REALIZING AN EFFICIENT DEEP LEARNING MODEL FOR PATIENT DIET RECOMMENDATION SYSTEM

Report submitted to the SASTRA Deemed to be University as the requirement for the course

BICCIC707: MINI PROJECT

Submitted by

DAVULURI SRI HANEESHA
(Reg. No.:121014014, B.Tech ICT)
MOVVA SRI SAI HARAVALI
(Reg. No.:121014062, B.Tech ICT)
RAMARAO SNEHA SINDHU
(Reg. No.:121014063, B.Tech ICT)

DECEMBER 2020

SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**Realising an efficient Deep Learning model for Patient Diet Recommendation System**" submitted as a requirement for the course, BICCIC707: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Ms. Davuluri Sri Haneesha (Reg. No. 121014014, B.Tech ICT)**, **Ms. Movva Sri Sai Haravali (Reg. No. 121014062, B.Tech ICT)** and **Ms. Ramarao Sneha Sindhu (Reg. No. 121014063, B.Tech ICT)** during the academic year 2020-21, in the School of Computing, under my supervision.

Signature of Project Supervisor :

Name with Affiliation :

Date :

Mini Project *Viva voce* held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS

It is a great pleasure for us to present this project to all of you. We would like to acknowledge each and every one who had a role to play in making our humble efforts an out-to-out success.

First of all, we express our gratitude to **Prof. Dr. S Vaidhyasubramaniam**, Vice-Chancellor, SASTRA Deemed University who provided all facilities and necessary encouragement during the course of our study. We extend our sincere thanks to **Prof. Dr. R. Chandramouli**, Registrar, SASTRA Deemed University for Providing the opportunity to pursue this mini project.

It is our privilege to express our sincerest regards to our **Dr.A.Uma Maheswari**, Dean(SOC) , Dr.Shankar Sriram VS, Associate Dean (CSE) and **Dr.B.Shanthi**, Associate Dean (Sponsored Research) who motivated us during the mini project work.We would also like to thank **Dr. Muthaiah R,** Associate Dean, SASTRA Deemed to be University, for providing the opportunity to pursue this mini project.

We are highly indebted to our guide **Dr. Subramaniya Swamy V,** Associate Professor(CSE), for his valuable inputs, guidance, encouragement, wholehearted cooperation and constructive criticism throughout the duration of our mini project. We would take this opportunity to thank our guide and all our professors who have directly or indirectly helped our project.

Lastly, we thank all the technical and non-technical staff of the School of Computing. We would like to express our gratitude to our parents and team members for their kind cooperation and motivation that enabled us to complete this project.

List Of Figures

List Of Tables

ABBREVIATIONS

The Abbreviations used in this report are listed below:

AI	Artificial Intelligence
ANN	Artificial Neural Networks
BMI	Body Mass Index
GRU	Gated Recurrent Unit
IoMT	Internet of Medical Things
API	Application Programming Interface
LSTM	Long Short Term Memory
MLP	MultiLayer Perceptron
RNN	Recurrent Neural Network

ABSTRACT

Health is of prime importance for any individual. In recent times, patients are exclusively relying on medication for improving health without including a healthy diet. Further research has shown that diets advised by Nutritionist or an automated AI cloud system based on medical diet can boost up immunity, vitality and enhance the overall health status for patients. This paper proposes a deep learning solution that detects which food should be given to a patient This paper presents a deep learning approach that detects which food a patient should consume based on their medical records, and other physical features like weight, age, gender, and nutrient values like calories, fiber, cholesterol, protein, fat, sodium. The medical dataset contains 30 records of patients consisting of 13 attributes and the product dataset consists of 1000 food products with 8 features. The dataset is preprocessed and optimal features are identified by Random Forest feature selection method and most important features are selected. Various machine learning and deep learning algorithms like Logistic Regression, Naive Bayes, Recurrent Neural Network(RNN), Multilayer Perceptron(MLP), Gated Recurrent Units(GRU), and Long Short-Term Memory(LSTM) are implemented. Cross validation method is implemented to estimate the test error of every model. To assess the performance of the models, evaluation metrics like precision, accuracy, recall and f1-measure are used. LSTM is found to outclass all other algorithms in terms of the evaluation metrics and produce good results.

Keywords: IoMT,LSTM,RNN,GRU,MLP,Naive Bayes,Logistic Regression

TABLE OF CONTENTS

CHAPTER 1

SUMMARY OF BASE PAPER

Base Paper Details:

Title: Realizing an Efficient IoMT-Assisted Patient Diet Recommendation System Through Machine Learning Model

Authors: Celestine Iwendi, Suleman Khan, Joseph Henry Anajemba, Ali Kashif Bashir, and Fazal Noor.

Published in: IEEE Access (Volume 8)

Date of Publication: 21 January, 2020

Publisher: IEEE

Indexed in: Scopus

As health became a major concern in this pandemic, the urge to improve one's immunity and have a good healthy body has bagged great concern. Many researchers have concluded that a proper diet can improve the quality of our health thereby increasing longevity and the immune system of our body. It also helps to fight the existing and future diseases that are to be fought by the body. The goal of this paper is to suggest a deep learning technique based on patient and product dataset which can detect the food that can be consumed by the patient considering the nutritional levels required by their body. The patients data is collected through hospitals which includes 30 records having the details of patients such as their age, gender, weight, disease, calories, protein, fat, sodium, carbohydrates, fiber and cholesterol with 1000 records. The other dataset contains product details such as calories, proteins, fat, sodium, carbohydrates, fiber and cholesterol with 1000 records in it collected through the internet. The two datasets are combined to form a dataset with 21 features in it. This data is preprocessed before inputting them to the models to get better results. The first preprocessing step is normalization. The dataset has outliers which decrease the model performance as they deviate from other observations. To avoid this, min-max normalization is performed on the dataset to scale the values in the range [0,1]. The categorical data is then encoded using one-hot encoding as categorical data cannot be sent directly to models for classification. After preprocessing the data, the data can be inputted to the model. However, to get better performance, feature selection algorithms are deployed to reduce the data dimensionality. Random Forest algorithm is used to remove irrelevant features leading to decrease the accuracy of the model. This algorithm gives the weight or the importance of each feature in classifying the data. Thus, the top important features are selected and sent to model for classification. The dataset is then divided into 70% for training purposes and 30% for testing purposes. K-Fold Cross validation is used for both. Both deep learning and machine learning models are used for classifying the data into allowed class and not allowed class. Deep

learning algorithms implemented are Recurrent Neural Network(RNN), Multilayer Perceptron(MLP), Gated Recurrent Units(GRU), and Long Short-Term Memory(LSTM) while the machine learning algorithms are Naive Bayes and also Logistic Regression. These models are compared using evaluation metrics such as accuracy, precision, recall, f1-measure.From the result analysis, it was concluded that LSTM has given significantly better results compared to other models and hence is proposed to be used for the recommendation system.

CHAPTER 2

MERITS AND DEMERITS

1. **S. Norouzi, A. K. Ghalibaf, and S. Sistani:**The key factor in this paper was the physical activity of patients/users that was included in the design and development of the application. On the basis of physical activity level, the authors estimated users' energy intake to identify the most suitable snack option for users based on their calorie needs by using the Harries Benedict equation, This approach was designed with a focus on the needs of patients and BMI rather than the conditions of the patients. It also included the limited sample size and the lack of key meals in the estimation process.

2. **M. Raut, K. Prabhu, R. Fatehpuria, S. Bangar, and S. Sahu:** This paper proposed a recommendation for a customised diet using the Ant bee Colony algorithm. Sadly, for the everyday activity and diet requirements of the user, the device relied on the Google Fit API and it also depended heavily on the past medical history of the user.

3. **R. Yera Toledo, A. A. Alzahrani, and L. Martinez:** This paper suggested using only physical user details to recommend the daily nutritional requirement. The study failed to include a strategy that simultaneously controlled the user's food and preference.

4. **V. Jaiswal:** The patient diet recommendation framework models the basic diet / nutritional needs of users based on individual eating habits and body statistics. While this research is useful for patients and nutritionists/doctors in predicting healthy diets, the drawback, however, is that it is void of a flexible model and achieves minimal solutions for patient needs design.

5. **N. Leipold, M. Lurz, and M. Bohm:** A nutrition assistance framework was developed in this paper, the system provides input on the dietary actions of a patient and accommodates behaviour improvement through various persuasive components such as self-monitoring, personalization, reflection, recommendations or tracking implementation. In order to achieve the desired effect in the long term as a mobile platform application for everyday use, the system built had to be improved according to the feedback provided.

6. **G. Agapito, B. Calabrese, P. H. Guzzi, M. Cannataro, M. Simeoni, I. Care, T. Lamprinoudi, G. Fuiano, and A. Pujia:**The authors suggested a Diet Organizer

System. Using a complex real-time survey prepared by medical physicians and compiled by users, they created a profile.The device referred to here As DIETOS can Not only prescribe individual foods with comparable health scores in the same group, but it can also provide dietary guidelines for some forms of health issues.However they could not integrate more health conditions into their proposed model.

7. **R. Priyadarshini, R. Barik, and H. Dubey:** This paper proposed a deep neural network model known as DeepFogIt collected people's data and forecast their health rules, but failed to reshape their method for realistic use, implementation and potential deployment for patients with health issues in the diet/food advice system.

8. **H. Fidan, A. Teneva, S. Stankov, and E. Dimitrova:** For personalised recommendations based on geographical, cultural, socioeconomic and distinct status, the user-based ontological AI profile was incorporated with a broad-based scientific experimental diabetes. This technique failed, however, to be clear and traceable.

CHAPTER 3

INTRODUCTION

Health is of utmost importance to any individual, without a healthy body one cannot function properly. To have a healthy immune system, cognitive and neurological development for proper body growth and organ formation, adequate nutrition is vital.A proper nutritious diet can be considered an important factor to improve the overall well being of an individual. Patients diagnosed with ailments should take special care for their health as even the minute parts of their daily life have to be adjusted to help them in their recovery and improve their overall health conditions. In the case of patients, studies have shown that most patients neglect including a healthy and nutritious diet in their journey to recovery. It is recognised that insufficient and inadequate food consumption causes different health problems and diseases. The lack of concise healthy diet knowledge causes people to rely on medicines exclusively instead of adjusting their diet to include more nutrients they require. Nutritious food has a lot of dietary fibres, good fats, antioxidants, vitamins, minerals and proteins which are the key factors for good health but For patients whose bodies are not tolerant of this intake due to the diseases they experience, it may also cause counter-effects. In particular, selecting the right diet is important for patients suffering from various diseases.

With time, there is rapid advancement in technology. These technologies are being implemented across various industries and practices to improve and make human life more efficient.Machine learning and models of deep learning are used in almost all fields. Machine learning models are a part of artificially intelligent models which get an understanding of the data and predict outcomes without being explicitly programmed to do so. The machine learning and models of deep learning can be used to identify if a user/patient is allowed to consume a particular food product. After the classification, the classified data can be used as an input to a recommendation model. Recommendation systems are another technological advancements which are used for suggesting a product depending on the user’s input. In recent times, food recommender systems have been receiving growing attention everyday for their significant role in recommending an efficient and healthy diet. These systems take in the users' nutrition requirement, their preferences and suggest a healthy meal that can satisfy the user while also balancing their nutrition intake

There are six working phases in the project:

1. Data Collection
2. Data Preprocessing
3. Optimal Feature selection
4. Training
5. Testing
6. Evaluation

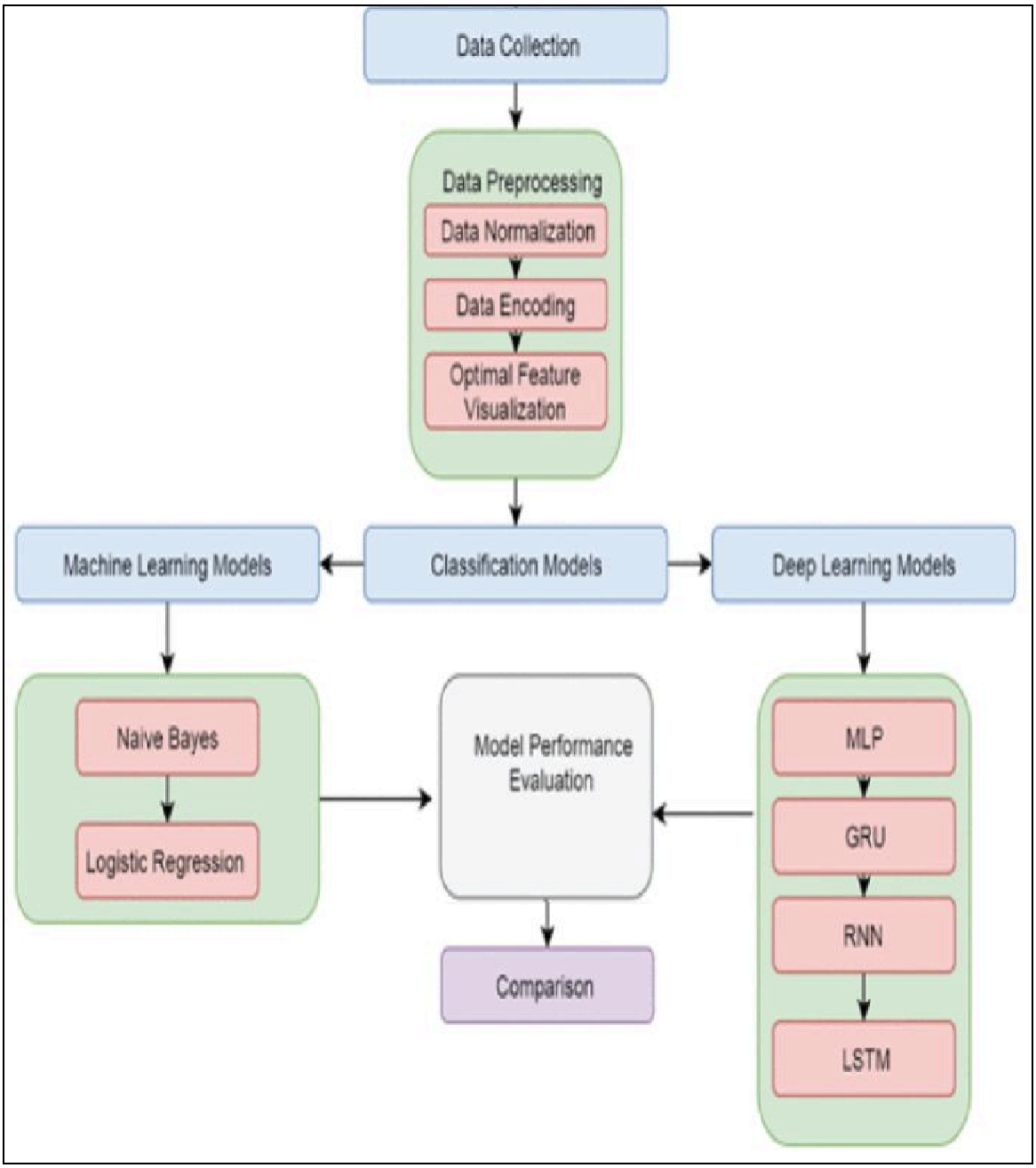

Figure 1: Workflow Chart

CHAPTER 4

METHODOLOG

Y

4.1 DATASETS

The datasets used in this project are a patient dataset, consisting of 30 patients' records with various features like age, gender, diagnosed disease and their nutritional information and a product dataset consisting of 1000 different food products with features like products' nutritional content and the barcode. These two datasets are combined such that each patient record is combined with each product record. This combined dataset has a total of 21 features and 30000 records.

No.	Feature	Feature Type
1	Product Barcode	Numeric
2	Product Calories	Numeric
3	Product Proteins	Numeric
4	Product Fat	Numeric
5	Product Sodium	Numeric
6	Product Carbohydrates	Numeric
7	Product Fiber	Numeric
8	Product Cholesterol	Numeric

Table 1: Product Features

No.	Feature	Feature Type
1	Number	Numeric
2	Age	Numeric
3	Gender	Categorical
4	Weight	Numeric
5	Disease	Categorical
6	Calories	Numeric
7	Protein	Numeric
8	Fat	Numeric
9	Sodium	Numeric
10	carbohydrate	Numeric
11	Fiber	Numeric
12	Cholesterol	Numeric
13	Target Class	Categorical

Table 2: Patient Features

4.2 PREPROCESSING

The dataset has to be preprocessed before it can be sent through a model. If the dataset is not preprocessed, the model will not fit correctly and there might be a drastic drop in accuracy. It has to be cleaned and checked for missing values, null values and noise. The preprocessing techniques applied on the dataset in this project are Normalisation, Encoding and Optimal feature visualization.

4.2.1 Data Normalization

Each feature in the dataset might have values in different ranges. The technique of converting all these values into one range for better model performance is called normalization.
The normalization technique used is the min-max normalization in which the ranges are scaled between [0,1]. The min-max normalization is given by

$$Z(i) = \frac{F(i) - min(F)}{max(F) - min(F)}$$

where,

F(i) - feature to be normalised
Z(i) - normalised feature
max(F) - greatest value of all features min(F) - lowest value of all features

4.2.2 Data Encoding

The conversion of the non-numerical attributes (categorical,nominal,etc..) into numerical attributes is referred to as Data Encoding.This is done because the computations inside a machine learning model is performed on numerical attributes.

4.2.3 Optimal Feature Visualization

In a dataset, all the features don't hold equal importance, some might be more important than the others. To understand and select the most important features, Random Forest feature importance is used as this facilitates the visualization of features giving us the percentage of cruciality of a particular feature in determining the result of a model.

4.3 ALGORITHMS

Two machine learning classifiers: Naive Bayes, Logistic Regression and four deep learning classifiers: MultiLayer Perceptron(MLP), Gated Recurrent Unit(GRU), Long Short Term Memory(LSTM) , Recurrent Neural Networks(RNN) are used for training and testing.

4.3.1 Naive Bayes :

Naive Bayes is a classification algorithm which works on probability. It's main features include the application of Bayes theorem and the assumption that each pair of features is conditionally independent to one another.
Bayes theorem is given as

$$P(A/B) = \frac{P(B/A).P(B)}{P(A)}$$

where,

P(A/B) is the probability of event A occurring given event B has already occurred P(B/A) is the probability of event B occurring given event A has already occurred P(A) is the probability of occurrence of event A
P(B) is the probability of occurrence of event B

4.3.2 Logistic Regression :

Logistic Regression is a Supervised Learning model which is used for predicting discrete-valued outputs. It uses a logistic function to model the probability of a particular event or class, happening or not happening.

4.3.3 MultiLayer Perceptron (MLP) :

The MultiLayer Perceptron is composed of multiple layers of perceptrons which include the input, output and hidden layers. It is a feedforward ANN(Artificial Neural Network) where each neuron computes an activation function.

4.3.4 Recurrent Neural Network (RNN) :

Recurrent Neural Networks are used to capture information from sequences or time-series data. It is the first neural network to remember the input given to it but couldn't retain long sequences which gave rise to long term dependencies and vanishing gradient problems which were rectified in the later modifications of RNN's like LSTM and GRU.

4.3.5 Long Short Term Memory (LSTM):

LSTM is an artificial neural network with not just feedforward but also includes the feedback connections in the field of Deep Learning. It has a gating mechanism to control the flow of information between cells in the neural network. It has 3 gates: the input gate, output gate and the forget gate. This facilitates to rectify the long-term dependencies and the vanishing gradient problems encountered in RNN.

4.3.6 Gated Recurrent Unit (GRU):

The GRU, with a gating mechanism to regulate the flow of information between the cells in the neural network, is similar to LSTM. It has two gates, the reset gate, which is used to decide how much information to forget from the past, and the update gate, which decides how much information can be passed on to the future from the past. The missing gradient issue observed in RNN can also be corrected by this algorithm

CHAPTER 5

SOURCE

CODE

```
#Datasets and preprocessing
cnt = 30
for i in range(30):
    cal = df1.loc[i,'CALORIES']
    chol = df1.loc[i,'CHOLESTROL']
    carbo = df1.loc[i,'CARBOHYDRATES']
    pro = df1.loc[i,'PROTEIN']
    fib = df1.loc[i,'FIBRE']
    ft = df1.loc[i,'FAT']
    sod = df1.loc[i,'SODIUM']
    for j in range(1000):
      pro_fat = df2.loc[j,'P_FAT']
      pro_chol = df2.loc[j,'P_CHOLESTROL']
      pro_carbo = df2.loc[j,'P_CARBOHYDRATES']
      pro_pro = df2.loc[j,'P_PROTEIN']
      pro_cal = df2.loc[j,'P_CALORIES']
      pro_fib = df2.loc[j,'P_FIBRE']
      pro_sod = df2.loc[j,'P_SODIUM']
      df1.loc[cnt] = df1.loc[i]
      df1.loc[cnt,'P_FAT'] = df2.loc[j,'P_FAT']
      df1.loc[cnt,'P_CHOLESTROL'] = df2.loc[j,'P_CHOLESTROL']
      df1.loc[cnt,'P_CARBOHYDRATES'] = df2.loc[j,'P_CARBOHYDRATES']
      df1.loc[cnt,'P_PROTEIN'] = df2.loc[j,'P_PROTEIN']
      df1.loc[cnt,'P_CALORIES'] = df2.loc[j,'P_CALORIES']
      df1.loc[cnt,'P_FIBRE'] = df2.loc[j,'P_FIBRE']
      df1.loc[cnt,'P_BARCODE'] = df2.loc[j,'P_BARCODE']
      df1.loc[cnt,'P_SODIUM'] = df2.loc[j,'P_SODIUM']
      if ft >= pro_fat and chol >= pro_chol and carbo >= pro_carbo and pro >= pro_pro and cal
>= pro_cal:
```

```
      df1.loc[cnt,'Class'] = "Yes"
    else:
      df1.loc[cnt,'Class'] = "No"
    cnt+=1

#Optimal Feature Visualisation
X = df1[df1.columns[:-1]]
y = df1.Class_Yes
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)
importance = model.feature_importances_
for i,v in enumerate(X_train.columns[0:19]):
print(i,v,importance[i]*100)
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()

#Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy:")
print(accuracy_score(y_test,y_pred)*100)
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
print("Recall:")
print(recall_score(y_test,y_pred)*100)
print("Precision:")
print(precision_score(y_test,y_pred)*100)
print("F1:")
print(f1_score(y_test,y_pred)*100)
```

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
y_pred = lr.fit(X_train, y_train).predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy:")
print(accuracy_score(y_test,y_pred)*100)
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
print("Recall:")
print(recall_score(y_test,y_pred)*100)
print("Precision:")
print(precision_score(y_test,y_pred)*100)
print("F1:")
print(f1_score(y_test,y_pred)*100)

#MultiLayer Perceptron
from tensorflow import
keras
from tensorflow.keras import layers

model = keras.Sequential()
model.add(layers.Dense(64,kernel_initializer='uniform',input_shape=(10,)))
model.add(layers.Activation('softmax'))

opt = keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer='adam')
from sklearn.neural_network import MLPClassifier
# creating an classifier from the model:
mlp = MLPClassifier(hidden_layer_sizes=(10, 10), max_iter=1000)

# fitting the training data to our model
mlp.fit(X_train, y_train)
```

```
y_pred = mlp.fit(X_train, y_train).predict(X_test)
from sklearn.metrics import accuracy_score
print("Accuracy:")
print(accuracy_score(y_test,y_pred)*100)
y_pred = mlp.fit(X_train, y_train).predict(X_test)
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
print("Recall:")
print(recall_score(y_test,y_pred)*100)
print("Precision:")
print(precision_score(y_test,y_pred)*100)
print("F1:")
print(f1_score(y_test,y_pred)*100)

#Recurrent Neural Network
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, BatchNormalization
from tensorflow.keras import layers
from keras import backend as K
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision
def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
from tensorflow.keras.utils import to_categorical
clinicalOutput = to_categorical(clinicalOutput)
clinicalInput = np.array(clinicalInput).reshape(4000,19,1)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Conv1D, Dropout, MaxPooling1D,
Flatten,SimpleRNN,BatchNormalization

def create_network():
    model = Sequential()
    model.add(SimpleRNN(512,input_shape=(nb_features,1)))
    model.add(Activation("relu"))
    model.add(BatchNormalization())
    model.add((Flatten()))
    model.add(Dropout(0.15))
    model.add(Dense(2048, activation="relu"))
    model.add(Dense(1024, activation="relu"))
    model.add(Dense(nb_classes, activation="softmax"))
    model.summary()
   model.compile(loss="categorical_crossentropy", optimizer = 'adam',
metrics=["accuracy",f1_m,precision_m,recall_m])
    return model
neural_network = KerasClassifier(build_fn=create_network,epochs=50)
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
clinicalOutput=np.array([number[0]fornumberinlb.fit_transform(clinicalOutput)])
results = cross_validate(neural_network, clinicalInput,
clinicalOutput,cv=10,scoring=("accuracy","f1","recall","precision"))
y_pred = cross_val_predict(neural_network, clinicalInput, clinicalOutput, cv=10)
#Provide AUC score
from sklearn.metrics import roc_auc_score
print("Accuracy result: ", np.mean(results["test_accuracy"]))
print("Recall result: ", np.mean(results["test_recall"]))
print("Precision result: ", np.mean(results["test_precision"]))
```

```
print("F1 result: ", np.mean(results["test_f1"]))
print("ROC: ", roc_auc_score(clinicalOutput, y_pred))
#Long Short Term Memory
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, BatchNormalization
from tensorflow.keras import layers
from keras import backend as K
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision
def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
from tensorflow.keras.utils import to_categorical
clinicalOutput = to_categorical(clinicalOutput)
clinicalInput = np.array(clinicalInput).reshape(4000,19,1)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Conv1D, Dropout, MaxPooling1D,
Flatten,LSTM,BatchNormalization
def create_network():
    model = Sequential()
    model.add(LSTM(512,input_shape=(nb_features,1)))
    model.add(Activation("relu"))
    model.add(BatchNormalization())
    model.add((Flatten()))
    model.add(Dropout(0.15))
```

```
    model.add(Dense(2048, activation="relu"))
    model.add(Dense(1024, activation="relu"))
    model.add(Dense(nb_classes, activation="softmax"))
    model.summary()
 model.compile(loss="categorical_crossentropy",optimizer='adam',
metrics=["accuracy",f1_m,precision_m,recall_m])
    return model
neural_network = KerasClassifier(build_fn=create_network,epochs=50)
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
clinicalOutput=np.array([number[0]fornumberinlb.fit_transform(clinicalOutput)])
results = cross_validate(neural_network, clinicalInput,
clinicalOutput,cv=10,scoring=("accuracy","f1","recall","precision"))
y_pred = cross_val_predict(neural_network, clinicalInput, clinicalOutput, cv=10)
#Provide AUC score
from sklearn.metrics import roc_auc_score
print("Accuracy result: ", np.mean(results["test_accuracy"]))
print("Recall result: ", np.mean(results["test_recall"]))
print("Precision result: ", np.mean(results["test_precision"]))
print("F1 result: ", np.mean(results["test_f1"]))
print("ROC: ", roc_auc_score(clinicalOutput, y_pred))

#Gated Recurrent Unit
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, BatchNormalization
from tensorflow.keras import layers
from keras import backend as K
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
```

```
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision
def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
from tensorflow.keras.utils import to_categorical
clinicalOutput = to_categorical(clinicalOutput)
clinicalInput = np.array(clinicalInput).reshape(4000,19,1)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Conv1D, Dropout, MaxPooling1D,
Flatten,GRU,BatchNormalization
def create_network():
    model = Sequential()
    model.add(GRU(512,input_shape=(nb_features,1)))
    model.add(Activation("relu"))
    model.add(BatchNormalization())
    model.add((Flatten()))
    model.add(Dropout(0.15))
    model.add(Dense(2048, activation="relu"))
    model.add(Dense(1024, activation="relu"))
    model.add(Dense(nb_classes, activation="softmax"))
    model.summary()
    model.compile(loss="categorical_crossentropy", optimizer = 'adam',
metrics=["accuracy",f1_m,precision_m,recall_m])
    return model
neural_network = KerasClassifier(build_fn=create_network,epochs=50)
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
clinicalOutput=np.array([number[0]fornumberinlb.fit_transform(clinicalOutput)])
results = cross_validate(neural_network, clinicalInput,
clinicalOutput,cv=10,scoring=("accuracy","f1","recall","precision"))
y_pred = cross_val_predict(neural_network, clinicalInput, clinicalOutput, cv=10)
```

```
#Provide AUC score
from sklearn.metrics import roc_auc_score
print("Accuracy result: ", np.mean(results["test_accuracy"]))
print("Recall result: ", np.mean(results["test_recall"]))
print("Precision result: ", np.mean(results["test_precision"]))
print("F1 result: ", np.mean(results["test_f1"]))
print("ROC: ", roc_auc_score(clinicalOutput, y_pred))
```

CHAPTER 6

RESULTS

Optimal Feature Visualisation:

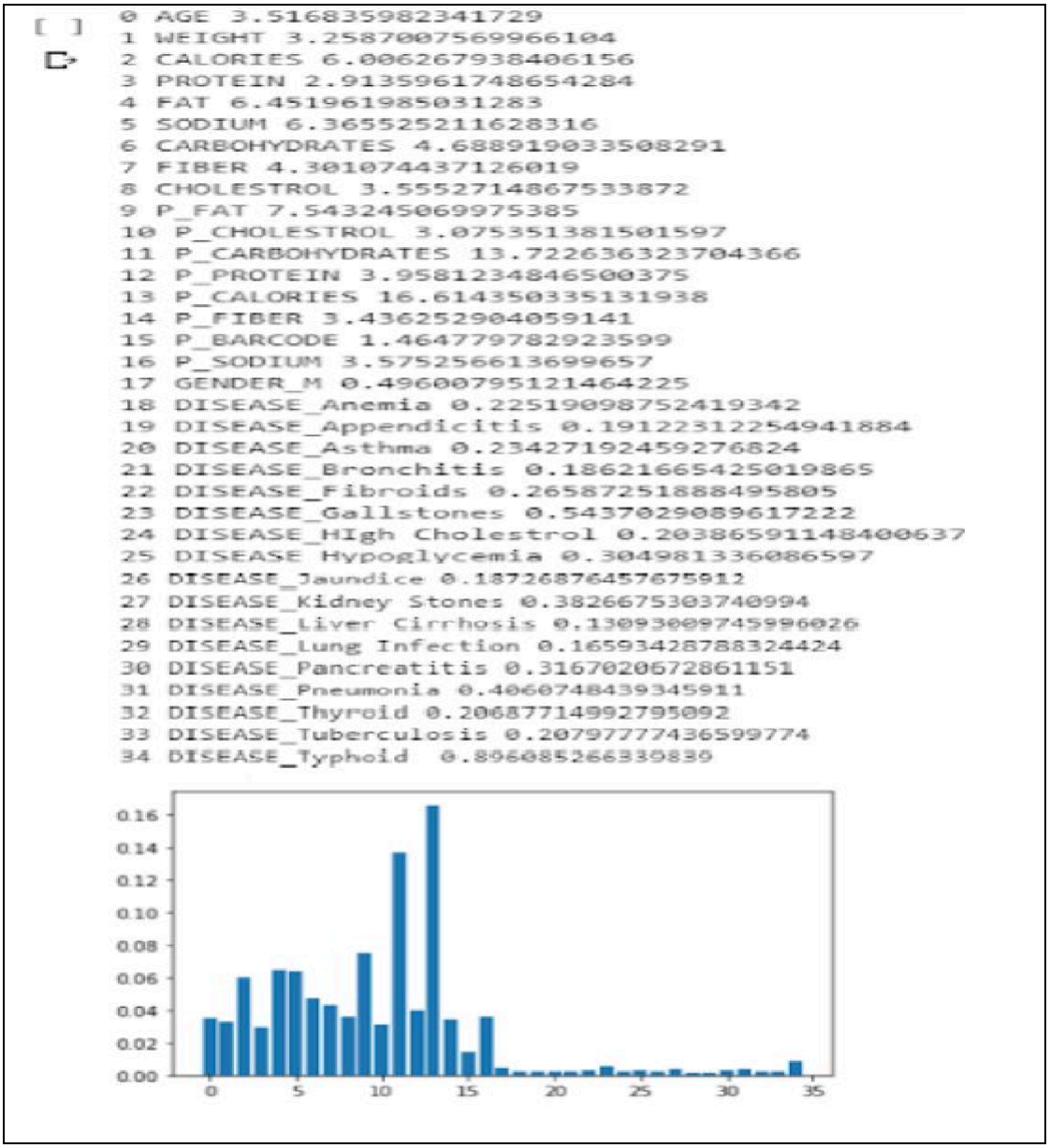

Figure 2: Feature Importance Graph

Naive Bayes :

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
print('Accuracy: ', accuracy_score(y_test,y_pred)*100)
print('Recall: ', recall_score(y_test,y_pred)*100)
print('Precision: ',precision_score(y_test,y_pred)*100)
print('F1 Score: ', f1_score(y_test,y_pred)*100)
```

```
Accuracy: 90.287656426378
Recall: 88.678560375413
Precision: 82.317234723846
F1 Score: 84.976548236426
```

Figure 3: Naive Bayes Evaluation Metrics

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
y_pred = lr.fit(X_train, y_train).predict(X_test)
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
print('Accuracy: ', accuracy_score(y_test,y_pred)*100)
print('Recall: ', recall_score(y_test,y_pred)*100)
print('Precision: ',precision_score(y_test,y_pred)*100)
print('F1 Score: ', f1_score(y_test,y_pred)*100)
```

```
Accuracy: 91.98746356787265
Recall: 90.78655435687654
Precision: 89.78766545637636
F1 Score: 89.76554354234567
```

Figure 4: Logistic Regression Evaluation Metrics

MultiLayer Perceptron :

```
y_pred = mlp.fit(X_train, y_train).predict(X_test)
print('Accuracy: ',accuracy_score(y_test,y_pred)*100)
print('Recall: ', recall_score(y_test,y_pred)*100)
print('Precision: ',precision_score(y_test,y_pred)*100)
print('F1 Score: ', f1_score(y_test,y_pred)*100)
```

```
Accuracy: 90.48888888888888
Recall: 88.675432345678909
Precision: 72.31726283048211
F1 Score: 81.44499178981938
```

Figure 5: MLP Evaluation Metrics

Recurrent Neural Network (RNN) :

```
results = cross_validate(neural_network, clinicalInput, clinicalOutput,cv=10,scoring=("accuracy","f1","recall","precision"))
```

```
Epoch 40/50
113/113 [==============================] - 9s 82ms/step - loss: 0.1489 - accuracy: 0.9442 - f1_m: 0.9441 - precision_m: 0.9441 - recall_m: 0.9441
Epoch 41/50
113/113 [==============================] - 9s 83ms/step - loss: 0.1388 - accuracy: 0.9458 - f1_m: 0.9461 - precision_m: 0.9461 - recall_m: 0.9461
Epoch 42/50
113/113 [==============================] - 9s 83ms/step - loss: 0.1664 - accuracy: 0.9397 - f1_m: 0.9397 - precision_m: 0.9397 - recall_m: 0.9397
Epoch 43/50
113/113 [==============================] - 9s 83ms/step - loss: 0.1460 - accuracy: 0.9442 - f1_m: 0.9441 - precision_m: 0.9441 - recall_m: 0.9441
Epoch 44/50
113/113 [==============================] - 10s 84ms/step - loss: 0.1483 - accuracy: 0.9444 - f1_m: 0.9447 - precision_m: 0.9447 - recall_m: 0.9447
Epoch 45/50
113/113 [==============================] - 9s 83ms/step - loss: 0.1361 - accuracy: 0.9486 - f1_m: 0.9477 - precision_m: 0.9477 - recall_m: 0.9477
Epoch 46/50
113/113 [==============================] - 9s 83ms/step - loss: 0.1199 - accuracy: 0.9519 - f1_m: 0.9516 - precision_m: 0.9516 - recall_m: 0.9516
Epoch 47/50
113/113 [==============================] - 9s 82ms/step - loss: 0.1394 - accuracy: 0.9514 - f1_m: 0.9511 - precision_m: 0.9511 - recall_m: 0.9511
Epoch 48/50
113/113 [==============================] - 9s 82ms/step - loss: 0.1790 - accuracy: 0.9283 - f1_m: 0.9284 - precision_m: 0.9284 - recall_m: 0.9284
Epoch 49/50
113/113 [==============================] - 9s 82ms/step - loss: 0.1558 - accuracy: 0.9472 - f1_m: 0.9466 - precision_m: 0.9466 - recall_m: 0.9466
Epoch 50/50
113/113 [==============================] - 9s 81ms/step - loss: 0.1958 - accuracy: 0.9244 - f1_m: 0.9242 - precision_m: 0.9242 - recall_m: 0.9242
```

Figure 6: RNN Evaluation Metrics

Long Short Term Memory (LSTM) :

```
#Determine the prediction
y_pred = cross_val_predict(neural_network, clinicalInput, clinicalOutput, cv=10)
```

```
Layer (type)                 Output Shape              Param #
=================================================================
gru_9 (GRU)                  (None, 512)               791040
_________________________________________________________________
activation_9 (Activation)    (None, 512)               0
_________________________________________________________________
batch_normalization_9 (Batch (None, 512)               2048
_________________________________________________________________
flatten_9 (Flatten)          (None, 512)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_27 (Dense)             (None, 2048)              1050624
_________________________________________________________________
dense_28 (Dense)             (None, 1024)              2098176
_________________________________________________________________
dense_29 (Dense)             (None, 2)                 2050
=================================================================
Total params: 3,943,938
Trainable params: 3,942,914
Non-trainable params: 1,024
_________________________________________________________________
Epoch 1/25
100/100 [==============================] - 16s 163ms/step - loss: 0.4458 - accuracy: 0.8094 - f1_m: 0.8094 - precision_m: 0.8094 - recall_m: 0.8094
Epoch 2/25
100/100 [==============================] - 16s 162ms/step - loss: 0.3276 - accuracy: 0.8647 - f1_m: 0.8647 - precision_m: 0.8647 - recall_m: 0.8647
Epoch 3/25
100/100 [==============================] - 16s 163ms/step - loss: 0.2650 - accuracy: 0.8950 - f1_m: 0.8950 - precision_m: 0.8950 - recall_m: 0.8950
Epoch 4/25
100/100 [==============================] - 16s 163ms/step - loss: 0.2348 - accuracy: 0.9087 - f1_m: 0.9087 - precision_m: 0.9087 - recall_m: 0.9087
Epoch 5/25
100/100 [==============================] - 16s 164ms/step - loss: 0.2094 - accuracy: 0.9216 - f1_m: 0.9216 - precision_m: 0.9216 - recall_m: 0.9216
Epoch 6/25
100/100 [==============================] - 17s 166ms/step - loss: 0.2084 - accuracy: 0.9166 - f1_m: 0.9166 - precision_m: 0.9166 - recall_m: 0.9166
Epoch 7/25
100/100 [==============================] - 16s 164ms/step - loss: 0.1941 - accuracy: 0.9259 - f1_m: 0.9259 - precision_m: 0.9259 - recall_m: 0.9259
Epoch 8/25
100/100 [==============================] - 16s 163ms/step - loss: 0.1872 - accuracy: 0.9297 - f1_m: 0.9297 - precision_m: 0.9297 - recall_m: 0.9297
Epoch 9/25
100/100 [==============================] - 16s 163ms/step - loss: 0.1814 - accuracy: 0.9278 - f1_m: 0.9278 - precision_m: 0.9278 - recall_m: 0.9278
Epoch 10/25
100/100 [==============================] - 16s 162ms/step - loss: 0.1679 - accuracy: 0.9391 - f1_m: 0.9391 - precision_m: 0.9391 - recall_m: 0.9391
Epoch 11/25
100/100 [==============================] - 17s 167ms/step - loss: 0.1656 - accuracy: 0.9434 - f1_m: 0.9434 - precision_m: 0.9434 - recall_m: 0.9434
Epoch 12/25
100/100 [==============================] - 16s 163ms/step - loss: 0.1534 - accuracy: 0.9353 - f1_m: 0.9353 - precision_m: 0.9353 - recall_m: 0.9353
Epoch 13/25
100/100 [==============================] - 16s 162ms/step - loss: 0.1375 - accuracy: 0.9491 - f1_m: 0.9491 - precision_m: 0.9491 - recall_m: 0.9491
Epoch 14/25
```

Figure 7: LSTM Evaluation Metrics

Gated Recurrent Unit (GRU):

```
results = cross_validate(neural_network, clinicalInput, clinicalOutput,cv=10,scoring=("accuracy","f1","recall","precision"))
```

```
Epoch 40/50
113/113 [==============================] - 32s 285ms/step - loss: 0.0665 - accuracy: 0.9742 - f1_m: 0.9743 - precision_m: 0.9743 - recall_m: 0.9743
Epoch 41/50
113/113 [==============================] - 32s 287ms/step - loss: 0.0646 - accuracy: 0.9725 - f1_m: 0.9726 - precision_m: 0.9726 - recall_m: 0.9726
Epoch 42/50
113/113 [==============================] - 32s 287ms/step - loss: 0.0511 - accuracy: 0.9794 - f1_m: 0.9793 - precision_m: 0.9793 - recall_m: 0.9793
Epoch 43/50
113/113 [==============================] - 32s 285ms/step - loss: 0.0656 - accuracy: 0.9778 - f1_m: 0.9776 - precision_m: 0.9776 - recall_m: 0.9776
Epoch 44/50
113/113 [==============================] - 32s 284ms/step - loss: 0.0511 - accuracy: 0.9744 - f1_m: 0.9746 - precision_m: 0.9746 - recall_m: 0.9746
Epoch 45/50
113/113 [==============================] - 33s 289ms/step - loss: 0.0521 - accuracy: 0.9794 - f1_m: 0.9795 - precision_m: 0.9795 - recall_m: 0.9795
Epoch 46/50
113/113 [==============================] - 33s 288ms/step - loss: 0.0761 - accuracy: 0.9706 - f1_m: 0.9704 - precision_m: 0.9704 - recall_m: 0.9704
Epoch 47/50
113/113 [==============================] - 33s 294ms/step - loss: 0.0611 - accuracy: 0.9775 - f1_m: 0.9776 - precision_m: 0.9776 - recall_m: 0.9776
Epoch 48/50
113/113 [==============================] - 33s 291ms/step - loss: 0.0765 - accuracy: 0.9733 - f1_m: 0.9735 - precision_m: 0.9735 - recall_m: 0.9735
Epoch 49/50
113/113 [==============================] - 33s 289ms/step - loss: 0.0537 - accuracy: 0.9792 - f1_m: 0.9790 - precision_m: 0.9790 - recall_m: 0.9790
Epoch 50/50
113/113 [==============================] - 33s 291ms/step - loss: 0.0505 - accuracy: 0.9814 - f1_m: 0.9815 - precision_m: 0.9815 - recall_m: 0.9815
```

Figure 8: GRU Evaluation Metrics

	Accuracy	Recall	Precision	F1 Measure
Naive Bayes	0.9028	0.8867	0.8231	0.8496
Logistic Regression	0.9167	0.9078	0.8978	0.8976
MLP	0.9048	0.8867	0.7321	0.8144
RNN	0.9272	0.8973	0.8564	0.8583
LSTM	0.9573	0.9278	0.8489	0.8372
GRU	0.9473	0.9020	0.8287	0.8472

Table 3: Result Analysis Table

CONCLUSION AND FUTURE WORK

The experiment is carried out in google colab laboratory. After analysing the evaluation metrics of each algorithm, it can be concluded that LSTM is the most efficient model with a significant accuracy of 95.73% .The other deep learning techniques like RNN, GRU, MLP has 92.72%, 94.73% and 90.48% accuracies respectively, while the accuracies of machine learning algorithms are 90.28% for Naive bayes and 91.67% for Logistic Regression. It is evident that the deep learning algorithms have more accuracy than machine learning algorithms. Also, LSTM has got the highest recall of 92.78. LSTM has comparatively less precision and f1-measure but its significantly higher accuracy has outshone the other algorithms in classifying the products as allowed and not allowed class. The proposed LSTM model can be used to build an effective patient-diet recommendation system where the patient can be recommended with products/food that are allowed to be consumed by the users in accordance with their past medical records.

REFERENCES

[1] S. Norouzi, A. K. Ghalibaf and S. Sistani, "A mobile application for managing diabetic patients' nutrition: A food recommender system", *Arch. Iranian Med.*, vol. 21, no. 10, pp. 466-472, 2018.

[2] M. Raut, K. Prabhu, R. Fatehpuria, S. Bangar and S. Sahu, "A personalized diet recommendation system using fuzzy ontology", *Int. J. Eng. Sci. Invention*, vol. 7, no. 3, pp. 51-55, 2018.

[3] R. Yera Toledo, A. A. Alzahrani and L. Martinez, "A food recommender system considering nutritional information and user preferences", *IEEE Access*, vol. 7, pp. 96695-96711, 2019.

[4] V. Jaiswal, "A new approach for recommending healthy diet using predictive data mining algorithm", *Int. J. Res. Anal. Rev.*, vol. 6, no. 2, pp. 58-65, 2019.

[5] N. Leipold, M. Lurz and M. Bohm, "Nutrilize a personalized nutrition recommender system: An enable study", *HealthRecSys*, vol. 3, no. 4, pp. 4-10, 2018.

[6] G. Agapito, B. Calabrese, P. H. Guzzi, M. Cannataro, M. Simeoni, I. Care, et al., "DIETOS: A recommender system for adaptive diet monitoring and personalized food suggestion", *Proc. IEEE 12th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, pp. 1-8, Oct. 2016.

[7] R. Priyadarshini, R. Barik and H. Dubey, "DeepFog: Fog computing-based deep neural architecture for prediction of stress types diabetes and hypertension attacks", *Computation*, vol. 6, no. 4, pp. 62, Dec. 2018.

[8] H. Fidan, A. Teneva, S. Stankov and E. Dimitrova, "Consumers' behavior of restaurant selection", *Proc. Int. Conf. High Technol. Sustain. Develop. (HiTech)*, pp. 1-3, 2018.