Sri Lanka Institute of Information Technology

Machine Learning

Semester 01 - Year 04 – 2020

Predicting of Liver Disease using Decision Tree Classifier Algorithm

Assignment 01

Submitted by:

P.M.C.P.Paththinisekara
IT17056212

April 18th 2020

Contents

1. Introduction

1) Problem Statement

The liver is a large organ that is on the right side of the body. And in fact, the liver is the largest organ inside the body. Furthermore, it is a vital organ that handles hundreds of operations in the body, which includes some metabolic functions, handling food digestion, storing energy of the body, removing poisons, etc. So, the liver disease is a dire situation. Even though liver disease is a non-transmutable disease, it can still be inherited through genetics. Many factors can cause liver failure, such as over usage of alcohol, obesity, or and diseases.

According to the latest analytics in 2019, Liver disease is responsible for 2 million deaths per year around the globe. Liver cancer has become the 16th of the cause of death globally. Liver transplants have become the second most frequent organ transplant due to those reasons. This is maybe due to the liver disease doesn’t always have noticeable common symptoms. Therefore, diseases have to be identified using the collection of lab tests.

The field of artificial intelligence and machine learning has become more and more advanced and widespread over the years to perform various tasks in various areas. The most popular method the machine learning is being used for is to make predictions on various events using past data. That can also be true for the healthcare fields. So, it’s possible to analyze the previous data on a selected disease and identify a disease in the present.

In this assignment, the Decision tree machine learning algorithms which is advanced classification algorithm has used for analyzing previous lab data of the liver disease and make accurate predictions. So, the machine learning model will be trained to get the lab data of a person and predict whether a specific person has liver disease or not.

2. Data Collection

2) Dataset

In order to create a machine learning model to predict likeliness to have liver disease for a person, there should be previous lab data that contains details of former patients. That dataset will be used to train the machine learning model. The dataset of the current assignment was found from www.kaggle.com. Kaggle.com is a web community in which companies and researchers publish datasets to the public to produce the best models for predicting and describing the data.

Table 2. 1 : Dataset

Dataset source URL	https://www.kaggle.com/andrewdemonbreun/liver-disease-lab-data
Dataset Name	liver-disease-lab-data
Dataset file type	CSV
Number of instances	483
Number of attributes	10
Classes	0 = liver disease negative 1 = liver disease positive

3) Description of dataset

This dataset has ten attributes that will be analyzed in the process of making the prediction by the decision tree machine learning model.

Table 2. 2 : Dataet description

Attribute	**Description**
Age	Age of the patient. (10 - 100)
Gender	Gender of the patient (Male or Female).

	(This attribute will be converted in to Male = 1 or Female = 0 when preprocessing data)
Total_Bilirubin	Total Bilirubin is a combination of direct and indirect bilirubin (0.1 – 35)
Direct_Bilirubin	Bilirubin in the blood or urine. (0.1 – 20)
Alkaline_Phosphotase	Alkaline phosphatase (ALP) enzyme in a person's blood. (50 - 2000)
Alamine_Aminotransferase	Alanine Aminotransferase (ALT) test measured amount of enzyme in the blood. (0 - 2000)
Aspartate_Aminotransferase	Aspartate Aminotransferase (AST) test measured amount of enzyme in the blood. (0 – 5000)
Total_Protiens	The total protein is the total amount of protein in the blood. (3 – 10)
Albumin	Albumin in the blood. (1 – 6)
Albumin_and_Globulin_Ratio	Albumin and Globulin Ratio is the ratio of albumin present in serum in relation to the amount of globulin. (0.1 – 2)
Liver disease? (1 = yes, 0 = no)	0 = liver disease negative 1 = liver disease positive

The “Liver disease? (1 = yes, 0 = no) “attribute represents the outcome of the classification model. Therefore, the classification result “0” represents liver disease negative person; in other words, a healthy person and “1” means a liver disease, positive person.

3. Methodology

1) Introduction to Decision tree algorithm

The decision tree algorithm is a popular supervised learning algorithm that is used to solve both regression and classification problems. Decision trees that are used to solve classification problems are called classification trees, and regression trees are used to solve regression problems where the outcome values are typically real numbers. A decision tree is basically a tree structure where nodes represent attributes and branches represent decision rules, and leaf nodes represent the possible outcomes of the tree, which are commonly considered as classes. So, the main features decision tree is,

1. Nodes: - Features or attributes of the tree.
2. Branch: - Decision rules which lead to the outcome or the next node.
3. Leaf nodes: - Final nodes that predict the result of the tree.

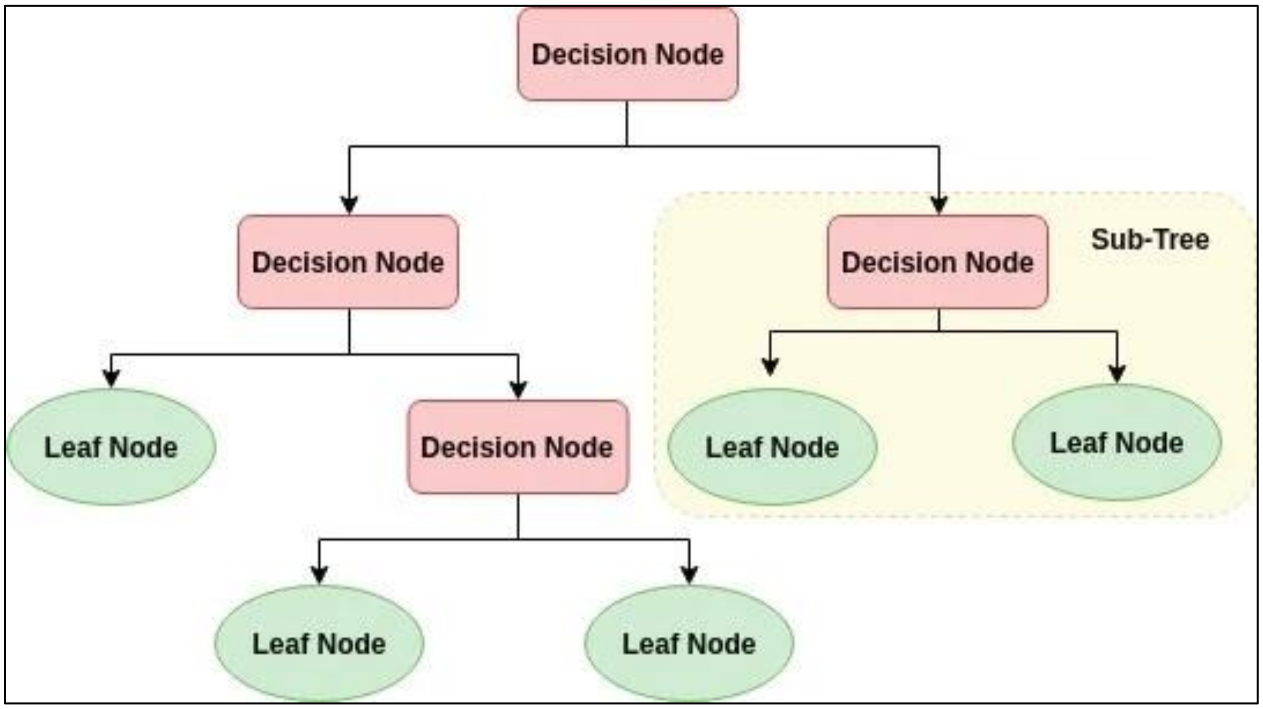


Figure 3. 1

The basic process behind the decision tree is,

1. First the algorithm selects the best attribute using information gain, Gini index and Gain Ratio, which are widely known as attribute selection measures.
2. Then the algorithm assigns the previously attribute chosen into a node and divides the data into small subsets.
3. Finally, the above process will be recursively run until the number of remaining attributes becomes 0, or no of remaining instances become 0.

2) Justification for the algorithm

In this assignment, a classification algorithm has to be used to predict liver disease likeliness of a person by analyzing given attributes. Therefore, the decision tree algorithm was selected as the machine learning algorithm. There are many for choosing this algorithm for this assignment. Mainly for this assignment, the chosen dataset has ten main attributes to in order to predict a binary outcome, and the assignment doesn't expect the use of extra-large datasets which could result in overfitting. Therefore, the selected dataset is complicated enough so that it can use the decision tree algorithm for predictions without a considerable downside.

The decision tree algorithm conducts a comprehensive analysis of each possible decision in the tree. Decision tree achieve this by performing the partition of data in a deeper level. So, each possible decision, conclusion or repetition much more comprehensive nature considering to other supervised learning classifiers such as logistic regression algorithm and support vector machines algorithm.

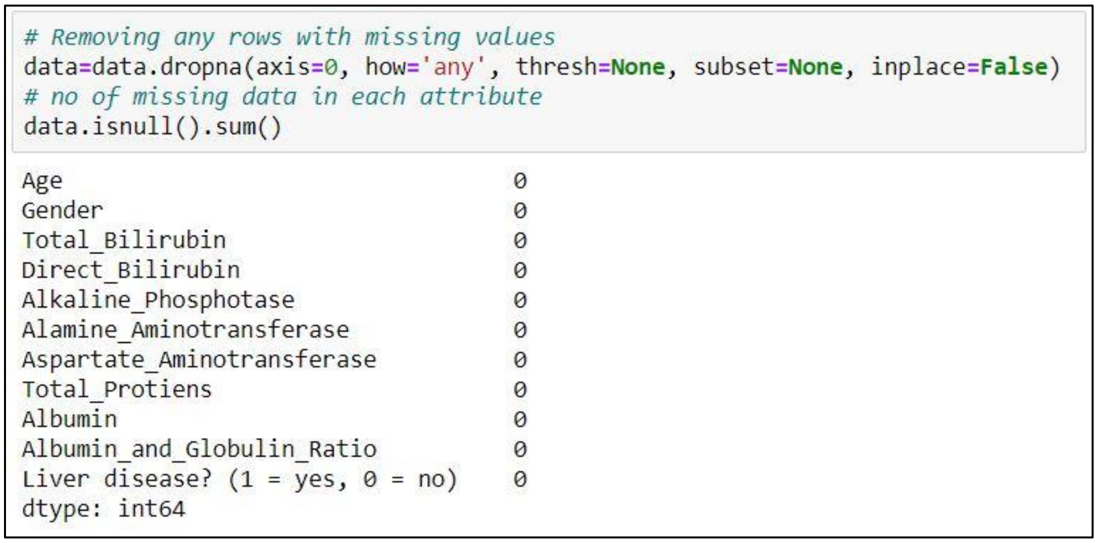
And another reason for the use of the decision tree algorithm in this assignment is a simple built of decision tree algorithm. Such as using decision tree the classification can be achieved without computing of the data; this also features that lacks in other machine learning models such as logistic regression algorithm and support vector machines algorithm.

Most importantly, when making predictions using a decision tree algorithm, it identifies the most relevant paths to makes a decision and provide a clear indication of the most important fields for prediction. Moreover, the decision tree algorithm reduces the uncertainty of the prediction or classification. So, this assignment is carried out using a sophisticated decision tree model that will consider the given attributes, which are lab test data of liver disease patients and classify them into two outcome classes as 0 or 1.

4. Implementation

1) Cleansing of the data

Data cleansing is the process of removing or correcting inaccurate records from the dataset. Simply it's the cleaning process of remodeling the dataset by identifying inaccurate or unfinished data. In this assignment there were some attributes which had missing data on few instances so they had to be removed from dataset through the data cleansing process.



```
# Removing any rows with missing values
data=data.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
# no of missing data in each attribute
data.isnull().sum()
```

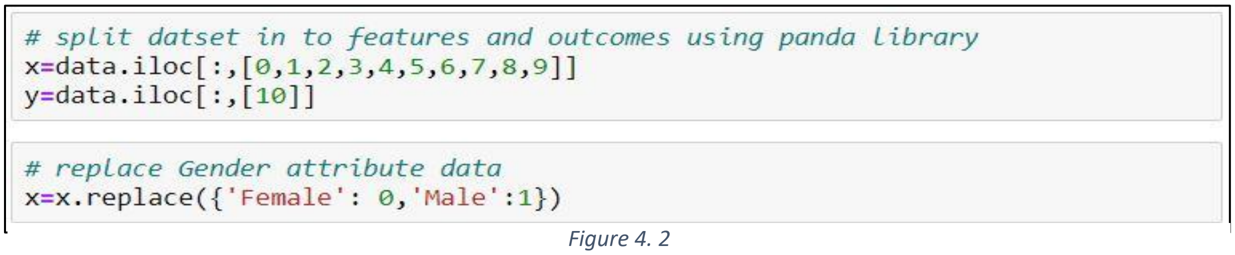
```
Age                               0
Gender                            0
Total_Bilirubin                   0
Direct_Bilirubin                  0
Alkaline_Phosphotase              0
Alamine_Aminotransferase          0
Aspartate_Aminotransferase        0
Total_Protiens                    0
Albumin                           0
Albumin_and_Globulin_Ratio        0
Liver disease? (1 = yes, 0 = no)  0
dtype: int64
```

Figure 4. 1

2) Data Processing

In order to apply the dataset in to the decision tree algorithm, data has to processed for the format of decision tree analyzing process to make the correct predictions

- First dataset should be separated in to attributes and outcomes.
- Then gender attribute data has to be reset as "0" or "1" instead of Male, Female.



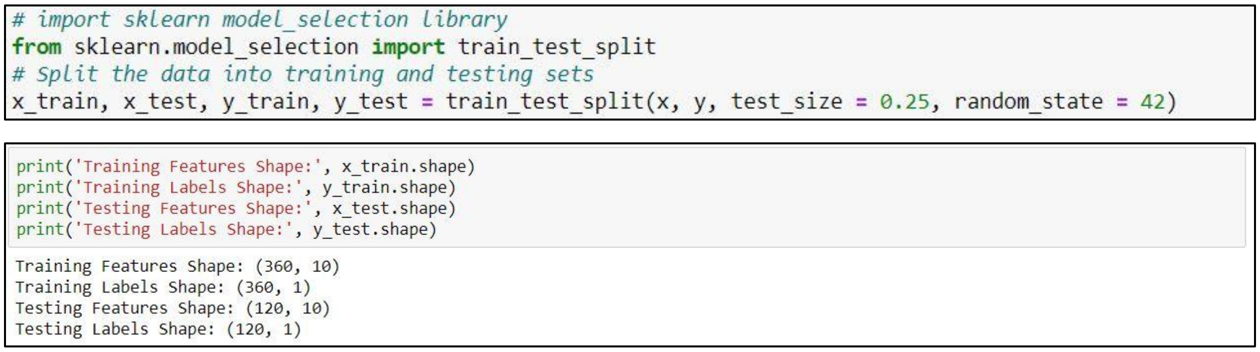
```
# split datset in to features and outcomes using panda library
x=data.iloc[:,[0,1,2,3,4,5,6,7,8,9]]
y=data.iloc[:,[10]]
```

```
# replace Gender attribute data
x=x.replace({'Female': 0,'Male':1})
```

Figure 4. 2

3) Split Dataset for training

In order for training the decision tree model the dataset has to be partition as train data which will be used to train the model and the test data which will be used to test the model. Here the dataset has been partitioned 1:3 ratio.



```
# import sklearn model_selection library
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 42)
```

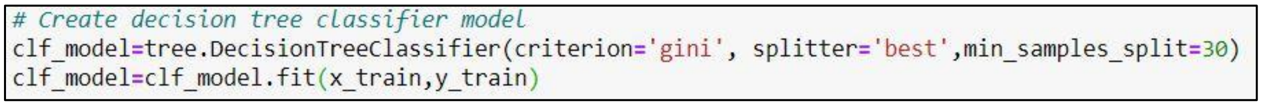
```
print('Training Features Shape:', x_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', x_test.shape)
print('Testing Labels Shape:', y_test.shape)
```

```
Training Features Shape: (360, 10)
Training Labels Shape: (360, 1)
Testing Features Shape: (120, 10)
Testing Labels Shape: (120, 1)
```

Figure 4. 3

4) Build decision tree classifier

Model will be created using decision tree classifier the criterion is assigned as "gini", that is the Gini impurity. And strategy is the method used to choose the split at each node. Strategy is changed as "best" to choose the best split. The minimum number of samples required to split an internal node is called min_samples_split it' has been set to 30.

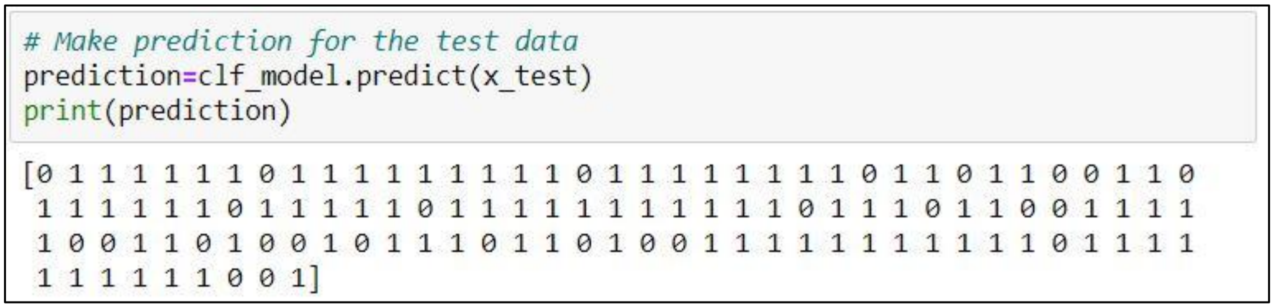


```
# Create decision tree classifier model
clf_model=tree.DecisionTreeClassifier(criterion='gini', splitter='best',min_samples_split=30)
clf_model=clf_model.fit(x_train,y_train)
```

Figure 4. 4

5) Make prediction on test set

After creating the decision tree model using the training data, the test data is applied to the model in order to make the prediction.



```
# Make prediction for the test data
prediction=clf_model.predict(x_test)
print(prediction)
```

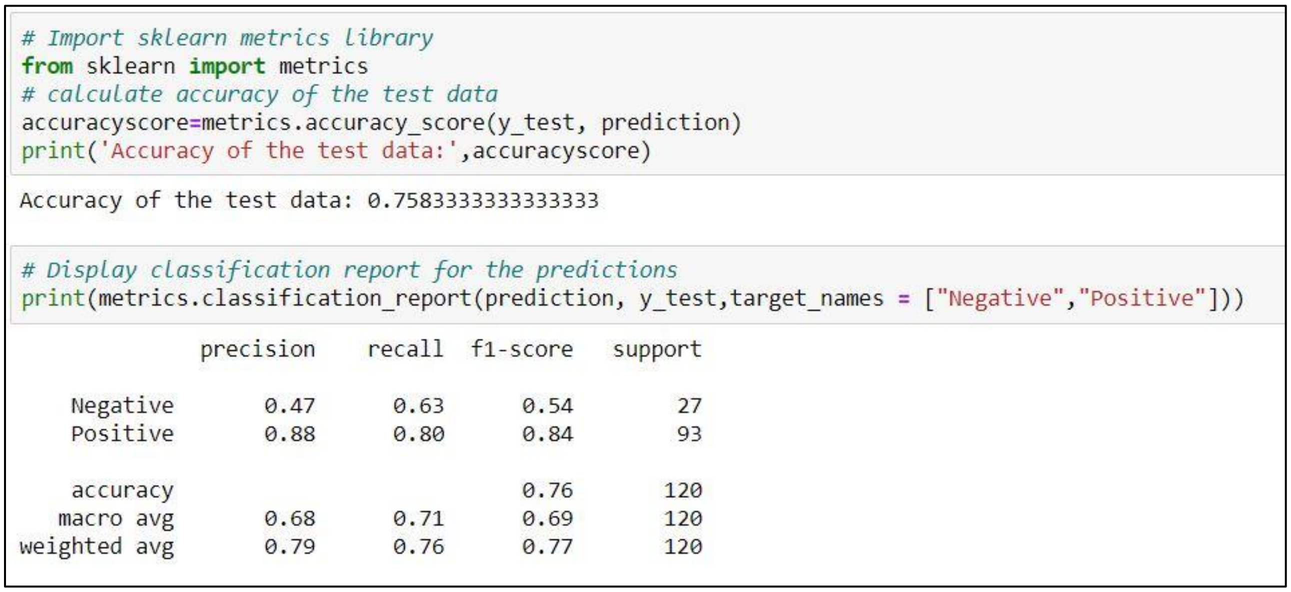
```
[0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 0
 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1
 1 0 0 1 1 0 1 0 0 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 0 0 1]
```

Figure 4. 5

5. Testing and Results

1) Accuracy of the predicted data

After running test data through the trained decision tree classification model, claasifier make predictions on the test data which was partitioned earlier for testing at the split data stage. The using the sklearn metrics function it's possible to get the accuracy of the trained machine learning model. Here the accuracy was 75% for the test data.



```
# Import sklearn metrics library
from sklearn import metrics
# calculate accuracy of the test data
accuracyscore=metrics.accuracy_score(y_test, prediction)
print('Accuracy of the test data:',accuracyscore)
```

```
Accuracy of the test data: 0.7583333333333333
```

```
# Display classification report for the predictions
print(metrics.classification_report(prediction, y_test,target_names = ["Negative","Positive"]))
```

```
              precision    recall  f1-score   support

    Negative       0.47      0.63      0.54        27
    Positive       0.88      0.80      0.84        93

    accuracy                           0.76       120
   macro avg       0.68      0.71      0.69       120
weighted avg       0.79      0.76      0.77       120
```

Figure 5. 1

2) Identify the types of errors made by the classifier

Confusion matrix is a table like measurement model which is also known as an error matrix that is often used to describe the performance of a classification model where output can be two or more classes. So in this case the confution metrics is shown as a heat map which shows the actual result instances count against predicted result instances count on each class.

```python
# Import seaborn library
import seaborn as sb
# Identify the types of errors made by the classifier using confusion matrix
mat_data = metrics.confusion_matrix(y_test, prediction)
# Create a heatmap for the confusion matrix
heat_map = sb.heatmap(mat_data, center=0, annot=True)
pl.xlabel('Actual Class')
pl.ylabel('Pedicted Class')
pl.show()
```

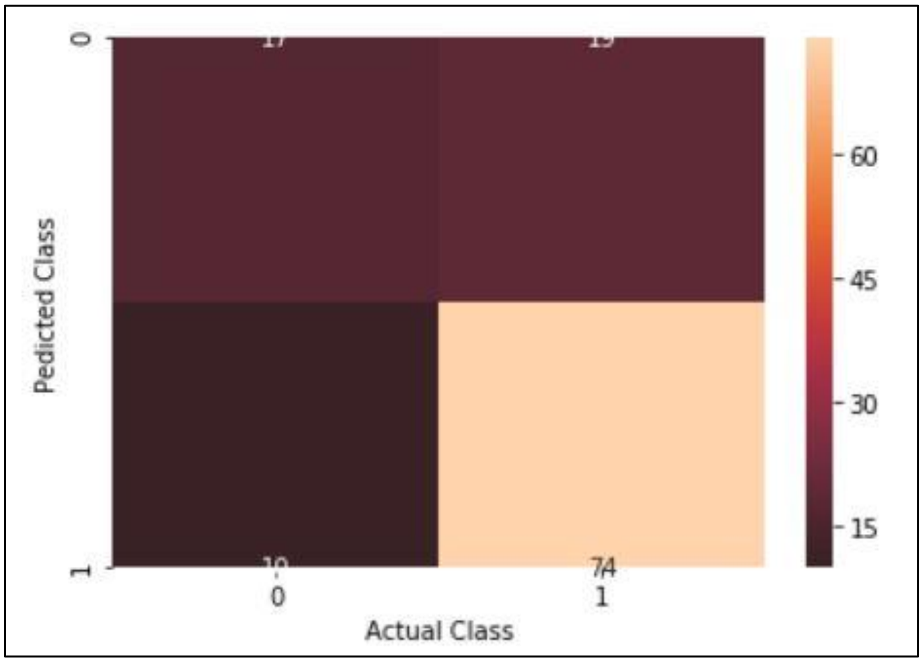


Figure 5. 2

6. Evaluation

1) Results and discussion

According to the results received after testing the model it seems like decision tree is a good classification method for classififying Liver disease lab datset because the prediction accuracy was 75%. Moreover when further anying the model it seems like adding more training data to the model could improve accuracy of the model explanentialy. Moreover fine tunning the minimum number of samples required to split an internal node also seems to increase the accuracy of the model.

2) Future work

In order to get more accurate and complex result in the future for liver disease indentifying, suggetions below will be used.

- Using multiple algorithms to check the performace with the dataset.
- Parameter tuning will be done to find the optimum value for each parameter to improve the accuracy of the model.
- Using a much more complex machine learning process like using a Neural Network to predict the result of the this datset.
- Using ensemble methods like Bagging (Bootstrap Aggregating) and Boosting.

7. Appendix

```
# Importing all necessary libraries
# Using numpy library for linear algebra
import numpy as np
# Using Panda library data manipulation of the dataset
import pandas as pd
# Using pylab library to plot 2D graphs
import pylab as pl
# import sklearn decision tree library
from sklearn import tree
# import sklearn preprocessing library
from sklearn import preprocessing

# import the dataset in the liver-disease-lab-data.csv file
data = pd.read_csv('/Users/Chamika Prabath/Desktop/ML 
assesment/IT17056212/dataset/liver-disease-lab-data.csv')

# count no of data in each attribute
data.count()
# Removing any rows with missing values
data=data.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
# no of missing data in each attribute
data.isnull().sum()
data.count()
# Display all the data
data.head()
# Summerize and show statistics of the dataset
data.describe()

# split datset in to features and outcomes using panda library
x=data.iloc[:,[0,1,2,3,4,5,6,7,8,9]]
y=data.iloc[:,[10]]
# replace Gender attribute data
x=x.replace({'Female': 0,'Male':1})
# Visualize histogram of liver disease

y.hist()
pl.suptitle("Histogram of Liver Disease")
pl.savefig('histogram_of_liver_disease_history')
pl.show()

# Visualize histogram for each numeric input variable
x.hist(bins=30, figsize=(12,12))
pl.suptitle("Histogram for each numeric input variable")
pl.savefig('liver_disease_each_numeric_input_variable')
pl.show()

# import sklearn model_selection library
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, 
random_state = 42)
```

```python
# Create decision tree classifier model
clf_model=tree.DecisionTreeClassifier(criterion='gini',
splitter='best',min_samples_split=30)
clf_model=clf_model.fit(x_train,y_train)
print('Training Features Shape:', x_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', x_test.shape)
print('Testing Labels Shape:', y_test.shape)

# Make prediction for the test data
prediction=clf_model.predict(x_test)
print(prediction)

# Import sklearn metrics library
from sklearn import metrics
# calculate accuracy of the test data
accuracyscore=metrics.accuracy_score(y_test, prediction)
print('Accuracy of the test data:',accuracyscore)

# Display classification report for the predictions
print(metrics.classification_report(prediction, y_test,target_names =
["Negative","Positive"]))

# Display decision tree
tree.plot_tree(clf_model)
# Import seaborn library

import seaborn as sb
# Identify the types of errors made by the classifier using confusion matrix
mat_data = metrics.confusion_matrix(y_test, prediction)
# Create a heatmap for the confusion matrix
heat_map = sb.heatmap(mat_data, center=0, annot=True)
pl.xlabel('Actual Class')
pl.ylabel('Pedicted Class')
pl.show()

#Calculate Classification Error
misclassification_Rate=1 - metrics.accuracy_score(y_test, prediction)

#Import ExtraTreesClassifier to boost accurcy
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier(n_estimators=100, random_state=0)
clf.fit(x_train, y_train)
ExtraTreesClassifier(random_state=0)
y_pred = clf.predict_proba(x_test)
print(y_pred)
```