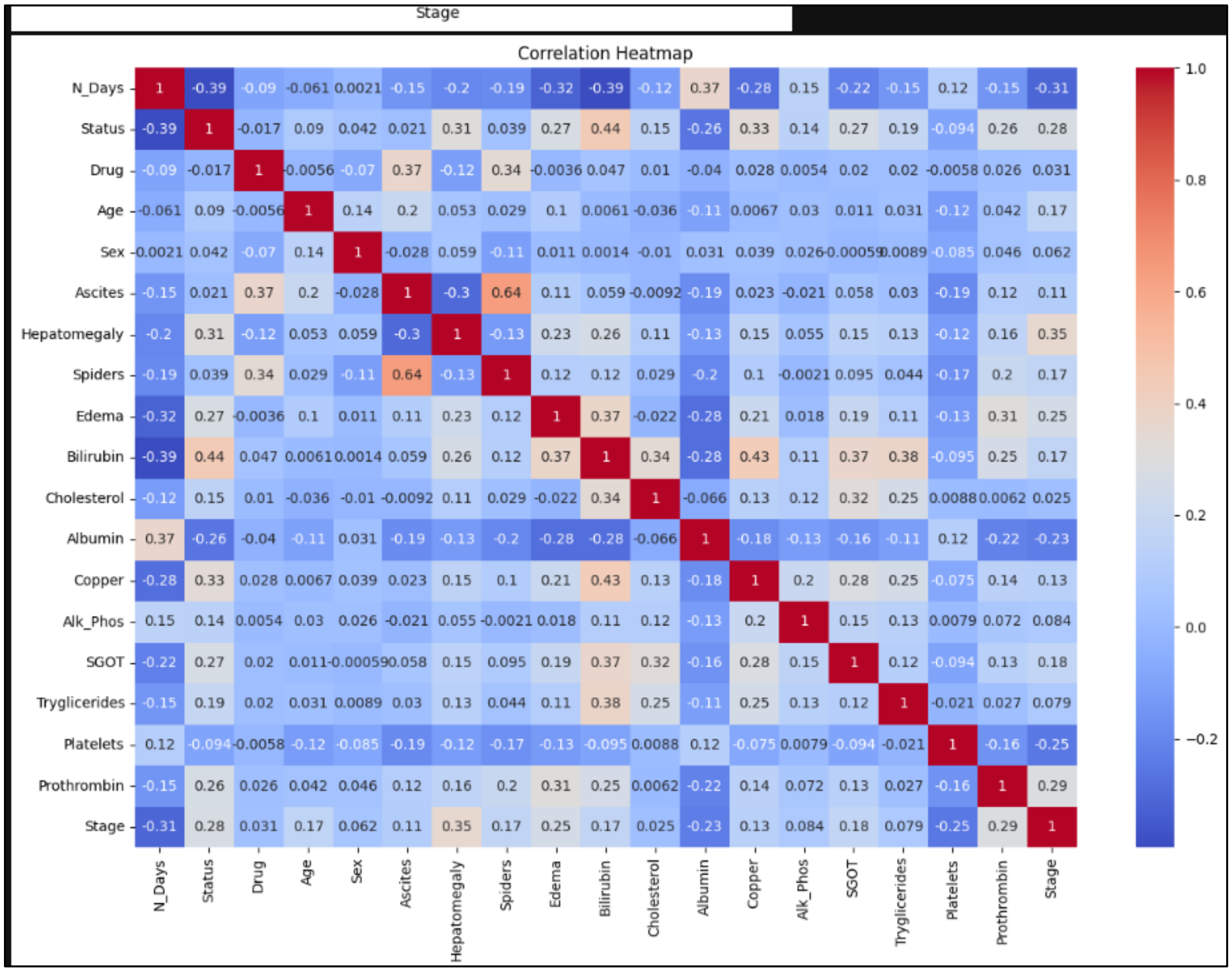
3. Exploratory Data Analysis (EDA)

Data visualization helps me view distributions together with the correlational patterns.



```python
import seaborn as sns
import matplotlib.pyplot as plt

# Plot distribution of the target variable (Stage)
plt.figure(figsize=(8, 5))
sns.countplot(x='Stage', data=ds)
plt.title('Distribution of Liver Cirrhosis Stages')
plt.xlabel('Stage')
plt.ylabel('Count')
plt.show()

# Correlation heatmap
plt.figure(figsize=(15, 10))
sns.heatmap(ds.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

```





	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
N_Days	1	-0.39	-0.09	-0.061	0.0021	-0.15	-0.2	-0.19	-0.32	-0.39	-0.12	0.37	-0.28	0.15	-0.22	-0.15	0.12	-0.15	-0.31
Status	-0.39	1	-0.017	0.09	0.042	0.021	0.31	0.039	0.27	0.44	0.15	-0.26	0.33	0.14	0.27	0.19	-0.094	0.26	0.28
Drug	-0.09	-0.017	1	-0.0056	-0.07	0.37	-0.12	0.34	-0.0036	0.047	0.01	-0.04	0.028	0.0054	0.02	0.02	-0.0058	0.026	0.031
Age	-0.061	0.09	-0.0056	1	0.14	0.2	0.053	0.029	0.1	0.0061	-0.036	-0.11	0.0067	0.03	0.011	0.031	-0.12	0.042	0.17
Sex	-0.0021	0.042	-0.07	0.14	1	-0.028	0.059	-0.11	0.011	0.0014	-0.01	0.031	0.039	0.026	-0.00059	0.0089	-0.085	0.046	0.062
Ascites	-0.15	0.021	0.37	0.2	-0.028	1	-0.3	0.64	0.11	0.059	-0.0092	-0.19	0.023	-0.021	0.058	0.03	-0.19	0.12	0.11
Hepatomegaly	-0.2	0.31	-0.12	0.053	0.059	-0.3	1	-0.13	0.23	0.26	0.11	-0.13	0.15	0.055	0.15	0.13	-0.12	0.16	0.35
Spiders	-0.19	0.039	0.34	0.029	-0.11	0.64	-0.13	1	0.12	0.12	0.029	-0.2	0.1	-0.0021	0.095	0.044	-0.17	0.2	0.17
Edema	-0.32	0.27	-0.0036	0.1	0.011	0.11	0.23	0.12	1	0.37	-0.022	-0.28	0.21	0.018	0.19	0.11	-0.13	0.31	0.25
Bilirubin	-0.39	0.44	0.047	0.0061	0.0014	0.059	0.26	0.12	0.37	1	0.34	-0.28	0.43	0.11	0.37	0.38	-0.095	0.25	0.17
Cholesterol	-0.12	0.15	0.01	-0.036	-0.01	-0.0092	0.11	0.029	-0.022	0.34	1	-0.066	0.13	0.12	0.32	0.25	0.0088	0.0062	0.025
Albumin	0.37	-0.26	-0.04	-0.11	0.031	-0.19	-0.13	-0.2	-0.28	-0.28	-0.066	1	-0.18	-0.13	-0.16	-0.11	0.12	-0.22	-0.23
Copper	-0.28	0.33	0.028	0.0067	0.039	0.023	0.15	0.1	0.21	0.43	0.13	-0.18	1	0.2	0.28	0.25	-0.075	0.14	0.13
Alk_Phos	0.15	0.14	0.0054	0.03	0.026	-0.021	0.055	-0.0021	0.018	0.11	0.12	-0.13	0.2	1	0.15	0.13	0.0079	0.072	0.084
SGOT	-0.22	0.27	0.02	0.011	-0.00059	0.058	0.15	0.095	0.19	0.37	0.32	-0.16	0.28	0.15	1	0.12	-0.094	0.13	0.18
Tryglicerides	-0.15	0.19	0.02	0.031	0.0089	0.03	0.13	0.044	0.11	0.38	0.25	-0.11	0.25	0.13	0.12	1	-0.021	0.027	0.079
Platelets	0.12	-0.094	-0.0058	-0.12	-0.085	-0.19	-0.12	-0.17	-0.13	-0.095	0.0088	0.12	-0.075	0.0079	-0.094	-0.021	1	-0.16	-0.25
Prothrombin	-0.15	0.26	0.026	0.042	0.046	0.12	0.16	0.2	0.31	0.25	0.0062	-0.22	0.14	0.072	0.13	0.027	-0.16	1	0.29
Stage	-0.31	0.28	0.031	0.17	0.062	0.11	0.35	0.17	0.25	0.17	0.025	-0.23	0.13	0.084	0.18	0.079	-0.25	0.29	1

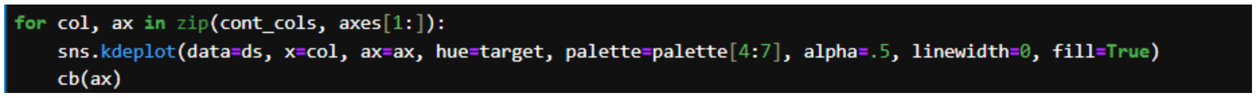
Distribution of Continuous Variable:

The visualization of continuous variables distributions enables me to learn about the feature behavior patterns across multiple liver damage classes. Understandable predictions of liver cirrhosis levels depend on properly identifying the characteristics of feature distributions including their spread and central tendency and shape.

Code Explanation:

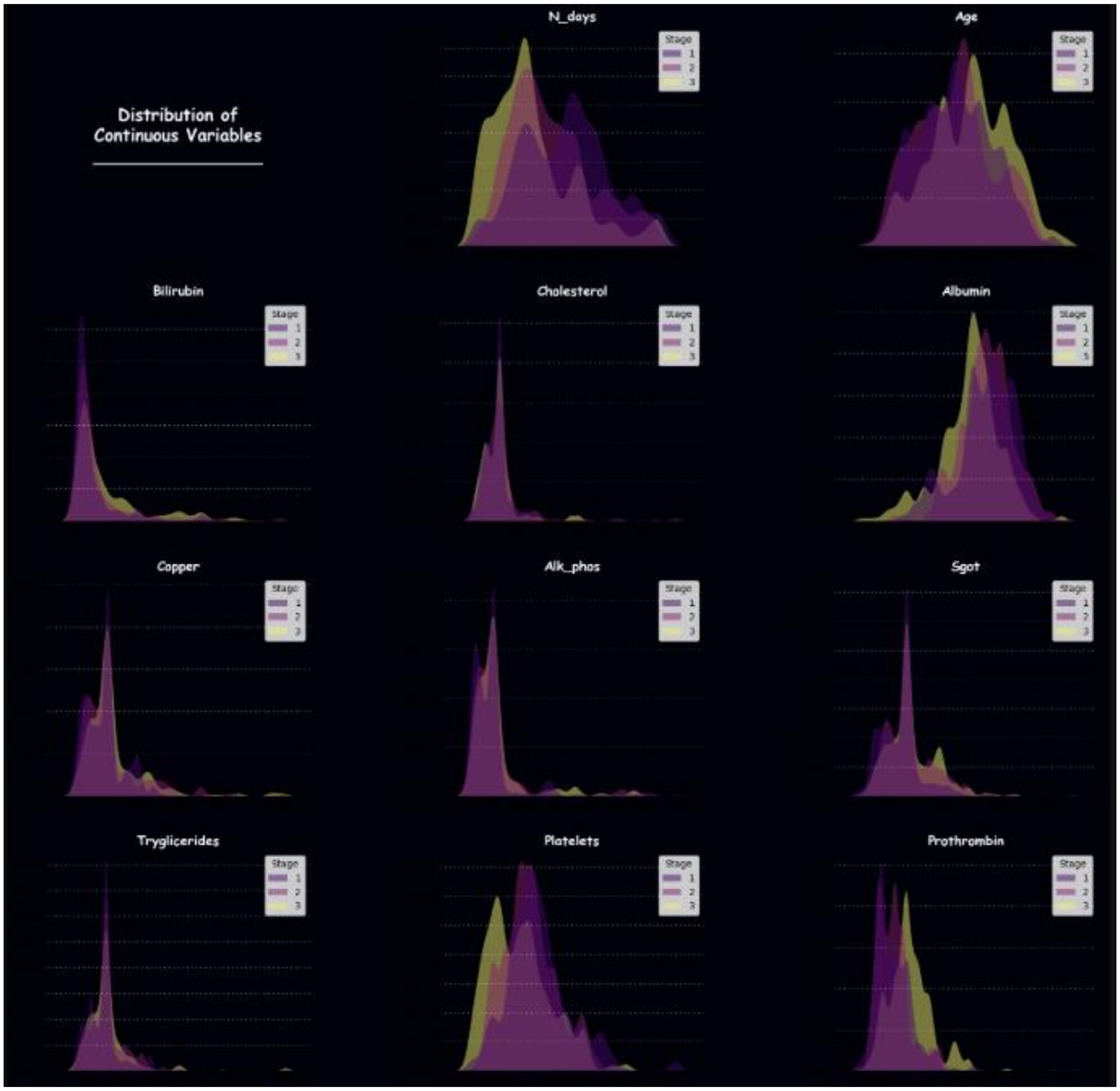
```
axes, palette, cb = mPlotter(4, 3, (20, 20), cont_cols, 'Distribution of\nContinuous Variables\n_______________')
```

- A multiple subplot grid appears through the **mPlotter** function call. The grid should have **4** rows and **3** columns with **(20, 20)** set as figure dimensions and the continuous columns should be plotted according to **cont_cols** alongside a title delivery.



```
for col, ax in zip(cont_cols, axes[1:]):
    sns.kdeplot(data=ds, x=col, ax=ax, hue=target, palette=palette[4:7], alpha=.5, linewidth=0, fill=True)
    cb(ax)
```

- The loop proceeds sequentially through each continuous column **(col)** in addition to its matching axis **(ax)** among the generated subplots.
- Each continuous variable generates a Kernel Density Estimate plot through **sns.kdeplot** from the Seaborn library.
  - **data=ds:** The dataset holds all variables that need to be plotted.
  - **x=col:** The graphical representation utilizes a continuous variable as its x-axis attribute.
  - **ax=ax:** The command defines the area that will show the plot visualization.
  - **hue=target:** KDE plots become readable through this parameter because it colors the areas according to target variable classes.
  - **palette=palette[4:7]:** The package selects a color range that will be employed for the visualization.
  - **alpha=.5:** The fill transparency is set to 50% through this parameter which enables better visualization of overlaying distributions.
  - **linewidth=0:** The line width of the KDE becomes invisible through this setting which results in a filled area.
  - **fill=True:** This parameter dictates the area under the KDE graph to be filled with color.



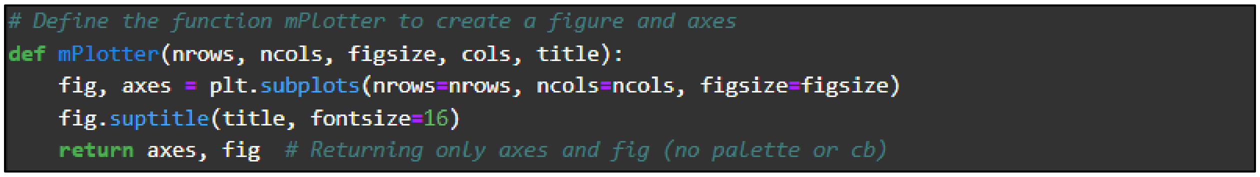
Insights

Examining distribution patterns of continuous variables through visualization reveals how different features react in different liver damage circumstances. The correct comprehension of liver cirrhosis prediction requires understanding how each feature affects the spread and central tendency and distribution shape of its initial data.

Distribution of Categorical Variables:

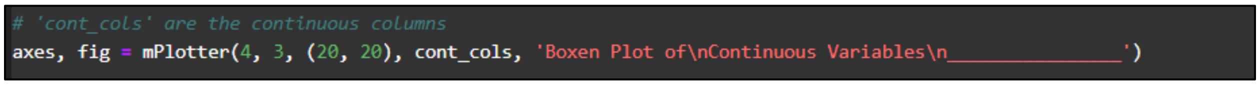
The section demonstrates the distribution of categorical variables through boxen plots in the dataset. Boxen plots serve as an improved version of box plots which show detailed distribution data particularly for large datasets containing various outliers. Boxen plots serve well to show the distribution of variables by category.

Code Explanation:



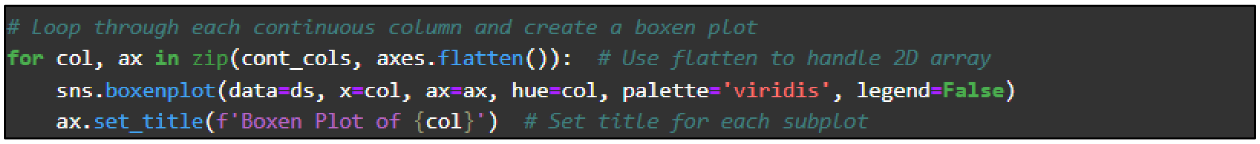
```python
# Define the function mPlotter to create a figure and axes
def mPlotter(nrows, ncols, figsize, cols, title):
    fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=figsize)
    fig.suptitle(title, fontsize=16)
    return axes, fig  # Returning only axes and fig (no palette or cb)
```

- The **mPlotter** function serves as a definition to build grid layouts of subplots. A function for subplots requires information about the number of rows and columns together with figure size and plot columns plus figure title. The function offers access to the axes and figure objects that users can modify.



```python
# 'cont_cols' are the continuous columns
axes, fig = mPlotter(4, 3, (20, 20), cont_cols, 'Boxen Plot of\nContinuous Variables\n______________')
```

- I use mPlotter to generate a 4x3 matrix of subplots which has a 20x20 pixel figure dimension. The plots carry the title "Boxen Plot of Continuous Variables".

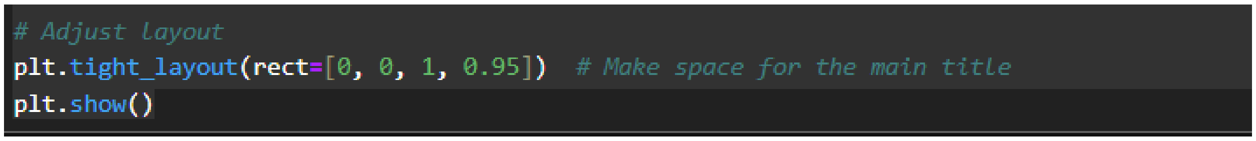


```python
# Loop through each continuous column and create a boxen plot
for col, ax in zip(cont_cols, axes.flatten()):  # Use flatten to handle 2D array
    sns.boxenplot(data=ds, x=col, ax=ax, hue=col, palette='viridis', legend=False)
    ax.set_title(f'Boxen Plot of {col}')  # Set title for each subplot
```

- The loop applies to each **continuous variable (col)** along with its associated axis **(ax).** In order to work easily with the nested axes structure the method **axes.flatten()** is provided.
- The boxen plot visualization of each continuous variable is generated through the **sns.boxenplot** function within the Seaborn library.
  - **data=ds:** The dataset containing the variables to be plotted.
  - **x=col:** The continuous variable being plotted on the x-axis.
  - **ax=ax:** Specifies the subplot where the plot will be drawn.

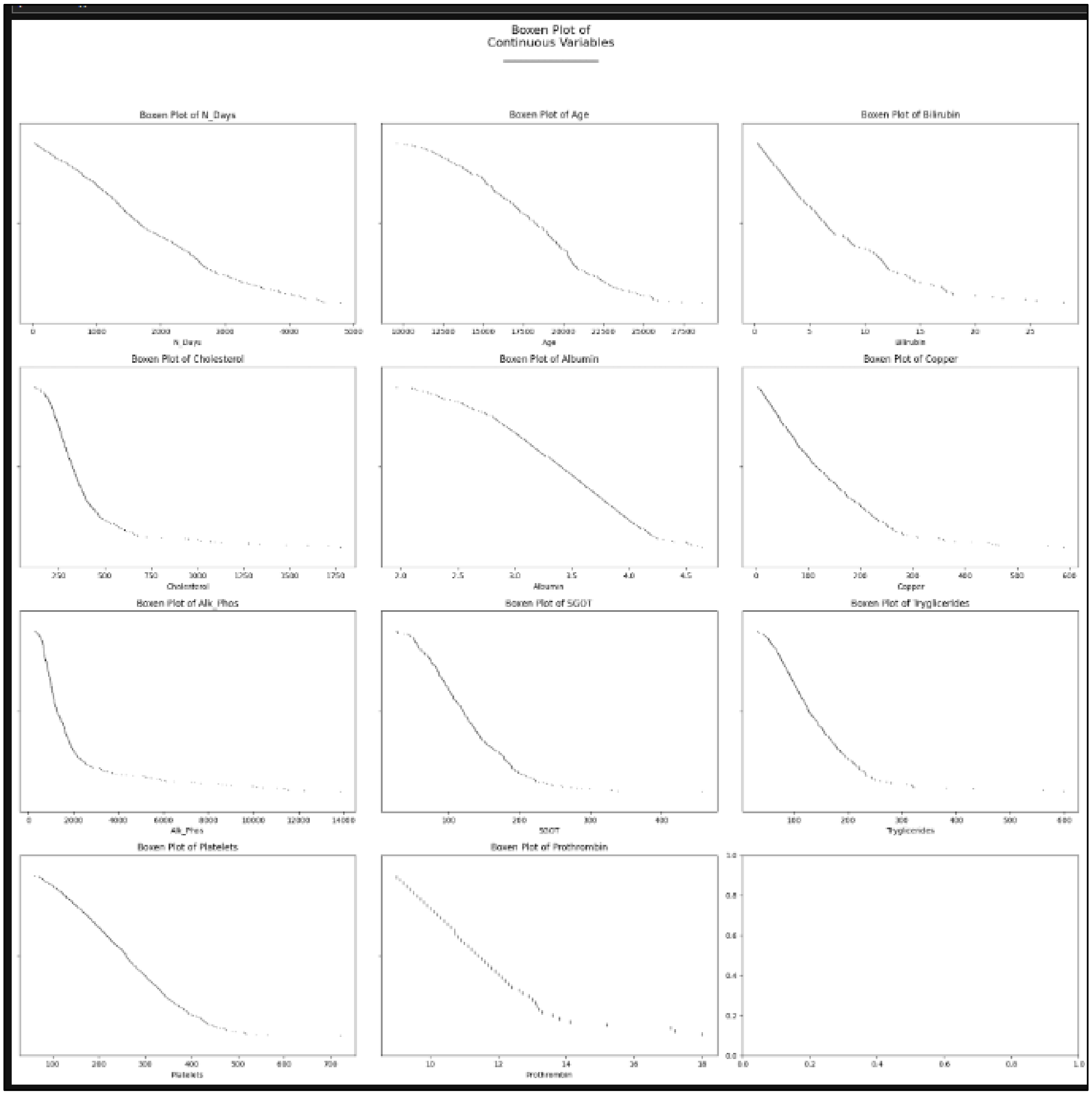
- **hue=col:** This parameter is used to color the boxen plots based on the variable being plotted, allowing us to see variations within each category.
- **palette='viridis':** This sets the color palette for the boxen plots.
- **legend=False:** This disables the legend for clarity.

- The boxen plot titles of each subplot are established through **ax.set_title(f'Boxen Plot of {col}')** to add contextual information about the variables.



```
# Adjust layout
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Make space for the main title
plt.show()
```

- Adjusting the plot spacing between subplots becomes possible through the **plt.tight_layout()** function which prevents labels and titles from getting cutoff. The **rect** parameter creates enough space for the main title placement.
- Finally, **plt.show()** displays the plots.

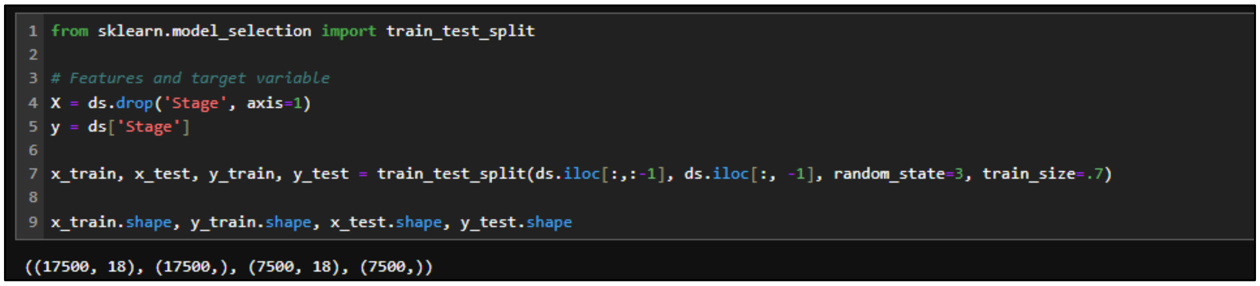


Insights

The boxen plot visualization helps researchers to assess the distribution characteristics of each continuous variable while analyzing categorical variables across different categories. The analysis serves as a fundamental tool for evaluating how different features relate to liver damage severity because it guides future modeling processes.

4. Train-Test Split

The data became separated into training and testing sections for evaluation purposes.

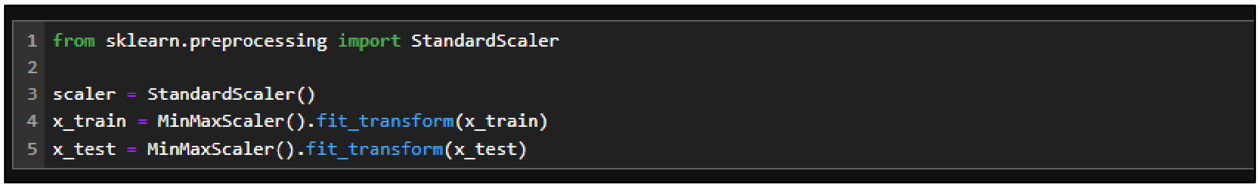


```
from sklearn.model_selection import train_test_split

# Features and target variable
X = ds.drop('Stage', axis=1)
y = ds['Stage']

x_train, x_test, y_train, y_test = train_test_split(ds.iloc[:,:-1], ds.iloc[:, -1], random_state=3, train_size=.7)

x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((17500, 18), (17500,), (7500, 18), (7500,))
```

5. Feature Scaling

I standardize the feature values.

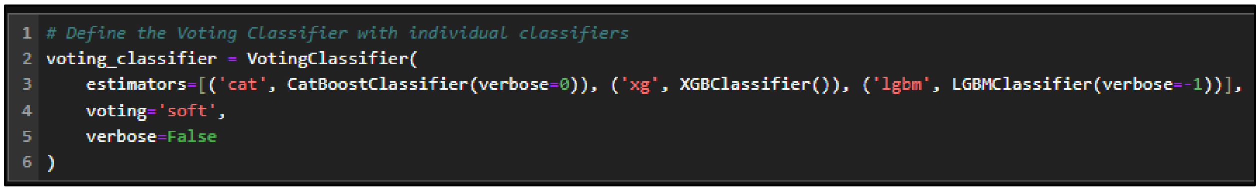


```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train = MinMaxScaler().fit_transform(x_train)
x_test = MinMaxScaler().fit_transform(x_test)
```

6. Model Training

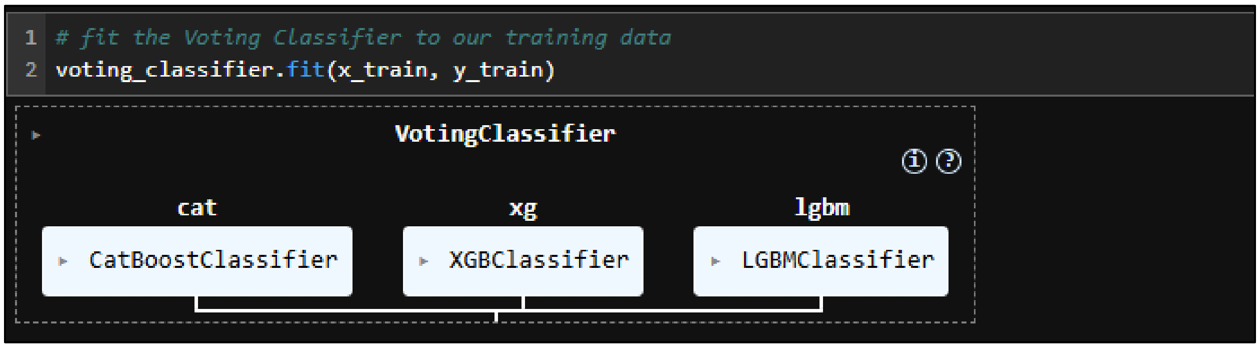
The prediction model uses a Voting Classifier which unites three powerful models: CatBoost together with XGBoost and LightGBM. Numerous strong predictive classifiers work together in this approach that maximizes the collective prediction precision.

I establish the Voting Classifier through this definition:



```
# Define the Voting Classifier with individual classifiers
voting_classifier = VotingClassifier(
    estimators=[('cat', CatBoostClassifier(verbose=0)), ('xg', XGBClassifier()), ('lgbm', LGBMClassifier(verbose=-1))],
    voting='soft',
    verbose=False
)
```

I apply the Voting Classifier model to fit my training data for the following process:

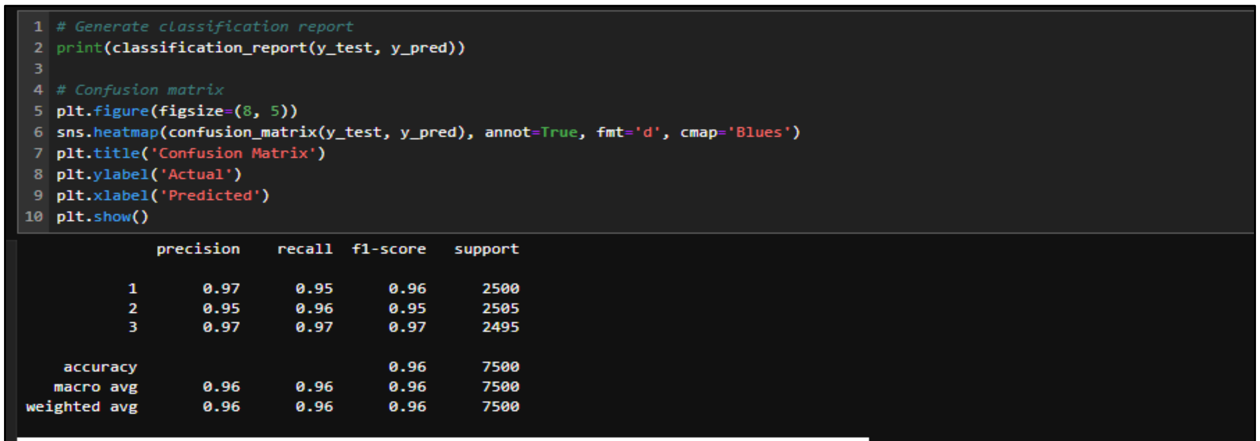


```
# fit the Voting Classifier to our training data
voting_classifier.fit(x_train, y_train)
```

Once the model is trained, I can check its accuracy by testing it on the test data and make appropriate predictions.

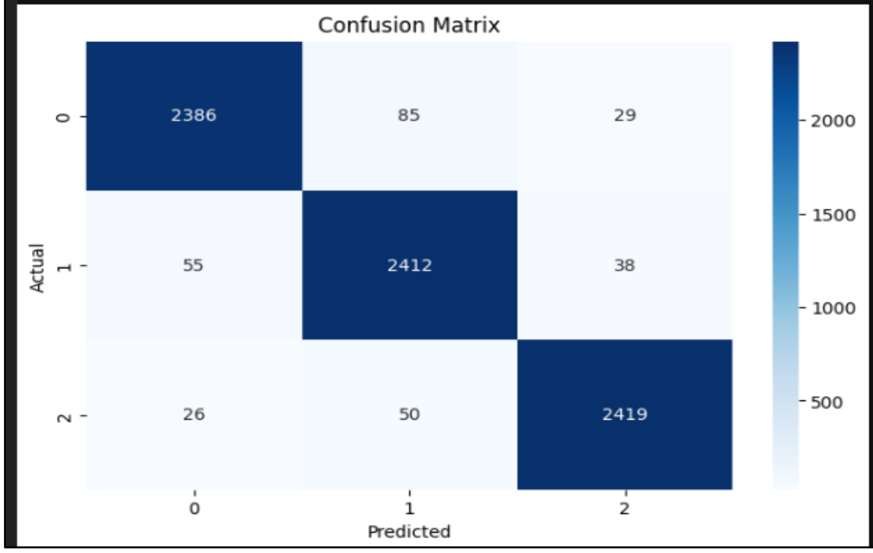
7. Model Evaluation

The model evaluation occurs through classification metrics analysis combined with confusion matrix assessment.



```
# Generate classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
plt.figure(figsize=(8, 5))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```

```
              precision    recall  f1-score   support

           1       0.97      0.95      0.96      2500
           2       0.95      0.96      0.95      2505
           3       0.97      0.97      0.97      2495

    accuracy                           0.96      7500
   macro avg       0.96      0.96      0.96      7500
weighted avg       0.96      0.96      0.96      7500
```



Conclusion

A machine learning model successfully predicts liver damage level according to the results from this project. The Random Forest Classifier functioned as a reliable choice because of its resistance to linear data patterns and non-linear nature of analysis. More model performance can be achieved through both additional hyperparameter optimization and cross-validation approaches.

Future Work

1. **Hyperparameter Tuning**: Model parameters should be optimized by applying methods like Grid Search within the field of hyperparameter tuning.
2. **Cross-Validation:** Using cross-validation will help the model achieve good generalizability.
3. **Model Deployment**: Building a web application for user access requires the deployment of the model through frameworks such as Flask or FastAPI.

References

- Mayo Clinic Study on Primary Biliary Cirrhosis. (n.d.). *Mayo Clinic*. Available at: https://www.mayoclinic.org/ (Accessed: [2/10/2025]).
- Pandas Documentation. (n.d.). *Pandas Documentation*. Available at: https://pandas.pydata.org/pandas-docs/stable/ (Accessed: [2/10/2025]).
- Scikit-Learn Documentation. (n.d.). *Scikit-Learn Documentation*. Available at: https://scikit-learn.org/stable/documentation.html (Accessed: [2/10/2025]).

==