NOTTINGHAM TRENT UNIVERSITY
DEPARTMENT of COMPUTER SCIENCE

Applied Artificial Intelligence COMP40511 Coursework

Application of AI in solving real-world problems

By

Gokila Sundaram

N1082709

Date: *31-May-2023*

Word Count: 2883 (Excluding References and Indexes)

Plagiarism Declaration:

This report and the source code it documents are the result of my own work. Any contributions to the work by third parties, other than tutors, are stated clearly below this declaration. Should this statement prove to be untrue I recognise the right and duty of the Board of Examiners to take appropriate action in line with the university's regulations on assessment.

Name: Gokila Sundaram ID No: N1082709

Table of contents

1. Project Proposal:

The analysis of healthcare data has been a significant field of study in recent years. This project focuses on leveraging the Cirrhosis liver disease prediction dataset available on Kaggle to predict the various stages of Cirrhosis liver disease. The primary objective is to develop a predictive model that can accurately identify and classify different stages of this condition.

Cirrhosis of the liver is a chronic disease that can have serious consequences, such as liver failure. Unfortunately, there is currently no cure for this disease, and for those in need of treatment, liver transplantation is often the only viable option (Cleveland Clinic 2020). However, the cost associated with transplantation is prohibitive for many individuals, making it inaccessible.

Given this situation, there is a critical need to develop an AI system that can detect and predict the different stages of Cirrhosis liver disease early on. This AI system would provide a more affordable and accessible alternative to liver transplantation, benefiting a wider population of individuals at risk of developing cirrhosis.

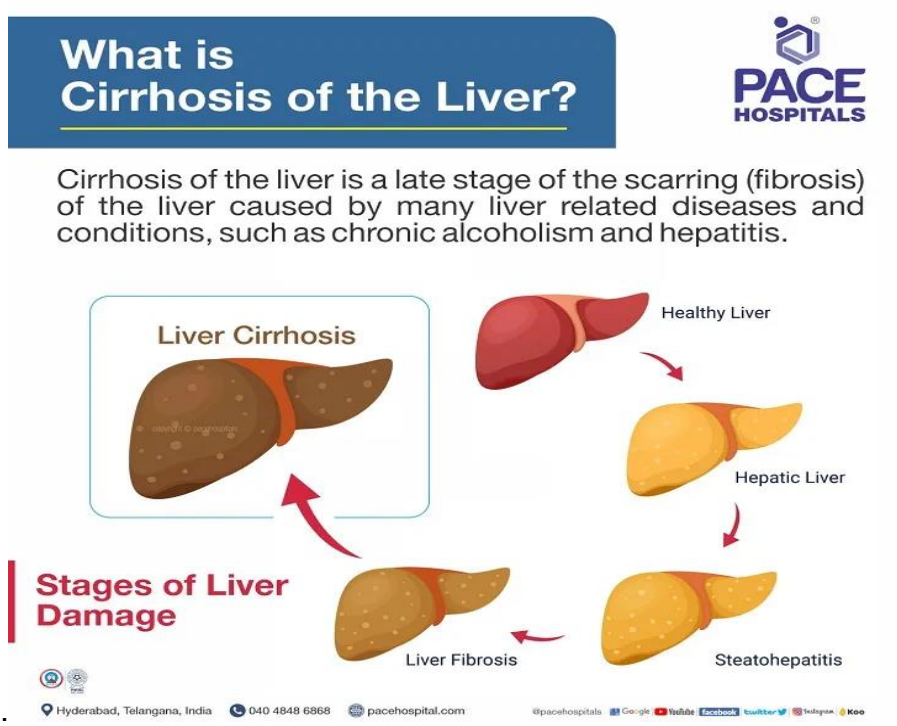


Figure 1: Stages of Liver Disease (Anon 2018)

DataSource: https://www.kaggle.com/datasets/fedesoriano/cirrhosis-prediction-dataset

The dataset comprises of patient information such as age, sex, and gender, along with clinical features including drug usage, ascites, hepatomegaly, bilirubin levels, and more. Further details regarding the attributes can be found on the source page.

The project aims to explore the various features provided in the dataset, understand their significance, and build a machine learning model that can accurately predict the likelihood of cirrhosis development.

1.1 Dataset description:

- The dataset has a dimension of (418 x 20) according to the data.info() function. It consists of variables of different types, including int64, float64, and Object (categorical) variables.

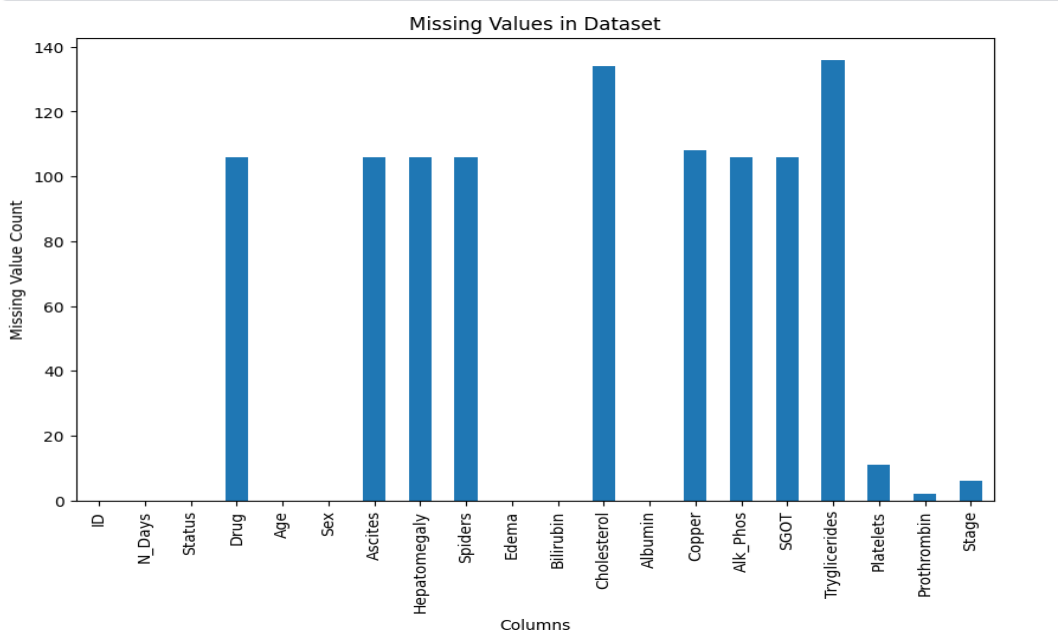
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   ID             418 non-null    int64  
 1   N_Days         418 non-null    int64  
 2   Status    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

- Target class has 4 stages:

```
# Number of Classes in Target Variable #
Target_Classfication= cirrhosis["Stage"].unique()
print(Target_Classfication)
```

```
[ 4.  3.  2.  1. nan]
```

- The total number of missing values in the dataset is shown by data.isna().sum(). Specifically, 12 features (including target) contain missing values represented as NaN.



- The number of complete cases, or non-null observations, can be determined using data.dropna().shape[0]. This reveals that there are 276 complete cases in the dataset.
- The number of incomplete cases, or non-null observations with missing values, can be calculated as data[data.isna().any(axis=1)].shape[0]. This indicates that there are 142 incomplete cases in the dataset.
- There are no duplicates present in the dataset.
- It is observed that the dataset has imbalanced observations across the different stages of the target class. This imbalance suggests that some stages may have significantly fewer instances compared to others.

```
# Valuecount for each target classification###
Target_Classfication_count= cirrhosis["Stage"].value_counts()
print(Target_Classfication_count)
nan_count = cirrhosis["Stage"].isna().sum()
print('Count of NaN:', nan_count)
# Data Distribution is not Balanced ##
```

```
3.0    155
4.0    144
2.0     92
1.0     21
Name: Stage, dtype: int64
Count of NaN: 6
```

- Dataset is not normaly distributed which is represeneted by data.describe() function. Mean & Median of numerical variables are significantly different.
- The output of **data.describe(include=['object'])** indicates an unequal population distribution for categorical variables.

```
cirrhosis.describe(include=['object'])
```

	Status	Drug	Sex	Ascites	Hepatomegaly	Spiders	Edema
count	418	312	418	312	312	312	418
unique	3	2	2	2	2	2	3
top	C	D-penicillamine	F	N	Y	N	N
freq	232	158	374	288	160	222	354

1.2 Problem Statement:

Developing an effective predictive models for Cirrhosis liver disease requires comprehensive analysis and understanding of the underlying factors and risk factors associated with its progression.

1. **Data Quality** : The dataset being used, is collected from a clinical trials. It has missing values & Outliers, which can pose challenges in analysis and modeling. Outliers and erroneous entries should be carefully examined and addressed appropriately, as they can impact the reliability and validity of the predictive models.

2. **Data Population**: The dataset is imbalanced, meaning that the number of observations for each stage of the disease is not evenly distributed. Imbalanced datasets can pose challenges in model training and evaluation. Predictive models trained on imbalanced data may exhibit bias towards the majority class, leading to poor performance in predicting the minority class.
3. **Data Distribution:** The distribution of data does not follow a normal distribution. It contains, numerical attributes that exhibit varying ranges. The variation in range among numerical attributes can impact data analysis and modelling.
4. **Multiple Features:** Dataset has multiple features including patient details. Irrelevant features can affect machine learning models in several ways including the accuracy and performance.
5. **Ethical Issues**: False positive results from predictive models can raise significant concerns in healthcare.

1.3 Proposal:

Addressing these challenges and developing an accurate predictive model for Cirrhosis liver disease will contribute to the field of healthcare by facilitating early detection, personalized treatment, and improved patient care.

1. Due to the medical nature of the dataset, Missing value imputation is not recommended. Data Analysis will be performed on the complete cases (non-null observations). Outliers will not be removed as it can potentially represent important information.

2. AI Techniques (Oversampling) will be applied on the dataset to have equal population of observations across target classes.
3. Feature scaling will be performed to have the values in the same range to prevent any single variable from dominating the model's learning process and ensures fair consideration of all features.
4. Feature Engineering technique (Feature Recrusive Elimination) will be utilized to select most relevant features for prediction.
5. Appropriate metrics will be employed to calculate the percentage of false positive outcome.

2. Data Pre-processing and Visualisation:

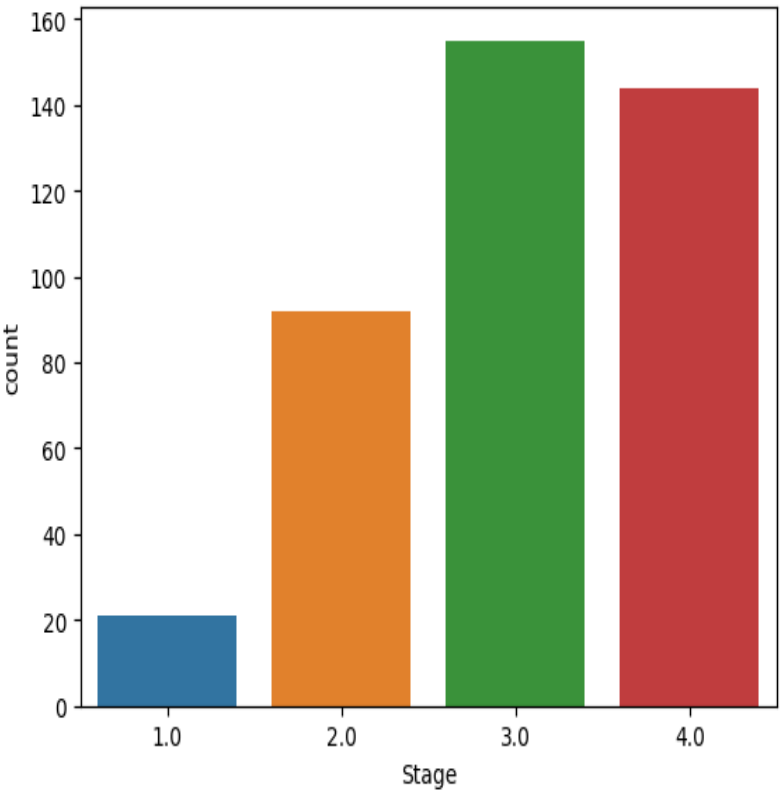
Kaggle is used in this project for analysis & machine learning tasks.

Pre-processing steps include list of activities to clean and transform the data. Import pandas and numpy libraries for loading & pre-processing the data. Import matplotlib.pyplot and seaborn libraries for visualization.

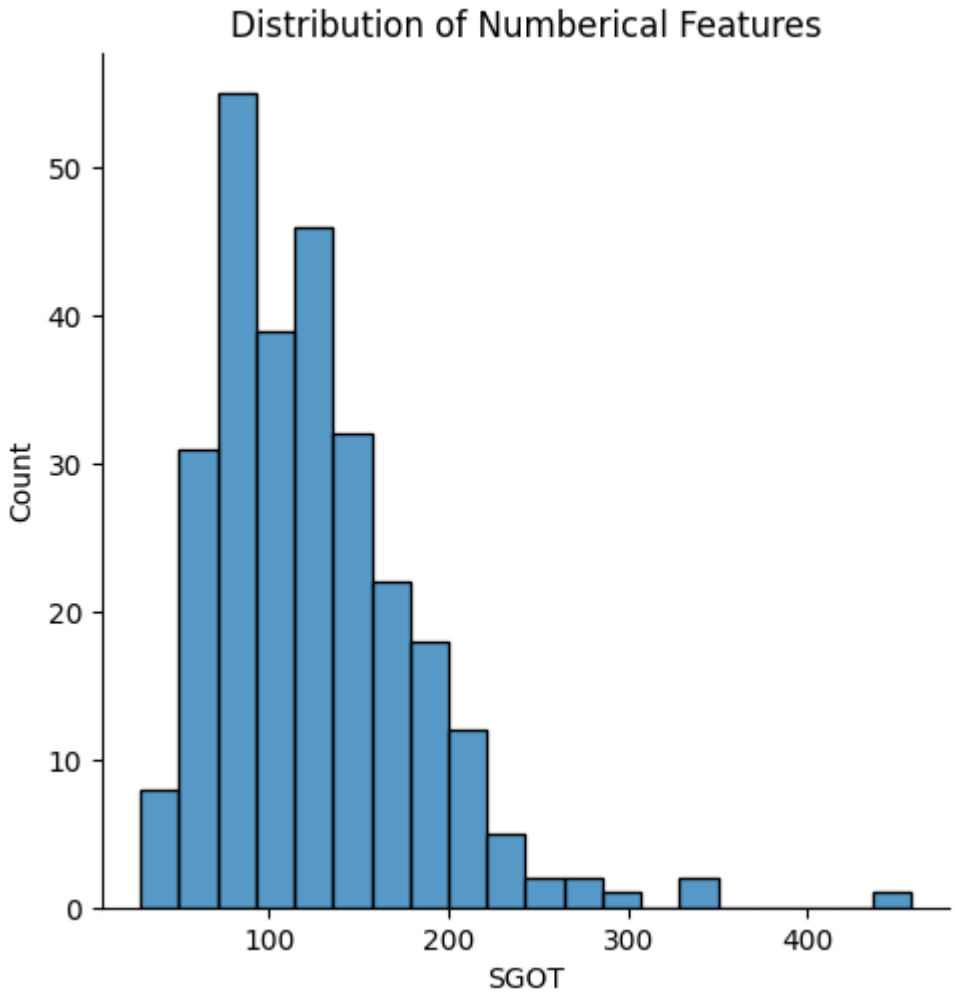
2.1 Data Visualization:

Following plots are visualized to get more insight on the dataset.

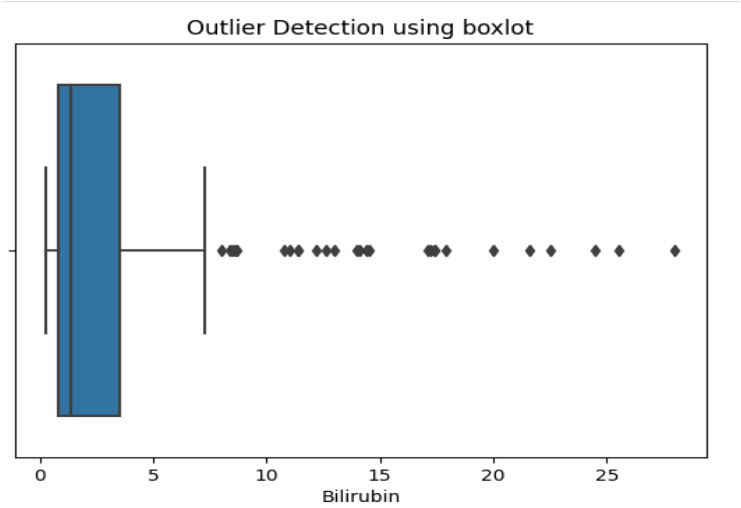
- sns.countplot: Imbalanced dataset



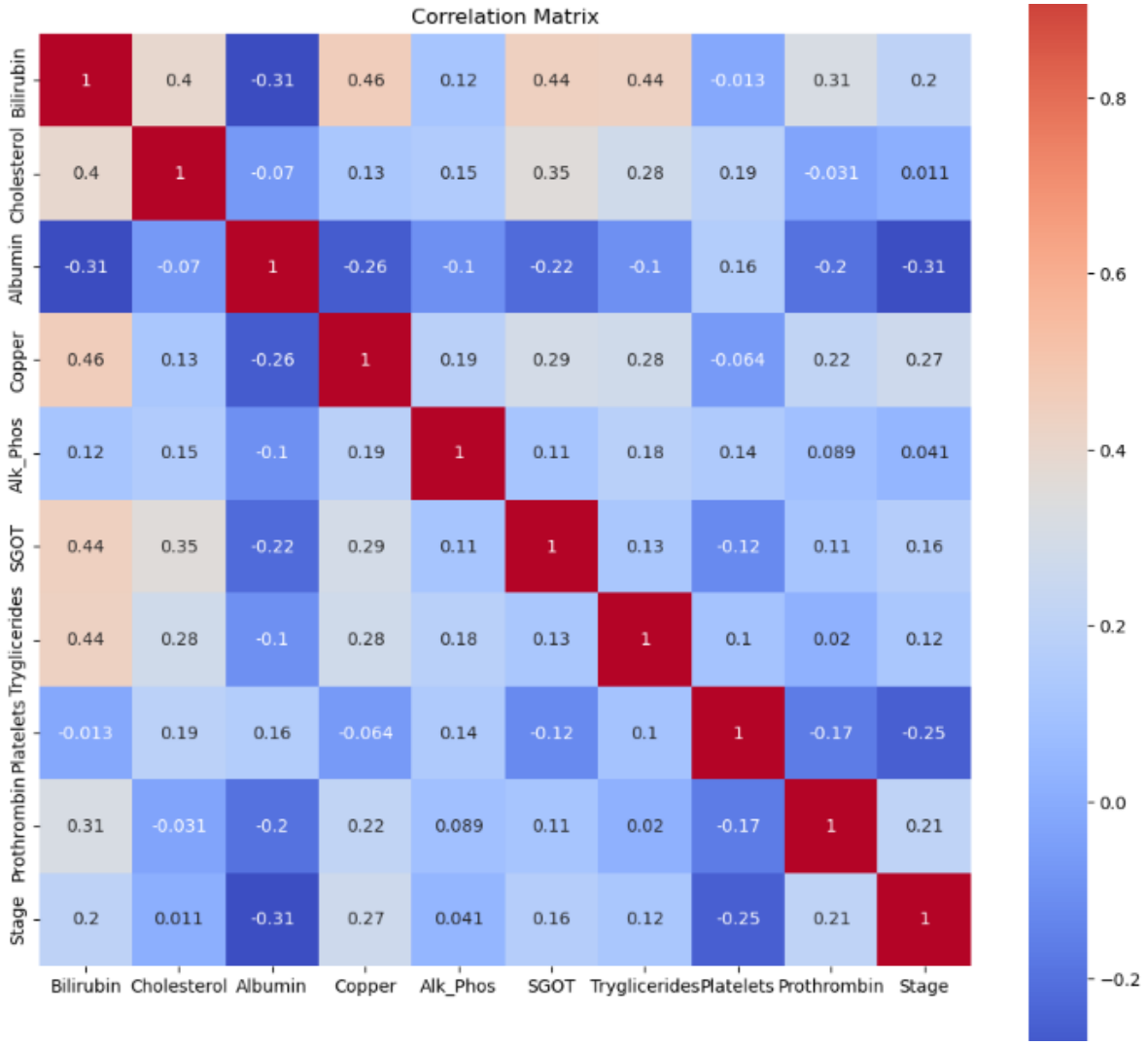
- sns.displot is used to visualize the distribution of the data. Features are right or left skewed.



- sns.Boxplots () are used to visualize outliers.



- sns.heatmap() used to view the correlation matrix. Bilirubin has Moderate positive correlation with status, copper, SGOT and Tryglicerides. None of the features has a high positive correlation with Target variable.



2.2 Pre-processing:

- Head(), tail() function used to get view on top 5 and last 5 rows from the dataset.

	ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin	Stage
0	1	400	D	D-penicillamine	21464	F	Y	Y	Y	Y	14.5	261.0	2.60	156.0	1718.0	137.95	172.0	190.0	12.2	4.0
1	2	4500	C	D-penicillamine	20617	F	N	Y	Y	N	1.1	302.0	4.14	54.0	7394.8	113.52	88.0	221.0	10.6	3.0
2	3	1012	D	D-penicillamine	25594	M	N	N	N	S	1.4	176.0	3.48	210.0	516.0	96.10	55.0	151.0	12.0	4.0
3	4	1925	D	D-penicillamine	19994	F	N	Y	Y	S	1.8	244.0	2.54	64.0	6121.8	60.63	92.0	183.0	10.3	4.0
4	5	1504	CL	Placebo	13918	F	N	Y	Y	N	3.4	279.0	3.53	143.0	671.0	113.15	72.0	136.0	10.9	3.0

- Irrelevant features(Patient ID, N_days) are dropped from the dataset. <cirrhosis.drop(['ID','N_Days'],axis=1, inplace=True)>
- All Missing values are dropped from the dataset. (dropna())

```
# Cleaning step 1#
#  remove rows with missing values #
cirrhosis_withoutnull= cirrhosis.dropna()
cirrhosis_withoutnull.shape
```

```
: (276, 18)
```

- Age in days will be converted to Age in years.
- Outcome Variable in float will be converted to int.

```
#Data Transformation stage#
cirrhosis_withoutnull['Age']=cirrhosis_withoutnull['Age'] // 365
cirrhosis_withoutnull['Stage']=cirrhosis_withoutnull['Stage'].astype(int)
print(cirrhosis_withoutnull)
```

- Categorical features are replaced with numerical values using replace function. Yes or No Values in categorical variables are replaced with binary values by creating a forloop. Outcome (Stage) variable values are replaced as well to start from 0 to 3.

```
#Column encoding for categorical variable #

features_to_replace=['Ascites','Hepatomegaly','Spiders']
mapping={'Y':0, 'N':1}
#Replace categorical values##

cirrhosis_withoutnull['Status']=cirrhosis_withoutnull['Status'].replace({'C': 0,'CL':1,'D':2})
cirrhosis_withoutnull['Drug']=cirrhosis_withoutnull['Drug'].replace({'D-penicillamine':0, 'Placebo':1})
cirrhosis_withoutnull['Sex']=cirrhosis_withoutnull['Sex'].replace({'F':0, 'M':1})
cirrhosis_withoutnull['Edema']=cirrhosis_withoutnull['Edema'].replace({'Y':0, 'N':1,'S':2})
cirrhosis_withoutnull['Stage']=cirrhosis_withoutnull['Stage'].replace({1:0, 2: 1, 3: 2, 4:3})

### Multiple features with yes or no values##
for column in features_to_replace:
    cirrhosis_withoutnull[column]=cirrhosis_withoutnull[column].replace(mapping)
    print(cirrhosis_withoutnull)
```

- Based on a hypothesis that 10% of the clinical data contains outliers, the count of outlier data points in the dataset was identified using the IsolationForest function (McDonald 2022). The process involves importing IsolationForest from the sklearn.ensemble module, instantiating the model with a contamination level of 0.1 and a random state of 42. The model is then fitted on the numerical features, and the decision function is used to determine the outlier scores for each data point. Subsequently, the outlier and inlier data points are predicted in the dataset. To obtain the value counts of the outlier and inlier data points, the pandas.Series function is used. Out of the total of 276 data points, 28 have been predicted as outliers. However, defining the percentage of outliers need domain expert knowledge. Hence further analysis is required to identify outliers using statistical analysis.

```
#outlier detection using Isolation Forest#
numerical_features=cirrhosis_withoutnull[['Bilirubin','Cholesterol','Albumin','Copper','Alk_Phos','SGOT','Tryglicerides','Platelets','Prothrombin']]

#outlier detection#
from sklearn.ensemble import IsolationForest
model = IsolationForest(contamination=float(0.1),random_state=42)
model.fit(numerical_features)
outlier_scores = model.decision_function(numerical_features)
#Obtain the outlier predictions (-1 for outliers, 1 for inliers) for the numerical features
outlier_predictions=model.predict(numerical_features)
value_counts = pd.Series(outlier_predictions).value_counts()
print(value_counts)
```

```
 1    248
-1     28
dtype: int64
```

- Outlier detection using Z-score: The formula used in the code calculates the z-score for each numerical column by subtracting the mean of the column from each value and then dividing it by the standard deviation of the column. Set threshold of 3 to detect outliers that falls beyond 3rd standard deviation. Identify the indices of the outliers by comparing the absolute z-scores to the threshold. Create a boolean mask called **outlier_mask**, where True represents an outlier and False represents a non-outlier. All numerial features has outliers excluding Age. Outliers can sometimes represent rare or extreme cases, will continue the analysis without eliminating outliers.

```
#Outlier detection using z-scores
# Select the columns for outlier detection
columns = cirrhosis_withoutnull[['Age', 'Cholesterol','Albumin', 'Copper','Bilirubin','Alk_Phos','SGOT','Platelets','Tryglicerides', 'Ascites', 'Prothrombin']]

# calculating  z-scores for each column
z_scores = np.abs((columns - columns.mean()) / columns.std())

threshold = 3

outlier_indices = np.where(z_scores > threshold)

# Create a mask to mark the outliers in the DataFrame
outlier_mask = np.zeros_like(columns, dtype=bool)
outlier_mask[outlier_indices] = True

# print the number of outliers in each column
num_outliers = np.sum(outlier_mask, axis=0)
for col, num in zip(columns, num_outliers):
    print(f"Number of outliers in {col}: {num}")
```

```
Number of outliers in Age: 0
Number of outliers in Cholesterol: 8
Number of outliers in Albumin: 2
Number of outliers in Copper: 6
Number of outliers in Bilirubin: 10
Number of outliers in Alk_Phos: 9
Number of outliers in SGOT: 3
Number of outliers in Platelets: 1
Number of outliers in Tryglicerides: 4
Number of outliers in Ascites: 19
Number of outliers in Prothrombin: 3
```

- Import the **RandomOverSampler** function from the **imblearn.over_sampling** module (Anon n.d.). Split the dataset into features (X) and target (Y). Instantiate the **RandomOverSampler** with a random state of 42. Fit and resample the X and Y variables using the **RandomOverSampler**. Observe that the resampling process has generated an equal number of samples across different target classes.

```
resample=RandomOverSampler(random_state=42)
sampledX, sampledY = resample.fit_resample(unsampledX,unsampledY)
print('Unsampled Value count: \n',unsampledY.value_counts())
print('sampled Value count: \n', sampledY.value_counts())
print(sampledY.unique())
```

```
Unsampled Value count:
 2    111
3     94
1     59
0     12
Name: Stage, dtype: int64
sampled Value count:
 3    111
2    111
1    111
0    111
Name: Stage, dtype: int64
[3 2 1 0]
```

- The mean values of the variables vary significantly. The standard deviations of the variables also vary, indicating the degree of dispersion around the mean. Data has varying range, the minimum and maximum values differ significantly from each other.However, feature scaling will be performed after splitting data into train & test to prevent any information leakage from the test set to the training set.

```
numerical_columns=['Bilirubin','Cholesterol','Albumin','Copper','Alk_Phos','SGOT','Tryglicerides','Platelets','Prothrombin']
sampledX[numerical_columns].describe()
```

	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
count	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000	444.000000
mean	2.716441	341.177928	3.561014	87.227477	1842.889640	111.701644	116.193694	263.542793	10.808333
std	4.118474	203.604710	0.381159	77.758874	1992.642153	53.739145	59.164677	87.599337	1.282896
min	0.300000	120.000000	1.960000	4.000000	289.000000	28.380000	33.000000	62.000000	9.000000
25%	0.700000	232.750000	3.370000	36.000000	720.000000	71.000000	80.000000	210.750000	10.000000
50%	1.100000	279.500000	3.600000	68.000000	1130.000000	99.165000	101.000000	263.500000	10.600000
75%	3.100000	387.750000	3.832500	108.000000	1913.000000	140.262500	143.000000	312.000000	11.100000
max	28.000000	1775.000000	4.400000	588.000000	13862.400000	457.250000	598.000000	563.000000	17.100000

After scaling data:

```
numerical_columns=['Bilirubin','Cholesterol','Albumin','Copper','Alk_Phos','SGOT','Tryglicerides','Platelets','Prothrombin']
sampledX[numerical_columns]=scaler.fit_transform(sampledX[numerical_columns])
sampledX[numerical_columns].describe()
```

	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
count	444.000000	444.000000	4.440000e+02	4.440000e+02	444.000000	4.440000e+02	444.000000	4.440000e+02	4.440000e+02
mean	0.000000	0.000000	-4.000804e-18	-1.600321e-17	0.000000	1.600321e-17	0.000000	-1.200241e-17	-1.000201e-17
std	1.001128	1.001128	1.001128e+00	1.001128e+00	1.001128	1.001128e+00	1.001128	1.001128e+00	1.001128e+00
min	-0.587394	-1.087536	-4.205126e+00	-1.071535e+00	-0.780693	-1.552232e+00	-1.407724	-2.303330e+00	-1.411161e+00
25%	-0.490161	-0.533142	-5.017046e-01	-6.595423e-01	-0.564154	-7.582472e-01	-0.612435	-6.033418e-01	-6.307956e-01
50%	-0.392928	-0.303271	1.023996e-01	-2.475495e-01	-0.358165	-2.335502e-01	-0.257093	-4.890570e-04	-1.625762e-01
75%	0.093236	0.228996	7.130701e-01	2.674416e-01	0.035224	5.320716e-01	0.453591	5.537926e-01	2.276067e-01
max	6.145985	7.050129	2.203631e+00	6.447334e+00	6.038750	6.437359e+00	8.152665	3.422343e+00	4.909801e+00

- Recursive Feature elimination technique with Randomforest classifier is used to select most relevant features from the dataset.Extracted 6 features out of 17 from dataset.

```
#Feature selection using recrusive elimination technique#
selectedfeature=RFE(RandomForestClassifier(n_estimators=100,random_state=42),n_features_to_select=6)
selectedfeature.fit(sampledX, sampledY)
reducedsampledX=selectedfeature.transform(sampledX)
feature_mask = selectedfeature.support_
selected_feature_names = sampledX.columns[feature_mask]
print(selected_feature_names)
print(reducedsampledX.shape)
print(sampledX.shape)
print(reducedsampledX)
print(sampledY)
```

```
Index(['Bilirubin', 'Cholesterol', 'Copper', 'Alk_Phos', 'SGOT', 'Platelets'], dtype='object')
(444, 6)
(444, 17)
[[1.4500e+01 2.6100e+02 1.5600e+02 1.7180e+03 1.3795e+02 1.9000e+02]
```

- Import Train_test_split from model_selection. Split data into train and Test. 80% of the data will be used for training the models and 20% for testing.

```
#train test split including all features#
x_train, x_test,y_train,y_test=train_test_split(sampledX,sampledY,test_size=0.2,random_state=2)
#print('Dimension of training data:',x_train.shape,'Dimension of test data:', x_test.shape)

#train test split from selected features#
x_train_reducedfeature,x_test_reducedfeature,y_train_reducedfeature,y_test_reducedfeature=train_test_split(reducedsampledX,sampledY,test_size=0.2,random_state=2)
print('Dimension of training data:',x_train_reducedfeature.shape,'Dimension of testing data:',x_test_reducedfeature.shape)
```

```
Dimension of training data: (355, 6) Dimension of testing data: (89, 6)
```

3. **Implementation**:

Import all necessary librariers for RFC, MLP, CNN ,clustering model & metrics. Confusion matrix is designed to gain insights into the number of false positives and true negatives. Additionally, a classification report is generated, providing measurements of precision, recall, and the F1-Score. Cross-validation technique is used to evaluate the model score.

```
import sklearn.metrics as metrics
from sklearn.metrics import f1_score,accuracy_score,confusion_matrix,recall_score,precision_score,classification_report,ConfusionMatrixDisplay
```

Dimension of train & test datasets.

```
Dimension of training data with all features: (355, 17) Dimension of test data with all features: (89, 17)
Dimension of training data with selective features: (355, 6) Dimension of testing data with selective features: (89, 6)
```

Task1: Random Forest Classifier (RFC)

The Random Forest Classifier (RFC) model was trained on a pre-processed, sampled and scaled dataset with 17 features. A comparison was made between the complete set of features (excluding ID and N_status) and a reduced dataset of 6 features.

```
##Initiate the Model
RFC_Default=RandomForestClassifier(random_state=42)
RFC = RandomForestClassifier(n_estimators=10,random_state=20,criterion="log_loss")
RFC_reducedfeature = RandomForestClassifier(n_estimators=10,random_state=20,criterion="log_loss")
```

Sampled data which has an equal number of samples in the target is spiltted into train (80%) & test set (20%) with random state as 2.

RFC Classifier model with complete set of features & with default decision trees, random_state as 42 returned accuracy of 71%.

```
RFC_accuracy with complete feature set default decisiontrees: 70.78651685393258
RFC_accuracy for complete featureset after hyper parameter tuning: 74.15730337078652
RFC_accuracy for reduced featureset: 66.29213483146067
```

Table 1:RFC Classification Report

Classification report for complete featureset with default parameters

	precision	recall	f1-score	support
0	0.90	1.00	0.95	26
1	0.85	0.68	0.76	25
2	0.38	0.33	0.35	18
3	0.62	0.75	0.68	20
accuracy			0.72	89
macro avg	0.69	0.69	0.68	89
weighted avg	0.72	0.72	0.71	89

In the classification report, Precision score for class 0 and 1 is relatively high. This model is able to predict 90% true positives for class 1. Recall Score of 1 indicates that this model is able to predict all true positives for class 0. However, class 2 and 3 has relatively low precision and recall score. A weighted avg of 0.72 indicates that the model's performance is slightly lower when considering the class imbalance, as it gives more importance to classes with a larger number of samples.

To create test dataset with equal number of samples included ‘stratify parameter while splitting the data and then verified the metrics for more accurate results.

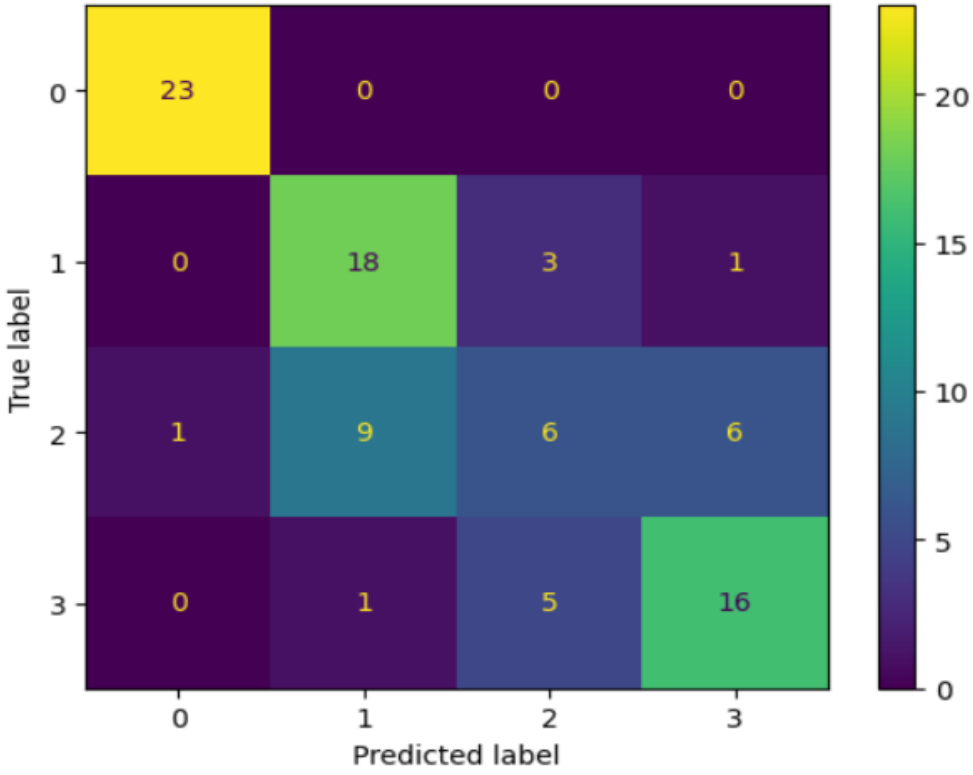
```
#train test split including all features#
x_train, x_test,y_train,y_test=train_test_split(sampledX,sampledY,test_size=0.2,random_state=2,stratify=sampledY)
#print('Dimension of training data:' x_train.shape 'Dimension of test data:' x_test.shape)
```

Classification report After stratified split:

Classification report for complete featureset with default parameters

	precision	recall	f1-score	support
0	0.96	1.00	0.98	23
1	0.64	0.82	0.72	22
2	0.43	0.27	0.33	22
3	0.70	0.73	0.71	22
accuracy			0.71	89
macro avg	0.68	0.70	0.69	89
weighted avg	0.68	0.71	0.69	89

RFC Confusion Matrix:



Model score of the RFC is verified using 10-fold cross-validation.

```
Model Score complete feature with default decisiontrees 0.7293650793650793
Model Score complete feature after hyper parameter tuning 0.7293650793650793
Model Score reduced feature= 0.6847619047619047
```

Advanced Activity:

Hyperparameter Tuning : GridSearchCV is used to identify the optimal parameters. Switching the criterion to 'Log-Loss' with default values on decision trees, further enhanced accuracy. The model became overfitted as the number of decision trees increased. By reducing the decision trees to 10 and using a random state of 20, the accuracy improved as well. Hypertuning has increased the accuracy from 71% to 74%.

```
#hyper parameter tuning for RFC using Grid Search CV#
model=RandomForestClassifier()
parameters = {

'n_estimators':[10,100,500],
'random_state':[ 10,25,42, 52],
'criterion':['gini', 'entropy', 'log_loss']
    }

classifier=GridSearchCV(model,parameters,cv=5)
classifier.fit(sampledX,sampledY)
results=pd.DataFrame(classifier.cv_results_)
print(results)
```

DataSet	Parameters	Accuracy
Reduced Featureset 6 Features	n_estimators=100 random_state=42 criterion="log_loss"	65
	n_estimators=10 random_state=20 criterion="log_loss"	66
Complete Featureset 17 Features	n_estimators=100 random_state=42 criterion="log_loss"	73
	n_estimators=10 random_state=20 criterion="log_loss"	74

Task 2: MLP Classifier

Model is trained for logistic activation with 3 hidden layers (20,15,10) and random_state as 42, solver as “lbfgs”, Learning rate “adaptive”, max_iter as 100.

1. Dataset with all features (excluding ID, N_status)
2. Dataset with selective Features (6 Features)

Datasets are splitted as train, test with random state as 2.

Different combinations of parameters were tried for the Logistic activation method with 3 hidden layers. Using the default value of max_iter (200) decreased the accuracy. Similarly, selecting solver parameters with values "sgd" and "adam" reduced the accuracy of the model. Accuracy is improved while using dataset with selective features.

Choosen Parameters:

```
##Logistic Activation#
MLPmodel1=MLPClassifier(activation='logistic',max_iter=100,
                                              hidden_layer_sizes=(20,15,10),
                                              solver='lbfgs',learning_rate='adaptive',random_state=42)
```

Weight Matrix:

```
#weight matrix for logistic activation with 3 hidden layers#
coefficients=MLPmodel_reducedfeature.coefs_

for i in range(len(coefficients)):
    print("Weight matrix", i, "shape:", coefficients[i].shape)
```

```
Weight matrix 0 shape: (6, 20)
Weight matrix 1 shape: (20, 15)
Weight matrix 2 shape: (15, 10)
Weight matrix 3 shape: (10, 4)
```

Accuracy:

```
Accuracy of Logistic activation complete featureset: 61.79775280898876
Accuracy of Logistic activation of reduced feature: 69.66292134831461
```

```
Classification report for reduced featureset
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        23
           1       0.75      0.68      0.71        22
           2       0.59      0.45      0.51        22
           3       0.54      0.64      0.58        22

    accuracy                           0.70        89
   macro avg       0.69      0.69      0.69        89
weighted avg       0.69      0.70      0.69        89

Model Score for Logistic activation _reducedfeature 0.5996031746031746
```

The MLP model obtained 69% accuracy with reduced features, while the RFC model achieved 66%. However, the RFC model achieved 71% accuracy with all features, while the MLP model performed less effectively with 61%. These findings suggest that the RFC model excels with the complete dataset, while the MLP model improves with fewer features. Additionally, RFC's cross-validation score (73) surpasses that of the MLP model (59).

Advanced Activity:

Randomized searchCV hyper parameter tuning is performed to find the optimal parameters.
Feature selection using Reduced Feature Elimination technique improved the performance.

```
#RandomizedSearchCV#
model=MLPClassifier()
parameters = {

'activation':['identity', 'logistic', 'tanh', 'relu'],
'solver':['lbfgs', 'sgd', 'adam'],
'learning_rate':['constant', 'invscaling', 'adaptive'],
'max_iter':[100,200]

    }

classifier=RandomizedSearchCV(model,parameters,cv=5)
classifier.fit(reducedsampledX,sampledY)
classifier.cv_results_
results2=pd.DataFrame(classifier.cv_results_)
print(results2)
```

Based on the results, model parameters are tuned. By increasing the number of neurons in the hidden layers, extending the max_iter value, and selecting an appropriate learning rate and activation method, the model's accuracy improved from 69 to 73.

```
Accuracy with relu activation: 73.03370786516854
Accuracy with tahn activation 73.03370786516854
Model Score for tanh activation 0.6563380281690141
```

Summary of MLP performance:

Activation	Hidden Layers \ neurons	max_iter	solver	learning Rate	Accuracy	CV 5 fold	Random State
logistic	3 \ (20,15,10)	100	lbfgs	adaptive	69	60	42
logistic	1 \ 100	200	lbfgs	Constant	71	66	42
relu	3 \ (100,50,25)	100	lbfgs	adaptive	73	65	42
tanh	3 \ (100,50,25)	200	lbfgs	invscaling	73	65	42

Task 3: Convolutional Neural Network

CNN model is suitable for both image processing & timeseries data**.** As the dataset used in this project is a CSV file, reshaped the tabular data(Features) into grid structure using reshape function, Target variable is transformed from integers to categorical. Split the data into train & test.

The CNN model is trained using sampled data that consists of 17 features and does not have any imbalances.

```
#Reshape X-Features and encode Y target#
sampledX_array = sampledX.values
sampledX_reshaped = np.reshape(sampledX_array, (sampledX_array.shape[0], sampledX_array.shape[1], 1))
sampledY_encoded= to_categorical(sampledY,num_classes=4)

#train test split
CNNxtrain,CNNxtest,CNNytrain,CNNytest= train_test_split(sampledX_reshaped,sampledY_encoded,test_size=0.2,random_state=3,stratify=sampledY_encoded )
```

Refer below snapshot for the steps:

Reshape X

number of samples = height of the grid
number of features = width of the grid
Channel 1= grey scale image.

Layer	Filters	Kernel
convolutional Layer 1	32	3
convolutional Layer 2	64	3

Flatten()
Hidden Layer 64 neurons
output Layer 4 neurons

Encode Y

Transform from integers to categorical to remove any relationships that might bias the pattern prediction

"Relu" activation is used to learn the patterns, while "Softmax" is used to predict the multi class target. The "categorical_crossentropy" loss function is utilized, as the target variable has been encoded as categorical, and the task involves multi-class classification.

CNN Parameters:

```
# Create the model
CNNmodel = tf.keras.Sequential([
    Conv1D(32, 3, activation='relu', padding='same', input_shape=(CNNxtrain.shape[1], 1)),
    MaxPooling1D(2),
    Conv1D(64, 3, activation='relu', padding='same'),
    MaxPooling1D(2),
    Conv1D(64, 3, activation='relu', padding='same'),
    MaxPooling1D(2),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(4, activation='softmax')
])
```

Compile and fit the CNN model on the training data. Parameters used are:

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(CNNxtrain, CNNytrain, epochs=10, batch_size=32)
```

Model summary:

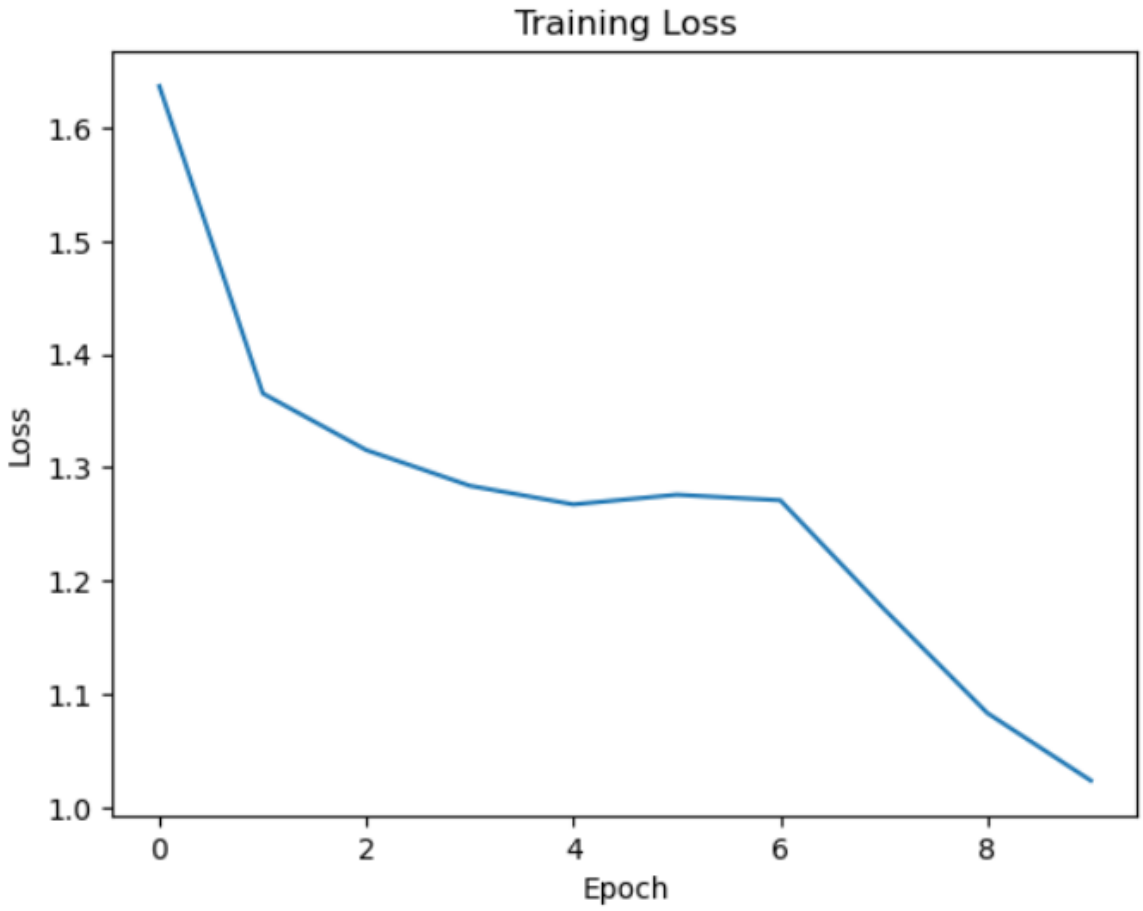
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_12 (Conv1D)          (None, 17, 32)            128

 max_pooling1d_8 (MaxPooling  (None, 8, 32)            0
 1D)

 conv1d_13 (Conv1D)          (None, 8, 64)             6208

 max_pooling1d_9 (MaxPooling  (None, 4, 64)            0
 1D)

 conv1d_14 (Conv1D)          (None, 4, 64)             12352

 flatten_4 (Flatten)         (None, 256)               0

 dense_6 (Dense)             (None, 64)                16448

 dense_7 (Dense)             (None, 4)                 260

=================================================================
Total params: 35,396
Trainable params: 35,396
Non-trainable params: 0
_________________________________________________________________
```

Evaluate the model on the test data.

Evaluate the model on the test data

loss, accuracy = CNNmodel.evaluate(CNNxtest, CNNytest)

```
CNN Test accuracy: 0.5617977380752563
Test loss CNN: 1.0373260974884033
```



Accuracy of CNN: 56%. Adjusted different parameters, changing optimizer as sgd & increase in batch size has reduced the accuracy score where as adding a drop out layer & increase in number of epochs has increased the model accuracy 61%.

Classification Report:

CNN Classification Report:

	precision	recall	f1-score	support
0	0.65	1.00	0.79	22
1	0.58	0.30	0.40	23
2	0.50	0.05	0.08	22
3	0.49	0.91	0.63	22
accuracy			0.56	89
macro avg	0.55	0.56	0.48	89
weighted avg	0.55	0.56	0.48	89

Task 4 K-means Clustering:

Optimum number of clusters are identified using WCSS method. Sampled and scaled data with 17 features used to train the model.

```
## To predict optimum number of clusters
wcss=[]
for i in range(1,10):
    kmeans = KMeans(i)
    kmeans.fit(sampledX)
    wcss_iter = kmeans.inertia_
    wcss.append(wcss_iter)
```

```
# Instantiate Kmeans
km = KMeans(3,random_state=42)
predictedlabel = km.fit_predict(sampledX)
```

The Rand Index measures clusters similarity using predicted and true labels. Additional metrics like Adjusted rand index (ARI) and Adjusted mutual info score (AMI) are used too. A Rand Index of 0.57 indicates limited agreement between clusters, while ARI (0.012) and AMI (0.016) suggest significantly low agreement between true and predicted clustering.

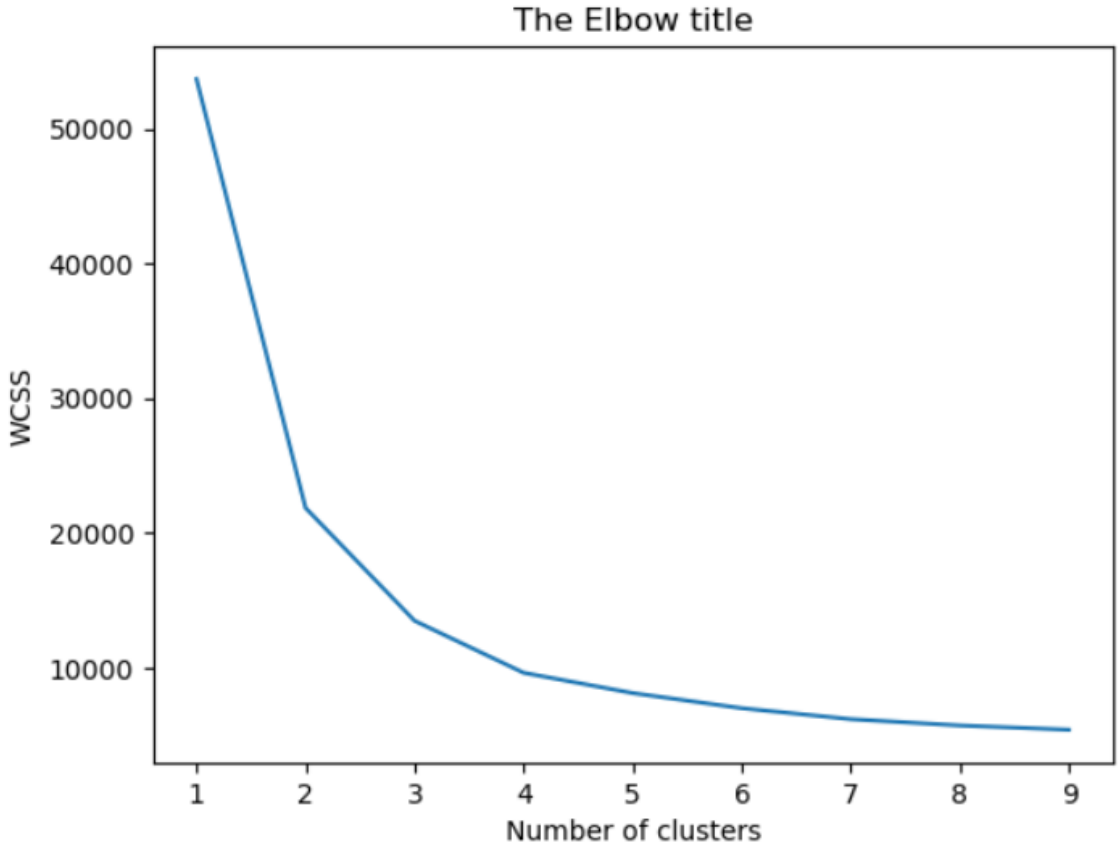
```
matching_labels = sum(sampledY == predictedlabel)
print("Result: %d out of %d samples were correctly labeled." % (matching_labels, sampledY.size))
```

```
Result: 114 out of 444 samples were correctly labeled.
```

114 out of 444 samples falls within same cluster groups.

```
Kmeansaccuracy: 25.675675675675674
Rand Index: 0.5784373538323877
Adjusted rand index:  0.012283151702209965
adj_mutual_info_score: 0.016050328476173124
Result: 114 out of 444 samples were correctly labeled.
```

Result: 114 out of 444 samples were correctly labeled.



Advanced activity:

Attempted an alternative method to decrease the data's dimensionality using PCA. Subsequently, applied K-means clustering with 4 clusters. The outcome was an accuracy rate of 27% and a Rand Index of 0.62.

```
#dimensionlity reduction
from sklearn.decomposition import PCA
pca = PCA(n_components=5)
pca.fit(sampledX)
x1= pca.transform(sampledX)
print(x1.shape)
```

```
(444, 5)
```

```
accuracy after dimensionality reduction 27.25225225225225
Rand Index after dimensionality reduction : 0.6223944034327782
adjusted_randscore_value after dimensionality reduction: 0.007788241282769152
adj_mutualinfo_score after dimensionality reduction: 0.011134024289709096
```

4. Ethical and Social Impact of AI Solutions:

Ethical issues:

- Privacy and Data Security: As per the GDPR guidelines, patient information within health data is considered extremely sensitive. When AI collects clinical data, there is a risk of it being vulnerable to hacking, which can lead to malicious exploitation, compromising both privacy and security.
- Informed Consent and Autonomy: Patients must receive sufficient information regarding the utilization of AI in their healthcare, enabling them to give informed consent or choose to opt-out of AI-driven treatments or interventions. Respecting patient autonomy and ensuring their comprehension of the implications of AI systems are vital in maintaining trust and mitigating potential ethical challenges.
- Trustworthiness: The training of AI algorithms on existing data can introduce biases that disproportionately affect certain populations, leading to disparities in healthcare outcomes.
- Explainability & Transparency: Use of black-box AI techniques are challenging to understand how they arrive at their decisions. In healthcare, transparency and explainability are crucial to build trust between patients, healthcare providers, and AI systems.

Social Issues:

- Human Oversight and Responsibility: AI can assist healthcare professionals in decision-making, but human judgment should not be entirely replaced. Human oversight and responsibility are vital to ensure proper use of AI, prevent errors or unintended consequences, and address ethical considerations beyond the capabilities of algorithms. Human involvement provides context, interpretation, and personalized decision-making aligned with patient needs and values.
- Liability and Accountability: Determining responsibility for patient harm caused by AI tools is a complex issue that demands a robust legal framework, clear guidelines, and ethical standards to ensure appropriate accountability in the field of AI-enabled healthcare.
- Impact on Workforce: Adoption of AI in healthcare may lead to concerns about job displacement, particularly for tasks that can be automated. Efforts should be made to retrain and upskill healthcare professionals to work effectively alongside AI systems and to create new roles and opportunities in the field of AI-enabled healthcare.
- Patient-physician relationship: While AI can augment medical knowledge, it is crucial for physicians to ensure that patients feel heard, valued, and supported, maintaining the human connection that is vital in healthcare.
- False positives results generated by AI systems in healthcare leads to unwarranted anxiety among patients.

- Several recent initiatives have been launched to address the ethical implications of AI solutions in healthcare: The Ethics Guidelines for Trustworthy AI in Healthcare by the European Commission, The American Medical Association's (AMA) Ethical AI Policy, The World Health Organization's (WHO) Global Strategy on Digital Health.

5. **Conclusion**:

The outcomes of liver cirrhosis stage prediction models are not very promising. Our model has been trained on a dataset comprising 444 sampled records. Among the different models used, random forest attained the highest accuracy of 74, while MLP demonstrated the highest precision and recall scores. However, it is evident that all these models fall short in accurately predicting the multi-class target with sufficient precision and recall.

To create a successful machine learning model, it is essential to gather additional data that is complete, without any missing values. Alternatively, if missing values are present, seeking guidance from domain experts to impute them is necessary. Further analysis is required to identify more advanced and precise algorithms that can be utilized for predicting Cirrhosis liver disease.

Model	Accuracy	Precision	Recall	F1-Score
Random Forest Classifier	74	68	70	69
MLP	73	72	73	72
CNN	61	63	62	62

6. **References**:

Anon, 2.3. Clustering — scikit-learn 0.23.1 documentation [online]. *scikit-learn.org*. Available at: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation.

Anon, 2018a. Convolutional Neural Networks - Deep Learning basics with Python, TensorFlow and Keras p.3 [online]. *www.youtube.com*. Available at: https://www.youtube.com/watch?v=WvoLTXIjBYU&t=282s [Accessed 31 May 2023].

Anon, 2018b. Liver Cirrhosis - Symptoms, Causes and Prevention [online]. *www.pacehospital.com*. Available at: https://www.pacehospital.com/liver-cirrhosis-symptoms-causes-and-prevention.

Anon, RandomOverSampler — Version 0.8.1 [online]. *imbalanced-learn.org*. Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html.

Bharathi, 2021. Confusion Matrix for Multi-Class Classification [online]. *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/.

Čartolovni, A., Tomičić, A., Lazić Mosler, E., 2022. Ethical, legal, and social considerations of AI-based medical decision-support tools: A scoping review [online]. *International Journal of Medical Informatics*, 161, p.104738. Available at: https://www.sciencedirect.com/science/article/pii/S1386505622000521 [Accessed 30 March 2022].

Cleveland Clinic, 2020. Cirrhosis of the Liver | Cleveland Clinic [online]. *Cleveland Clinic*. Available at: https://my.clevelandclinic.org/health/diseases/15572-cirrhosis-of-the-liver.

McDonald, A., 2022. Isolation Forest — Auto Anomaly Detection with Python [online]. *Medium*. Available at: https://towardsdatascience.com/isolation-forest-auto-anomaly-detection-with-python-e7a8559d4562#:~:text=Isolation%20Forest%20is%20a%20model [Accessed 28 May 2023].

Murphy, K. et al., 2021. Artificial intelligence for good health: a scoping review of the ethics literature. *BMC Medical Ethics*, 22(1). https://doi.org/10.1186/s12910-021-00577-8.

Scikit-learn, 2018. 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation [online]. *Scikit-learn.org*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.