HEPATITIS C DETECTION

USING

MACHINE LEARNING

Submitted By:

Livya Eldho

MAC23MCA-2036

Faculty Guide:

Prof. Sonia Abraham

Assistant Professor

MCA Department, MACE

INTRODUCTION

Hepatitis C is a blood-borne infection of the liver caused by the hepatitis C virus (HCV). Hepatitis C virus is a major cause for happening liver disease all over the world. In this work, a machine learning based model has been proposed that can classify hepatitis C virus infected patient's stages. This study explores the effectiveness of machine learning techniques in the prediction of treatment response in hepatitis c patients.

Instances of hepatitis C disease from Egyptian patients were gathered by me from the UCI Machine Learning Repository. The dataset contains 1385 sample observations and has 29 columns including 1 class variable and 28 features.

The proposed system is the comparative study of three algorithms Random Forest (RF), Artificial Neural Network (ANN), Support Vector Machine(SVM). The performance of these algorithms are compared to classify hepatitis c disease under four classes which are no fibrosis, mild fibrosis, moderate fibrosis and severe fibrosis/cirrhosis. The project will follow a structured methodology encompassing data preprocessing, model development, training, evaluation and deployment. An automated system can be very helpful to assist medical experts and even make automated disease predictions without any human mistakes. Patients can diagnose their condition without the assistance of a medical expert. Additionally , a web interface will be created to allow users to input parameters and receive the result.

A blood test is one of the ways to diagnose Hepatitis C disease. But after a lab blood test, a medical expert needs to examine the test stats to diagnose the disease. There is very little difference in the blood test stats, which refer to different hepatitis c stages. Such minor differences can lead to the wrong diagnosis even by medical experts as human error is expected. Incorrect diagnoses can result in inappropriate medication and additional complications. Therefore, an automated system can be highly beneficial in assisting medical experts and providing disease predictions without the risk of human error.

LITERATURE REVIEW

Paper 1 : Effective factors in diagnosing the degree of hepatitis c using machine learning

The paper "Effective factors in diagnosing the degree of hepatitis c using machine learning" explores the application machine learning techniques for detection and classification of hepatitis c disease using selective features.

In this study, a dataset was used which contains 27 features and 1385 records of patients with different grades of HCV, which can be categorized into four categories: Demographic, Symptoms, Liver Function Tests and Blood Panels before the start of treatment. The dataset was clean and preprocessed to ensure accuracy and consistency. To reduce the dimension of the dataset and determine the effective features, three feature selection techniques such as Pearson Correlation, ANOVA, and Random Forest were applied. Among all the algorithms, KNN, Random forests, and Deep Neural Networks were selected to be utilized, and then their evaluation metrics, such as Accuracy and Recall.

Various machine learning algorithms were applied to build predictive models. In all the modeling algorithms, Holdout was used to split up the dataset into "Train" and "Test" sets. The models were trained on the training set and evaluated on the testing set. This study utilizes a retrospective observational design which has been conducted to find the effective factors for diagnosing the degree HCV. This study showed that using data-mining algorithms can be helpful to for HCV diagnosing. The proposed model in this study can help physicians to diagnose the degree of HCV at an affordable and with high accuracy.

Performance evaluation of these models based on accuracy showed that Deep Learning with Accuracy = 93.51 had the highest performance. Random Forest had almost the same performance as Deep Learning. This performance was achieved on dataset containing features that were selected by ANOVA feature selection.

Title of the paper	Sayadi M, Varadarajan V, Gozali E, Sadeghi M. Effective factors in diagnosing the degree of hepatitis C using machine learning. Frontiers in Health Informatics. 2023 Apr 16;12:137.
Area of work	Detection and classification of hepatitis c disease using selective features.
Dataset	Dataset was taken from UCI repository. The dataset contains 1385 sample observations and each sample is represented by 29 features.
Methodology / Strategy	The dataset was clean and preprocessed to ensure accuracy and consistency. Three feature selection techniques were first used to select the most useful features for HVC analyzing. Then for each selected feature set, three machine-learning algorithms were applied, and a diagnosing model for detecting Hepatitis C was reported. The models were trained on the training set and evaluated on the testing set.
Algorithm	RF, KNN, DNN
Result/Accuracy	Results indicate that DNN and RF achieved the highest accuracy of 93.51 and 93.029 respectively. DNN – 93.51 RF – 93.029 KNN – 89.283

Paper 2 : Diagnosing the Stage of Hepatitis C Using Machine Learning

The paper " Diagnosing the Stage of Hepatitis C Using Machine Learning " aims to precision performance evaluation of the proposed Intelligent Hepatitis C Stage Diagnoses System (IHSDS) empowered with machine learning is presented to detect the Stage of Hepatitis C in a human using Artificial Neural Network (ANN).

In this research, the Hepatitis C patient dataset titled "HCV-Egy-Data" is obtained from the UCI machine learning repository. It includes 1385 observations, where each sample has 29 properties, out of which 19 properties are selected. The value of attribute "histological staging" in this dataset indicates the Stage of the patient. There are 336(24.26%) cases in class 1, 332 (23.97%) cases in class 2, 355 (25.63%) cases in class 3, and 362 (26.14%) cases in class 4.

969 samples (70% of the dataset) are used for training the model, and the remaining 416 samples (30% of the dataset) are used for validation purposes. This research work predicts the Stage of Hepatitis C by using the back-propagation algorithm of the ANN model. This model is used to gain the maximum precision in the prediction of the Stage of Hepatitis C.

Different stages involved in the back-propagation algorithm include reading the training data, building and connecting the ANN layers (this includes preparing weights, biases, and activation function of each layer), predicting error, updating parameters, and prediction precision.

To evaluate the performance of the proposed model, precision, miss rate, and mean square error (MSE) are calculated. The proposed IHSDS was trained and validated using the dataset and shows 98.89% and 94.44% precision during training and validation phases, respectively. The validation precision value of previous methods is compared with the proposed model to state that the results provided by the proposed IHSDS empowered with machine learning are better than the results provided by other proposed methods(RF,SVM,LR) with regard to precision.

Title of the paper	Butt MB, Alfayad M, Saqib S, Khan MA, Ahmad M, Khan MA, Elmitwally NS. Diagnosing the stage of hepatitis C using machine learning. Journal of Healthcare Engineering. 2021;2021(1):8062410.
Area of work	Performance evaluation of the proposed Intelligent Hepatitis C Stage Diagnoses System (IHSDS) empowered with machine learning in detection of stages of Hepatitis C using ANN.
Dataset	Dataset was taken from UCI repository. The dataset contains 1385 sample observations and each sample is represented by 29 features.
Methodology / Strategy	This model is used to gain the maximum precision in the prediction by using the back-propagation algorithm of the Artificial Neural Network model. It includes reading the training data, building and connecting the ANN layers (this includes preparing weights, biases, and activation function of each layer), predicting error, updating parameters, and prediction precision. To evaluate the performance of the proposed model, precision, miss rate, and mean square error (MSE) are calculated.
Algorithm	ANN(back-propagation algorithm)
Result/Accuracy	The proposed IHSDS shows 98.89% and 94.44 % precision during training and validation phases

Paper 3 : Hepatitis C Virus (HCV) Prediction by Machine Learning Techniques

This study aims to know the performance comparisons between multi and binary class labels of the same dataset, not limited to tool comparison, and to know which selected features play a key role in the prediction of HCV by using Egyptian patient's dataset. The management of HCV infected patients can be monitored by the assessment of liver fibrosis staging in Chronic Hepatitis C (CHC).

In this paper, the objective is to identify the best model, the selected features which play a key role in the prediction of HCV disease, and to compare the performances of Python and R tools by using Hepatitis C Virus (HCV) from Egyptian patient's dataset from UCI in the content of multi and binary class labels. The multivariate datatype consists of 1385 instances with 29 attributes. The multiclass label (i.e., Baseline histological staging) dataset instances consist of distinct values and frequencies, which are as follows F1, F2, F3 and F4. The binary class label dataset consists of mild to moderate fibrosis as class label = 0 (F1 and F2) and advanced fibrosis as class label =1 (F3-F4).

A total of seven machine learning techniques followed by different feature selection methods were used to evaluate the performance of the classifiers on dataset. The analytical tools such as Python-based Scikit learn and R based CARET package were used to perform data analysis. In Spyder (Python IDE)-based scikit-learn package, before to classification model building, the pre-processing steps such as data normalization, total dataset split into 70–30% as a training and testing data respectively with set.seed was used.

In Python, the multiclass dataset shows the highest accuracy (28.36%) in random forest, followed by KNN with 26.44%. The similar accuracy performance of KNN accuracy has been observed in NN, but when coming to precision and recall the KNN shows a better performance. Similarly, the binary dataset shows the highest accuracy (53.12%) in NN with the exemption to precision in comparison with SVM where the accuracy shows 52.64%. The individual performances of each classifier to accuracy, precision, and recall in binary class label almost shows the double score of multiclass label respectively.

On the other hand, the R multiclass dataset shows the highest accuracies with similar performances in SVM and RF (51.31%) followed by KNN (50.83%), and NB (50.65%). In the binary class label, boosting shows the highest accuracy (54.23%), followed by KNN (53.06%). Similar performance of accuracy also been noticed in NN and Bagging (51.73%) respectively. The accuracies of multiclass and binary class labels show similar performance, but whereas precision and recall show different performances. Within the comparison of 3 datasets features,

the 12 selected features overall average accuracy, precision, and recall shows similar performances to 29 and 21 selected features in both class labels and tools. Thus, indicating 12 selected features can be useful for the prediction of the HCV dataset. Despite class labels and tools used, average performances of evaluation metrics of the classifiers are in the order of SVM, RF, KNN, NN, and NB.

Title of the paper	Nandipati SC, XinYing C, Wah KK. Hepatitis C virus (HCV) prediction by machine learning techniques. Applications of modelling and simulation. 2020 Mar 15;4:89-100.
Area of work	Performances of classifiers and tools on multi and binary class labels of the same HCV datasets.
Dataset	The data was taken from the UCI machine learning repository. Dataset contains 1385 instances with a total of 29 features.
Methodology / Strategy	The selected targeted dataset is imbalanced, so to make the dataset balanced, resampling techniques are used. Feature selection methods like univariate feature selection approach and the feature importance method is implemented and the best set of characteristics for building effective models are selected.
Algorithm	Random Forest Classifer, Naive Bayes Classifer, Adaboost, K-Nearest Neighbor, SupportVectorMachine, NN, Bagging
Result/Accuracy	Performance comparison of multi and binary class labels Despite class labels and tools used, average performances of evaluation metrics of the classifiers are in the order of SVM, RF, KNN, NN, and NB. 12 selected features can be useful for the prediction of the HCV dataset.

LITERATURE SUMMARY

PAPER	TITLE	DATASET	ALGORITHM	ACCURACY
PAPER1	Sayadi M, Varadarajan V, Gozali E, Sadeghi M. Effective factors in diagnosing the degree of hepatitis C using machine learning. Frontiers in Health Informatics. 2023 Apr 16;12:137.	Dataset of Egyptian patients from UCI repository containing 1385 instance,29 features	DNN RF KNN	93.51 93.029 89.283
PAPER 2	Butt MB, Alfayad M, Saqib S, Khan MA, Ahmad M, Khan MA, Elmitwally NS. Diagnosing the stage of hepatitis C using machine learning. Journal of Healthcare Engineering. 2021;2021(1):8062410.	Dataset of Egyptian patients from UCI repository containing 1385 instance,29 features	ANN	94.44
PAPER 3	Nandipati SC, XinYing C, Wah KK. Hepatitis C virus (HCV) prediction by machine learning techniques. Applications of modelling and simulation. 2020 Mar 15;4:89-100.	Dataset of Egyptian patients from UCI repository containing 1385 instance,29 features	KNN SVM RF NB NN Bagging Boosting	Average performances of evaluation metrics of the classifiers are in the order of SVM, RF, KNN, NN, and NB

PROJECT PROPOSAL

The project aims to enhance the early detection of Hepatitis C using machine learning algorithms. From the above three papers, we get to know that different models were used for the detection of hepatitis c disease. First paper is the detection and classification of hepatitis disease using selective features. Second paper focuses on performance evaluation of the proposed Intelligent Hepatitis C Stage Diagnoses System (IHSDS) empowered with machine learning in detection of stages of Hepatitis C using ANN. Third paper aims at performances of classifiers and tools on multi and binary class labels of the same HCV datasets.

The proposed system is the comparative study of three algorithms Random Forest (RF), Artificial Neural Network (ANN), Support Vector Machine(SVM). The performance of these algorithms are compared to classify hepatitis c disease under four classes which are no fibrosis, mild fibrosis, moderate fibrosis and severe fibrosis/cirrhosis. An automated system can be very helpful to assist medical experts and even make automated disease predictions without any human mistakes. Patients can diagnose their condition without the assistance of a medical expert. Additionally , a web interface will be created to allow users to input parameters and receive the result.

Objectives

- Develop a machine learning model to detect and classify hepatitis c using RF, ANN, SVM.
- Compare the performance of these models to determine the most accurate one.
- Create a user-friendly web interface for users to input data and obtain the result.

Motivation for choosing RF, ANN, SVM

- Handling complex relationships
- High accuracy
- Robustness to Noise and Variability

DATASET

In this study, the dataset titled “HCV-Egy-Data” is taken from the UCI machine learning repository. The dataset contains 29 attributes including 1 class variable and 28 features and has 1385 records of patients with different stages of HCV, which can be categorized into different categories before the start of treatment such as:

(1) Demographic

(2) Symptom

(3) Liver Function Tests (LFTs)

(4) Blood Panels

The features are age, gender, BMI, fever, nausea, headache, diarrhea, fatigue & generalized boneache, jaundice, epigastricpain, WBC, RBC, HGB, Plat, AST1, ALT1, ALT4, ALT12, ALT24, ALT36, ALT48, ALT, RNA Base, RNA 4, RNA12, RNA EOT, RNA EF, Baseline histological Grading and Baseline histological staging.

Class variable : Baseline histological staging

The class variable include numbers from 1 to 4 which indicates different stages in hepatitis c disease which are :

1. no fibrosis
2. mild fibrosis
3. moderate fibrosis
4. severe fibrosis/cirrhosis.

Dataset:https://archive.ics.uci.edu/dataset/503/hepatitis+c+virus+hcv+for+egyptian+patients

EXPLORATORY ANALYSIS

```
df.shape
```

```
(1385, 29)
```

Dataset has 1385 rows and 29 features

```
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Age                               1385 non-null   int64
 1   Gender                            1385 non-null   int64
 2   BMI                               1385 non-null   int64
 3   Fever                             1385 non-null   int64
 4   Nausea/Vomting                    1385 non-null   int64
 5   Headache                          1385 non-null   int64
 6   Diarrhea                          1385 non-null   int64
 7   Fatigue & generalized bone ache   1385 non-null   int64
 8   Jaundice                          1385 non-null   int64
 9   Epigastric pain                   1385 non-null   int64
 10  WBC                               1385 non-null   int64
 11  RBC                               1385 non-null   float64
 12  HGB                               1385 non-null   int64
 13  Plat                              1385 non-null   float64
 14  AST 1                             1385 non-null   int64
 15  ALT 1                             1385 non-null   int64
 16  ALT4                              1385 non-null   float64
 17  ALT 12                            1385 non-null   int64
 18  ALT 24                            1385 non-null   int64
 19  ALT 36                            1385 non-null   int64
 20  ALT 48                            1385 non-null   int64
 21  ALT after 24 w                    1385 non-null   int64
 22  RNA Base                          1385 non-null   int64
 23  RNA 4                             1385 non-null   int64
 24  RNA 12                            1385 non-null   int64
 25  RNA EOT                           1385 non-null   int64
 26  RNA EF                            1385 non-null   int64
 27  Baseline histological Grading     1385 non-null   int64
 28  Baselinehistological staging      1385 non-null   int64
dtypes: float64(3), int64(26)
memory usage: 313.9 KB
```

Dataset contains numerical attributes

```
4    360
3    355
1    335
2    331
```

The Baseline histological Staging class contains numbers from 1 to 4 which indicates different hepatitis conditions

Data Preprocessing

Missing Values

There are no missing values in the dataset

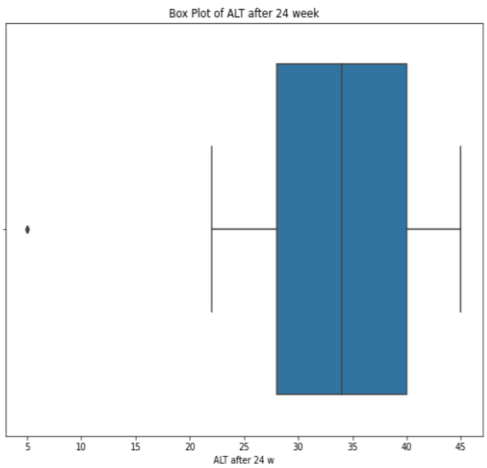
```
df.isnull().sum()
```

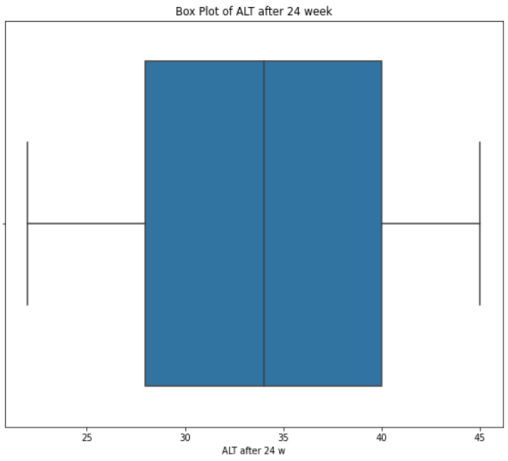
```
Age                                    0
Gender                                 0
BMI                                    0
Fever                                  0
Nausea/Vomting                         0
Headache                               0
Diarrhea                               0
Fatigue & generalized bone ache        0
Jaundice                               0
Epigastric pain                        0
WBC                                    0
RBC                                    0
HGB                                    0
Plat                                   0
AST 1                                  0
ALT 1                                  0
ALT4                                   0
ALT 12                                 0
ALT 24                                 0
ALT 36                                 0
ALT 48                                 0
ALT after 24 w                         0
RNA Base                               0
RNA 4                                  0
RNA 12                                 0
RNA EOT                                0
RNA EF                                 0
Baseline histological Grading          0
Baselinehistological staging           0
dtype: int64
```

Handling Outliers

- **ALT after 24 week**

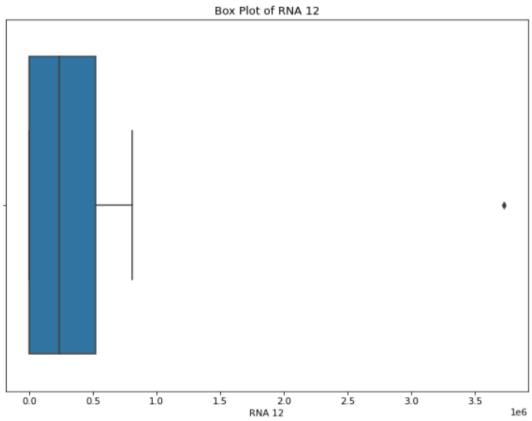
  ALT after 24 week shows one outlier. So value equals to 5 is removed.

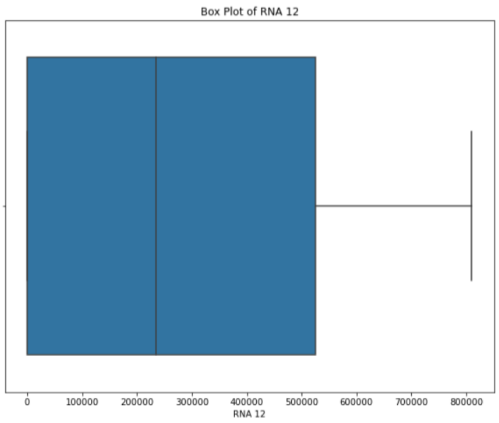




- **RNA 12**

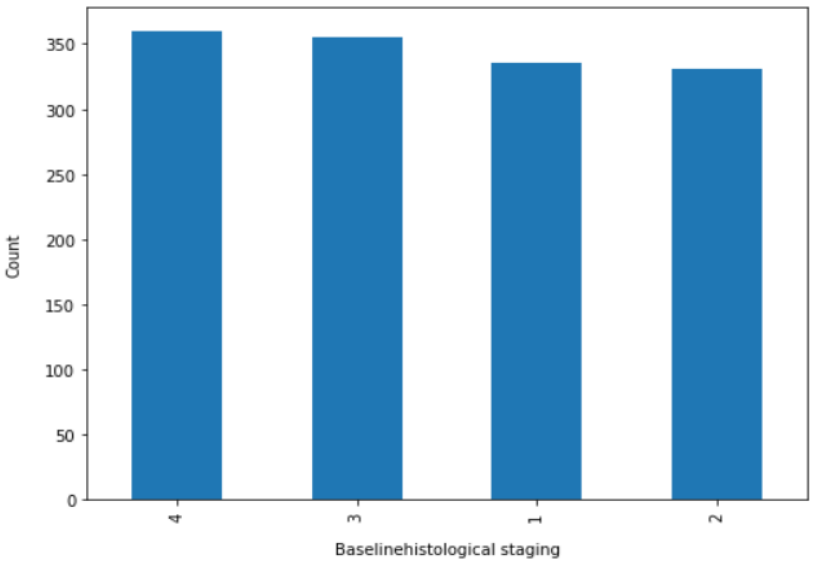
```
df['RNA 12'] = np.where(df['RNA 12'] == 3731527, np.nan, df['RNA 12'])
df.dropna(subset=['RNA 12'], inplace=True)
plt.figure(figsize=(10, 8))
sns.boxplot(x=df['RNA 12'])
plt.title('Box Plot of RNA 12')
plt.show()
```





Handling class imbalance

Baseline histological staging class is balanced. There is no large variation between number of people in the different stages

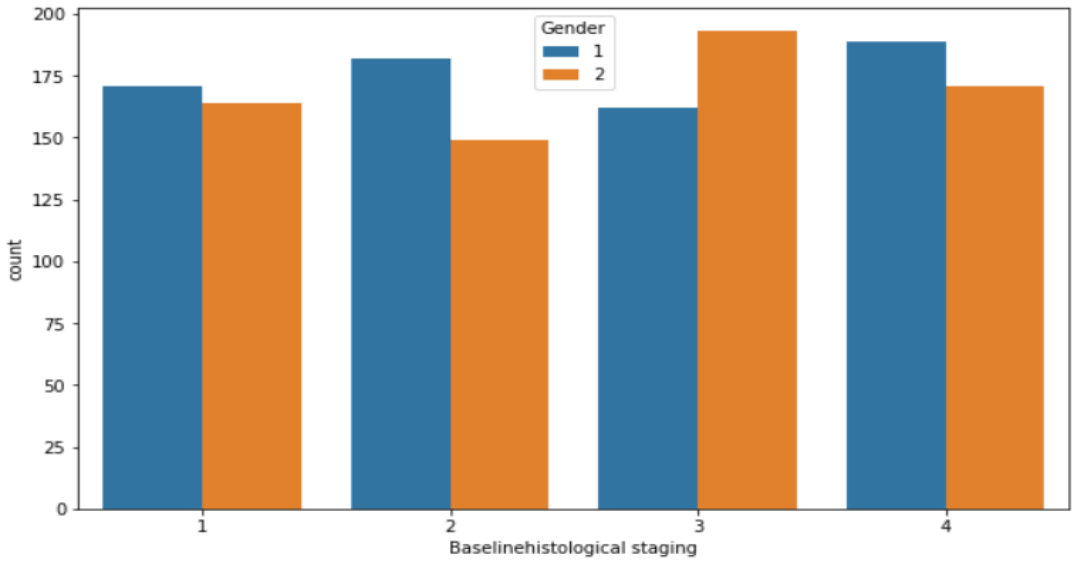


TRENDS AND PATTERNS

- **Gender**

```
plt.figure(figsize=(10, 6))
sns.countplot(data=df,x='Baselinehistological staging', hue='Gender')
```

```
<AxesSubplot:xlabel='Baselinehistological staging', ylabel='count'>
```

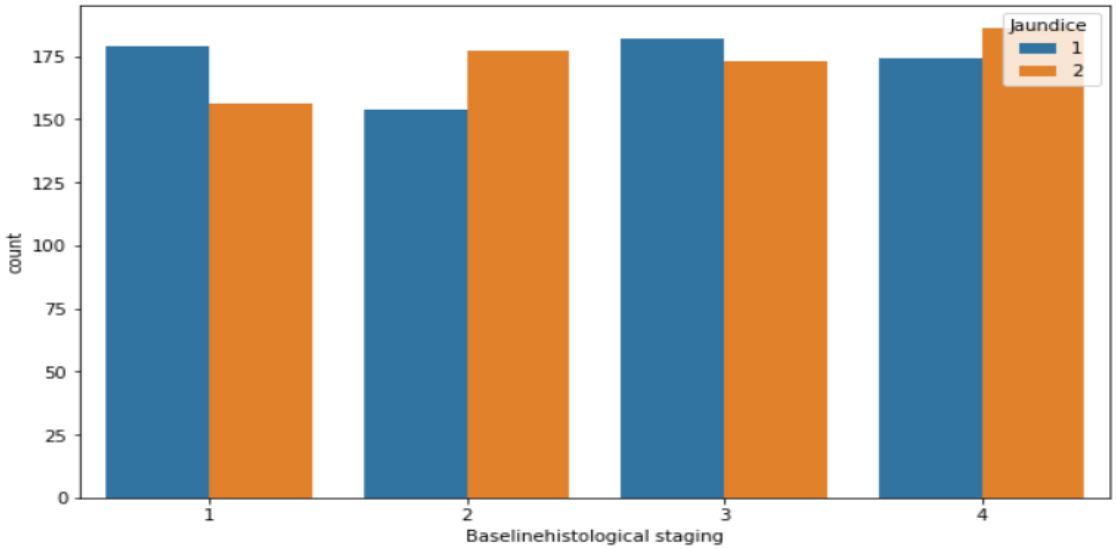


The plot indicates that for most of the stages (1,2 and 4),Male has a higher count compared to female. For stage 3, female is more represented. The higher count of female in stage 3 suggests that individuals in this gender may be more likely to reach or be diagnosed at this stage.

- **Jaundice**

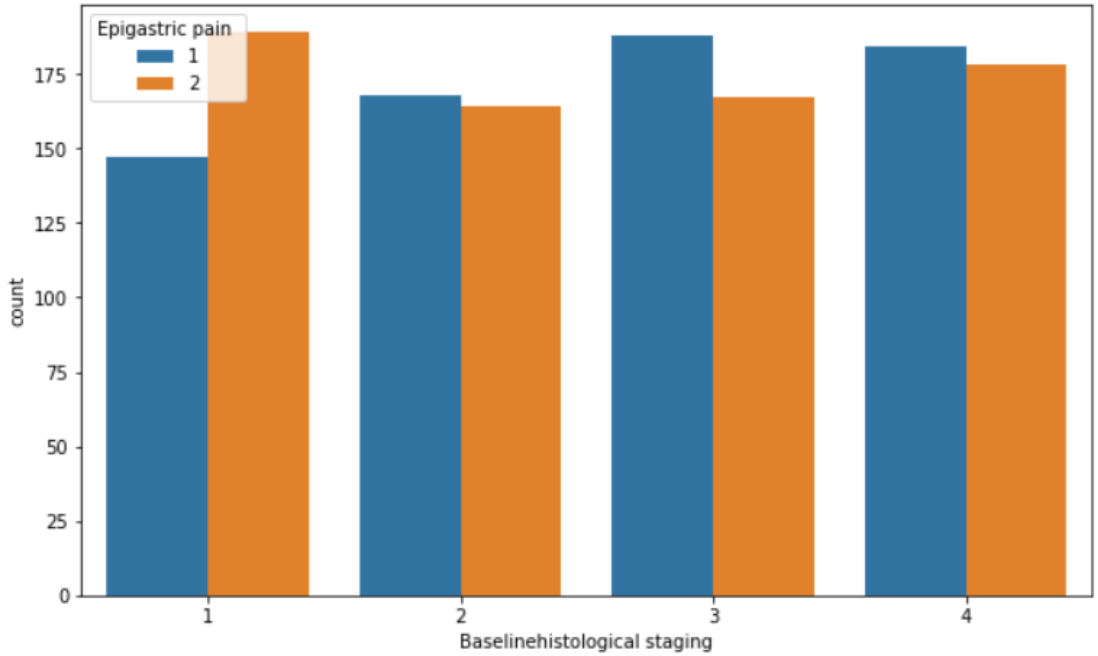
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df,x='Baselinehistological staging', hue='Jaundice ')
```

```
<AxesSubplot:xlabel='Baselinehistological staging', ylabel='count'>
```



Patients in stage 4 (Cirrhosis stage) are more likely to be affected by jaundice .

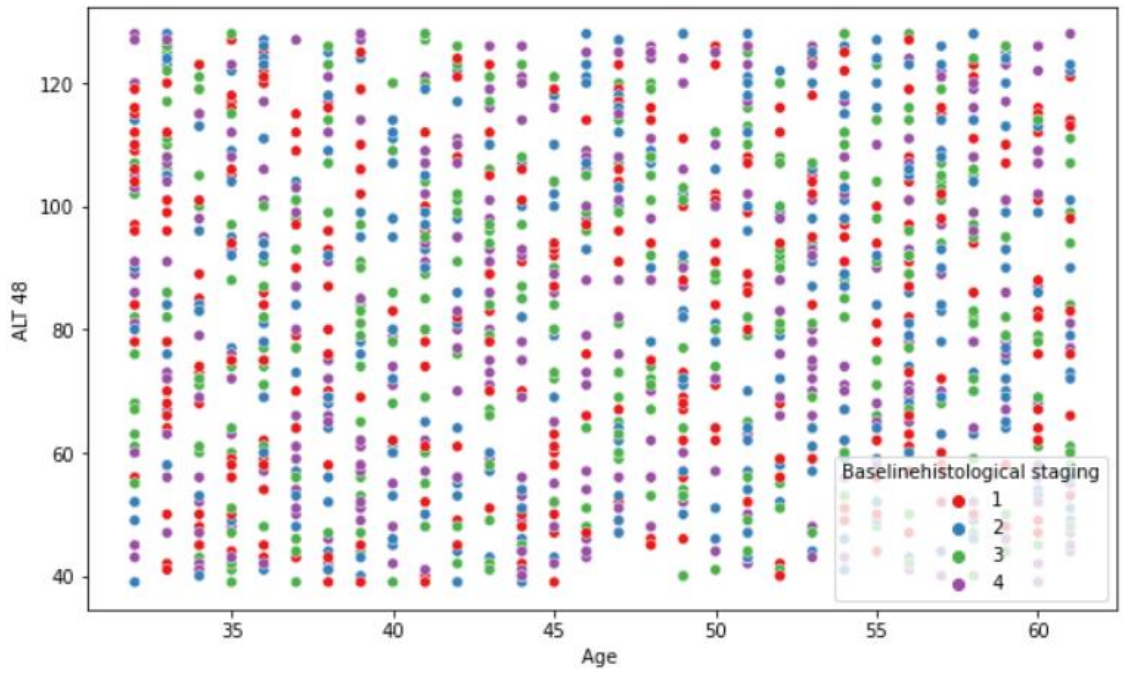
- **Epigastric pain**



Patients in stage 1 in hepatitis c are more likely to report experiencing epigastric pain.

- **Age**

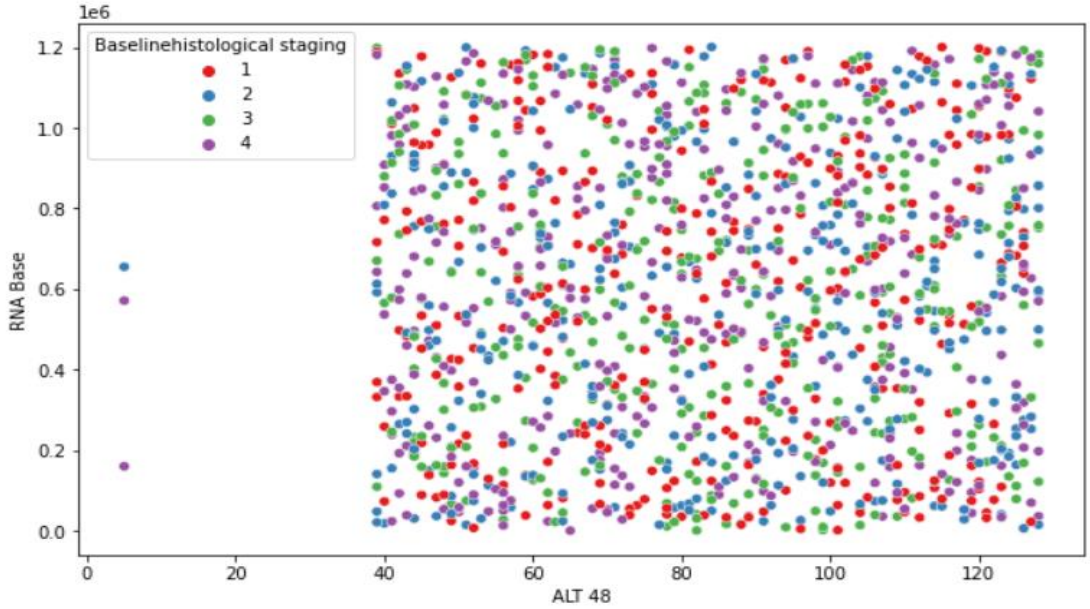
```
<AxesSubplot:xlabel='Age ', ylabel='ALT 48'>
```



Patients below age 35 show a slight increase in the number of stage 1 cases.

- **ALT 48**

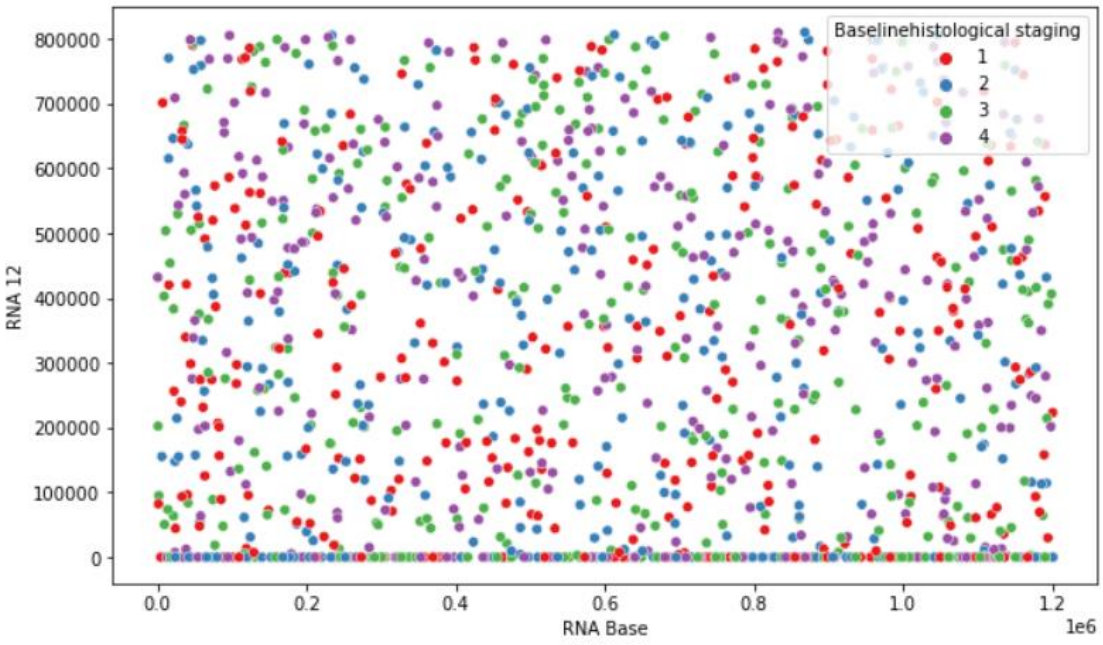
```
<AxesSubplot:xlabel='ALT 48', ylabel='RNA Base'>
```



The absence of data between 0 to 40 in ALT 48 might imply that disease state only manifest at ALT 48 levels above 40. It suggests that ALT 48 values below 40 may not associated with specific histological stages under study

- **RNA Base**

```
<AxesSubplot:xlabel='RNA Base', ylabel='RNA 12'>
```



Elevated level of RNA Base are more likely to be considered to be stage 3

WORKING OF ALGORITHMS

Algorithms used in 'Hepatitis C Detection Using Machine Learning' are Artificial Neural Networks, Support Vector Machine and Random Forest.

1.Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the human brain's neural networks. They consist of interconnected nodes (neurons) arranged in layers: an input layer, one or more hidden layers, and an output layer. Each neuron processes input data through weighted connections and activation functions to produce output. ANNs are particularly powerful in capturing complex patterns in data, making them suitable for a wide range of tasks, including classification, regression, and more. Algorithm:

1. Initialization:
    - Initialize the weights and biases for all layers of the network randomly.
2. Forward Propagation:
    - For each input sample, calculate the output of each neuron by computing the weighted sum of inputs and applying the activation function.
    - Pass the output from the input layer through the hidden layers and finally to the output layer.
3. Loss Calculation:
    - Compute the loss by comparing the predicted output with the actual target using a loss function (e.g., Mean Squared Error for regression or Cross-Entropy for classification).
4. Backpropagation:
    - Calculate the gradient of the loss function with respect to each weight and bias using the chain rule.
    - Update the weights and biases by moving them in the direction that reduces the loss, typically using an optimization algorithm like Gradient Descent.
5. Iteration:
    - Repeat the forward propagation, loss calculation, and backpropagation steps for multiple epochs until the model converges (i.e., the loss stops decreasing significantly).
6. Prediction:
    - After training, use the trained network to make predictions on new data.

Pseudocode:

Initialize weights and biases

For each epoch:

For each input sample:

Perform forward propagation through all layers

Calculate the loss Perform backpropagation to update weights and biases

End For

End For

Use the trained model to make predictions

2. Support Vector Machines (SVM)

Support Vector Machines (SVMs) are supervised learning models used for classification and regression tasks. SVMs work by finding the optimal hyperplane that best separates data points of different classes in a feature space. The data points that are closest to the hyperplane are called support vectors. SVMs are effective in high-dimensional spaces and are especially useful for binary classification.

Algorithm:

1. Data Preparation:
   o Represent the input data as feature vectors.
2. Hyperplane Selection:
   - Find the hyperplane that maximizes the margin between the two classes. The margin is the distance between the hyperplane and the nearest data point from either class.
3. Optimization:
   - Use optimization techniques (like Quadratic Programming) to minimize the classification error and maximize the margin.
4. Support Vectors:
   o Identify the support vectors, which are the data points closest to the hyperplane.
5. Kernel Trick (Optional):
   - If the data is not linearly separable, apply a kernel function (e.g., RBF, Polynomial) to transform the data into a higher-dimensional space where a linear separation is possible.
6. Prediction:
   o Use the optimal hyperplane (or decision boundary) to classify new data points.

Pseudocode:

Transform the input data using a kernel function (if needed)

Find the optimal hyperplane that maximizes the margin

Identify support vectors

For each new input sample:

Calculate the decision function based on the hyperplane

Assign the sample to a class based on the sign of the decision function

3. Random Forest (RF)

Random Forest is an ensemble learning method that builds multiple decision trees and merges their outputs to make a more accurate and stable prediction. Each tree is built using a random subset of features and data samples, making the model less prone to overfitting and more robust. Random Forest is widely used for both classification and regression tasks.

Algorithm:

- Bootstrap Sampling:
  - Randomly select samples from the training data (with replacement) to create multiple datasets (bootstrap samples).
- Tree Construction:
  - For each bootstrap sample, construct a decision tree by recursively splitting the data based on the best feature that maximizes the separation of classes (or minimizes the error in regression).
  - During the construction, at each split, randomly select a subset of features to choose the best split from.
- Aggregation:
  - After all trees are built, aggregate their predictions:
  - For classification, use majority voting (i.e., the class that gets the most votes is the final prediction).
  - For regression, average the predictions from all trees.
- Prediction:

  o Use the aggregated output from all trees to make the final prediction for new data.

Pseudocode:

For each tree in the Random Forest:

Bootstrap sample the data

Randomly select a subset of features for each split

Build the decision tree using the selected features and bootstrap sample

Repeat until the maximum depth or minimum sample size is reached

End For

For each new input sample

Pass the sample through each decision tree in the forest

Record the output (class or regression value) from each tree

Aggregate the outputs:

- For classification: Use majority voting to determine the final class label - For regression: Compute the average of the outputs to get the final prediction Return the aggregated result as the final prediction

PACKAGES AND FUNCTIONS

1. **Pandas**
   - read_csv()`: Reads a CSV file into a DataFrame.
   - DataFrame.shape`: Returns the dimensions of the DataFrame.
   - DataFrame.info()`: Provides a concise summary of the DataFrame.
   - DataFrame.replace()`: Replaces values in a DataFrame.
   - value_counts()`: Returns the counts of unique values.
   - concat()`: Concatenates two or more DataFrames.
   - isnull()`: Checks for missing values in the DataFrame.
   - fillna()`: Fills NA/NaN values using the specified method.
   - DataFrame.dtypes()`: Selects columns based on data types.
   - DataFrame.copy()`: Creates a deep copy of the DataFrame.
   - quantile()`: Returns values at the given quantile.
   - DataFrame.drop()`: Drops specified labels from rows or columns.
   - DataFrame.columns`: Accesses the column labels of the DataFrame.

2.Seaborn

   - heatmap()`: Plots a heatmap.
   - pairplot()`: Plots pairwise relationships in a dataset.

3. Matplotlib

   - figure()`: Creates a new figure.
   - show()`: Displays the figure.
   - title()`: Sets the title of the plot.
   - subplots()`: Creates a figure and a set of subplots.

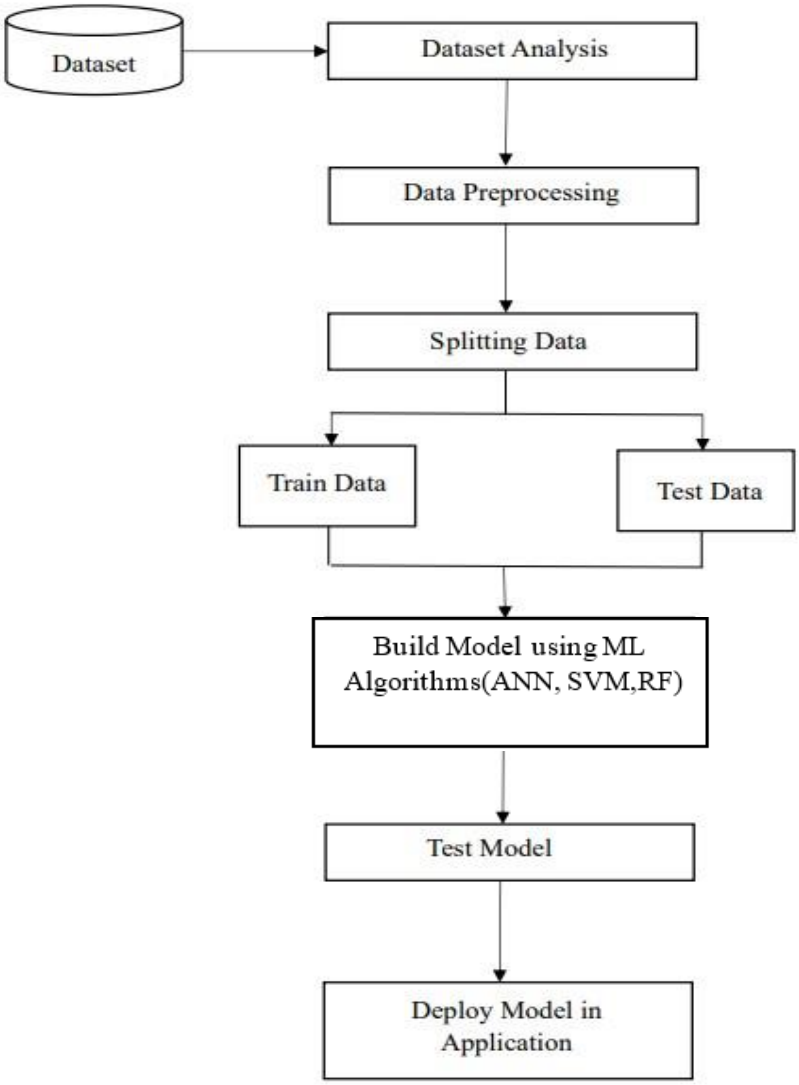
4. **Scikit-learn**
   - utils.resample(): Resamples arrays or sparse matrices.
   - preprocessing.LabelEncoder(): Encodes labels with a value between 0 and n_classes-1.
   - preprocessing.MinMaxScaler(): Scales features to a given range, typically between 0 and 1.
   - feature_selection.SelectKBest(): Selects features according to the k highest scores.
   - feature_selection.chi2(): Computes the chi-squared statistic for feature selection.

- feature_selection.SelectFromModel(): Selects features based on importance weights.
- ensemble.RandomForestClassifier(): A random forest classifier.
- experimental.enable_iterative_imputer(): Enables the `IterativeImputer`.
- impute.IterativeImputer(): Imputes missing values by modeling each feature as a function of the others.

5.Numpy

- triu(): Returns the upper triangle of an array.
- ones_like(): Returns an array of ones with the same shape and type as a given array

PROJECT PIPELINE



TIMELINE

Submission of project synopsis with Journal Papers - 22.07.2024

Project proposal approval - 26.07.2024

Presenting project proposal before the Approval Committee -29.07.2024 & 30.07.2024

Initial report submission - 12.08.2024

Analysis and design report submission - 16.08.2024

First project presentation - 21.08.2024 & 23.08.2024

Sprint Release I - 30.08.2024

Sprint Release II - 26.09.2024

Internal project presentation - 30.09.2024 & 01.10.2024

Sprint Release III - 18.10.2024

Submission of the project report to the guide - 28.10.2024

Final project presentation - 28.10.2024 & 29.10.2024

Submission of project report after corrections - 01.11.2024

SYSTEM DESIGN

Model Building Dataset is split in to two sets: 80% of datapoints for training and 20% for testing. train_test_split function is used to divide the data into training and testing sets.

1.Training and testing of ANN model

```
from sklearn.neural_network import MLPClassifier

ann_model = MLPClassifier(
    hidden_layer_sizes=(250, 150, 100, 50),
    random_state=55,
    max_iter=500,
    learning_rate_init=0.001,
    alpha=0.005,
    early_stopping=True,
    validation_fraction=0.10,
    n_iter_no_change=10,  # This line should be properly indented
)    # Number of iterations with no improvement before stopping


ann_model.fit(X_train, y_train)
print(ann_model.score(X_train,y_train))
y_pred = ann_model.predict(X_test)
ann_accuracy = accuracy_score(y_test, y_pred)
report= classification_report(y_test, y_pred)
print("ANN Accuracy:", ann_accuracy)
print(report)
```

Creates an ANN model with the following parameters:

- Hidden_layer_sizes=(250,150, 100,50) : Defines a network with two hidden layers containing 100 and 50 neurons, respectively.
- random_state=55: Ensures consistent results for reproducibility.
- Max_iter=500
- learning_rate_init=0.001
- alpha=0.005
- early_stopping=True
- validation_fraction=0.10
- n_iter_no_change=10

Trains the ANN model on the training data and this model is used to predict the target variable for the test data .The accuracy of the model is checked by comparing the predicted values (y_pred) to the actual values (y_test).

Test accuracy obtained for ANN model : 0.9433628318584071

Training accuracy obtained for ANN model: 0.9840637450199203

```
0.9840637450199203
ANN Accuracy: 0.9433628318584071
              precision    recall  f1-score   support

           1       0.95      0.94      0.95       121
           2       0.96      0.94      0.95       162
           3       0.94      0.94      0.94       141
           4       0.92      0.96      0.94       141

    accuracy                           0.94       565
   macro avg       0.94      0.94      0.94       565
weighted avg       0.94      0.94      0.94       565
```

```
models = {
    "ANN": ann_model
}
user_data_list = [
    [[5500, 4200000, 150000, 30, 50, 55, 60, 65, 70, 1000000, 950000, 600000, 400000]],
                #mild fibrosis
    [[4000, 3500000, 80000, 60, 100, 110, 120, 130, 140, 3000000, 2800000, 2000000, 1500000]],

        # cirrhosis
class_labels = {
    1: "Mild Fibrosis",
    2: "Mild-to-Moderate Fibrosis",
    3: "Spreading of Bridged Scarring",
    4: "Severe Cirrhosis"
}
# Iterate through each user data and make predictions
for user_data in user_data_list:
    print(f"Testing on data: {user_data}")
    for model_name, model in models.items():
        user_data1=pd.DataFrame(user_data, columns=['WBC', 'RBC', 'Plat', 'AST 1', 'ALT 1', 'ALT4', 'ALT 24',
       'ALT 48', 'RNA Base', 'RNA 4', 'RNA 12', 'RNA EOT'])
        prediction = model.predict(user_data1)
        class_label = class_labels[prediction[0]]
        print(f"{model_name} Prediction: {prediction[0]} - {class_label}")
    print("\n")
```

```
Testing on data: [[5500, 4200000, 150000, 30, 50, 55, 60, 65, 70, 1000000, 950000, 600000, 400000]]
ANN Prediction: 2 - Mild-to-Moderate Fibrosis


Testing on data: [[4000, 3500000, 80000, 60, 100, 110, 120, 130, 140, 3000000, 2800000, 2000000, 1500000]]
ANN Prediction: 4 - Severe Cirrhosis
```

2 . Training and testing of SVM model

```
from sklearn.svm import SVC
svm_model = SVC(C=1,                    # Regularization parameter (lower value)
    kernel = 'rbf',
    gamma= 'scale' ) #kernel='linear', C=1, random_state=48)
svm_model.fit(X_train, y_train)
print(svm_model.score(X_train,y_train))
y_pred = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test,y_pred)
print("SVM Accuracy:", svm_accuracy)
```

```
0.7972554227534308
SVM Accuracy: 0.6442477876106195
```

Creates an SVM model with the following parameters:

- C=1 : Sets the regularization parameter (a higher value implies a smaller margin and potentially more overfitting).
- kernel='rbf : Uses a rbf kernel for the SVM.
- gamma=’scale’

Trains the SVM model on the training data and this model is used to predict the target variable for the test data .The accuracy of the model is checked by comparing the predicted values (y_pred) to the actual values (y_test).

Test accuracy obtained for SVM model : 0.7972554227534308

Training accuracy obtained for SVM model : 0.6442477876106195

3. Training and testing of RF model

```
# 3. Random Forest (RF)
rf_model = RandomForestClassifier(
    n_estimators=375,           # Adjusted
    max_depth=9,                # Adjusted
    min_samples_split=10,
    min_samples_leaf=2,
    max_features=0.5,
    bootstrap=True,
    random_state=42
)

rf_model.fit(X_train, y_train)
print(rf_model.score(X_train,y_train))
y_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test,y_pred)
report=classification_report(y_test,y_pred)
print("Random Forest Accuracy:", rf_accuracy)
print(report)
```

```
0.983178397521027
Random Forest Accuracy: 0.7380530973451327
              precision    recall  f1-score   support

           1       0.72      0.71      0.71       139
           2       0.77      0.71      0.74       154
           3       0.74      0.72      0.73       139
           4       0.73      0.82      0.77       133

    accuracy                           0.74       565
   macro avg       0.74      0.74      0.74       565
weighted avg       0.74      0.74      0.74       565
```

Creates an RF model with the following hyperparameters:

- max_depth: 9
- min_samples_leaf: 2
- min_samples_split: 10
- n_estimators: 375
- random_state: 42
- bootstrap=True

Trains the RF model on the training data and this model is used to predict the target variable for the test data .The accuracy of the model is checked by comparing the predicted values (y_pred) to the actual values (y_test).

Test accuracy obtained for RF model : 0.7380530973451327

Training accuracy obtained for RF model : 0.983178397521027