

Applied AI and data mining

Project proposal: In this section, I will highlight all the key aspects related to my project and the dataset.

Problem Statement: The aim of this project is to analyze a dataset related to Cirrhosis, which is scarring (fibrosis) of the liver caused by long-term liver damage, (NHS, 2020) and develop a model for predicting the survival state of patients with liver cirrhosis on various clinical and demographic factors. This project aims to provide accurate and comprehensive solutions for medical professionals in the early detection, diagnosis and treatment of Cirrhosis.

Dataset Description: Dataset used for this project is “**cirrhosis_data.csv**” file, which contains all the relevant information about patients listed below:

ID	Patient ID
N_DAYS	Number of days between registration and the last consultation
Status	status of the patient C (censored), CL (censored due to liver tx), or D (death) (class label)
Drug	Drug used for treatment
Age	Age in days
Sex	Male or female
Ascites	Presence or absence of ascites
Hepatomegaly	Presence or absence of an enlarged liver
Spiders	Presence or absence of spider angiomas
Edema	Presence or absence of edema
Bilirubin	Bilirubin level in mg/dl
Cholesterol	Cholesterol level in mg/dl
Albumin	Albumin level in gm/dl
Copper	Copper level in ug/day
Alk_Phos	Alkaline phosphatase level in U/liter
SGOT	Serum glutamic-oxaloacetic transaminase level in U/ml
Tryglicerides	Triglyceride level
Platelets	Platelet count in ml/1000
Prothrombin	Prothrombin time in seconds
Stage	Stage of the disease

This dataset meets all the requirements :

- Has a column 'Status' containing labels representing the classes (status of the patient C (censored), CL (censored due to liver tx), or D (death)) has at least 3 classes.
- Contains more than 4 features (19 features in total)
- Has 616 samples (rows in the dataset)
- Dataset has been taken from **UCI Repository**.

AI techniques used in this project to solve the problem of predicting the presence, severity, and survival outcome of Cirrhosis, I will use all the mentioned techniques in the assignment such as:-

1. **Multiclass classification:** SVM with kernel tuning and hyperparameter optimization, Multilayer perceptron with advanced architecture and regulation, Convolutional Neural Network.
2. **Clustering:** K-Means Clustering with elbow method, Hierarchical Clustering, DBSCAN, Gaussian Mixture Models (GMM), KMeans with Hyperparameter Tuning
3. **Advanced techniques:** Ensemble methods (eg: Data Augmentation with SMOTE, Gradient Boosting) and many more techniques.

The project will require considerable data preparation, feature engineering, model training, evaluation, and results interpretation. Missing values, categorical variables, numerical characteristics, and any data quality issues will be handled using appropriate procedures. The models' performance will be assessed using appropriate measures, such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve. In addition, the research will investigate the ethical and societal implications of employing AI algorithms to predict and manage cirrhosis, taking into account issues such as privacy, bias, and the potential impact on patient treatment. Strategies for reducing potential hazards and ensuring responsible AI implementation will be presented.

Data Cleaning and visualization: Below are the steps taken into consideration for the data cleaning task:

- 1. Data loading and Inspection:** In the beginning, I loaded the dataset of Cirrhosis using the pandas library and then thoroughly understood the data structure and identified any missing values.


Data Overview

```
[24]: data = pd.read_csv('cirrhosis_data.csv')
      data = data.drop('ID', axis=1)
      data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 19 columns):
#   Column              Non-Null Count  Dtype
---  -
0   N_Days              418 non-null   int64
1   Status              418 non-null   object
2   Drug                312 non-null   object
3   Age                 418 non-null   int64
4   Sex                 418 non-null   object
5   Ascites             312 non-null   object
6   Hepatomegaly        312 non-null   object
7   Spiders             312 non-null   object
8   Edema               418 non-null   object
9   Bilirubin           418 non-null   float64
10  Cholesterol          284 non-null   float64
11  Albumin              418 non-null   float64
12  Copper              310 non-null   float64
13  Alk_Phos            312 non-null   float64
14  SGOT                312 non-null   float64
15  Tryglicerides        282 non-null   float64
16  Platelets           407 non-null   float64
17  Prothrombin          416 non-null   float64
18  Stage               412 non-null   float64
dtypes: float64(10), int64(2), object(7)
memory usage: 62.2+ KB
```

(self)

Below is the statistical representation of data:

 Jupyter Jagpreet Singh_N1230672 Last Checkpoint: 6 hours ago

File Edit View Run Kernel Settings Help

 Code

data statistical description

```
[3]: # Get descriptive statistics
print(data.describe())
```

	N_Days	Age	Bilirubin	Cholesterol	Albumin	\
count	418.000000	418.000000	418.000000	284.000000	418.000000	
mean	1917.782297	18533.351675	3.220813	369.510563	3.497440	
std	1104.672992	3815.845055	4.407506	231.944545	0.424972	
min	41.000000	9598.000000	0.300000	120.000000	1.960000	
25%	1092.750000	15644.500000	0.800000	249.500000	3.242500	
50%	1730.000000	18628.000000	1.400000	309.500000	3.530000	
75%	2613.500000	21272.500000	3.400000	400.000000	3.770000	
max	4795.000000	28650.000000	28.000000	1775.000000	4.640000	

	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	\
count	310.000000	312.000000	312.000000	282.000000	407.000000	
mean	97.648387	1982.655769	122.556346	124.702128	257.024570	
std	85.613920	2140.388824	56.699525	65.148639	98.325585	
min	4.000000	289.000000	26.350000	33.000000	62.000000	
25%	41.250000	871.500000	80.600000	84.250000	188.500000	
50%	73.000000	1259.000000	114.700000	108.000000	251.000000	
75%	123.000000	1980.000000	151.900000	151.000000	318.000000	
max	588.000000	13862.400000	457.250000	598.000000	721.000000	

	Prothrombin	Stage
count	416.000000	412.000000
mean	10.731731	3.024272
std	1.022000	0.882042
min	9.000000	1.000000
25%	10.000000	2.000000
50%	10.600000	3.000000
75%	11.100000	4.000000
max	18.000000	4.000000

(self)

Checking the missing values

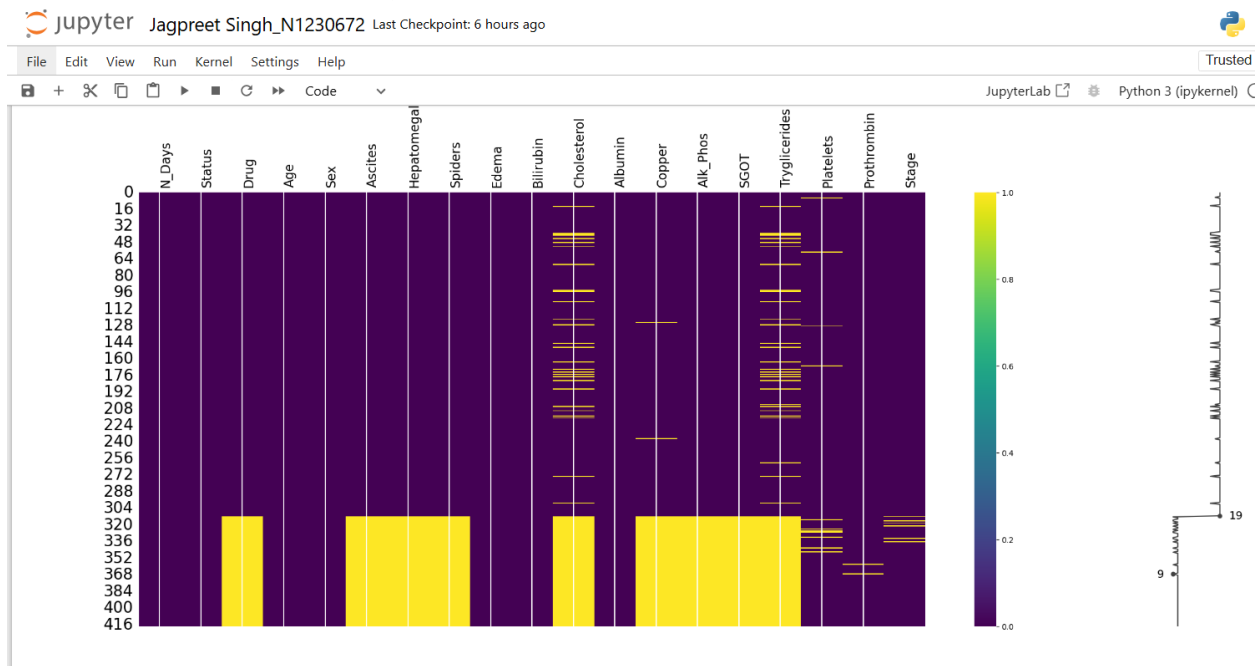
```
[4]: print(data.isnull().sum())
```

N_Days	0
Status	0
Drug	106
Age	0
Sex	0
Ascites	106
Hepatomegaly	106
Spiders	106
Edema	0
Bilirubin	0
Cholesterol	134
Albumin	0
Copper	108
Alk_Phos	106
SGOT	106
Tryglicerides	136
Platelets	11
Prothrombin	2
Stage	6

dtype: int64

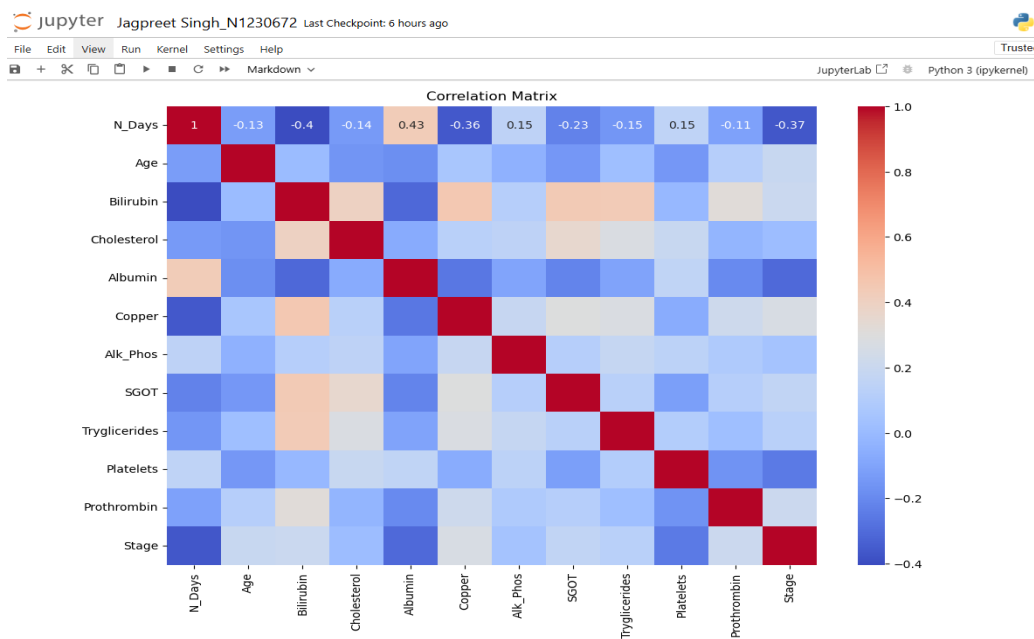
(self)

Visualization of missing number:

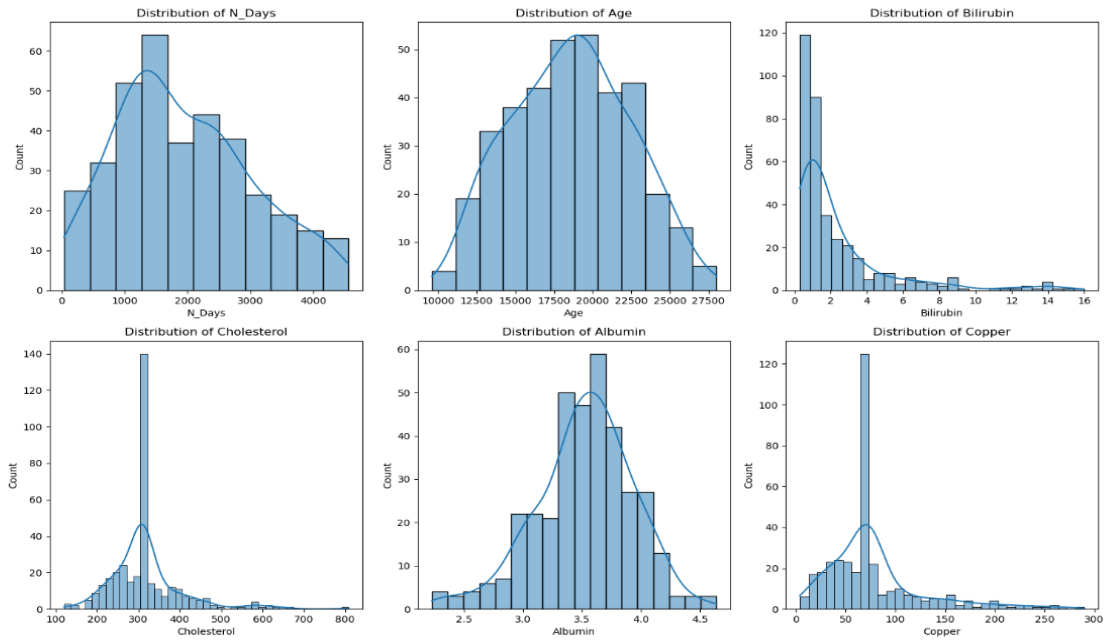


(self)

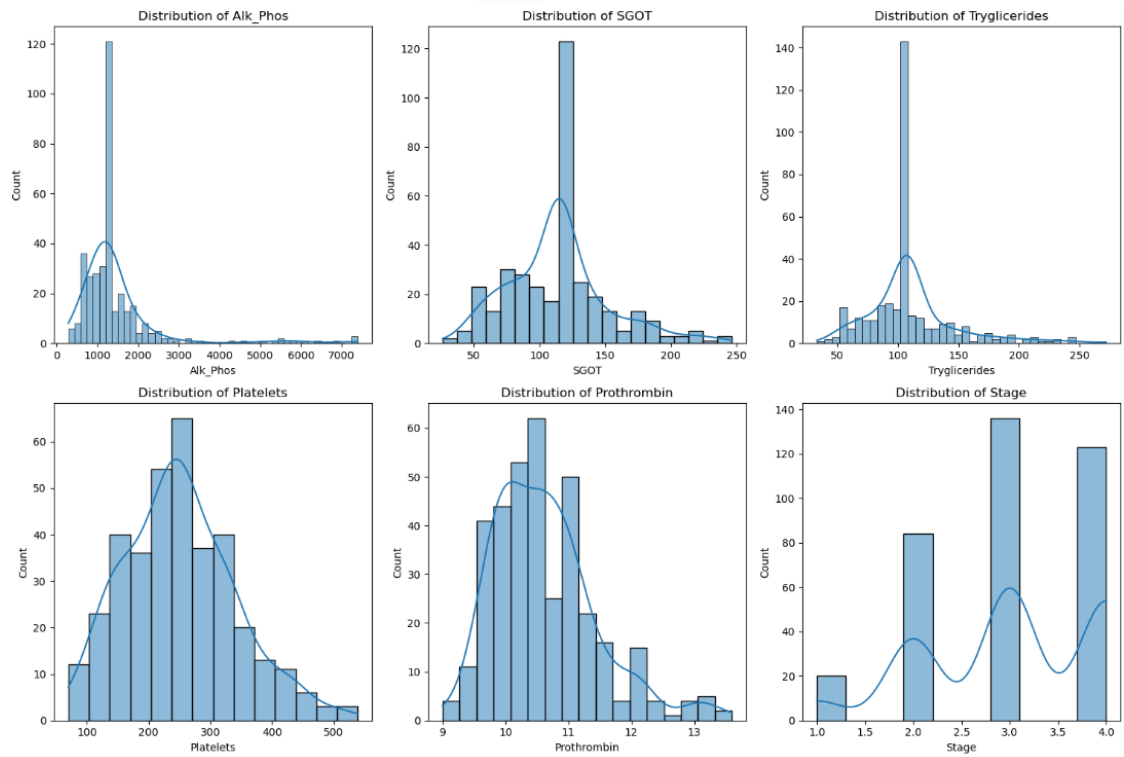
- Visual representation:** First I divided the data into numerical and categorical data and then processed visual representation for both.



(self)

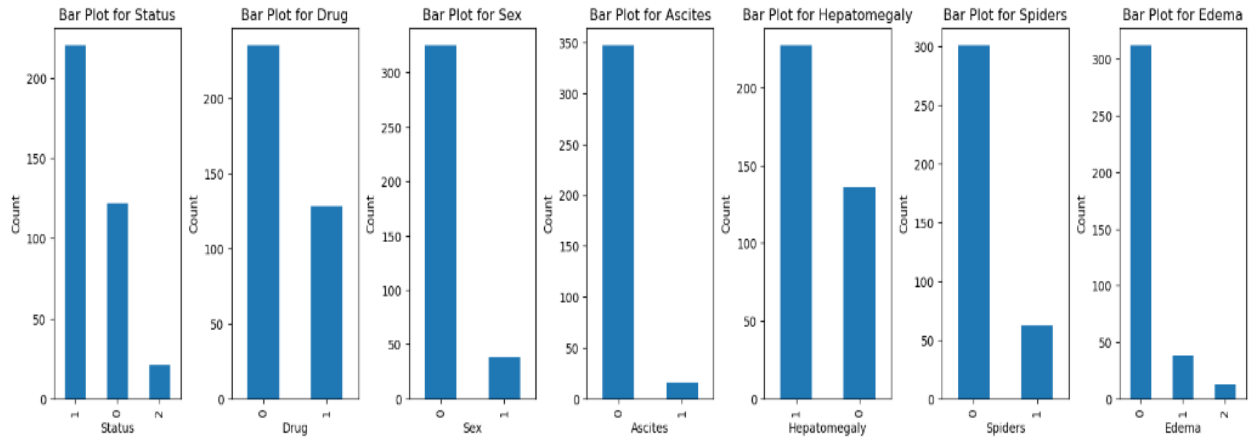


(self)



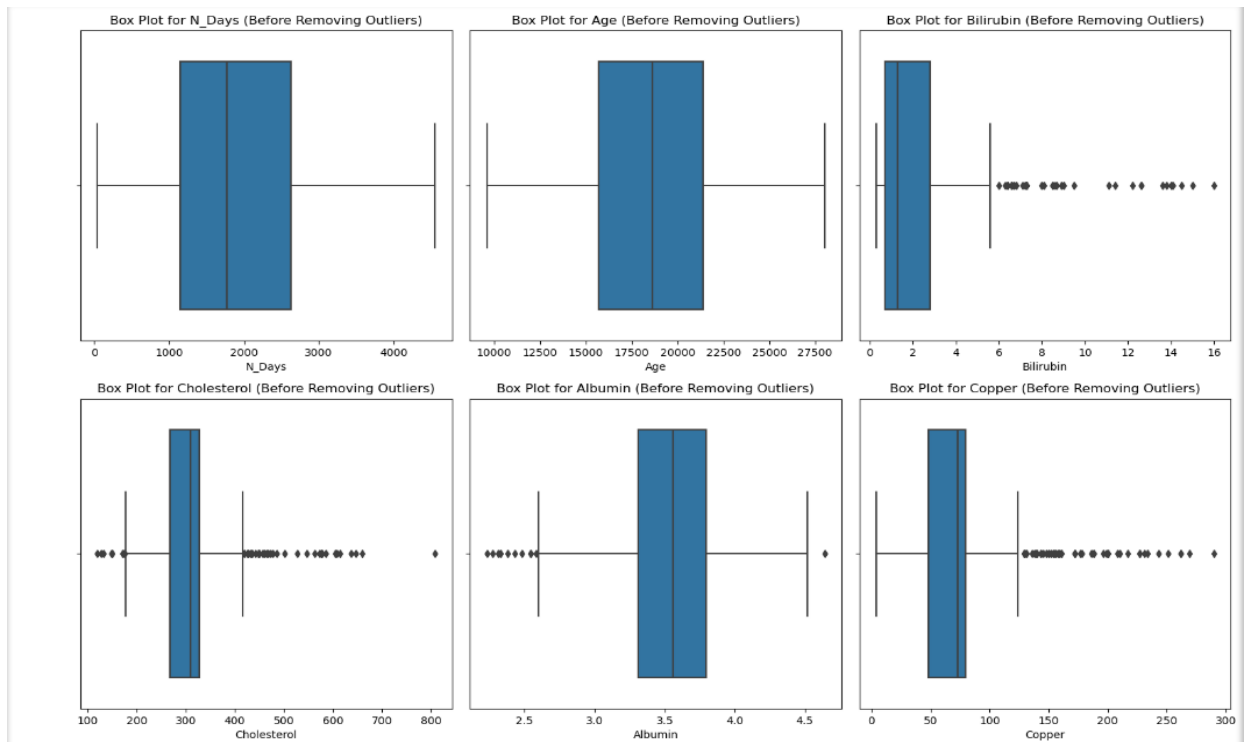
(self)

Visual representation for categorical data:

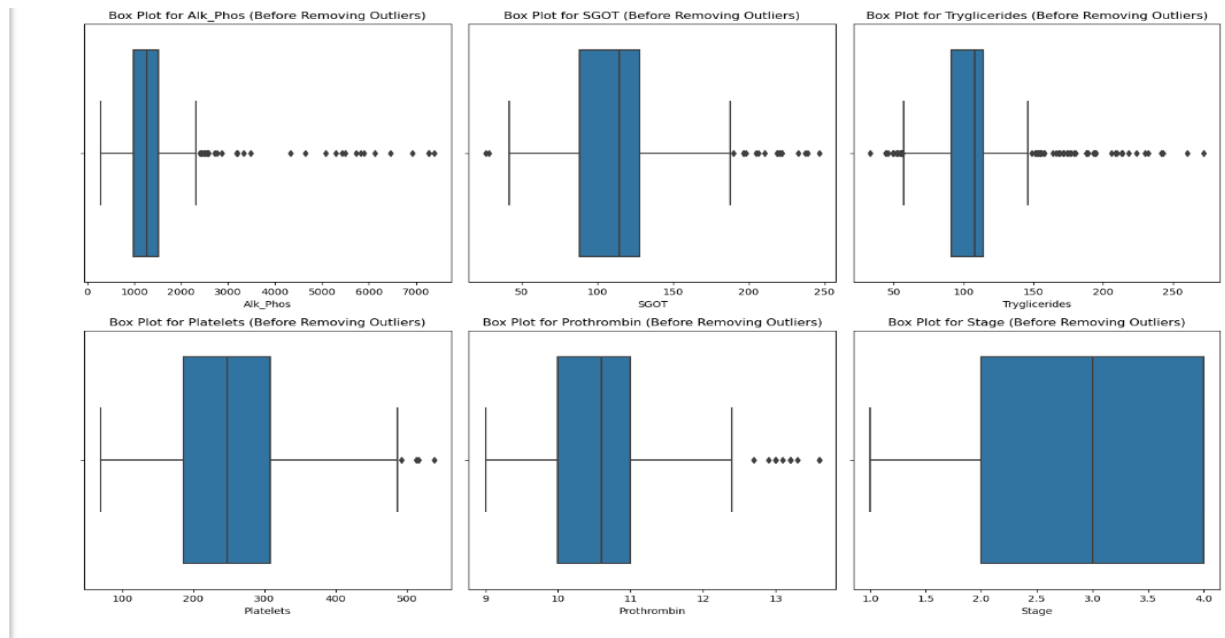


3. Handling Missing Values:

- Check for duplicates
- Plotted box plot before imputation to check the distribution, central tendency, and variability of data.



(self)



(self)

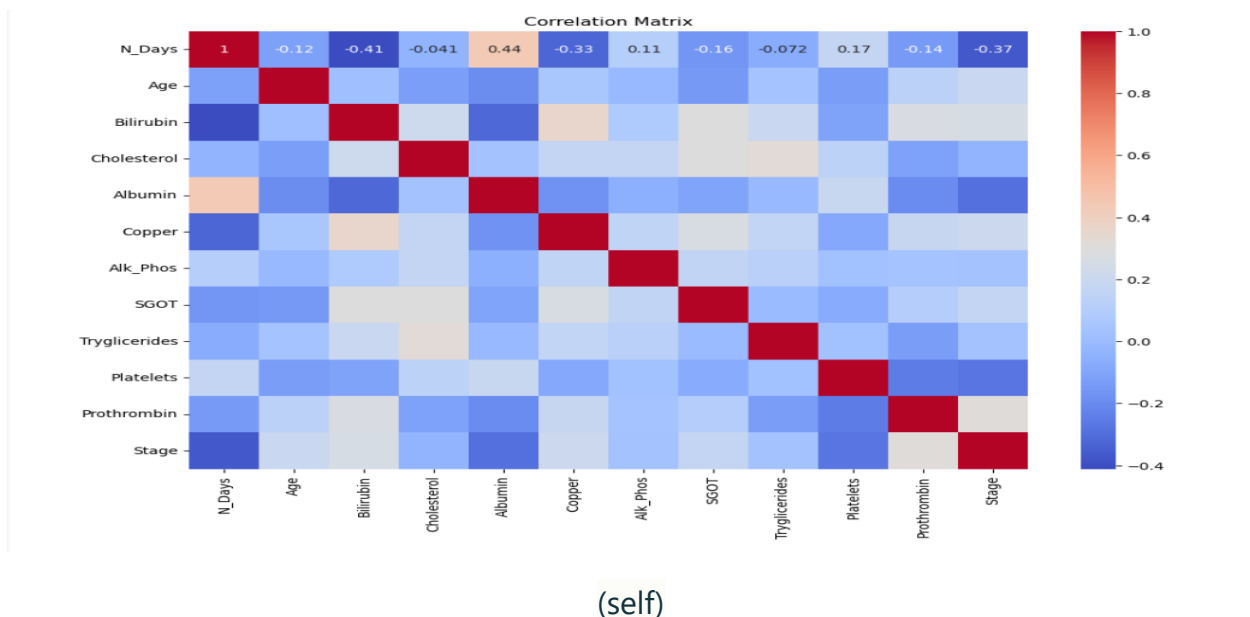
- c. Missing values were detected using the `isnull().sum()` method. Two strategies were employed to handle missing values:
 - d. **For categorical columns**, the **most frequent** value was imputed using the `SimpleImputer` from `Scikit-learn`, as most frequent value helps to preserve the distribution of the categorical variable in the data set and instead of losing a lot of data it is the best way to impute the categorical data with most frequent value.
 - e. **For numerical columns**, the **median** value was imputed using the same imputer strategy, as using the median value is more effective than the mean as median value is least effected with extreme values makes it better for central tendency in the presence of outliers.
4. **Duplicate Row Detection:** Duplicate rows were identified using the `duplicated()` function. If duplicate rows were found, they were displayed to alert the user for further action.

Data Preprocessing: This is the crucial step for the project after data cleaning now we will use several preprocessing techniques to make the data ready for further machine learning models.

- **Handling Categorical Variables:** Categorical variables were encoded using label encoding (`LabelEncoder`) to convert them into numerical format, enabling their

utilization in ML models, as this for using in the machine learning model it is must to convert all categorical data to numeric data

- **Handling Outliers:** Outliers were detected using z-scores, with a threshold of 3 standard deviations. Data points exceeding this threshold were considered outliers and subsequently removed from the dataset.
- **Correlation Analysis:** A correlation matrix was generated to identify relationships between numerical variables. This matrix provides insights into the strength and direction of correlations, aiding in feature selection and model interpretation.



- **Data Splitting:** The dataset was split into features and labels (dependent variable). This separation facilitates supervised learning tasks, with the dataset further split into training and testing sets using the `train_test_split()` function.

Implementation and Evaluation

TASK-1

In the analysis, a multiclass Support Vector Machine (SVM) was implemented using scikit-learn, with a focus on data preprocessing and model optimization techniques. The following steps were undertaken:

Data Preprocessing:

Step	Description
Feature Scaling	Utilize StandardScaler to maintain uniformity and mitigate scale-related issues.
Feature Selection	Employ Recursive Feature Elimination with Cross-Validation (RFECV) to select pertinent features.
Model Construction	Instantiate an SVM classifier with a linear kernel for classification tasks.
Pipeline Construction	Create a pipeline to sequentially apply feature scaling and SVM classification.
Hyperparameter Tuning	Utilize GridSearchCV for exhaustive hyperparameter tuning of the SVM model.
Model Evaluation	Train the best-performing SVM model obtained from GridSearchCV on the scaled training set and evaluate on the test set using accuracy as the metric.

```
selected Features for SVM: Index(['N_Days', 'Drug', 'Age', 'Sex', 'Ascites', 'Hepatomegaly', 'Edema',
                                'Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT',
                                'Tryglicerides', 'Platelets', 'Prothrombin', 'Stage'],
                                dtype='object')
Best SVM Accuracy on Test Set: 0.726027397260274
      precision    recall  f1-score   support

      0         0.84        0.55        0.67         29
      1         0.69        0.93        0.79         40
      2         0.00        0.00        0.00          4

 accuracy                   0.73         73
 macro avg              0.51        0.49        0.48         73
 weighted avg           0.71        0.73        0.70         73
```

(self)

Data Augmentation with SMOTE:

SMOTE Application: SMOTE was again employed to augment the minority class samples in the scaled training set, tackling class imbalance.

Model Training and Evaluation: An SVM classifier with a linear kernel was trained on the augmented training set and evaluated on the scaled test set to assess the impact of data augmentation on classification performance.

The **results** of the analysis demonstrated the effectiveness of SVM and ensemble methods in classifying the target variable.

SMOTE SVM Accuracy on Test Set: 0.5205479452054794

Ensemble Method: Gradient Boosting:

Data Resampling: The Synthetic Minority Over-sampling Technique (SMOTE) was utilized to address class imbalance by oversampling the minority class.

Feature Engineering: Polynomial features of degree 2 were generated to capture potential nonlinear relationships between features.

Model Training: A Gradient Boosting Classifier was trained on the resampled and engineered features to enhance predictive performance.

Gradient Boosting Accuracy on Test Set: 0.6438356164383562

Task2 - MLP

I had updated the Task 1's solution to utilize a multilayer perceptron (MLP) classifier with particular parameters: four hidden layers consisting of 25, 18, 10, and 5 neurons, respectively. However, we cannot use the kernels here in MLP as it is specific for kernel based models such as SVM, in MLP there is no need for kernel as the model learns from the input features to output directly through hidden layers.

Here is a thorough explanation of the **MLP model's parameters and training procedure:**

Parameters of MLP Model and Training Method:

Hidden Layers and Neurons: The MLP architecture includes four hidden layers, each comprising a different number of neurons. This design allows the model to learn hierarchical representations of the input data, capturing increasingly complex features at each layer ([Goodfellow et al., 2016](#)).

Activation Function: The Rectified Linear Unit (ReLU) activation function is used for all hidden layers. ReLU is chosen due to its simplicity, computational efficiency, and ability to alleviate the vanishing gradient problem during training by allowing the model to learn non-linear relationships in the data ([Chollet, 2017](#)).

Solver: The 'adam' optimizer is employed as the solver for training the MLP model. Adam is an adaptive learning rate optimization algorithm that efficiently adjusts the learning rates for each parameter during training, leading to faster convergence and improved performance ([Kingma & Ba, 2014](#)).

Results and Comparison:

After training the MLP model with the specified parameters, the accuracy of the resulting method is reported as 0.59 on the test set. This accuracy represents the model's ability to correctly classify instances in the test data.

To Increase the accuracy then I had used several different AI ML methods mentioned below:

- 1. MLP with Dropout Regularization :** This technique used to prevent overfitting in neural networks by randomly dropping neurons during training. his helps in reducing co-adaptation of neurons and encourages the network to learn more robust features. In the context of multilayer perceptron (MLP) models, dropout regularization can be applied by adding dropout layers after each hidden layer.

MLP regularization applied with below specification:

Architecture: The MLP model has 4 hidden layers with 25, 18, 10, and 5 neurons respectively, followed by a softmax output layer.

Activation Function: Rectified Linear Unit (ReLU) activation function was used for all hidden layers.

Solver: Adam optimizer was used as the solver for training the model.

Max Iterations: The model was trained for a maximum of 500 iterations.

Dropout Rate: Dropout layers with a dropout rate of 0.2 were added after each hidden layer.

```
3/3 ————— 0s 26ms/step  
MLP with Dropout Regularization Accuracy: 0.71
```

MLP with Augmentation and Learning Rate Decay: It is a technique used to artificially increase the size of the training dataset by applying various transformations to the existing data samples.

Used with below specification:

- **Architecture:** The same MLP architecture as before, with 4 hidden layers and ReLU activation.
- **Data Augmentation:** Synthetic Minority Over-sampling Technique (SMOTE) was used to augment the minority class samples in the training data, addressing the class imbalance.
- **Learning Rate Decay:** Learning rate scheduling was implemented using a learning rate scheduler, which gradually reduces the learning rate during training.

After training the model I got the accuracy which is shown below:

```
3/3 ————— 0s 25ms/step  
MLP with Augmentation and Learning Rate Decay Accuracy: 0.68
```

Justification of Parameters:

- **Depth and Width of the Network:** The chosen architecture with four hidden layers and specific numbers of neurons in each layer allows for capturing complex patterns in the data while avoiding overfitting.
- **Activation Function and Solver:** ReLU activation function and the 'adam' optimizer were selected based on their effectiveness in training deep neural networks and their widespread adoption in the deep learning community.

Comparison with Other Models:

Performance: Out of all the models, the MLP model with dropout regularization had the highest accuracy, closely followed by Task 1's SVM. This suggests that, when compared to more conventional machine learning models like SVM and Gradient

Boosting, the unique architecture and regularization strategies in MLP produced better classification accuracy.

Customization: Task 2 made it possible to modify the MLP design and investigate other setups in an effort to boost performance. The model's performance was further improved by the use of dropout regularization and data augmentation approaches.

Complexity: In general, MLP models are more complicated than more conventional machine learning models, such as Gradient Boosting and SVM. Better performance may result from increasing complexity, but it may also necessitate greater processing power and longer training periods.

In summary, the MLP classifier with the specified parameters offers a competitive performance compared to other models, demonstrating its potential for learning complex patterns in the data. Further optimization and fine-tuning may be explored to improve the accuracy and generalization of the MLP model.

TASK 3-Convolutional Neural Network (CNN) Model:

Model Architecture:

```
# Build CNN model
cnn_model = Sequential([
    Conv1D(32, kernel_size=3, padding='same', activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Conv1D(64, kernel_size=3, padding='same', activation='relu'),
    MaxPooling1D(pool_size=2),
    Conv1D(128, kernel_size=3, padding='same', activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(5, activation='softmax')
])
```

(self)

1. **Convolutional layers:** Conv1d is the first layer with 32 filters and kernel size of 3 and same padding. Since the training data is not a picture, the number of features in the data (X_train.shape[1]) and a single channel (1) determine the input shape.
2. **Max Pooling Layer 1:** It is the layer which reduces the spatial dimension of the feature map, I had used pool size of 3.

3. **Convolutional layer 2:** This is the second layer with same properties as the first layer.
4. **Max Pooling Layer 2:** Another max pooling layer with a pool size of 2.
5. **Convolutional Layer 3:** It is third convolutional layer with 128 filters, a kernel size of 3, and the 'same' padding.
6. **Flatten Layer:** This layer flattens the output of the convolutional layers into a 1D vector.
7. **Dense Layer 1:** This is a fully connected dense layer with 128 units and the ReLU activation function.
8. **Output Layer:** This is the output layer with the number of units equal to the number of unique classes in the target variable (y_train). The softmax activation function is used for multi-class classification.

Training Method

The CNN model is trained using the following steps:

The input data (X_train and X_test) is reshaped to match the expected input shape for the CNN model.

The target variable (y_train and y_test) is converted to integer type: `y_train = y_train.astype('int')` and `y_test = y_test.astype('int')`.

The CNN model is compiled with the Adam optimizer and sparse categorical cross-entropy loss: `cnn_model.compile`.

The model is trained using the fit method with the training data (X_train_cnn, y_train), validation data .

Below is the screenshot of the output for CNN

```

Epoch 39/50
10/10 ----- 0s 5ms/step - accuracy: 0.7211 - loss: 3.1239 - val_accuracy: 0.4521 - val_loss: 9.3181
Epoch 40/50
10/10 ----- 0s 6ms/step - accuracy: 0.5439 - loss: 7.4366 - val_accuracy: 0.5479 - val_loss: 18.1052
Epoch 41/50
10/10 ----- 0s 6ms/step - accuracy: 0.6303 - loss: 12.0739 - val_accuracy: 0.6575 - val_loss: 6.9231
Epoch 42/50
10/10 ----- 0s 6ms/step - accuracy: 0.6496 - loss: 5.5270 - val_accuracy: 0.6438 - val_loss: 5.5110
Epoch 43/50
10/10 ----- 0s 6ms/step - accuracy: 0.6218 - loss: 3.4144 - val_accuracy: 0.6301 - val_loss: 4.1943
Epoch 44/50
10/10 ----- 0s 7ms/step - accuracy: 0.6329 - loss: 4.7481 - val_accuracy: 0.5753 - val_loss: 2.4344
Epoch 45/50
10/10 ----- 0s 5ms/step - accuracy: 0.5963 - loss: 2.4397 - val_accuracy: 0.5753 - val_loss: 3.7151
Epoch 46/50
10/10 ----- 0s 5ms/step - accuracy: 0.6505 - loss: 2.6404 - val_accuracy: 0.3836 - val_loss: 14.6003
Epoch 47/50
10/10 ----- 0s 4ms/step - accuracy: 0.4100 - loss: 14.1352 - val_accuracy: 0.4110 - val_loss: 5.6211
Epoch 48/50
10/10 ----- 0s 4ms/step - accuracy: 0.5247 - loss: 5.6065 - val_accuracy: 0.3014 - val_loss: 7.4603
Epoch 49/50
10/10 ----- 0s 5ms/step - accuracy: 0.5838 - loss: 6.1686 - val_accuracy: 0.6849 - val_loss: 5.4566
Epoch 50/50
10/10 ----- 0s 4ms/step - accuracy: 0.6772 - loss: 4.8721 - val_accuracy: 0.5890 - val_loss: 6.1698
3/3 ----- 0s 36ms/step
CNN Classification Report:
      precision    recall  f1-score   support

     0       1.00      0.10      0.19         29
     1       0.59      1.00      0.74         40
     2       0.00      0.00      0.00          4

 accuracy
macro avg       0.53      0.37      0.31         73
weighted avg       0.72      0.59      0.48         73

CNN Confusion Matrix:
[[ 3 24  2]
 [ 0 40  0]
 [ 0  4  0]]
CNN Accuracy: 58.90%

```

(self)

Justification of Parameters:

Convolutional Layers: The model can learn ever more complicated features from the input data by employing three convolutional layers with progressively more filters (32, 64, and 128). According to [LeCun et al. \(1998\)](#), a popular choice for capturing local patterns in the data is the kernel size of 3.

Max Pooling Layers: The model is made more resilient to slight changes or distortions in the input data by using the max pooling layers to downsample the feature maps and add translation invariance ([Scherer et al., 2010](#)).

Activation Functions: In order for the model to learn complicated patterns, non-linearity must be introduced via the ReLU activation function in the convolutional and dense layers ([Hinton et al., 2012](#)). For multi-class classification, the output layer uses the softmax activation function ([Bridle, 1990](#)).

Dropout Regularization: To reduce overfitting and enhance generalization, dropout regularization can be applied to the thick layers of this CNN model, even if it is not specifically included in the model ([Srivastava et al., 2014](#)).

Optimizer and Loss Function: A popular and effective optimization approach for neural network training is the Adam optimizer ([Kingma & Ba, 2014](#)). For multi-class classification problems involving target variables that are represented using integers, the sparse categorical cross-entropy loss is appropriate ([Lin et al., 2017](#)).

Batch Size & Epochs: 32 is a popular batch size for neural network training because it strikes a compromise between convergence and computing efficiency ([Brownlee, 2018](#)). In order to allow the model to converge, 50 epochs were chosen; however, this number can be changed in accordance with the model's performance and convergence behavior ([Goodfellow et al., 2016](#)).

Effect of adjusting parameters:

Parameter	Description
Convolutional Layers	Additional layers capture more detailed characteristics but may increase training complexity.
Kernel Size	Larger sizes capture global patterns but slow down training and increase model complexity.
Activation Function	Different functions impact the model's ability to learn and generalize from the data.
Dropout Rate	Adjustments reduce overfitting but high rates may lead to underfitting.
Batch Size & Epochs	Alterations affect training time and convergence, with larger epochs risking overfitting.

Accuracy and Comparison:

Model	Accuracy
TASK1 SVM	72%
TASK1 SVM WITH GRADIENT BOOSTING	65%
TASK1 Data Augmentation with SMOTE	57%
TASK2 MLP	55%
TASK2 MLP DROPOUT	72%
TASK2 MLP with augmentation and Learning Rate Decay	68%
TASK3 CNN	58.90%

TASK4 - CLUSTERING

Clustering is a fundamental unsupervised learning technique used to discover inherent patterns and groupings within data. In this task, we apply the K-Means clustering algorithm to the dataset and evaluate the clustering performance.

NECESSARY STEPS BEFORE APPLYING CLUSTERING:

Feature Scaling: Before clustering, it's essential to scale the features to a similar range to prevent any single feature from dominating the distance calculations. Here, we employ standard scaling to normalize the features.

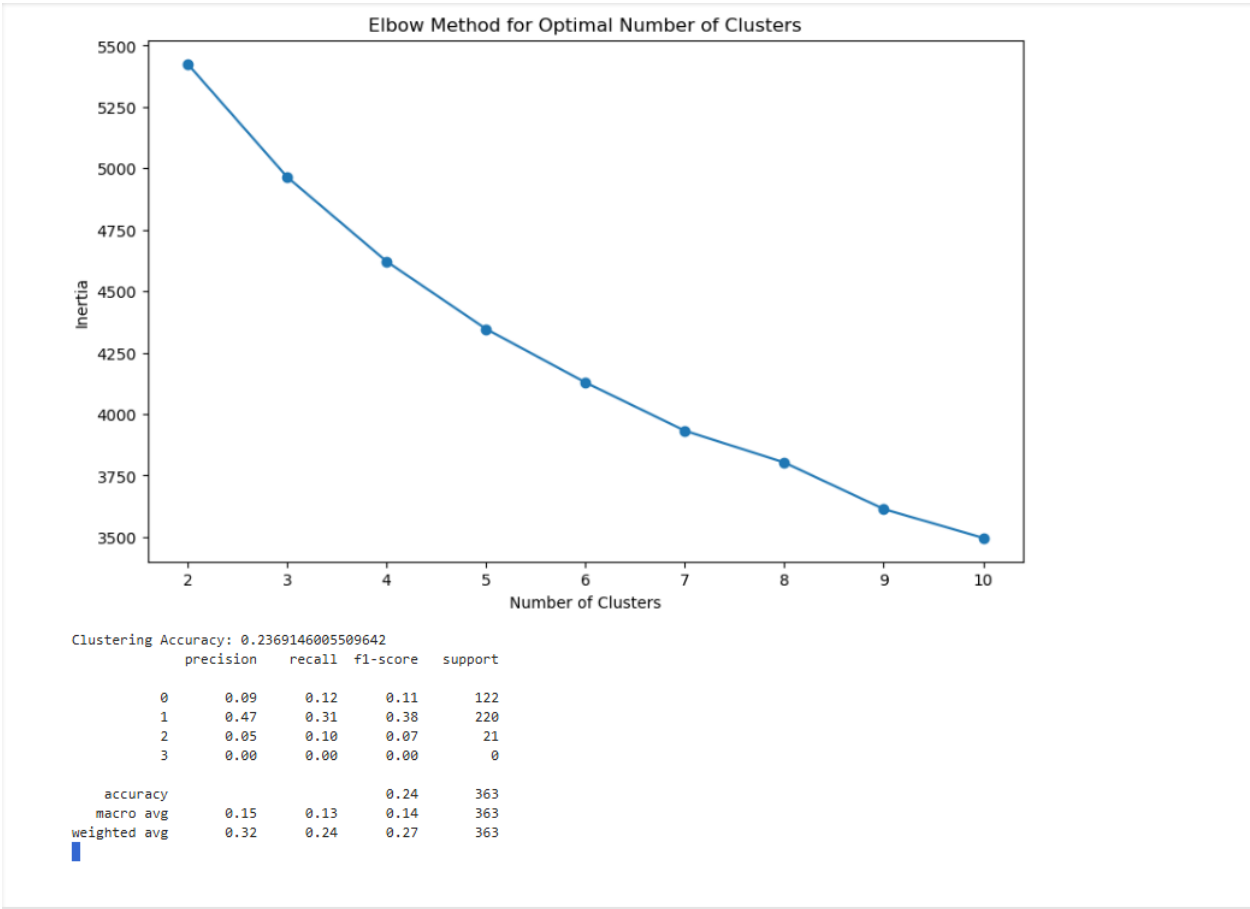
Dimensionality Reduction: As the dataset may contain high-dimensional features, dimensionality reduction techniques such as Principal Component Analysis (PCA) are applied to reduce the dataset's dimensionality while retaining most of the variance. PCA helps in visualizing the data in lower dimensions and can enhance clustering performance.

Finding the Ideal Number of Clusters: Effective clustering depends on selecting the ideal number of clusters. We use the elbow approach, which plots the within-cluster sum of squares (inertia) against the number of clusters, to ascertain this. The ideal number of clusters is indicated by the elbow point, which is the point at which the rate of inertia decreases more slowly.

K-Means Clustering: Based on the optimal number of clusters identified from the elbow plot, K-Means clustering is performed. K-Means is a centroid-based clustering algorithm that aims to partition the data into K clusters by minimizing the within-cluster sum of squares.

Evaluation of Clustering Performance: To evaluate the clustering performance, we compare the cluster labels assigned by K-Means to the true class labels. Metrics such as accuracy, precision, recall, and F1-score are calculated to assess how well the clusters correspond to the true classes.

Output:



(self)

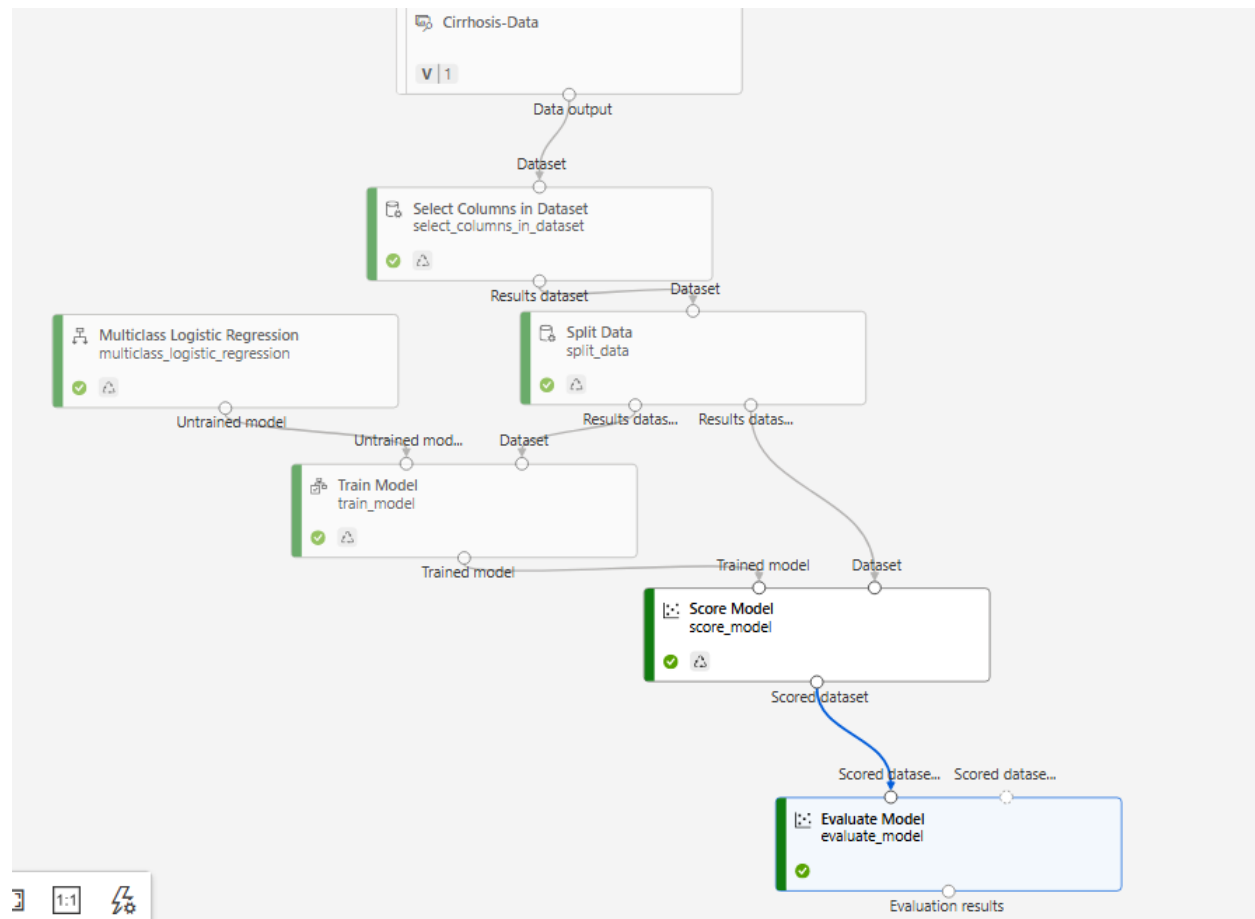
Advanced task

Other methods to improve the accuracy in TASK4

Clustering Method	Accuracy	Silhouette Score	Davies-Bouldin Score	Calinski-Harabasz Score
Hierarchical Clustering	0.23	0.10	2.31	34.67
DBSCAN	0.47	0.04	2.78	26.68
Gaussian Mixture Models (GMM)	0.38	0.11	2.09	39.16
KMeans with Hyperparameter Tuning	0.48	0.13	2.05	41.80

Used MS-Azure cloud based on my pre trained data:

Please find link of my MS-AZURE work - [ML-Cirrhosis - Azure Machine Learning](#)



Please find the evaluation matrix for the same:

Evaluation_results				
Rows ? Columns ?				
1 5				
Overall_Accuracy	Micro_Precision	Macro_Precision	Micro_Recall	Macro_Recall
0.779817	0.779817	0.524603	0.779817	0.510088

Ethical and Social Impact of AI Solutions: Artificial intelligence (AI) has revolutionized various industries, including healthcare, offering tremendous potential to improve patient outcomes, streamline processes, and enhance medical decision-making. However, the integration of AI in healthcare also raises significant ethical and social considerations that need to be carefully addressed.

Patient privacy and data integrity: In a world full of data there is an increase in hackers as well, so data safety is the foremost importance for any organization now. With massive AI mechanism and access to data there is a risk of unwanted access and data breach and misuse of data.

Fairness and Bias: AI systems are prone to bias, which can exacerbate already-existing inequalities in healthcare results. Certain demographic groups may be treated unfairly due to biases in training data, algorithmic decision-making, and model interpretations that disproportionately influence them. In order to discover and eliminate biases in AI-driven healthcare systems, addressing algorithmic bias necessitates transparent and accountable AI development procedures, varied and representative training data, and continuous monitoring.

Informed Consent and Autonomy: The use of AI technologies in clinical decision-making raises questions about patient autonomy and informed consent. Patients may not fully understand or have visibility into how AI algorithms influence their diagnosis, treatment recommendations, or access to care. Ensuring that patients are adequately informed about the use of AI in healthcare, including its limitations and potential risks, is essential to uphold patient autonomy and promote shared decision-making between patients and healthcare providers.

Accountability and Liability: Determining accountability and liability becomes difficult as AI systems get more independent and capable of making important judgments in healthcare settings. If there are mistakes, unfavorable results, or malfunctions with AI-driven treatment algorithms or diagnostic instruments, who bears the blame? Regulatory frameworks, procedures for recourse, and clear lines of accountability must all be established in order to handle the ethical and legal issues that arise from the use of AI in healthcare.

In conclusion, AI's ethical and societal ramifications cannot be disregarded, even while it has the potential to completely transform the way healthcare is delivered and enhance patient outcomes. In order to promote responsible and equitable AI use in healthcare and eventually advance the shared aim of providing high-quality, patient-centered care, it is

imperative that concerns like as patient privacy, bias mitigation, informed consent, responsibility, and professional adaptability be addressed.

References

Dataset - [Cirrhosis Patient Survival Prediction - UCI Machine Learning Repository](#)

NHS. (2020). Retrieved from NHS: <https://www.nhs.uk/conditions/cirrhosis/>

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique (Anon.).

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System- [XGBoost | Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining](#).

Chollet, F. (2017). Deep Learning with Python- [Deep Learning with Python, Second Edition - Francois Chollet - Google Books](#).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning - [Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books](#).

KINGMA, D.P. and LEI BA, J., 2017. *Published as a conference paper at ICLR 2015 ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python - [pedregosa11a.pdf \(jmlr.org\)](#).

LECUN, Y., BENGIO, Y. and HAFFNER, P., *Gradient-Based Learning Applied to Document Recognition*.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. International Conference on Artificial Neural Networks (pp. 92-101). Springer- [Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition | SpringerLink](#).

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv

preprint arXiv:1207.0580- [\[1207.0580\] Improving neural networks by preventing co-adaptation of feature detectors \(arxiv.org\)](#).

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing: Algorithms, Architectures, and Applications*, 40(1), 227-236 - [Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition | SpringerLink](#).

SRIVASTAVA, N., et al., *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*

Lin, H. T., Lin, C. J., & Weng, R. C. (2017). A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3), 267-276- [A note on Platt's probabilistic outputs for support vector machines | Machine Learning \(springer.com\)](#).

Brownlee, J. (2018). Deep learning for computer vision. *Machine Learning Mastery* - [Deep Learning for Time Series Forecasting: Predict the Future with MLPs ... - Jason Brownlee - Google Books](#).