Classification of infectious state of Hepatitis C virus affected patients using classification algorithms

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

SAI ROHITH PASHAM	**221710315046**
ACHYUT RANJAN	**221710315002**
SAI KRISHNA	**221710315029**
VIHAR DASARI	**221710315061**

Under the esteemed guidance of

Mr. R. Jethya

Assistant professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM

(DEEMED TO BE UNIVERSITY)

HYDERABAD CAMPUS

MAY-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

HYDERABAD CAMPUS

DECLARATION

We herby declare that the major project entitled "**Classification of infectious state of Hepatitis C virus affected patients using classification algorithms**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s)	Name(s)	Signature(s)
221710315046	**SAI ROHITH.P**	
221710315002	**ACHYUT RANJAN**	
221710315029	**SAI KRISHNA**	
221710315061	**VIHAR DASARI**	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that Project entitled "**Classification of infectious state of Hepatitis C virus affected patients using classification algorithms**" is submitted by **SAI ROHITH (221710315046), ACHYUT RANJAN (221710315002), SAI KRISHNA (221710315029), VIHAR DASARI (221710315061)** in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering. The Major Project has been approved as it satisfies the academic requirements.

Project Guide

Mr. R.Jethya

Assistant Professor

Head of the Department

Dr. S. Phani Kumar

Professor

ACKNOWLEDGMENT

Our Project would not have been successful without the help of several people. we would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are extremely thankful to our honourable Pro-Vice Chancellor, **Prof. N. Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of our project.

We are highly indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the project.

We are very much obliged to our beloved **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in completion of this project.

We hereby wish to express our deep sense of gratitude to **Dr. S Aparna**, Assistant Professor, Department of Computer Science and Engineering, School of Technology and to **Guide**, **Mr. R Jethya,** Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the project.

We are also thankful to all the staff members of Computer Science and Engineering department who have cooperated in making our project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our project work.

Sincerely,

SAI ROHITH(221710315046)

ACHYUT RANJAN(221710315002)

SAI KRISHNA(221710315029)

VIHAR DASARI(221710315061)

ABSTRACT

Hepatitis C is caused by the Hepatitis C virus (HCV) and it affects the liver organ of human body. It can cause acute as well as chronic hepatitis that can be mild or can be lifelong and life-threatening. It ultimately leads to liver cancer. It is estimated that 71 million people are effected by the HCV disease annually and resulted in 4,00,000 deaths per year[1]. After getting affected by the virus, the patient could go through 4 stages: Portal fibrosis(F1), few septa(F2), many Septa(F3), cirrhosis(F4).

So, in this project, we propose a machine learning model that classifies the stages of HCV affected patients. The dataset is taken from UCI repository that contain instances of Egyptian patients. Synthetic minority oversampling technique(SMOTE) is applied and classification is done using classification algorithms like KNN, Random Forest, etc and results are observed.

Keywords:
SMOTE, Random Forest, K-nearest neighbours.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 MOTIVATION

Since it is a highly infectious disease that spreads through exposure from even small amount of blood, it is very important to diagnose it and check for the stage the patient is in out the four stages: portal fibrosis, few septa, many septa, and cirrhosis. The region that is most affected is the Mediterranean and especially Egypt which has shown a prevalence rate of 13 to 15 percent. Liver-biopsy is the medical procedure that is generally used to detect such diseases such as cirrhosis, any infection or any cancerous cells. But, this method is very expensive and not so feasible for the patients with mild conditions. So, in modern day many non-invasive ways of detecting the stages are put forward. Machine learning is currently the upcoming field to find the condition of the affected patients. We can get results with some lab tests and these results are both fast and accurate by using machine learning.

1.2 BACKGROUND

This research is focussed on avoiding the use of invasive methods such as liver biopsy to predict the stage of HCV in a patient. This can also help the doctors in determining quickly and accurately, so that they can focus more on the treatment part. Since there are about 160 million people who are annually infected by this HC virus and also 350000 people losing their lives, it is the matter of importance that we have to put forth more efficient ways in diagnosing the virus.

1.3 EXISTING SYSTEM

The existing system to find the stage of HCV affected patient is through invasive procedure that are very expensive and it require a surgery. Another way is through machine learning and the accuracy of the current model that is built using the decision tree algorithm has less accuracy and hence it is not recommended by the doctors

1.4 OBJECTIVE

So, our main objective is to analyse the dataset that contains the instances of the Egyptian patients who are effected by HCV and build a model that has high enough accuracy so as to help the doctors an

1.5 STRUCTURE OF PROJECT

- Requirement gathering
- Preprocessing the data
- Building the model
- Testing
- Output and Results.

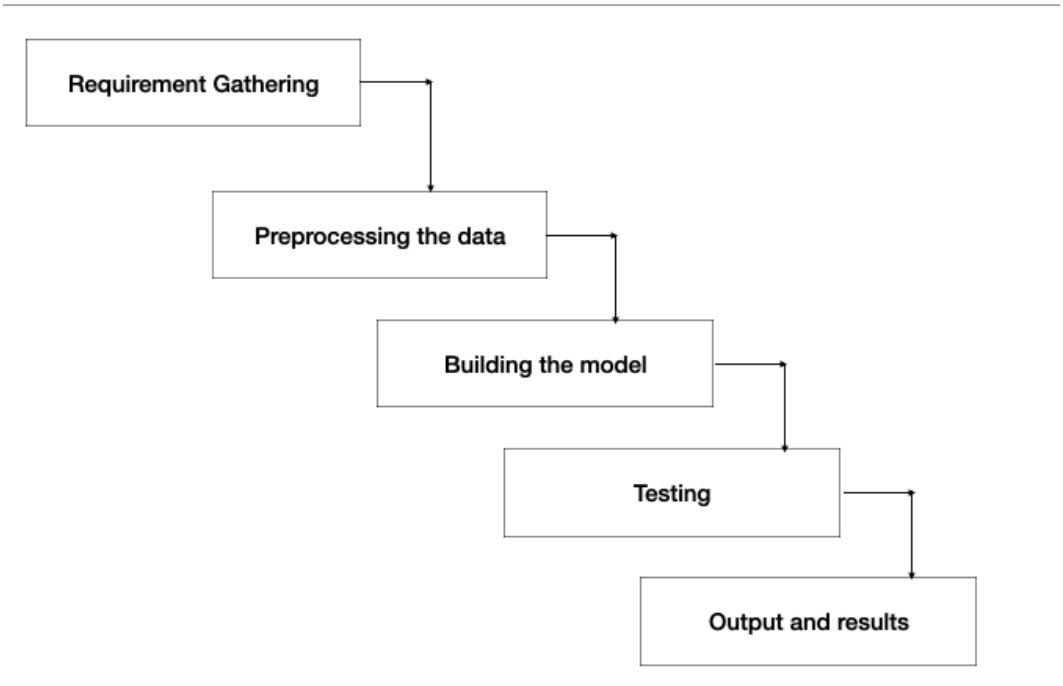

Figure 1.5.1 Project structure

1.5.1 REQUIREMENT GATHERING

It is the first and foremost step of any project. The first requirement of our project is gathering the dataset. Our dataset is available on the internet and we have taken it from the University of California Irvine repository[2]. It has 29 columns and 1385 rows. We have referred through various similar paper which are published on the IEEE papers and analysed the steps of how to create the model. we took references from those papers and did literature survey of some papers and amassed all the Requisites of the project in this stage.

1.5.2 PREPROCESSING THE DATA

The next step that we have performed is to make the dataset ready for building the models. Here we check for null values and duplicate values. If present we need to remove them and also check if it is required to scale the data.

1.5.3 IMPLEMENTATION

The implementation of the project is performed by building the model on dataset using various classification algorithms. We used algorithms like k-nearest neighbours, random forest and logistic regression. After fine-tuning the algorithms by hyper parameters tuning, we use the best result giving algorithm.

1.5.4 TESTING

After splitting the dataset to test and train and building the model on train, we use the test data to see how accurate our model is performing on unseen data. Now, any sample input can be given and output can be derived with accuracy based on the result we got in the testing.

1.5.5 OUTPUT MODEL

After testing the model which is implemented by following the required steps and tested by using test dataset, we have successfully created a model that accurately predicts in which stage the HCV affected patient is in. A windows system with python along with several packages should be installed in order to be able to run the model.

1.6 ALGORITHMS USED

This project uses the k-nearest neighbours, random forest and logistic regression along with python and some its packages like numpy, pandas, sklearn, matplotlib, etc. which will be discussed in this section.

1.6.1 KNN

Knearest neighbour(KNN) algorithm is a machine learning algorithm that is supervised and it can be used to solve both both kind of problems: classification as well as regression but it is predominantly use for classification problems. It is a lazy learner algorithm which means that it will use all of the train data to make the prediction about the query instance. It uses feature similarity for prediction. It means it uses the measure how similar the new input is to the features in the dataset and it assumes that similar classes occur together and does the prediction. It could be understood from the following steps:

1. First thing that is needed is a dataset, so, we have to give the training data to the build the model.
2. Next thing is to choose the k-value. It can be any value and it should preferably be odd. It is because in odd value we can get one class as majority and other as minority. So, it needs to be odd.
3. Next step is to find the distance from the new input to its nearest k-neighbours.
4. To find the distance, we can use many distance metrics namely, Euclidean distance, Manhattan distance, Hamming distance, etc.
5. Now it will predict the class as the class which is greater in the nearest neighbours.

In the example below, we can clearly understand the concept of K-nearest neighbours algorithm.

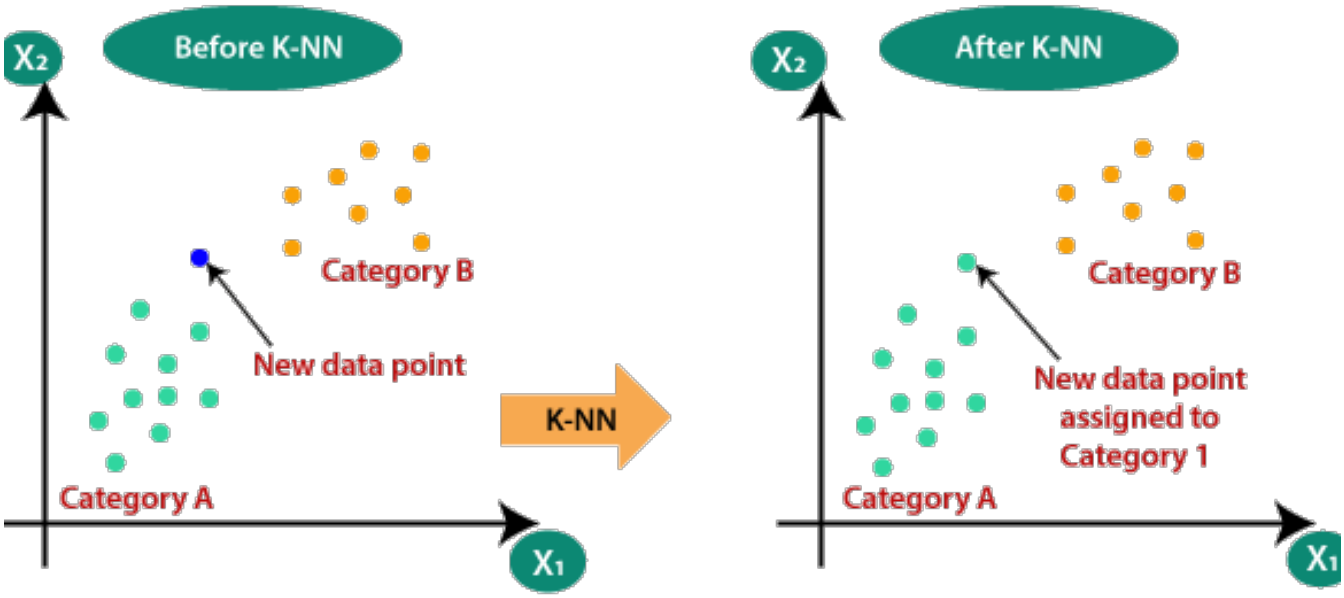

Fig 1.6.1.1 KNN algorithm

From the left side in fig 1.6.1.1 we can see that the new data point has been plotted in the graph and its nearest 3 neighbours are considered by using the euclidean distance and hence category A was in majority and the model predicts that new datapoint belongs to category A as shown in right side of the figure.

1.6.2 RANDOM FOREST

Random forest is another supervised machine learning algorithm that is flexible and easy to use. It delivers good performance even without hyper-parameter tuning most of the times. It is an ensemble technique which means that it builds many number of decision trees and it is a combination of learning models that improve the overall accuracy.

In simple terms, it builds multiple decision trees and then combines them together to improve the accuracy of the model.

It can also be used for both classification and regression problems. While increasing the trees, the random forest adds more randomness to the model. When splitting a node, it looks for the best feature among a random subset of features rather than the most important feature. As a consequence, there is a lot of variety, which leads to a better model. The higher the number of trees, the higher the accuracy and prevents the problem of overfitting.

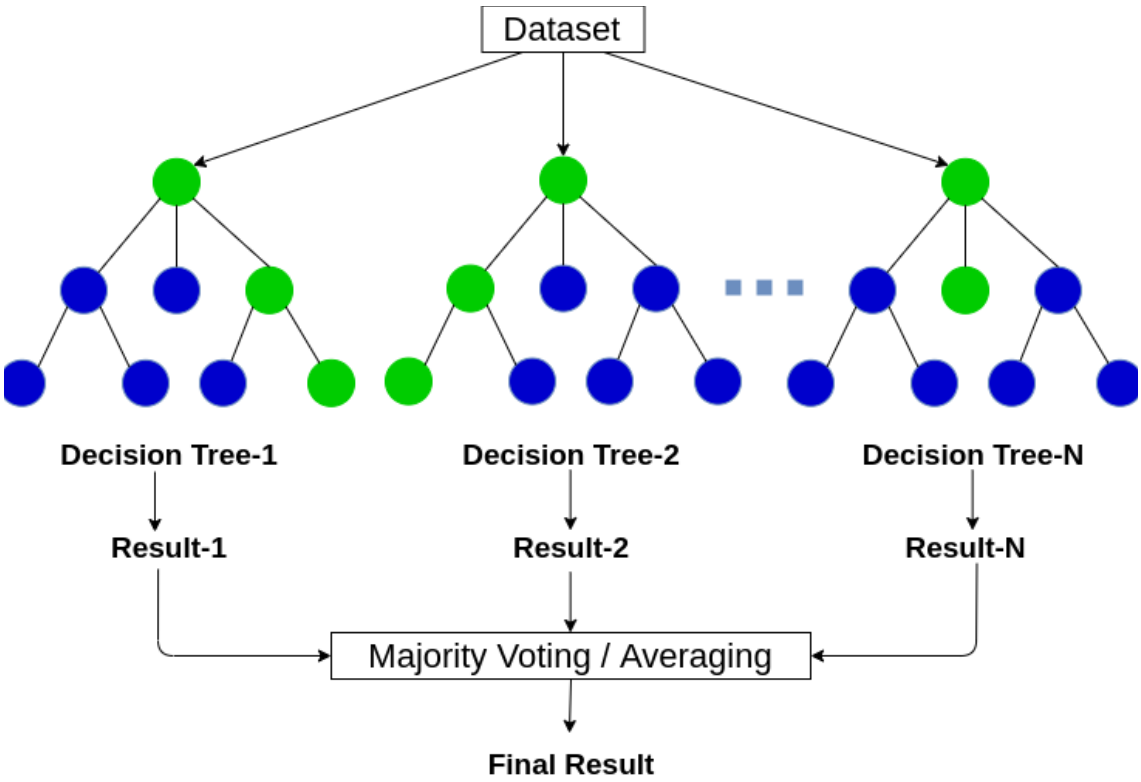

Fig 1.6.2.1 Random Forest

From the above figure 1.6.2.1, we can see that from the dataset, we generate many decision trees and the features are selected randomly. Then next we combine all the results of the decision tress and build the best model that gives us the best accuracy.

1.6.3 SMOTE

While working with imbalanced datasets, the problem is that the model will perform poorly on the classes that are in minority. So one solution is increase the number of minority instances and SMOTE which stands for synthetic minority oversampling technique is used to create those synthetic minority instances.

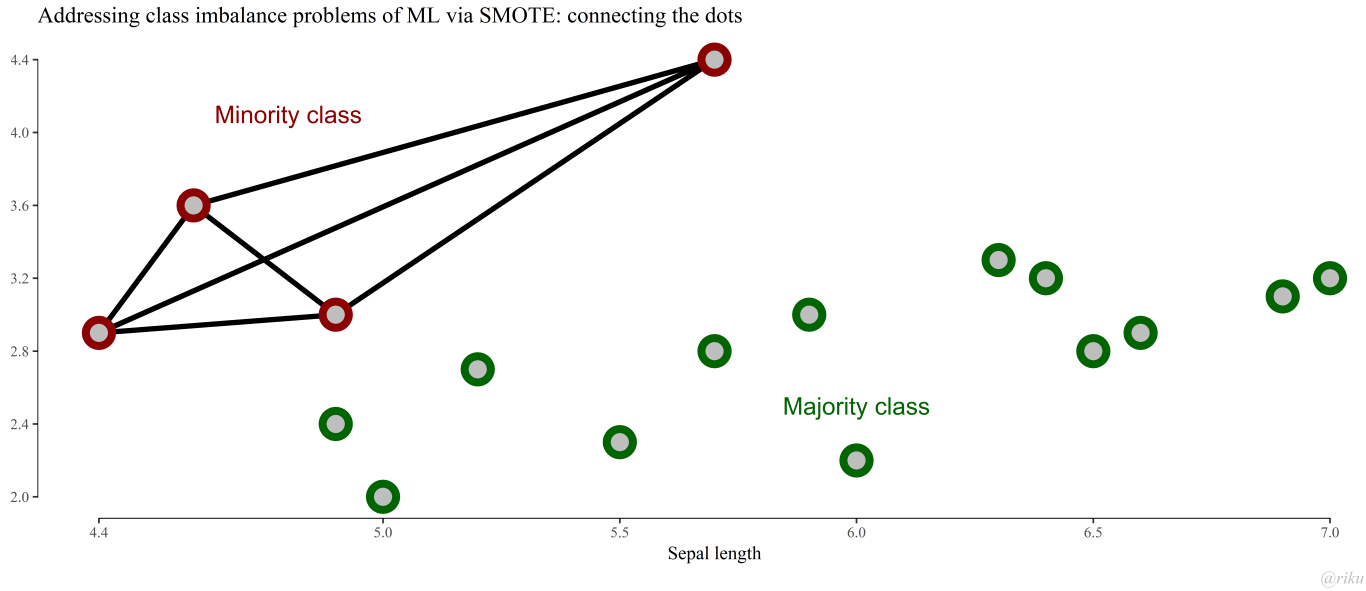

Fig 1.6.3.1 Before SMOTE

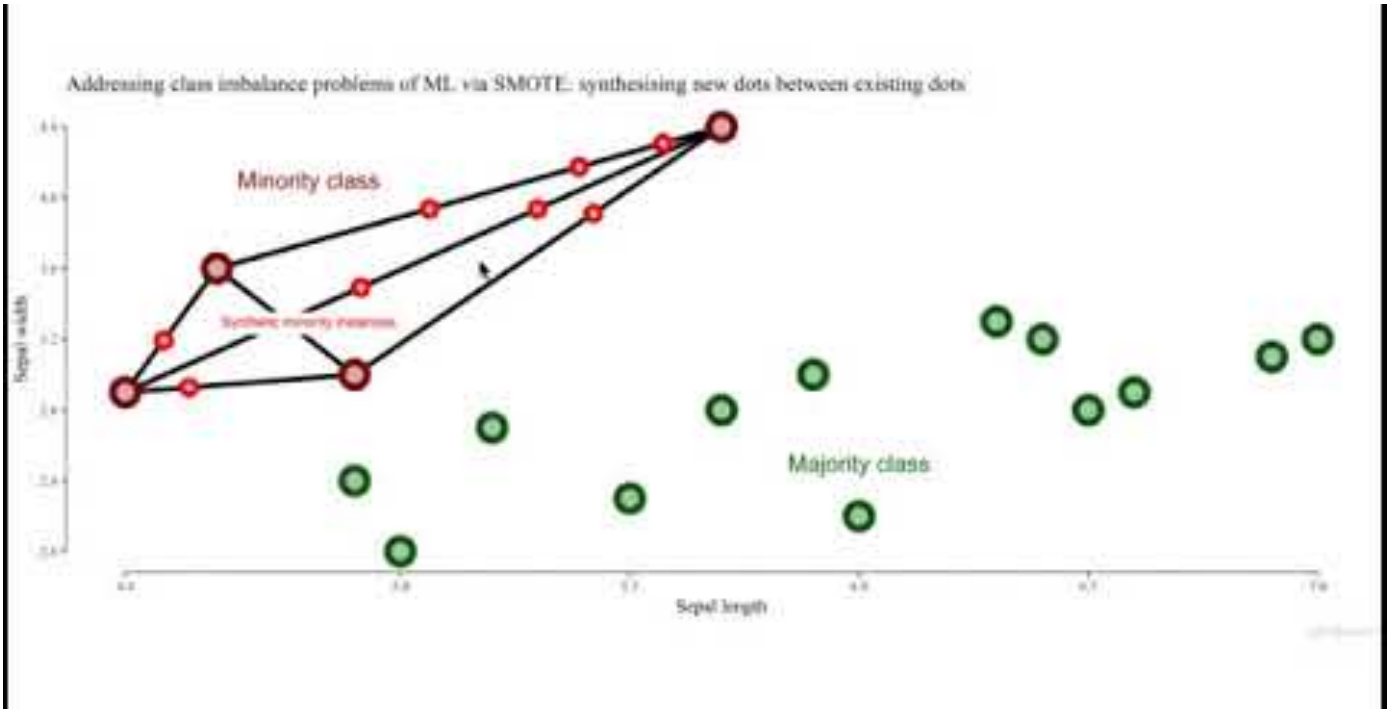

Fig 1.6.3.2 After SMOTE

From the figure 1.6.3.1, we can see that the green class is in majority and the red class is in minority. So, this leads to class imbalance and the model won't show good results because of this. So, what smote does is that it use nearest neighbours and creates new synthetic instances to match the number of majority classes as shown in the figure 1.6.3.2.

In order to use this algorithm, we used a software called Weka. Weka is a open source software that provides tools for data preprocessing and also to apply smote. We applied it 8 times and that's where we found the best balance of classes and the size of the dataset improved highly.

2. LITERATURE SURVEY

N.Zayed, A. B.Awad, W.El-Akel, W.Doss, T.Awad, A.Radwan, and M.Mabrouk, "The assessment of data mining for the prediction of therapeutic outcome in 3719 egyptian patients with chronic hepatitis c," *Clinics and research in hepatology and gastroenterology*, vol. 37, no. 3, pp. 254–261, 2013 [3]

- The authors: N.Zayed, A.Awad, W.Doss, T.Awad, A.Radwan, M.mabrouk have developed a model that uses decision tree for prediction.
- The dataset contains 3719 rows and 19 attributes of Egyptian patients of HCV.
- Using the data they had available then, they could only identify if the patient is effected by HCV or not.
- They developed a model that predicts for the end virological response to PEGIFN and ribavirin therapy in HCV patients.
- The DT model was issued to a 10 fold cross validation which is done internally and externally.
- The accuracy of the model that they developed was 73%.

T. Orczyk and P. Porwik, "Liver fibrosis diagnosis support system using machine learning methods," in *Advanced Computing and Systems for Security*. Springer, 2016, pp. 111–121.[4]

- The authors T.Orczyk and P.Porwik have carried out various experiments on a dataset that is built by them that proved usability and provided decent accuracy.
- The dataset that they used had 290 instances and 26 columns of the patients data at Silesian Medical University.
- They used various feature selection algorithms such as CFS, ReliefF, SVM, Genetic Wrapper, Single separate, Single accuracy.
- Then they used classification algorithms such as Random forest, oneR, Decision Table.
- After applying all the steps and algorithms, they obtained an accuracy of around 70%.

- The drawback of this model is that there are very few instances for the model to give accurate results.
- Also the accuracy they got is not suitable for sensible medical applications that require highest accuracy possible.

S. Hashem, G. Esmat, W. Elakel, S. Habashy, S. Abdel Raouf, S. Dar- weesh, M. Soliman, M. Elhefnawi, M. El-Adawy, and M. ElHefnawi, "Accurate prediction of advanced liver fibrosis using the decision tree learning algorithm in chronic hepatitis c egyptian patients," *Gastroen- terology research and practice*, vol. 2016, 2016.[5]

- The authors S.Hashem, G.Esmat, W.Elakel, S.Habashy, S.Abdel Raouf, S.Darwesh,M.Soliman, M.ELhefnawi, M.El-Adawy have developed a model that tries to accurately predict the advanced stages of fibrosis and also starting stages using machine learning algorithms such as decision tree.
- They collected the data from about 39,567 patients from the Egyptian national committee for control of Viral hepatitis.
- They used MedCalc, excel, and weka tools for analysing and applying the algorithm.
- The output were 5 stages- F0 to F4, where F3 and F4 are the advanced stages of liver fibrosis.
- They used about 66% for train i.e 22690 records for training and the rest 16877 records for test data.
- They used alternating decision tress to build the model and when they have tested the model, they got 84% accuracy.

3.PROBLEM ANALYSIS

3.1 EXISTING APPROACH

In the existing approaches, the decision tree based classifiers are the mostly used classification techniques. Various classification and feature selection algorithms are used and the prediction is done.

3.2 DRAWBACKS

Due to more focus on decision tree based models, experiments using other classifiers is missing and they were not explored and the results obtained were not high enough to be used in the real world application because it is a highly sensible matter regarding health.

3.3 PROPOSED MODEL

We propose a model that classifies the stages of HCV affected patients using various classification algorithms. We intend to improve upon the accuracy that the previous research have established and create a more reliable model with high accuracy. In order to accomplish this, we have used class balancing techniques like SMOTE and used classifier like k-nearest neighbours classifier, Random forest classifier and logistic regression.

3.4 SOFTWARE AND HARDWARE

SOFTWARE REQUIRED

- Anaconda Navigator
- Jupyter Notebook
- Python 3.8
- Packages like numpy, pandas, sklearn etc

HARDWARE

- OS - windows, linux, Macintosh
- Processor - min intel i3
- Ram - 4 GB
- Hard disk - minimum 250 GB

3.5 ABOUT DATASET

The dataset used in this project has 1385 records and 29 attributes. The dataset is taken from the university of California Irvine repository. It contains the data about the Egyptian patients. A description about the attributes is given below.

1. Age - It mentions the age of the patient.
2. Gender - It has 2 values, 1 represents Male and 2 represents Female.
3. BMI - It means Body mass index and it tell us the measure of body fat on the basis of height and weight.
4. Fever - It has 2 values, 1 for absent and 2 for present.
5. Nausea - It means the sensation to vomit and it also has 2 values 1 for absent and 2 for present.
6. Headache - It also has 2 values. 1-absent, 2-present.
7. Diarrhoea - It is distinguished by loose stool and also liquid stool. 1-condition absent and 2- condition present.
8. Fatigue - it is characterized by feeling of tiredness. 1 -condition absent and 2-condition present.
9. Bone ache - it is characterised by the sensation of pain in the bones, especially near the joints. 1 - absent and 2- present.
10. Jaundice - It is a medical condition where skin and eyes turn yellow in color due to increase in levels of bilirubin. 1- absent, 2- present.
11. Epigastria pain - It is the pain in the upper abdomen region. 1- absent, 2- present.
12. WBC - it refers to white blood cells. They help in fighting against the foreign bodies that enter the body. The normal range of them is between 4,500 to 11,000.

13. RBC - It refers to red blood cells. They are the cells that help in the transportation of oxygen throughout the body. The normal range is between 4.5-5.9 x 10^6/microliter for men and 4.1-5.1 x 106 microliter for women.

14. HGB - It refers to Haemoglobin. It is what gives the blood red color. It is a protein and it also helps in the movement of oxygen. The normal count is 14 to 18 gm/lit for men and 12 to 16 gm/lit for women.

15. Platelet - They are the tiny cells that help in clotting the blood whenever there is cut on the body. The normal count is 150,000 to 450,000 per microliter of blood.

16. AST(1 week) - It refers to Aspartate transaminase ratio after 1 week. It is an enzyme that is found in liver. It can be linked to liver damage as when damage to liver occurs it is released to blood. So higher the value, higher the injury severity.

17. ALT(1 week) - It refers to Alanine transaminase. It is also an enzyme and similar to AST, higher levels of ALT indicate that there is damage done to the liver.

18. ALT(4 weeks) - ALT levels in blood are measure after 4 weeks.

19. ALT(12 weeks) - ALT levels in blood are measure after 12 weeks.

20. ALT(24 weeks) - ALT levels in blood are measure after 24 weeks

21. ALT(36 weeks) - ALT levels in blood are measure after 36 weeks

22. ALT(48 weeks) - ALT levels in blood are measure after 48 weeks

23. RNA Base - it is important in synthesis of proteins.

24. RNA(4) - RNA measured in blood after 4 weeks.

25. RNA(12) - RNA measured in blood after 12 weeks.

26. RNA(EOT) - RNA measured at the end of treatment.

27. RNA EF - RNA Elongation factor.

28. Baseline Histological Grading - It has 1 to 16 grades based upon the tumour size.

29. Baseline Histological - It has the 4 stages F1:portal fibrosis, F2: few septa, F3: many septa, F4: Cirrhosis.

The Baseline Histological is the output attribute and the rest all are considered as input.

4. IMPLEMENTATION

4.1 OVERALL FLOW OF PROJECT

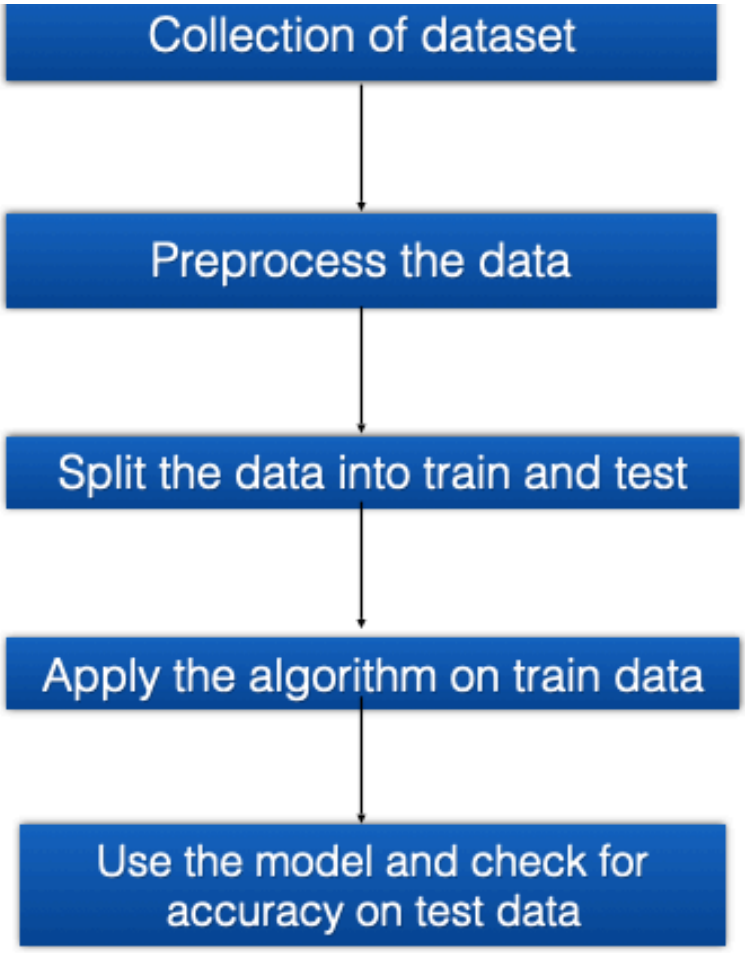

Fig 4.1.1 Steps involved in project

Here in the figure we can see the steps involved in the project. So, first step is to collect the dataset and store it in CSV(comma separated values) format. Then the next step is to preprocess the data i.e make the data ready for building the model.

Then after preprocessing the data, the next step is to split the dataset in to 2 parts: training and testing. Then use the train data to build the classification model by using classification algorithms. Then the last step of the project is to test the model on test data and report the accuracy.

4.2 IMPORTING DATASET

Before importing the dataset the first step of any machine learning project is to import the standard libraries that are required for any project as shown in figure.

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
```

Fig-4.2.1 Importing the libraries

The next step now is to import the dataset and it is done as shown in the figure 4.2.2 below

Reading the dataset

```
In [2]: df = pd.read_csv("dataset.csv")
        df
```

Out[2]:

	Age	Gender	BMI	Fever	Nausea/Vomting	Headache	Diarrhea	Fatigue & generalized bone ache	Jaundice	Epigastric pain	...	ALT 36	ALT 48	ALT after 24 w	RNA Base	RNA 4	RNA 12	R E
0	56	1	35	2	1	1	1	2	2	2	...	5	5	5	655330	634536	288194	
1	46	1	29	1	2	2	1	2	2	1	...	57	123	44	40620	538635	637056	3368
2	57	1	33	2	2	2	2	1	1	1	...	5	5	5	571148	661346	5	7359
3	49	2	33	1	2	1	2	1	2	1	...	48	77	33	1041941	449939	585688	7444
4	59	1	32	1	1	2	1	2	2	2	...	94	90	30	660410	738756	3731527	3389
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1380	44	1	29	1	2	2	2	1	1	1	...	63	44	45	387795	55938	5	
1381	55	1	34	1	2	2	1	1	1	1	...	97	64	41	481378	152961	393339	735
1382	42	1	26	2	2	1	1	1	2	1	...	87	39	24	612664	572756	806109	3437
1383	52	1	29	2	1	1	2	2	2	1	...	48	81	43	139872	76161	515730	24
1384	55	2	26	1	2	2	2	1	2	1	...	64	71	34	1190577	628730	5	

Fig-4.2.2 Importing the dataset

read_csv function of the pandas library is used to read the dataset and store it as a data frame with the variable name as df.

Now to know the dimensionality of the dataset we used shape property. We have 1385 rows and 29 attributes.

4.3 PREPROCESSING THE DATASET

Preprocessing of dataset is the process of making the data ready for machine learning model. So, here we have to check if there are any null values and duplicate values and remove them if present.

```
import missingno as msno
msno.matrix(df)
plt.show()
```

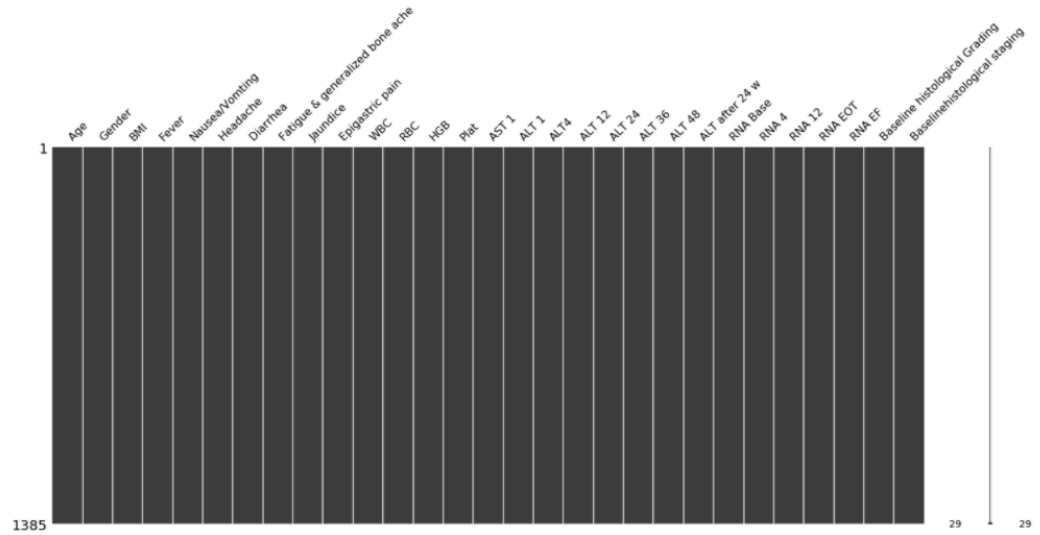

Fig-4.3.1 Checking for null values

```
df.duplicated().sum()
```

```
0
```

Fig-4.3.2 Checking for duplicate records.

From both the figures above it can be concluded that there are no null values or duplicate records in our dataset.

```
df.dtypes
```

```
Age                                  int64
Gender                               int64
BMI                                  int64
Fever                                int64
Nausea/Vomting                       int64
Headache                             int64
Diarrhea                             int64
Fatigue & generalized bone ache      int64
Jaundice                             int64
Epigastric pain                      int64
WBC                                  int64
RBC                                float64
HGB                                  int64
Plat                               float64
AST 1                                int64
ALT 1                                int64
ALT4                               float64
ALT 12                               int64
ALT 24                               int64
ALT 36                               int64
ALT 48                               int64
ALT after 24 w                       int64
RNA Base                             int64
RNA 4                                int64
RNA 12                               int64
RNA EOT                              int64
RNA EF                               int64
Baseline histological Grading        int64
Baselinehistological staging         int64
dtype: object
```

Fig-4.3.3 Datatypes of Attributes

```
df.describe()
```

	Age	Gender	BMI	Fever	Nausea/Vomting	Headache	Diarrhea	Fatigue & generalized bone ache	Jaundice	Epigastric pain	...	ALT 3
count	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	1385.000000	...	1385.00000
mean	46.319134	1.489531	28.608664	1.515523	1.502527	1.496029	1.502527	1.498917	1.501083	1.503971	...	83.11769
std	8.781506	0.500071	4.076215	0.499939	0.500174	0.500165	0.500174	0.500179	0.500179	0.500165	...	26.39903
min	32.000000	1.000000	22.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	5.00000
25%	39.000000	1.000000	25.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	61.00000
50%	46.000000	1.000000	29.000000	2.000000	2.000000	1.000000	2.000000	1.000000	2.000000	2.000000	...	84.00000
75%	54.000000	2.000000	32.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	...	106.00000
max	61.000000	2.000000	35.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	...	128.00000

8 rows × 29 columns

Fig 4.3.4 Statistical analysis

From the above figure 4.3.3, we can see that we have 26 integer and 3 floating type attributes. From the figure 4.3.4, we can see that some of the attributes have values that are too high in value and because of this we need to scale the values. We need to scale the data because the

machine learning algorithms use Euclidean distance and it will be more biased to attribute with high magnitude than the attribute with low magnitude.

```
df['Baselinehistological staging'].value_counts()
```

```
4    362
3    355
1    336
2    332
Name: Baselinehistological staging, dtype: int64
```

Fig-4.3.5 Number of patients in each stages.

From the figure 4.3.5, we can see that there are 336, 332, 355, 362 records of patients who are in 1,2,3 and 4th stage of the HCV respectively.

4.4 SPLITTING INTO TRAIN AND TEST

```
X = df.drop(['Baselinehistological staging'],axis=1)
y = df['Baselinehistological staging']
```

Spliting in to train and test

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=18)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1108, 28)
(277, 28)
(1108,)
(277,)
```

Fig-4.4.1 Splitting into train and test

From the above Figure 4.4.1, we can see that X is taken as input and y is the output to the model. In X, we gave every column except the baseline histological staging which is the output, as input.

Then we imported train_test_split function from sklearn package and performed the dataset splitting. We have 1108 rows in train and 277 in test as we gave test size as 20%. Random state is taken as 18.

4.5 SCALING THE DATA

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# Scaling for training data
scaled_X_train = pd.DataFrame(scaler.fit_transform(X_train),columns=X_train.columns)
scaled_X_train

# Scaling for test data
scaled_X_test = pd.DataFrame(scaler.transform(X_test),columns=X_test.columns)
scaled_X_test
```

Fig-4.5.1 Scaling the data

Now we are scaling the data using the standard scaler where the mean is taken as 0 and all the values of an attribute are within one standard deviation away from the mean.

4.6 BUILDING THE MODEL

Logistic regression algorithm is used to predict the stages in this multi class classification. First we need to import logistic regression from sklearn.linear_model.

Then the next step is to create an object for that model.

Now we need to fit the input and output of the training data into the algorithm and build the model.

Next we need to predict the accuracy of the train data and check how the model performs for the test data.

```
# importing the model class
from sklearn.linear_model import LogisticRegression
# creating a object for that model
lm = LogisticRegression()
# fitting the input and output of training data to the object and building the model
lm.fit(scaled_X_train,y_train)
```

```
LogisticRegression()
```

```
# predicting the output of the training data
y_train_pred = lm.predict(scaled_X_train)
y_train_pred
```

```
array([3, 4, 4, ..., 2, 2, 3])
```

```
from sklearn.metrics import accuracy_score
print("Training data accuracy:",accuracy_score(y_train,y_train_pred))
log_train = accuracy_score(y_train,y_train_pred)
```

```
Training data accuracy: 0.33574007220216606
```

```
y_test_pred = lm.predict(scaled_X_test)
print("Testing data accuracy:",accuracy_score(y_test,y_test_pred))
log_test = accuracy_score(y_test,y_test_pred)
```

```
Testing data accuracy: 0.2490974729241877
```

Fig-4.6.1 Logistic regression

From the figure 4.6.1, we can see that the accuracy for train data is 33% and for test it is 24%

Next we will try using K-NN

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train, y_train)
training_score = clf.score(X_train, y_train)
print("Train accuracy:",training_score)
test_score = clf.score(X_test, y_test)
print("Test accuracy",test_score)
```

```
Train accuracy: 0.5406137184115524
Test accuracy 0.2490974729241877
```

Fig-4.6.2 KNN

For KNN, we got around 54% for train and 25% for test data.

Now, trying on random forest classifier.

```
# importing the model class
from sklearn.ensemble import RandomForestClassifier
# creating a object for that model
rfc = RandomForestClassifier(n_estimators = 100)
# fitting the input and output of training data to the object
rfc.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
from sklearn.metrics import classification_report
y_pred_train = rfc.predict(X_train)
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00       278
           2       1.00      1.00      1.00       260
           3       1.00      1.00      1.00       291
           4       1.00      1.00      1.00       279

    accuracy                           1.00      1108
   macro avg       1.00      1.00      1.00      1108
weighted avg       1.00      1.00      1.00      1108
```

```
y_pred_test = rfc.predict(X_test)
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           1       0.23      0.33      0.27        58
           2       0.25      0.24      0.24        72
           3       0.19      0.22      0.21        64
           4       0.41      0.28      0.33        83

    accuracy                           0.26       277
   macro avg       0.27      0.26      0.26       277
weighted avg       0.28      0.26      0.27       277
```

Fig-4.6.3 Random Forest

For random forest we got an accuracy of 100% for training data and only 26% for the test data. This means that the model has overfitted i.e the model works well on the train data or seen data but it doesn't show the same performance when it come to test data or unseen data.

So, from our observations, we can see that the all three of the models have performed poorly. There can be 2 reasons for this: class imbalance and not enough records in the dataset for the models to show good performance.

4.7 APPLYING SMOTE

To balance the classes, we use SMOTE(synthetic minority oversampling technique). We use this technique by using weka which is an open source machine learning tool. We apply it 8 times where we found the best balance of classes.

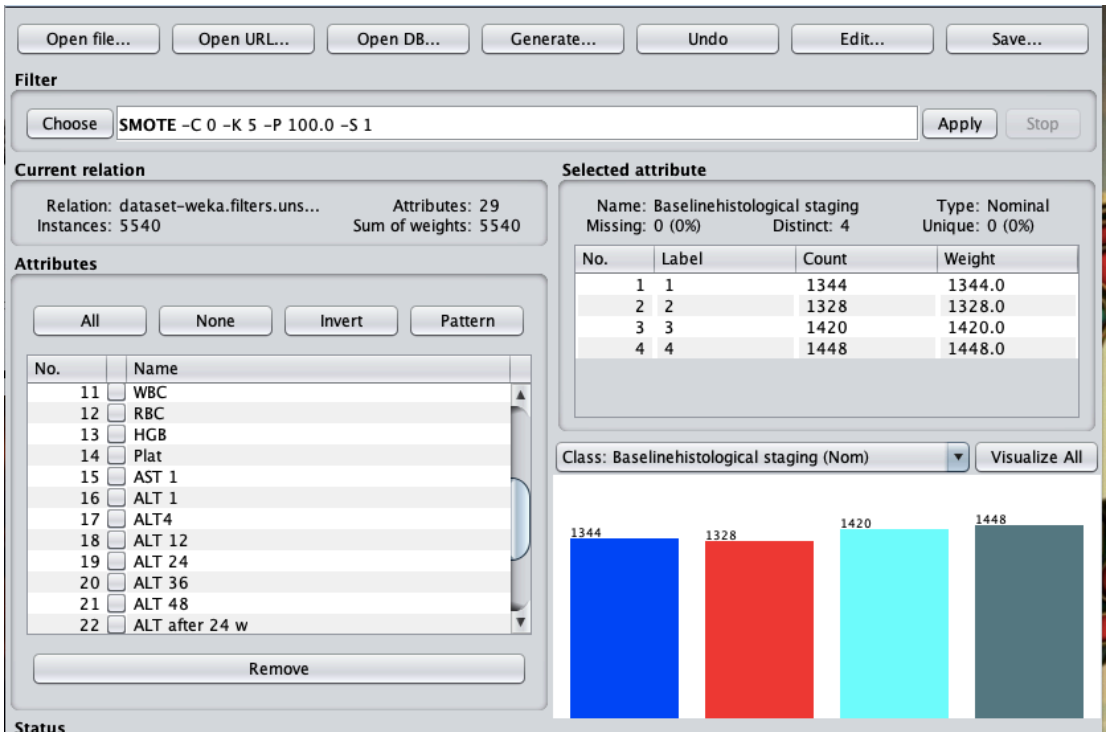

Fig-4.7.1 Applying SMOTE

After applying the SMOTE, the number of instances have increased to 5540 in which 1344, 1328, 1420, 1448 are the number of records in 1st,2nd,3rd,4th stages respectively.

Now the next step is to save this newly acquired dataset and try to build our model.

4.8 BUILDING THE MODEL AFTER SMOTE

We perform all the preprocessing steps that we followed before.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=18)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4432, 28)
(1108, 28)
(4432,)
(1108,)
```

Fig-4.8.1 Splitting new dataset

We again used the train_test_split function to split the dataset into training and testing. This time we got 4432 records in train and 1108 in test.

Now using KNN-classifier:

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(scaled_X_train, y_train)
training_score = knn.score(scaled_X_train, y_train)
print("training accuracy:",training_score)
test_score = knn.score(scaled_X_test, y_test)
print("test accuracy:",test_score)
```

```
training accuracy: 0.9842057761732852
test accuracy: 0.9395306859205776
```

Fig-4.8.2 KNN on new dataset

From the above figure we can see that the we chose the k value that is the number of nearest neighbours as 3 and we got an accuracy of 98% on train and 93.9% on test data.

To further improve the accuracy, we used hyper parameter tuning. Hyper parameter tuning is choosing of the optimal parameters that improve the performance of the model.

```
from sklearn.model_selection import GridSearchCV
leaf_size = list(range(1,50))
p=[1,2]
hyperparameters = dict(leaf_size=leaf_size, p=p)
clf = GridSearchCV(knn, hyperparameters, cv=10)
best_model = clf.fit(scaled_X_train,y_train)
print('Best leaf_size:', best_model.best_estimator_.get_params()['leaf_size'])
print('Best p:', best_model.best_estimator_.get_params()['p'])
#Predict testing set
y_pred = best_model.predict(scaled_X_test)
#Check performance using accuracy
print(accuracy_score(y_test, y_pred))
```

```
Best leaf_size: 1
Best p: 1
0.9693140794223827
```

Fig-4.8.3 Hyper parameter tuning

GridSearchCV is used to find those optimal parameter. So, we took leaf_size and p as parameters where leaf_size affects the speed of computation and p is the distance metric. Cv refers to cross validation and we have given it 10 fold. We obtained the best leaf_size value as 1 and also distance metric as 1. The accuracy of the model has also improved to 96.9%.

```
# Classification report
from sklearn.metrics import classification_report
y_pred_test = best_model.predict(scaled_X_test)
print(classification_report(y_test,y_pred_test))
```

	precision	recall	f1-score	support
1	0.95	0.98	0.97	260
2	0.97	0.96	0.97	265
3	0.99	0.96	0.97	286
4	0.97	0.98	0.97	297
accuracy			0.97	1108
macro avg	0.97	0.97	0.97	1108
weighted avg	0.97	0.97	0.97	1108

Fig-4.8.4 Classification report of KNN test

From figure 4.8.4, we can see the classification report for the knn test data. We can see that we got 97% accuracy and good recall and precision values for every class.
The recall tells us how well our model correctly identifies true positives.
Precision is the ratio of true positives and all positives. On the whole, the higher the precision and recall values, the better.

```
#confusion matrix for train data
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred_test)
```

```
array([[253,   2,   3,   2],
       [  7, 249,   5,   4],
       [  8,   6, 261,  11],
       [  8,   6,   5, 278]])
```

Fig-4.8.5 Confusion matrix

From the figure 4.8.5, we can see that the confusion matrix for the model. The diagonal elements are the correctly classified ones where as the other elements are misclassified ones in each class.

Next step is trying the Random forest classifier. The n_estimators parameter as used in figure 4.8.6 is the number of decision trees the model will build and then merge together to average out the results and give us the best performance.

After building the model on train and predicting on the training data, we can see from the classification report from the figure 4.8.6 that we got an accuracy of 100% along with precision and recall also accounting for 100%.

```
# importing the model class
from sklearn.ensemble import RandomForestClassifier
# creating a object for that model
rfc = RandomForestClassifier(n_estimators = 100)
# fitting the input and output of training data to the object
rfc.fit(scaled_X_train,y_train)
```

```
RandomForestClassifier()
```

```
from sklearn.metrics import classification_report
y_pred_train = rfc.predict(scaled_X_train)
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00      1084
           2       1.00      1.00      1.00      1063
           3       1.00      1.00      1.00      1134
           4       1.00      1.00      1.00      1151

    accuracy                           1.00      4432
   macro avg       1.00      1.00      1.00      4432
weighted avg       1.00      1.00      1.00      4432
```

Fig-4.8.6 Random Forest on train data after smote

```
y_pred_test = rfc.predict(scaled_X_test)
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           1       0.88      0.93      0.90       260
           2       0.92      0.87      0.90       265
           3       0.90      0.90      0.90       286
           4       0.91      0.90      0.90       297

    accuracy                           0.90      1108
   macro avg       0.90      0.90      0.90      1108
weighted avg       0.90      0.90      0.90      1108
```

Fig-4.8.7 Random forest on test after smote

```
#confusion matrix for train data
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred_test)
```

```
array([[242,   4,   7,   7],
       [ 10, 231,  10,  14],
       [ 15,   8, 256,   7],
       [  8,   8,  13, 268]])
```

Fig-4.8.8 Random Forest Confusion matrix

From the classification report we can see that there are 997 classes that are correctly classified and 111 misclassified classes.

Now moving on to logistic regression,

```
# importing the model class
from sklearn.linear_model import LogisticRegression
# creating a object for that model
lm = LogisticRegression()
# fitting the input and output of training data to the object and building the model.
lm.fit(scaled_X_train,y_train)
```

```
LogisticRegression()
```

```
# predicting the output of the training data
y_train_pred = lm.predict(scaled_X_train)
y_train_pred
```

```
array([4, 3, 1, ..., 2, 4, 1])
```

Figure- 4.8.9 Logistic regression after smote

```
from sklearn.metrics import accuracy_score
print("Training data accuracy:",accuracy_score(y_train,y_train_pred))
log_train = accuracy_score(y_train,y_train_pred)
```

```
Training data accuracy: 0.3418321299638989
```

```
y_test_pred = lm.predict(scaled_X_test)
print("Testing data accuracy:",accuracy_score(y_test,y_test_pred))
log_test = accuracy_score(y_test,y_test_pred)
```

```
Testing data accuracy: 0.33393501805054154
```

Fig-4.8.10 logistic regression accuracy

The logistic regression algorithm is applied on the newly acquired dataset after applying smote. After building a model that is based on the train data, we got an accuracy of about 34% for the train and 33% for test data.

5. RESULTS REPORT

All the three classifiers - knn classifier, random forest classifier and logistic regression have been successfully implemented. Now, let's compare the three classification algorithms and select the model that performed the best.

```
plt.figure(figsize=(12,8))
plt.xlabel("Classifier")
plt.ylabel("Accuracy")
plt.title("Accuracy Comparition")
plt.bar(list(x),list(y))
```

```
<BarContainer object of 6 artists>
```

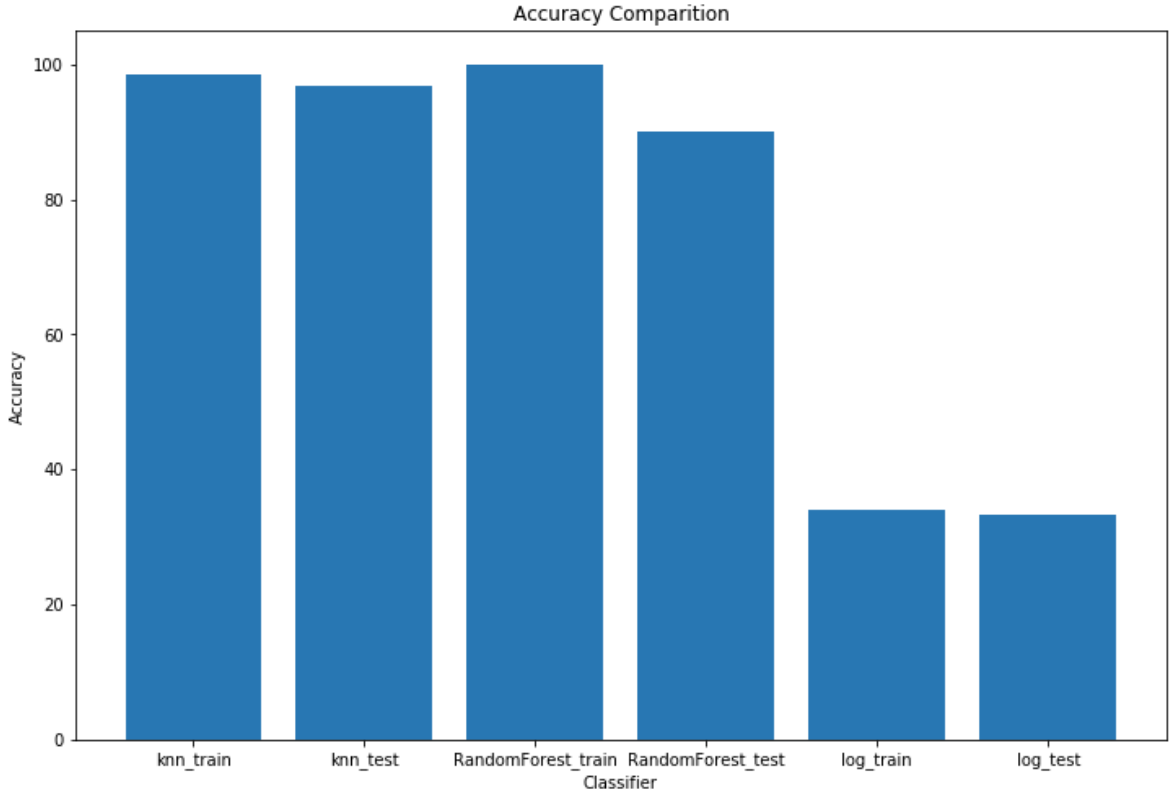

Fig-5.1 Comparing the models

From the above figure 5.1, it is clear that knn performed the best with accuracy of 97% on test data. Although Random forest has got 100% in train data but the unseen data or test data accuracy is preferred. Logistic regression is the weakest performer of all.

```
plt.figure(figsize=(18,8))
plt.xlabel("Classifier")
plt.ylabel("Accuracy")
plt.title("Accuracy Comparition")
plt.bar(list(x),list(y))
```

```
<BarContainer object of 8 artists>
```

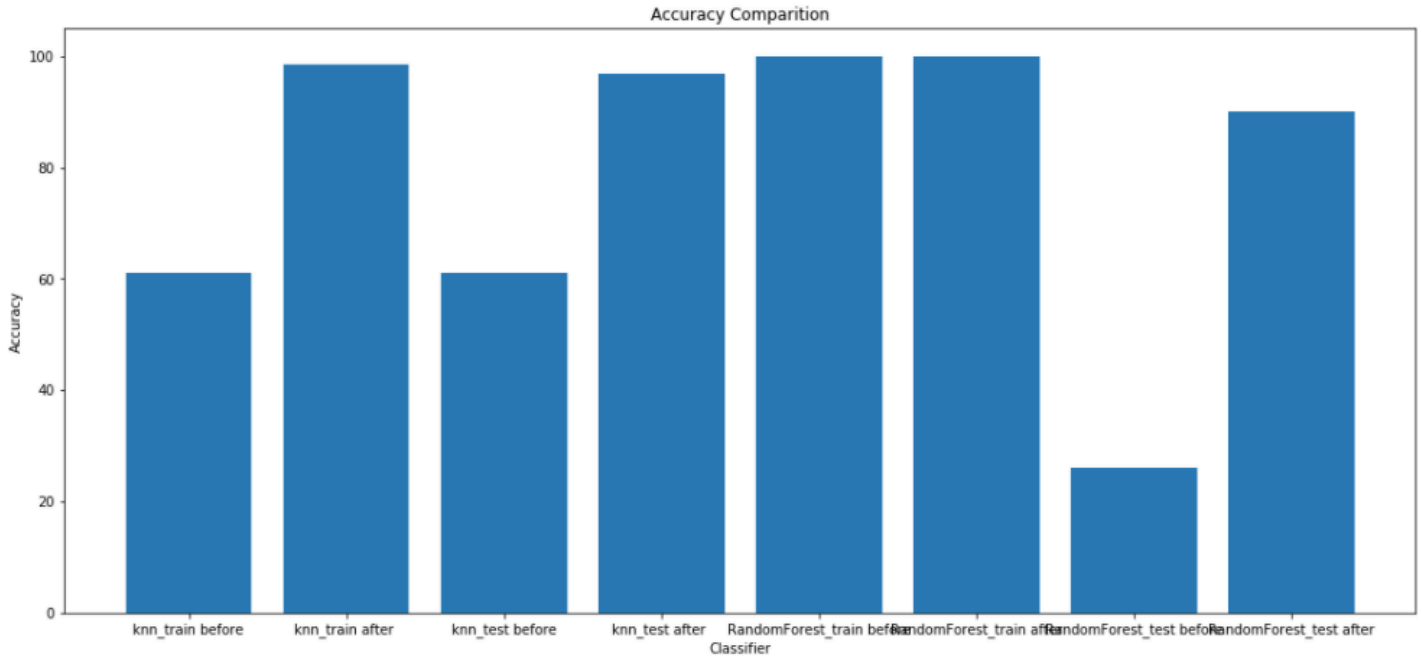

Fig-5.2 Results comparison before and after smote

From the above figure 5.2, we can see that the accuracy of every classification has improved with the application of smote. The figure gives us the comparison between before and after the smote of each algorithms train and test data.

6. CONCLUSION

In this project, we have successfully implemented a model that predicts the stages of HCV patients using various classification techniques. From the results we obtained in this work, we can say that knn showed the best results. This work of research will help medical field since this is a non-invasive procedure and it will save a lot of time. The only limitation that we found in this project is the lack of a huge dataset with more records. In the future, we can may be overcome this by collecting data from more patients.

7. REFERENCES

- [1] https://www.who.int/news-room/fact-sheets/detail/hepatitis-c
- [2] https://archive.ics.uci.edu/ml/datasets/Hepatitis+C+Virus+%28HCV%29+for+Egyptian+patients
- [3] N.Zayed, A. B.Awad, W.El-Akel, W.Doss, T.Awad, A.Radwan, and M.Mabrouk, "The assessment of data mining for the prediction of therapeutic outcome in 3719 egyptian patients with chronic hepatitis c," *Clinics and research in hepatology and gastroenterology*, vol. 37, no. 3, pp. 254–261, 2013
- [4] T. Orczyk and P. Porwik, "Liver fibrosis diagnosis support system using machine learning methods," in *Advanced Computing and Systems for Security*. Springer, 2016, pp. 111–121
- [5] S. Hashem, G. Esmat, W. Elakel, S. Habashy, S. Abdel Raouf, S. Dar- weesh, M. Soliman, M. Elhefnawi, M. El-Adawy, and M. ElHefnawi, "Accurate prediction of advanced liver fibrosis using the decision tree learning algorithm in chronic hepatitis c egyptian patients," *Gastroen- terology research and practice*, vol. 2016, 2016.
- https://builtin.com/data-science/random-forest-algorithm
- https://rikunert.com/SMOTE_explained