HEALTHCARE RECOMMENDATION SYSTEM USING ANN ALGORITHM

A PROJECT REPORT

Submitted by

S.BALA SUBIRAMANIAN	**(910619205014)**
S.BOGAR	**(910619205016)**
R.NAVANEETHAN	**(910619205033)**

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

K.L.N.COLLEGE OF ENGINEERING, POTTAPALAYAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"HEALTHCARE RECOMMENDATION SYSTEM USING ANN ALGORITHM"** is the bona-fide work of **"S.BALA SUBIRAMANIAN (Reg.No.910619205014), S.BOGAR (Reg.No.910619205016), R.NAVANEETHAN (Reg.No.910619205033)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
DR.P.GANESH KUMAR	**MRS N.VISHNU DEVI**
M.E.,Ph.D.,	**M.E.,**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR 2**
INFORMATION TECHNOLOGY	**INFORMATION TECHNOLOGY**
K.L.N.COLLEGE OF ENGINEERING	**K.L.N.COLLEGE OF ENGINEERING**
(An ISO 9001-2008 Certified Institution)	(An ISO 9001-2008 Certified Institution)
POTTAPALAYAM, SIVAGANGAI,	**POTTAPALAYAM, SIVAGANGAI,**
TAMIL NADU, INDIA.	**TAMIL NADU, INDIA.**

Submitted for the Project Viva-Voce held on______________

Internal Examiner **External Examiner**

Acknowledgement

Any work would be unfulfilled without a word of thanks. We hereby take pleasure in acknowledging the persons who guided me throughout our work.

First and foremost, thanks are to the omnipotent for providing us with his abundant blessings all throughout. We all extend our heartfelt thanks to **Er.K.N.K.KARTHIK, B.E.,** President of our college and **Dr.A.V.RAMPRASAD, M.E., Ph.D.,** Principal for provisioning us with the all required.

We esteem our self to articulate our sincere thanks to **DR. P.GANESH KUMAR, M.E.,PH.D.** Head of the Department for leading us towards the zenith of success.

We express our grateful thanks to our **Project Guide MRS N.VISHNU DEVI, M.E** and **Project Coordinator DR.J.S.KANCHANA, M.E., PH.D.,** for their invaluable guidance and motivation. Their assistance and advices had been very helpful throughout our project. I would like to thank all teaching and non-teaching staffs of our department who have been the sources of encouragement and ideas. I thank them for lending their support whenever needed.

ABSTRACT

Prediction of diabetes and recommend a medicine to a patient is a critical challenge in the area of clinical data analysis.These issues raise the need to apply recommender systems in the healthcare domain to help medical professionals, make more efficient and accurate health-related decisions.Recommender system has the ability to predict whether a particular user would prefer an item or not based on the user’s profile.. Diabetes, also known as chronic illness, is a group of metabolic diseases due to a high level of sugar in the blood over a long period. The risk factor and severity of diabetes can be reduced significantly if the precise early prediction is possible. Machine learning (ML) has been shown to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry. We have also seen ML techniques being used in recent developments in different areas of the Internet of Things (IoT). Various studies give only a glimpse into predicting diabetes with ML techniques. The prediction model is introduced with different combinations of features and several known classification techniques.The system is developed based on classification algorithms includes Random Forest, Logistic Regression, Gradient Boosting and Artificial Neural Network algorithms have been used. The performance measuring metrics are used for assessment of the performances of the classifiers. The performances of the classifiers have been checked on the selected features as selected by features selection algorithms.

TABLE OF CONTENTS

1.INTRODUCTION

1.1 INTRODUCTION CHAPTER 1

It is difficult to identify diabetes because of several contributory risk factors such as high blood pressure, high cholesterol, abnormal pulse rate and many other factors. Various techniques in data mining and neural networks have been employed to find out the severity of diabetes among humans. The severity of the disease is classified based on various methods like Random Forest and Logistic Regression. The nature of diabetes is complex and hence, the disease must be handled carefully.We have also seen hybrid ml model is used in predicting the accuracy of events related to diabetes. Various methods have been used for knowledge abstraction by using known methods of data mining for prediction of diabetes. Diagnosis of diabetes is traditionally done by the analysis of the medical history of the patient, physical examination report and analysis of concerned symptoms by a physician. But the results obtained from this diagnosis method are not accurate in identifying the patient of diabetes. Moreover, it is expensive and computationally difficult to analyse. Thus, to develop a non-invasive diagnosis system based on classifiers of machine learning (ML) to resolve these issues. Expert decision system based on machine learning classifiers and the application of artificial fuzzy logic is effectively diagnosis the diabetes as a result, the ratio of death decreases. The main objective of this research is to improve the performance accuracy of diabetes prediction. We proposed a machine learning based diagnosis method for the identification of diabetes in this research work.

1.2 Problem Statement

It is evident from the literature that the incidence of diabetes mellitus is increasing and that although there is evidence that the complications of diabetes can be prevented, there are still patients who lack the required knowledge and skills to manage and control their condition.

1.3 Project Objectives

- To effectively classify and predict the data.
- To enhance the performance of the overall Accurate results.

1.4 Scope of the Project

1.4.1 Existing System

Diabetes is one of the most significant causes of mortality in the world today. Prediction of cardiovascular disease is a critical challenge in the area of clinical data analysis. Diabetes is very dangerous if not immediately treated on time. The existing system doesn’t effectively classify and predict the disease in human body. Practical use oh healthcare database systems and kn0owledge discovery is difficult in diabetes diagnosis.

1.4.2 Proposed System

The proposed model is introduced to overcome all the disadvantages that arises in the existing system. This system will increase the accuracy of the Supervised classification results by classifying the data based on the diabetic prediction and others using Random Forest classification algorithm. It enhances the performance of the overall classification results. Apply hybrid data mining techniques to the dataset to investigate if ML & DL techniques can achieve equivalent (or better) results in identifying suitable treatments as that achieved in the diagnosis.

1.4 Project Plan

A project plan for a software project outlines the scope, timeline, budget, resources, and milestones for completing the project. It includes a detailed breakdown of the tasks and activities required to develop and deliver the software. The plan also includes a risk management strategy and quality assurance processes to ensure that the final product meets the client's requirements and specifications.

REVIEW NO.	DESCRIPTION OF ACTIVITY	DATE OF COMPLETION	DURATION	FIRST MONTH				SECOND MONTH				THIRD MONTH				FOURTH MONTH			
				WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 1	WEEK 2	WEEK 3	WEEK 4
0	Abstract, Requirements specification and Design	30/01/2023	--																
1	Detailed Module Description and Implementation	07/03/2023	35																
2	Demo and Draft Project Report	10/04/2023	32																
3	Final Project Demo	27/04/2023	17																

2. LITERATURE REVIEW

LITERATURE SURVEY CHAPTER 2

2.1 INTRODUCTION

The literature review has been done to know about the different types of communications used according to time and technology changes

2.2 LITERATURE REVIEW

[1] William L. Galanter, Andrew D. Boyd, and Houshang Darab "Improving the In-Hospital Mortality Prediction of Diabetes ICU PatientsUsingaProcessMining/DeepLearningArchitecture"VOL.26,NO.1,JANUARY2022

Diabetes intensive care unit (ICU) patients are at increased risk of complications leading to in-hospital mortality. Assessing the likelihood of death is a challenging and time-consuming task due to a large number of influencing factors. Healthcare providers are interested in the detection of ICU patients at higher risk, they are commonly based on a snapshot of the health conditions of a patient during the ICU stay and do not specifically consider a patient's prior medical history In this paper, a process mining/deep learning architecture is proposed to improve established severity scoring methods by incorporating the medical history of diabetes patients. First, health records of past hospital encounters are converted to event logs suitable for process mining. The event logs are then used to discover a process model that describes the past hospital encounters of patients. An adaptation of Decay Replay Mining is proposed to combine medical and demographic information

[2] MD. KAMRUL HASAN 1 , MD. ASHRAFUL ALAM1 , DOLA DAS2, ‘‘Diabetes Prediction Using Ensembling of Different Machine Learning Classifiers,’’ in Proc. Int. Conf. Comput. Netw.Informat.(ICCNI),Apr.2021.

Diabetes, also known as chronic illness, is a group of metabolic diseases due to a high level of sugar in the blood over a long period. The risk factor and severity of diabetes can be reduced significantly if the precise early prediction is possible. we are proposing a robust framework for diabetes prediction where the outlier rejection, filling the missing values, data standardization, feature selection, K-fold cross-validation, and different Machine Learning (ML) classifiers (k-nearest Neighbour, Decision Trees, Random Forest, AdaBoost, Naive Bayes, and XGBoost) and Multilayer Perceptron (MLP) were employed.

[3] I. Jenhani, N. B. Amor, and Z. Elouedi, ‘‘Decision trees as possibilistic classifiers,’’ Int. J. Approx. Reasoning, vol. 48, no. 3, pp. 784–807, Aug. 2019.

Diabetes is a major public health issue, with a prevalence of over 5.8 million in the USA, and over 23 million worldwide, and rising. The lifetime risk of developing Diabetes is one in five. Although promising evidence shows that the age-adjusted incidence of Diabetes may have plateaued, Diabetes still carries substantial morbidity and mortality, with 5-year mortality that rival those of many cancers. Diabetes represents a considerable burden to the health-care system, responsible for costs of more than $39 billion annually in the USA alone, and high rates of hospitalizations, readmissions, and outpatient visits.

[4] M. Maniruzzaman, M. J. Rahman, M. Al-MehediHasan, H. S. Suri, M. M. +Abedin, A. El-Baz, and J. S. Suri, ''Accurate diabetes risk stratification using machine learning: Role of missing value and outliers,'' J. Med. Syst.,vol.42,no.5,p.92,May2018

Diabetes may perhaps consequence in debility, severe disorder, and meager quality of lifespan. Furthermore, it could also be lethal. Hence inferring Diabetes has turn into foremost distress currently. This paper centers on various machine learning practices which assist ascertaining and perceiving innumerable Diabetes. Multifarious machine learning approaches conversed here are Hidden Markov Models, Support Vector Machine, Feature Selection, Computational intelligent classifier, prediction system, data mining techniques and genetic algorithm

[5]A. K. Dewangan and P. Agrawal (May-2020) "Classification of diabetes mellitus using machine learning techniques," IEEE Access. 8. 1-1. 10.1109/ACCESS. May-2020.2994056

Diabetes is a major public health issue, with a prevalence of over 5.8 million in the USA, and over 23 million worldwide, and rising. The lifetime risk of developing Diabetes is one in five. Although promising evidence shows that the age-adjusted incidence of Diabetes may have plateaued, Diabetes still carries substantial morbidity and mortality, with 5-year mortality that rival those of many cancers. Diabetes represents a considerable burden to the health-care system, responsible for costs of more than $39 billion annually in the USA alone.

2.3 SUMMARY

The paper here has used different classification. The merits and the drawbacks in these papers are considered for our project.

3. SYSTEM ANALYSIS

3.1REQUIREMENT ANALYSIS

3.1.1 HARDWARE REQUIREMENT

- System : Pentium IV 2.4 GHz
- Hard Disk : 200 GB
- Mouse : Logitech.
- Keyboard : 110 keys enhanced
- Ram : 4GB

3.1.2 SOFTWARE REQUIREMENTS

- O/S : Windows 7.
- Language : Python
- Front End: Anaconda Navigator – Spyder

3.1.3 MODULE SPECIFICATION

3.1.3.1 DATA PREPROCESSING

➢ Data preprocessing is the process of removing the unwanted data from the dataset.

✓ Missing data removal

✓ Encoding Categorical data

- Missing data removal: In this process, the null values such as missing values are removed using imputer library.
- Encoding Categorical data: That categorical data is defined as variables with a finite set of label values. That most machine learning algorithms require numerical input and output variables. That an integer and one hot encoding is used to convert categorical data to integer data.

3.1.3.2 CLASSIFING

- Random Forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.
- Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions. The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes. Artificial neural network tutorial covers all the aspects related to the artificial neural network

3.1.3.3 SPLITTING DATASET INTO TRAIN AND TEST DATA

- Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.

- One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
- Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.
- To train any machine learning model irrespective what type of dataset is being used you have to split the dataset into training data and testing data.

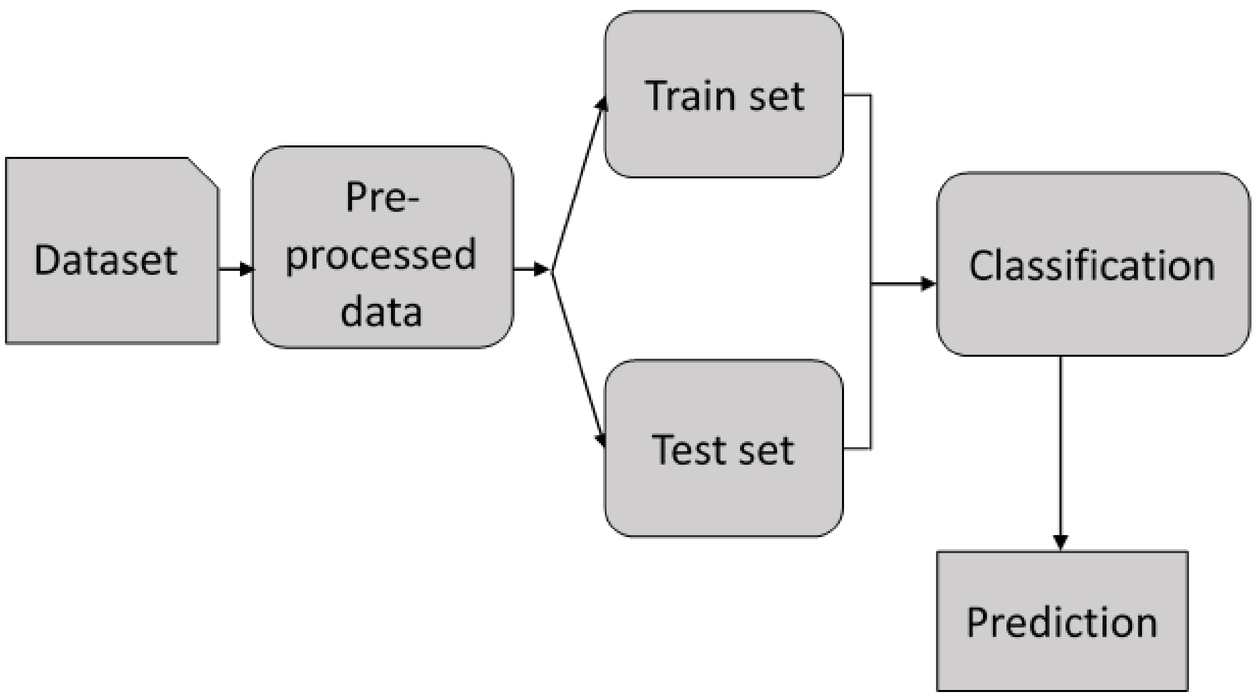
3.1.3.4 RESULT GENERATION

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

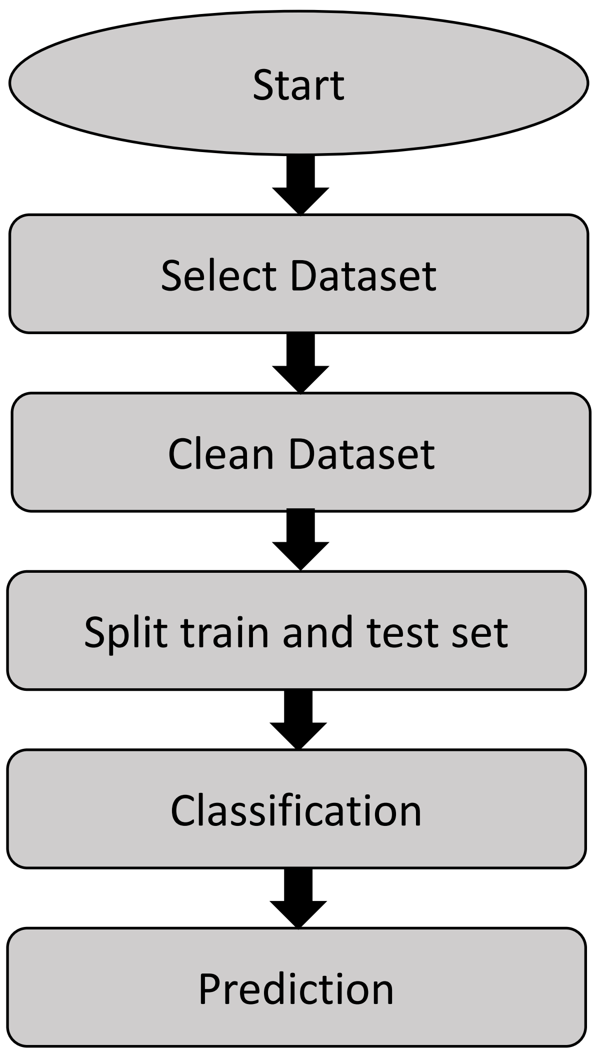
- Accuracy   Accuracy= $\frac{TP+TN}{TP+TN+FP+FN}$
- Precision   Precision=$\frac{TP}{TP+FP}$
- Recall   Recall=$\frac{TP}{TP+FN}$
- F1-Score   F1-score=$\frac{2TP}{2TP+FP+FN}$

4.SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

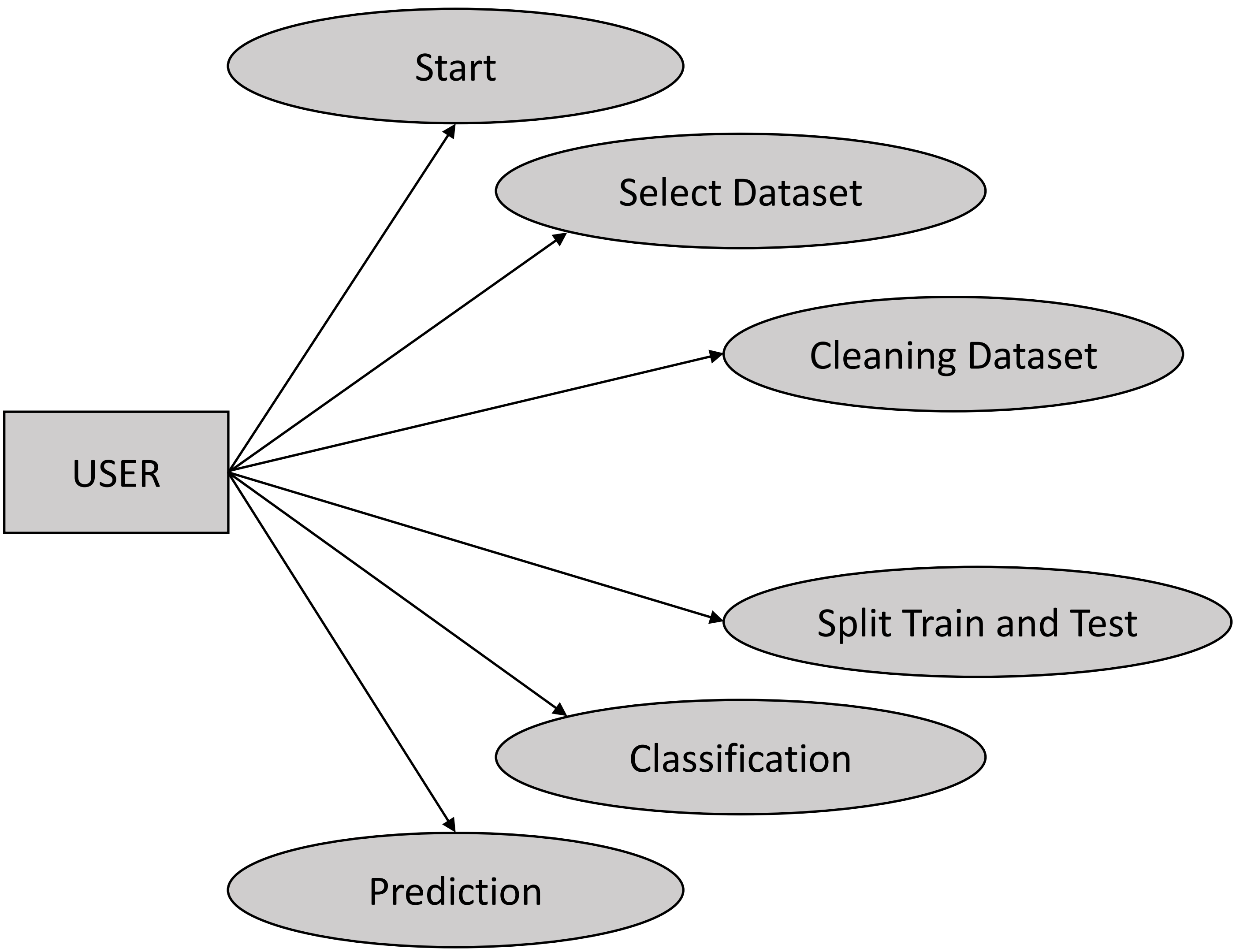


4.2 FLOW DIAGRAM

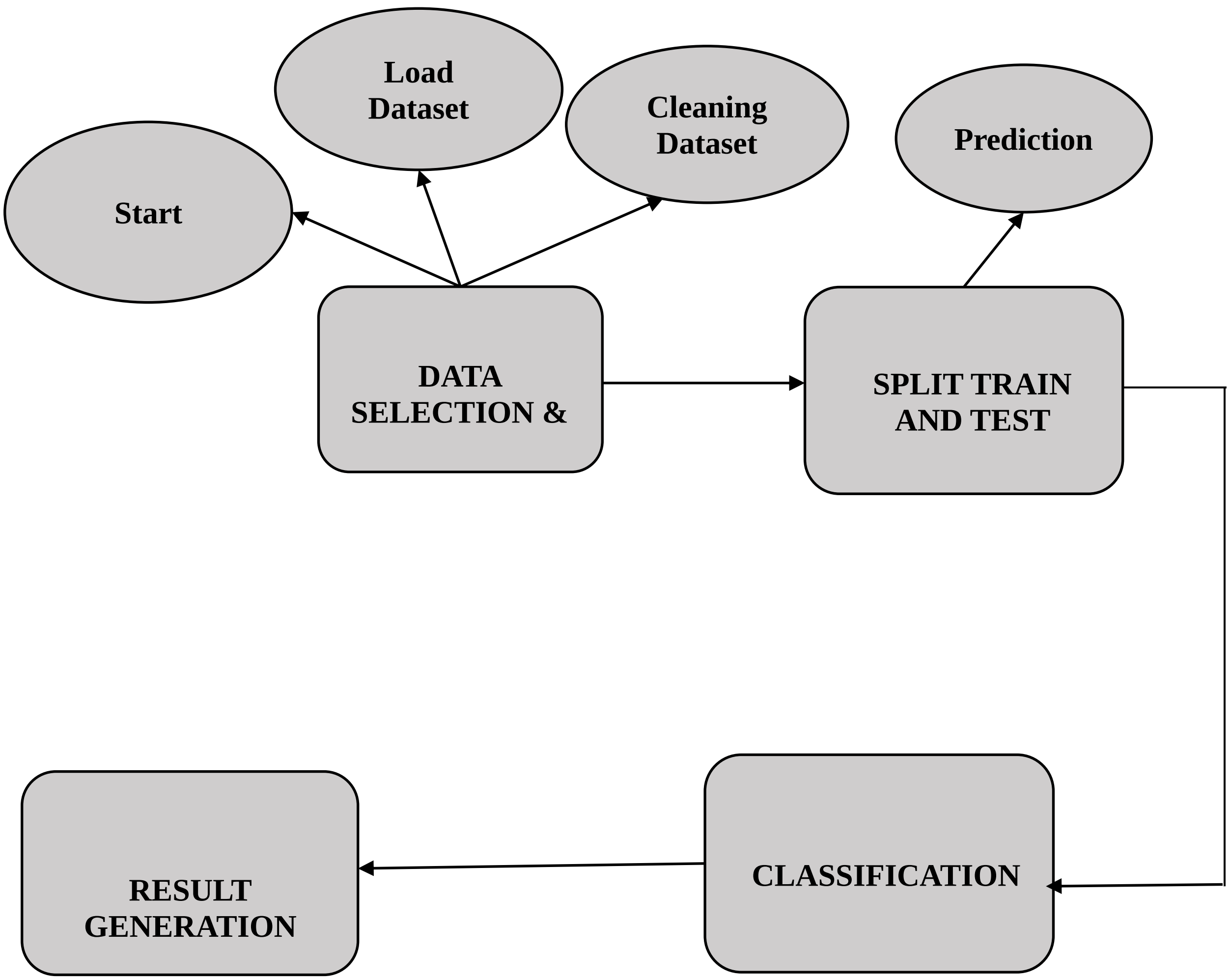


4.3 UML Diagram

USECASE DIAGRAM

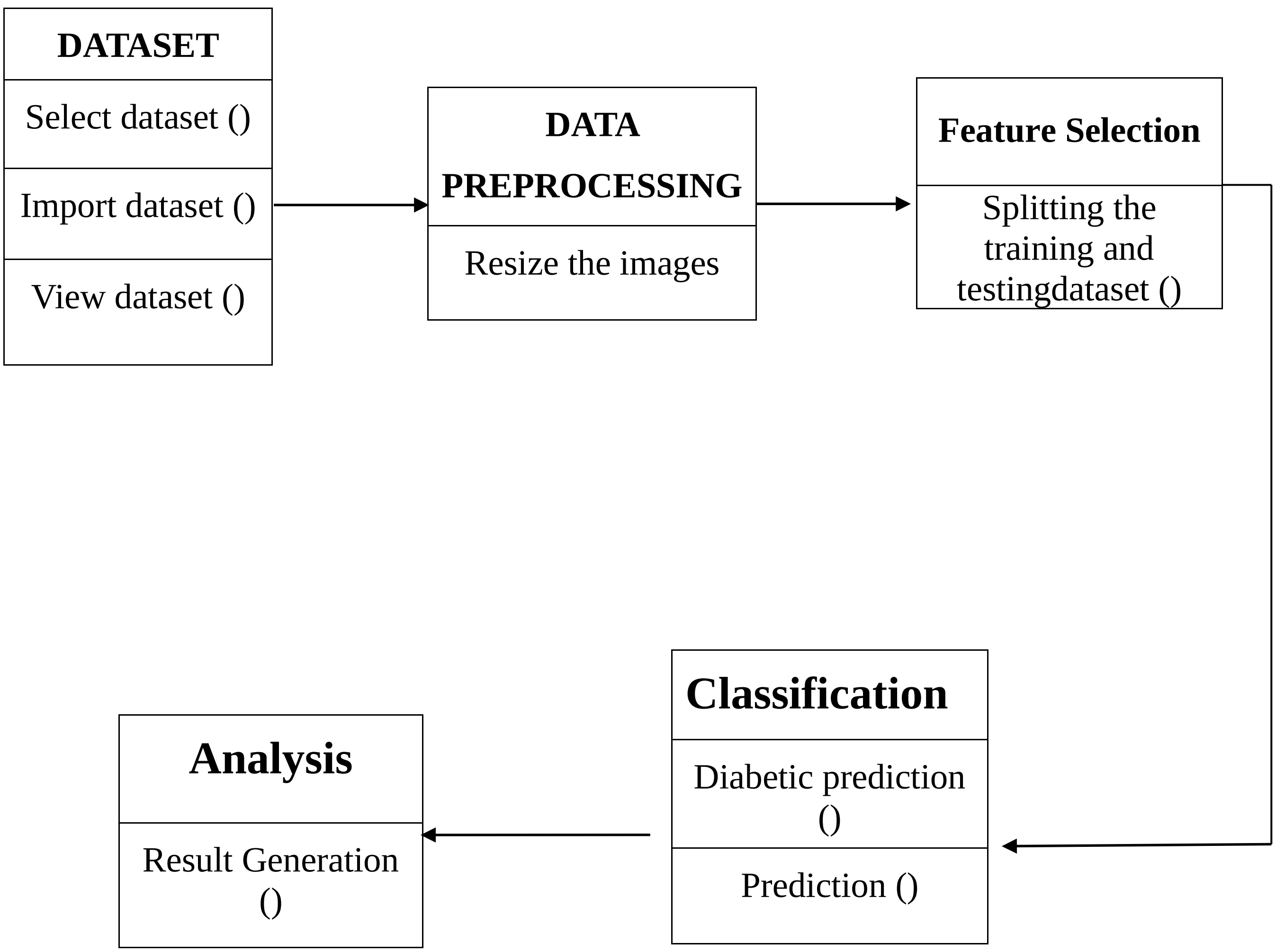


ER DIAGRAM

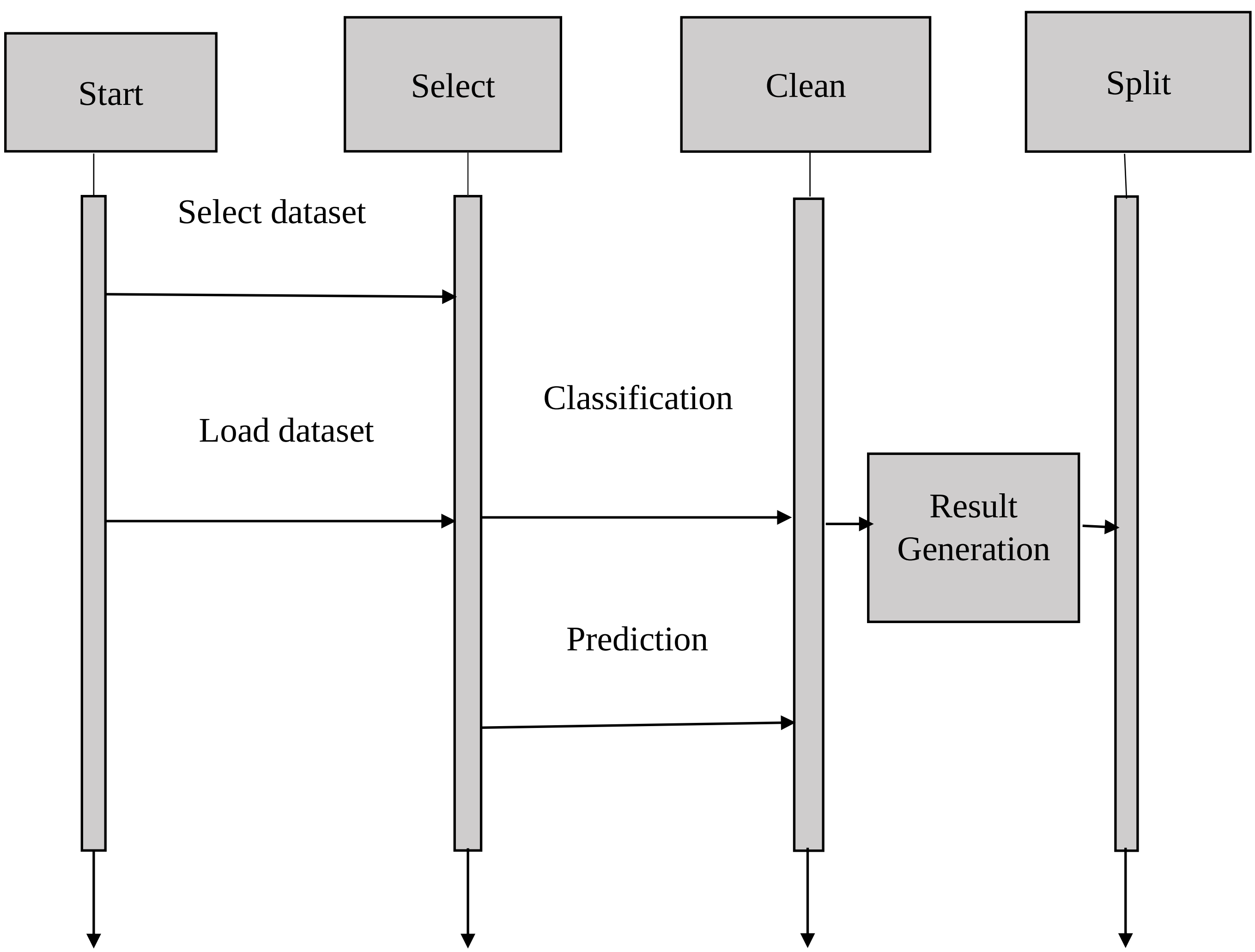


Load Dataset
Cleaning Dataset
Prediction
Start
DATA SELECTION &
SPLIT TRAIN AND TEST
RESULT GENERATION
CLASSIFICATION

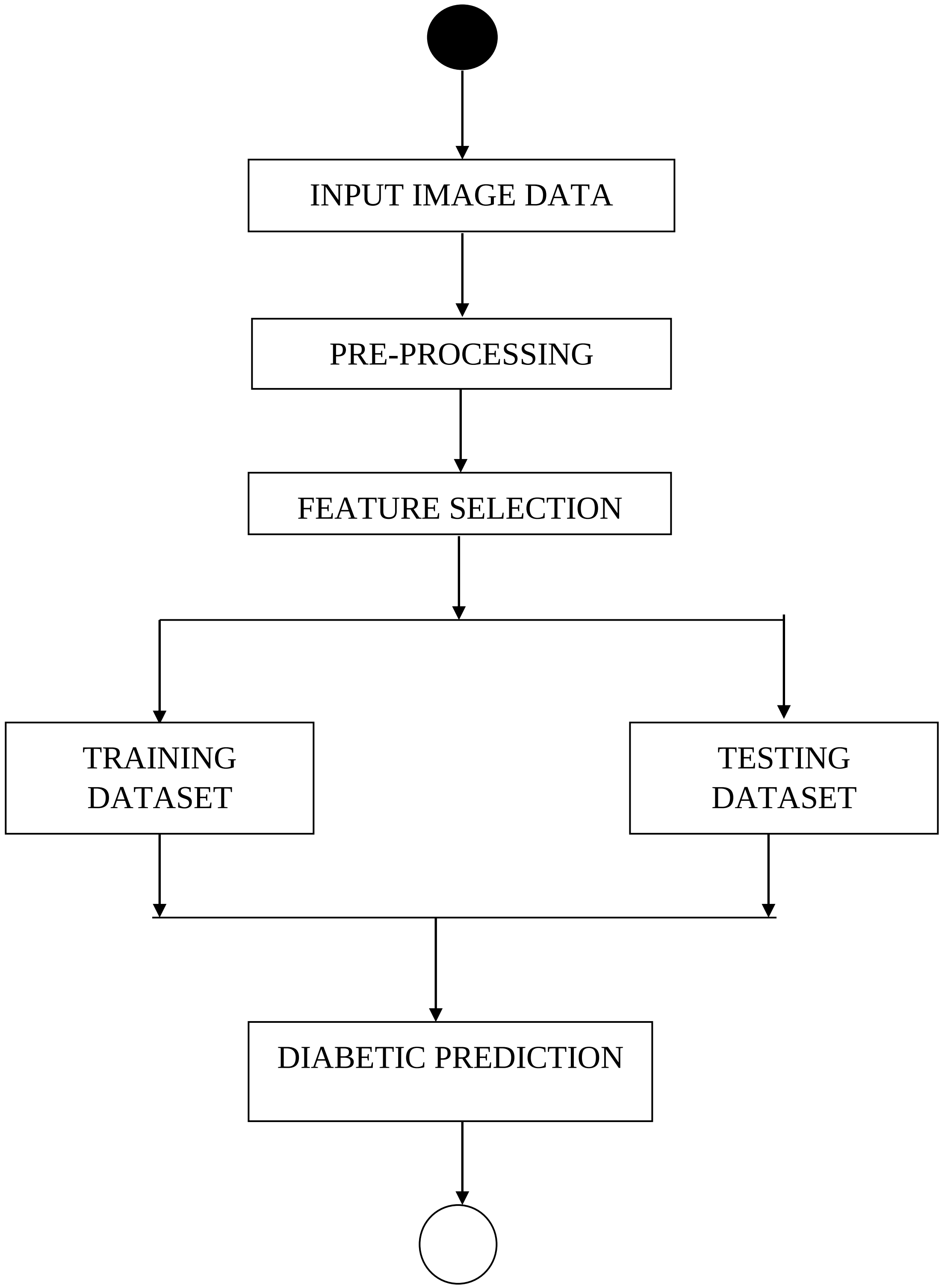
CLASS DIAGRAM



SEQUENCE DIAGRAM



ACTIVITY DIAGRAM



5. IMPLEMENTATIONS

```
import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

import scikitplot as skplt

import keras



#Read dataset

dataset = pd.read_csv('diabetes.csv')

print(dataset.head())

print(dataset.info())

print(dataset.describe())
```

```
#pre-processing

print("Checking NULL values:",dataset.isnull().any())


#EDA

"""Data visualization"""


"""heat map"""

corr = dataset.corr(method='pearson')

mask = np.triu(np.ones_like(corr, dtype=np.bool))

fig = plt.subplots(figsize=(25, 15))

sns.heatmap(dataset.corr(), annot=True,fmt='.2f',mask=mask)

plt.show()


"""Box plot"""

data1=dataset.drop('Outcome',axis=1)

data1.plot(kind='box', subplots=True, layout=(4,4), sharex=False,sharey=False
,figsize =(15,15))

plt.show()
```

```
"""pie graph"""

dataset['Outcome'].value_counts().plot(kind='pie',colors=['Brown',
'Green'],autopct='%1.1f%%',figsize=(9,9))

plt.show()


#model selection

X = dataset.iloc[:,:-1].values

y = dataset.iloc[:,-1].values


# Splitting the dataset into the Training set and Test set

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state =9)


'''RANDOM FOREST'''


#Create a Gaussian Classifier

rf_clf=RandomForestClassifier(n_estimators=100)

rf_clf.fit(x_train,y_train)


rf_ypred=rf_clf.predict(x_test)
```

```
print('\n')

print("------Accuracy------")

rf=accuracy_score(y_test, rf_ypred)*100

RF=('RANDOM FOREST Accuracy:',accuracy_score(y_test, rf_ypred)*100,'%')

print(RF)

print('\n')

print("------Classification Report------")

print(classification_report(rf_ypred,y_test))

print('\n')

print('Confusion_matrix')

rf_cm = confusion_matrix(y_test, rf_ypred)

print(rf_cm)

print('\n')

tn = rf_cm[0][0]

fp = rf_cm[0][1]

fn = rf_cm[1][0]

tp = rf_cm[1][1]

Total_TP_FP=rf_cm[0][0]+rf_cm[0][1]

Total_FN_TN=rf_cm[1][0]+rf_cm[1][1]
```

```
specificity = tn / (tn+fp)

rf_specificity=format(specificity,'.3f')

print('RF_specificity:',rf_specificity)

print()


plt.figure()

skplt.estimators.plot_learning_curve(RandomForestClassifier(), x_train, y_train,

                                     cv=7, shuffle=True, scoring="accuracy",

                                     n_jobs=-1, figsize=(6,4), title_fontsize="large",
text_fontsize="large",

                                     title="Random Forest Digits Classification Learning
Curve");


plt.figure()

sns.heatmap(confusion_matrix(y_test,rf_ypred),annot = True)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.show()
```

```
'''ANN'''
print("Artificial Neural Network")
print()
ann = keras.models.Sequential()


ann.add(keras.layers.Dense(units=7, activation='relu'))
ann.add(keras.layers.Dense(units=132, activation='relu'))
ann.add(keras.layers.Dense(units=279, activation='relu'))
ann.add(keras.layers.Dense(units=423, activation='relu'))
ann.add(keras.layers.Dense(units=579, activation='relu'))
ann.add(keras.layers.Dense(units=456, activation='relu'))
ann.add(keras.layers.Dense(units=303, activation='relu'))
ann.add(keras.layers.Dense(units=154, activation='relu'))
ann.add(keras.layers.Dense(units=1, activation='sigmoid'))


ann.compile(loss='binary_crossentropy',
            optimizer='adam',
            metrics=[
               keras.metrics.BinaryAccuracy(name='accuracy'),
```

```
            keras.metrics.Precision(name='precision'),
             keras.metrics.Recall(name='recall')
          ])
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score,fbeta_score, f1_score

n_split=10
for train_index,test_index in KFold(n_split).split(x_train):
  x_train,x_test=X[train_index],X[test_index]
  y_train,y_test=y[train_index],y[test_index]

  sc = StandardScaler()
  x_train = sc.fit_transform(x_train)
  x_test = sc.transform(x_test)

ann.fit(x_train, y_train, epochs=20)
  y_pred = ann.predict(x_test)
  y_pred = np.array([0  if i<0.5 else 1 for i in y_pred])
```

```
  # ann.evaluate(x_test, y_test)


  print(accuracy_score(y_test,y_pred))
  print(precision_score(y_test, y_pred))
  print(fbeta_score(y_test, y_pred, beta=0.5))
  print(f1_score(y_test, y_pred))


from sklearn.metrics import accuracy_score, precision_score,fbeta_score, f1_score
print(f'accuracy    : {accuracy_score(y_test,y_pred)}')
print(f'precision   : {precision_score(y_test,y_pred)}')
print(f'fBeta score : {fbeta_score(y_test,y_pred, beta=0.5)}')
print(f'f1 score    : {f1_score(y_test,y_pred)}')


trainScore = ann.evaluate(x_test, y_test, verbose=1)
ann=trainScore[1]*100
print('ANN Accuracy:',trainScore[1]*100,'%')
print('\n')
print('Confusion_matrix')
cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)

print('\n')

TP = cm[0][0]

FP = cm[0][1]

FN = cm[1][0]

TN = cm[1][1]

Total_TP_FP=cm[0][0]+cm[0][1]

Total_FN_TN=cm[1][0]+cm[1][1]

specificity = TN / (TN+FP)

ann_specificity=format(specificity)

print('ANN_specificity:',ann_specificity)

print('\n')


sns.heatmap(confusion_matrix(y_test,y_pred),annot = True)

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.show()
```

```
#comparision

vals=[ rf, ann]

inds=range(len(vals))

labels=["RF", "ANN"]

fig,ax = plt.subplots()

rects = ax.bar(inds, vals)

ax.set_xticks([ind for ind in inds])

ax.set_xticklabels(labels)

plt.show()
```

6. TESTING

UNIT TESTING:

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

INTEGRATION TESTING:

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing.

ii) Bottom-up integration testing.

WHITE BOX TESTING:

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

BLACK BOX TESTING:

- Black box testing is done to find incorrect or missing function
- Interface error
- Errors in external database access
- Performance errors.
- Initialization and termination errors

In 'functional testing', is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called 'black box testing'. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

SOFTWARE TESTING STRATEGIES

VALIDATION TESTING:

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many,But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

USER ACCEPTANCE TESTING:

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

OUTPUT TESTING:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

7. SCREENSHOTS

data - DataFrame

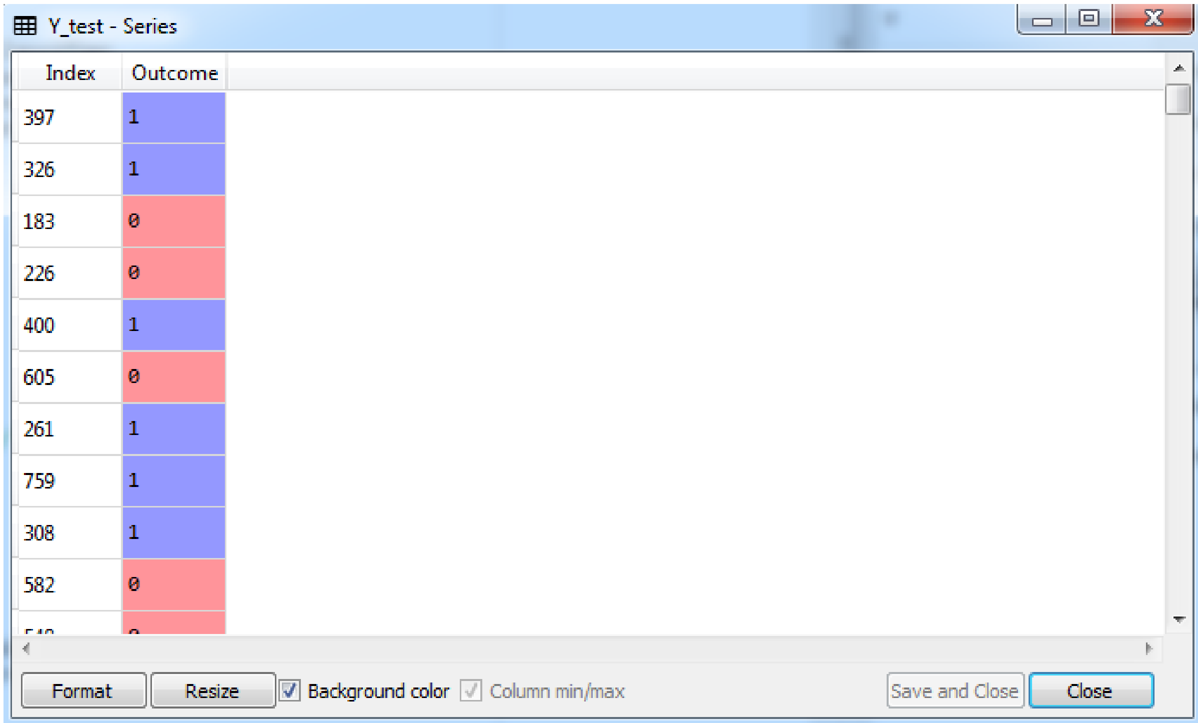
Index	Glucose	odPressu	inThickne	Insulin	BMI	PedigreeF	Age	Outcome
1	85	66	29	0	26.6	0.351	31	0
3	89	66	23	94	28.1	0.167	21	0
5	116	74	0	0	25.6	0.201	30	0
7	115	0	0	0	35.3	0.134	29	0
10	110	92	0	0	37.6	0.191	30	0
12	139	80	0	0	27.1	1.441	57	0
18	103	30	38	83	43.3	0.183	33	0
20	126	88	41	235	39.3	0.704	27	0
21	99	84	0	0	35.4	0.388	50	0
27	97	66	15	140	23.2	0.487	22	0

Format Resize Background color Column min/max Save and Close Close

X_train - DataFrame

Index	Glucose	odPressu	inThickne	Insulin	BMI	PedigreeF	Age
109	95	85	25	36	37.4	0.247	24
0	148	72	35	0	33.6	0.627	50
209	184	84	33	0	35.5	0.355	41
667	111	70	27	0	27.5	0.141	40
589	73	0	0	0	21.1	0.342	25
508	84	50	23	76	30.4	0.968	21
585	93	56	11	0	22.5	0.417	22
194	85	55	20	0	24.4	0.136	42
556	97	70	40	0	38.1	0.218	30
232	79	80	25	37	25.4	0.583	22

Format Resize Background color Column min/max Save and Close Close


Y_test - Series
Index
Outcome
397 1
326 1
183 0
226 0
400 1
605 0
261 1
759 1
308 1
582 0
Format
Resize
Background color
Column min/max
Save and Close
Close


y_pred - NumPy object array
0
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 1
8 1
Format
Resize
Background color
Save and Close
Close

corr - DataFrame

Index	Glucose	3loodPressur	SkinThickness	Insulin	BMI	esPedigreeFu	Age
Glucose	1	0.220699	0.0845541	0.332712	0.291421	0.163684	0.310394
BloodPressure	0.220699	1	0.226704	0.100708	0.343193	0.0708484	0.28573
SkinThickness	0.0845541	0.226704	1	0.439518	0.403183	0.193493	-0.08768
Insulin	0.332712	0.100708	0.439518	1	0.205065	0.189918	-0.02870
BMI	0.291421	0.343193	0.403183	0.205065	1	0.168178	0.10245
DiabetesPedigreeFunction	0.163684	0.0708484	0.193493	0.189918	0.168178	1	0.05777
Age	0.310394	0.285733	-0.0876813	-0.0287075	0.10245	0.0577701	1
Outcome	0.462712	0.0786622	0.0802831	0.133391	0.296	0.179108	0.24505

Format Resize Background color Column min/max Save and Close Close

df_majority - DataFrame

Index	Glucose	oodPressu	inThickne	Insulin	BMI	Pedigreef	Age	Outcome
1	85	66	29	0	26.6	0.351	31	0
3	89	66	23	94	28.1	0.167	21	0
5	116	74	0	0	25.6	0.201	30	0
7	115	0	0	0	35.3	0.134	29	0
10	110	92	0	0	37.6	0.191	30	0
12	139	80	0	0	27.1	1.441	57	0
18	103	30	38	83	43.3	0.183	33	0
20	126	88	41	235	39.3	0.704	27	0
21	99	84	0	0	35.4	0.388	50	0
27	97	66	15	140	23.2	0.487	22	0

Format Resize Background color Column min/max Save and Close Close

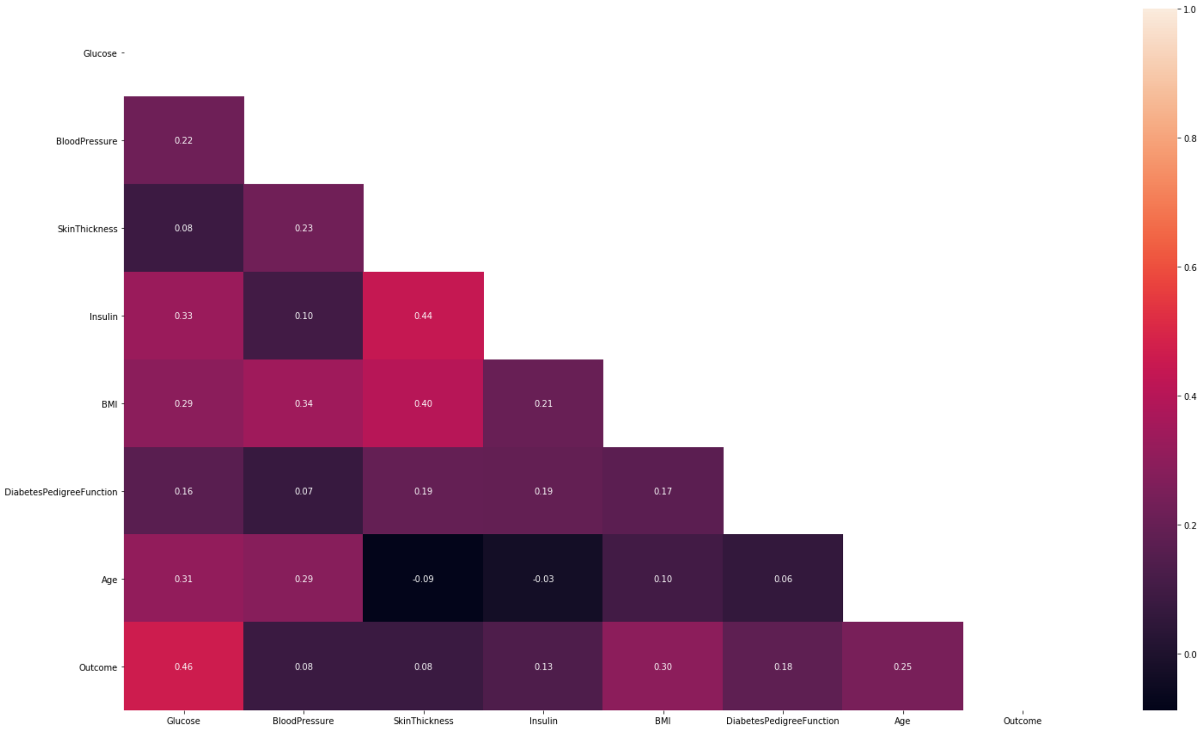
IPython console

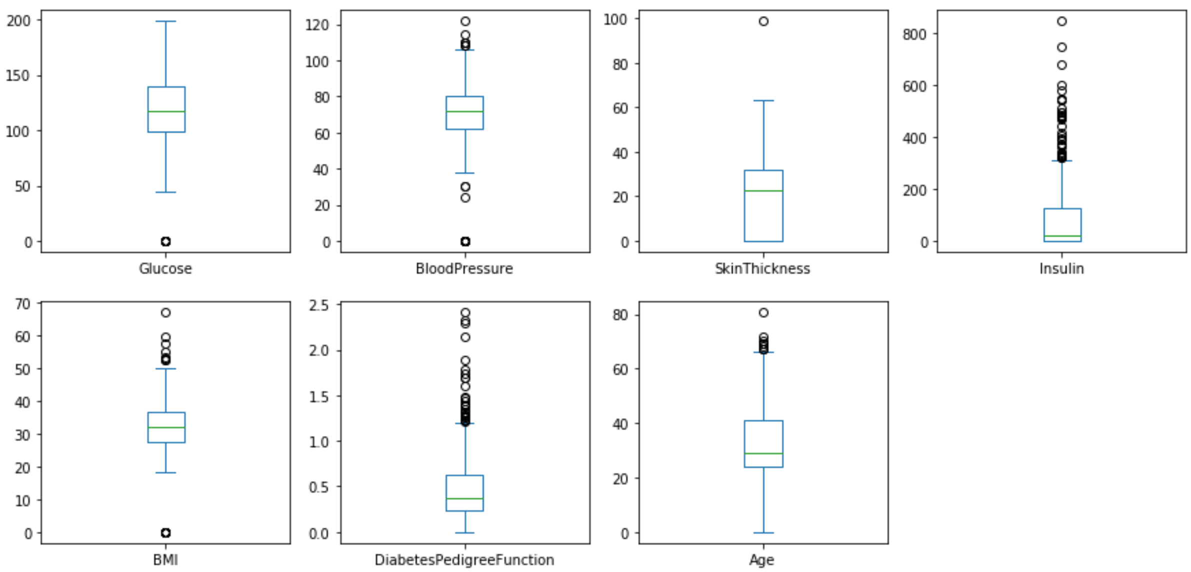
Console 1/A | Console 2/A

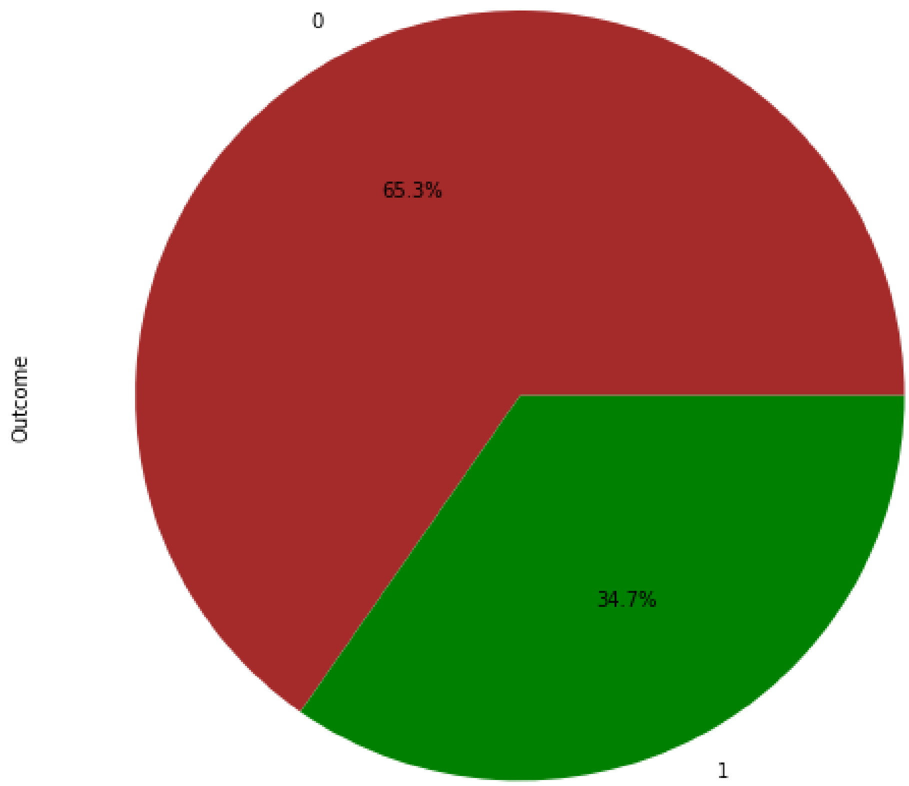
```
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:550:
FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
   Glucose  BloodPressure  ...  Age  Outcome
0      148             72  ...   50        1
1       85             66  ...   31        0
2      183             64  ...   32        1
3       89             66  ...   21        0
4      137             40  ...   33        1

[5 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 773 entries, 0 to 772
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Glucose                   773 non-null    int64
 1   BloodPressure             773 non-null    int64
 2   SkinThickness             773 non-null    int64
 3   Insulin                   773 non-null    int64
 4   BMI                       773 non-null    float64
 5   DiabetesPedigreeFunction  773 non-null    float64
 6   Age                       773 non-null    int64
 7   Outcome                   773 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 48.4 KB
None
          Glucose  BloodPressure  ...         Age     Outcome
count  773.000000     773.000000  ...  773.000000  773.000000
mean   120.112549      68.658473  ...   33.025873    0.346701
std     33.311787      20.073629  ...   12.021542    0.476228
min      0.000000       0.000000  ...    0.000000    0.000000
25%     99.000000      62.000000  ...   24.000000    0.000000
50%    117.000000      72.000000  ...   29.000000    0.000000
75%    140.000000      80.000000  ...   41.000000    1.000000
max    199.000000     122.000000  ...   81.000000    1.000000

[8 rows x 8 columns]
Checking NULL values: Glucose                     False
BloodPressure               False
SkinThickness               False
```


Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
0.22
0.08 0.23
0.33 0.10 0.44
0.29 0.34 0.40 0.21
0.16 0.07 0.19 0.19 0.17
0.31 0.29 -0.09 -0.03 0.10 0.06
0.46 0.08 0.08 0.13 0.30 0.18 0.25
Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
1.0
0.8
0.6
0.4
0.2
0.0


Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age

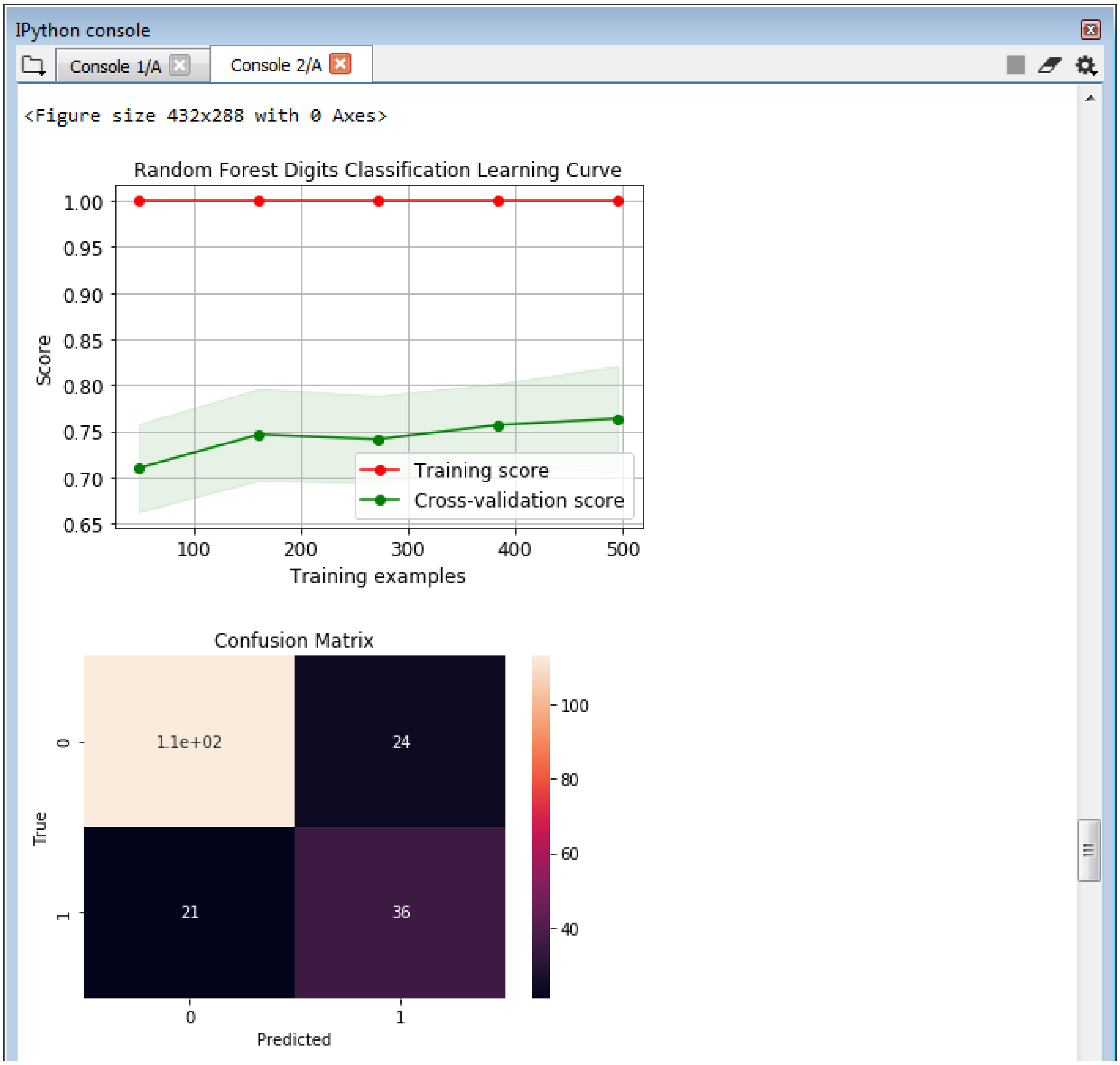



```
------Accuracy------
('RANDOM FOREST Accuracy:', 76.80412371134021, '%')


------Classification Report------
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       134
           1       0.63      0.60      0.62        60

    accuracy                           0.77       194
   macro avg       0.73      0.72      0.72       194
weighted avg       0.77      0.77      0.77       194
```


IPython console
Console 1/A
Console 2/A
<Figure size 432x288 with 0 Axes>
Random Forest Digits Classification Learning Curve
Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
100
200
300
400
500
Training examples
Training score
Cross-validation score
Confusion Matrix
True
0
1
1.1e+02
24
21
36
Predicted
100
80
60
40

```
Artificial Neural Network

WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\backend
\tensorflow_backend.py:3172: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\backend
\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use
tf.compat.v1.global_variables instead.

Epoch 1/20
521/521 [==============================] - 2s 4ms/step - loss: 0.6338 - accuracy: 0.6372 -
precision: 0.2258 - recall: 0.0407
Epoch 2/20
521/521 [==============================] - 0s 541us/step - loss: 0.5565 - accuracy: 0.6699 -
precision: 0.5000 - recall: 0.2035
Epoch 3/20
521/521 [==============================] - 0s 536us/step - loss: 0.5227 - accuracy: 0.7255 -
precision: 0.6142 - recall: 0.4535
Epoch 4/20
521/521 [==============================] - 0s 923us/step - loss: 0.5149 - accuracy: 0.7294 -
precision: 0.5896 - recall: 0.5930
```

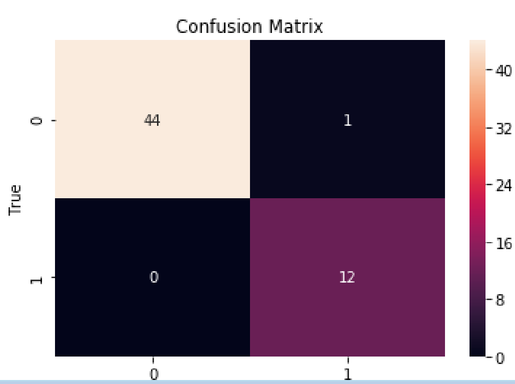
IPython console

Console 1/A | Console 2/A

```
522/522 [==============================] - 1s 1ms/step - loss: 0.0035 - accuracy: 0.9981 -
precision: 1.0000 - recall: 0.9947
Epoch 20/20
522/522 [==============================] - 1s 1ms/step - loss: 0.0029 - accuracy: 0.9981 -
precision: 1.0000 - recall: 0.9947
0.9824561403508771
0.9230769230769231
0.9375
0.9600000000000001
accuracy     : 0.9824561403508771
precision    : 0.9230769230769231
fBeta score  : 0.9375
f1 score     : 0.9600000000000001
57/57 [==============================] - 0s 4ms/step
ANN Accuracy: 98.24561476707458 %


Confusion_matrix
[[44  1]
 [ 0 12]]


ANN_specificity: 0.9230769230769231
```



8.CONCLUSION

8.1 Conclusion

In this process, we present the hybrid predictive models by using machine learning method Random Forest (RF) and Artificial Neural Network (ANN) to predict diabetes disease. By method to Improve Expected Output of Semi-structured Sequential Data. It will enhance the performance of the predicted result. Finally generate the result based on accuracy, precision, recall and f1-score.

8.2 Future Enhancement

In future, it is possible to provide extensions or modifications to the proposed clustering and classification algorithms to achieve further increased performance. Apart from the experimented combination of data mining techniques, further combinations such as artificial intelligence, soft computing and other clustering algorithms can be used to improve the detection accuracy and to reduce the rate diabetes.

9.APPENDIX

9.1 ABOUT THE SOFTWARE

9.1.1 PYTHON

Python is an interpreted, high-level, general-purpose programming language. It provides constructsthat enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object- oriented, imperative, functional and procedural, it also has a comprehensive standard library. Python is a multi-paradigm programming language. Object- oriented programming and structured programming are fully supported, and many of its features support functional programming and aspectoriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycledetecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has filter(), map(), and reduce() functions; list comprehensions, dictionaries, sets and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. 51 The language's core philosophy is including aphorisms such as:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated

• Readability counts Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal. Indentation Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.Thus, the program's visual structure accurately represents the program's semantic structure. This feature is also sometimes termed the off-side rule. Statements and control flow Python's statements include (among others):

• The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language.

• Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it.

However at agiven time a name will be bound to some object, which will have a type; thus there is dynamic typing.

• The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).

• The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.

• The while statement, which executes a block of code as long as its condition is true.

• The try statement, which allows exceptionsraised in its attached code block to be caught and handled by except clauses; it also ensures that clean- up code in a finally block will always be run regardless of how the block exits.

• The raise statement, used to raise a specified exception or re-raise a caught exception. • The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.

• The def statement, which defines a function or method.

• The with statement, from Python 2.5 released on September 2006, which encloses a code block within a context manager, allowing Resource Acquisition Is Initialization (RAII)- like behavior and replaces a common try/finally idiom.

• The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.

• The assert statement, used during debugging to check for conditions that ought to apply.

• The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator.

• The import statement, which is used to import modules whose functions or variables can be used in the current program. There are three ways of using import: import [as] or from import * or from import [as], ...

• The print statement was changed to the print() function in Python 3. Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels

10.REFERENCES

- Misra, H. Gopalan, R. Jayawardena, A. P. Hills, M. Soares, A. A. RezaAlbarrán, and K. L. Ramaiya, “Diabetes in developing countries,” Journal of Diabetes, vol. 11, no. 7, pp. 522-539, Mar. 2019.
- R. Vaishali, R. Sasikala, S. Ramasubbareddy, S. Remya, and S. Nalluri, “Genetic algorithm based feature selection and MOE Fuzzy classification algorithm on Pima Indians Diabetes dataset,” in Proc. International Conference on Computing Networking and Informatics, Oct. 2017, pp. 1-5.
- Emerging Risk Factors Collaboration and other, “Diabetes mellitus, fasting blood glucose concentration, and risk of vascular disease: a collaborative meta-analysis of 102 prospective studies,” The Lancet, vol. 375, no. 9733, pp. 2215-2222, Jul. 2010.
- N. H. Choac, J. E. Shaw, S. Karuranga, Y. Huang, J. D. R. Fernandes, A. W. Ohlrogge, and B. Malandaa, “IDF Diabetes Atlas: Global estimates of diabetes prevalence for 2017 and projections for 2045,” Diabetes Research and Clinical Practice, vol. 138, pp. 271-281, Apr. 2018.
- P. Saeedi, I. Petersohn, P. Salpea, B. Malanda, S. Karurangaa, N. Unwin, S. Colagiuri, L. Guariguata, A. A. Motala, K. Ogurtsova, J. E. Shaw, D. Bright, R. Williams, and IDF Diabetes Atlas Committee, “Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the International Diabetes Federation,” Diabetes Research and Clinical Practice, vol. 157, pp. 107843, Nov. 2019.

- J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus,"
- in Proc. Annual Symposium on Computer Application in Medical Care, Nov. 1988, pp. 261-265.
- M. Maniruzzaman, M. J. Rahman, M. A. M. Hasan, H. S. Suri, M. M. Abedin, A. El-Baz, and J. S. Suri, "Accurate diabetes risk stratification using machine learning: role of missing value and outliers," Journal of Medical Systems, vol. 42, no. 5, pp. 92, May 2018.
- G. J. McLachlan, "Discriminant analysis and statistical pattern recognition," Journal of the Royal Statistical Society: Series A (Statistics in Society), vol. 168, no. 3, pp. 635-636, Jun. 2005.
- T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," IEEE Transactions on Electronic Computers, vol. 14, no. 3, pp. 326-334, Jun. 1965.
- G. I. Webb, J. R. Boughton, and Zhihai Wang, "Not So Naive Bayes: Aggregating one-dependence estimators," Machine learning, vol. 58, no. 1, pp. 5-24, Jan. 2005.
- S. B. Belhouari and A. Bermak, "Gaussian process for nonstationary time series prediction," Computational Statistics & Data Analysis, vol. 47, no. 4, pp. 705-712, Feb. 2004.
- C. Cortes and V. Vapnik , "Support-vector networks," Machine Learning, vol. 20, pp. 237-297, Sep. 1995.

- A. Reinhardt and T. Hubbard, “Using neural networks for prediction of the subcellular location of proteins,” Nucleic Acids Research, vol. 26, no. 9, pp. 2230-2236, Mar. 1998.
- B. Kégl, “The return of AdaBoost. MH: Multi-class Hamming trees,” arXiv:1312.6086, Dec. 2013.
- T. BP and H. WH, “A multivariate logistic regression equation to screen for diabetes: development and validation,” Diabetes Care, vol. 25, no. 11, pp. 1999-2003, Nov. 2002.
- I. Jenhani, N. B. Amor, and Z. Elouedi, “Decision trees as possibilistic classifiers,” International Journal of Approximate Reasoning, vol. 48, no. 3, pp. 784-807, Aug. 2008.
- L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5-32, Oct. 2001.
- D. Sisodia and D. S. Sisodia, “Prediction of diabetes using classification algorithms,” Procedia Computer Science, vol. 132, pp. 1578-1585, Jan. 2018.
- S. Perveen, M. Shahbaz, A. Guergachi, and K. Keshavjee, “Performance analysis of data mining classification techniques to predict diabetes,” Procedia Computer Science, vol. 82, pp. 115-121, Mar. 2016.
- M. Pradhan and G. R. Bamnote, “Design of Classifier for Detection of Diabetes Mellitus Using Genetic Programming,” in Proc. third International Conference on Frontiers of Intelligent Computing: Theory and Applications, Nov. 2015, pp. 763-770.

- N. Nai-arun and R. Moungmai, “Comparison of classifiers for the risk of diabetes prediction,” Procedia Computer Science, vol. 69, pp. 132-142, Dec. 2015.
- M. Maniruzzaman, N. Kumar, M. M. Abedin, M. S. Islam, H. S. Suri, A. S. El-Baz, and J. S. Suri, “Comparative approaches for classification of diabetes mellitus data: Machine learning paradigm,” Computer Methods and Programs in Biomedicine, vol. 152, pp. 23-34, Dec. 2017.
- R. Bansal, N. Gaur, and S. N. Singh, “Outlier Detection: Applications and techniques in data mining,” in Proc. sixth International Conference-Cloud System and Big Data Engineering, Jan. 2016, pp. 373-377.
- D. Cousineau and S. Chartier, “Outliers detection and treatment: A review,” International Journal of Psychological Research, vol. 3, no. 1, pp. 58-67, Mar. 2010.
- C. R. Rao, “The use and interpretation of principal component analysis in applied research,” Sankhya: The Indian Journal of Statistics, Series A, pp. ¯ 329-358, Dec. 1964.
- A. Hyvärinen and E. Oja, “Independent Component Analysis: Algorithms and applications,” Neural Networks, vol. 13, no. 4-5, pp. 411-430, Jun. 2000.
- F. Han and H. Liu, “Statistical analysis of latent generalized correlation matrix estimation in transelliptical distribution,” Bernoulli: official journal of the Bernoulli Society for Mathematical Statistics and Probability, vol. 23, no. 1, pp. 23-57, Feb. 2017.

- S. Arlot and A. Celisse, “A survey of cross-validation procedures for model selection,” Statistics surveys, vol. 4, pp. 40-79, Jul. 2010.
- D. Krstajic, L. Buturovic, D. E. Leahy, and S. Thomas, “Cross-validation pitfalls when selecting and assessing regression and classification models,” Journal of Cheminformatics, vol. 6, no. 1, pp. 10, Mar. 2014.
- X. Zeng and T. R. Martinez, “Distribution-balanced stratified crossvalidation for accuracy estimation,” Journal of Experimental & Theoretical Artificial Intelligence, vol. 12, no. 1, pp. 1-12, Nov. 2000.
- P. Cunningham and S. J. Delany, “k-Nearest neighbour classifiers,” Multiple Classifier Systems, vol. 34, no. 8, pp. 1-17, Mar. 2007.
- T. Chen and C. Guestrin, “XGboost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 2016, pp. 785-794.
- S. Hsieh, S. Hsieh, P. Cheng, C. Chen, K. Hsu, I. Wang, and F. Lai, “Design ensemble machine learning model for breast cancer diagnosis,” Journal of Medical Systems, vol. 36, no. 2012, pp. 2841-2847, Jul. 2011.
- B. Harangi, “Skin lesion classification with ensembles of deep convolutional neural networks,” Journal of Biomedical Informatics, vol. 86, pp. 25-32, Oct. 2018.
- A. S. Miller, B. H. Blott, and others, “Review of neural network applications in medical imaging and signal processing,” Medical and Biological Engineering and Computing, vol. 30, no. 5, pp. 449-464, Jan. 1992.

- D. E. Rumelhart, G. E. Hinton, and J. Ronald, “Review of neural network applications in medical imaging and signal processing,” Nature, vol. 323, no. 6088, pp. 533-536, Oct. 1986.
- A. S. Glas, J. G. Lijmer, M. H. Prins, G. J. Bonsel, and P. M. M. Bossuyt, “The Diagnostic Odds Ratio: A single indicator of test performance,” Journal of Clinical Epidemiology, vol. 56, no. 11, pp. 1129-1135, Nov. 2003.
- P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” arXiv:1710.05941, Oct. 2017.
- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” The journal of machine learning research, vol. 15, no. 1, pp. 1929-1958, Jan. 2014.
- J. J. Deeks, “Systematic reviews of evaluations of diagnostic and screening tests,” BMJ, vol. 323, no. 7305, pp. 157-162, Jul. 2001.
- L. Li, “Diagnosis of Diabetes Using a Weight-Adjusted Voting Approach,” in Proc. IEEE International Conference on Bioinformatics and Bioengineering, Nov. 2014, pp. 320-324.
- A. K. Dewangan and P. Agrawal, “Classification of diabetes mellitus using machine learning techniques,” International Journal of Engineering and Applied Sciences, vol. 2, no. 5, pp. 145-148, May 2015.
- S. Bashir, U. Qamar, and F. H. Khan, “IntelliHealth: A medical decision support application using a novel weighted multi-layer classifier ensemble framework,” Journal of Biomedical Informatics, vol. 59, pp. 185-200, Feb. 2016.

- H. Kaur and V. Kumari, “Predictive modelling and analytics for diabetes using a machine learning approach,” Applied Computing and Informatics, Dec. 2018.
- Q. Wang, W. Cao, W. Guo, J. Ren, Y. Cheng, and D. N. Davis, “DMP_MI: An effective diabetes mellitus classification algorithm on imbalanced data with missing values,” IEEE Access, vol. 7, pp. 102232-102238, Jul. 2019.
- S. P. Chatrati, G. Hossain, A. Goyal, A. Bhan, S. Bhattacharya, D. Gaurav, and S. M. Tiwari, “Smart Home Health Monitoring System for Predicting Type 2 Diabetes and Hypertension,” Journal of King Saud Universitycomputer and Information Sciences, Jan. 2020.
- L. Jiang, Z. Cai, D. Wang, and S. Jiang, “Survey of Improving K-NearestNeighbor for Classification,” in Proc. Fourth International Conference on Fuzzy Systems and Knowledge Discovery, Aug. 2007, pp. 679-683.
- R. E. Schapire, “Explaining AdaBoost,” in Empirical Inference, pp. 37-52, Oct. 2013
- S. Taheri and M. Mammadov, “Learning the naive Bayes classifier with optimization models,” International Journal of Applied Mathematics and Computer Science, vol. 23, no. 4, pp. 787-795, Dec. 2013.