**PREDICTION OF DIABETES USING**

**MACHINE LEARNING**

By

**KARANAM VENKATA SAI ESWAR** **20BEC1266**

**ACHANTA SAMPATH MIHIR**  **20BEC1351**

A project report submitted to

**DR SATHIYA NARAYANAN S**

**SCHOOL OF ELECTRONICS ENGINEERING**

in partial fulfilment of the requirements for the course of

**CSE3506 – ESSENTIALS OF DATA ANALYTICS**

in

**B.TECH. - ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**APRIL 2023**

**BONAFIDE CERTIFICATE**

Certified that this project report e­­­­ntitled “**Prediction of Diabetes using Machine Learning ”** is a bonafide work of **Karanam Venkata Sai Eswar – 20BEC1266, Achanta Sampath Mihir -20BEC1351** who carried out the project work under my supervision and guidance for **CSE3506 – ESSENTIALS OF DATA ANALYTICS**

**DR SATHIYA NARAYANAN S**

Associate Professor

School of Electronics Engineering (SENSE),

Vellore Institute of Technology (VIT Chennai)

Chennai – 600 127.

**ABSTRACT**

Machine Learning is used across many ranges around the world. The healthcare industry is no exclusion. Machine Learning can play an essential role in predicting presence/absence of locomotors disorders, heart diseases and more. Such information, if predicted well in advance, can provide important intuitions to doctors who can then adapt their diagnosis and dealing per patient basis. We work on predicting diabetes h diseases in people using Machine Learning algorithms. In this project we perform the comparative analysis of different machine algorithms such as Logistic Regression, Adaboost, Decision tree, Gradient Boosting, XGB, SVC and Random Forest and we have taken different health parameters into consideration such as Age, Gender, Polyuria, Polydipsia, Sudden Weight loss, weakness, polyphagia and many more. By taking all such factors into consideration, we propose an ensemble classifier which perform hybrid classification by taking strong and weak classifiers since it can have multiple number of samples for training and validating the data so we perform the analysis of existing classifier and proposed classifier.

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and a deep sense of gratitude to our project guide, **DR SATHIYA NARAYANAN S,** Associate Professor Senior Grade, School of Electronics Engineering, for his consistent encouragement and valuable guidance pleasantly offered to us throughout the course of the project work.

We are extremely grateful to **Dr. Susan Elias,** Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

### We thank our Head of the Department, **Dr. Mohana prasad K,** for his support throughout this project.

We also take this opportunity to thank all the faculty of the school for their support and the wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

**KARANAM VENKATA SAI ESWAR ACHANTA SAMPATH MIHIR**

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **SR. NO.** |  | **TITLE** | **PAGE NO.** |
|  |  | ABSTRACT | 3 |
|  |  | ACKNOWLEDGEMENT | 4 |
|  |  |  |  |
| 1 |  | INTRODUCTION | 6 |
|  |  |  |  |

2 LITERATURE REVIEW 8

3 PROPOSED SYSTEM 9

4 MAIN RESULTS AND INFERENCES 30

5 CONCLUSION AND FUTURE WORK 40

6 REFERENCES 41

**CHAPTER 1**

**INTRODUCTION**

**1.1 BACKGROUND AND MOTIVATION**

Motivation behind doing this project, Prediction of diabetes disease using machine learning algorithms is to improve the accuracy and efficiency of diagnosing and treating this chronic disease. Diabetes is a widespread and costly disease that affects millions of people worldwide. Early diagnosis and intervention are critical to managing the disease and preventing complications such as heart disease, kidney disease, and blindness.

Machine learning algorithms can be used to analyze large amounts of data, including medical records, lifestyle factors, and genetic information, to identify patterns and predict the risk of diabetes. By using these algorithms, healthcare providers can better predict the likelihood of a patient developing diabetes and take proactive steps to prevent or manage the disease.

Furthermore, machine learning algorithms can help to identify subgroups of patients with different diabetes risk profiles, allowing for personalized treatment plans that are tailored to the individual's specific needs. This approach has the potential to significantly improve the management of diabetes, reduce healthcare costs, and ultimately improve patient outcomes.

**1.2 DATASET DESCRIPTION**

This dataset consists of 16 features and a target variable. The detailed description of all the features are as follows:[5]

1. **Age:** Defines patient’s age.
2. **Gender:** Describes whether patient is male or female.
3. **Polyuria:** Monitors the amount of glucose present in patient’s blood.
4. **Polydipsia:** Monitors patient’s thirst and dehydration level.
5. **Sudden Weight Loss:** Checks whether patient have lost sudden weight.
6. **Weakness:** Determine whether patient is physically weak or strong,
7. **Polyphagia:** Monitors patient’s insulin levels and hunger levels.
8. **Genital thrush:** Checks whether there is any fungal infection occurred.
9. **Visual Blurring:** Checks whether patient is suffering from difficulty in his/her vision.
10. **Itching:** Checks whether patient is suffering from any itching issues.
11. **Irritability:** Check whether patient is suffering from any irritation.
12. **Delayed Healing:** Monitors the healing time of a patient.
13. **Partial paresis:** Monitors patient’s bone movement.
14. **Muscle stiffness:** Monitors patient’s muscle movement whether it is impaired or weekend.
15. **Alopecia:** Monitors patient’s hair loss and checks whether he is suffering from type 1 or type 2 diabetes.
16. **Obesity:** Describes whether patient is obese or not.

**CHAPTER 2**

**LITERATURE REVIEW**

After conducting a literature review on healthcare datasets, researchers have employed different techniques and methods to perform analysis and predictions. Several prediction models have been developed using a combination of data mining techniques and machine learning algorithms. For example, Dr. Saravana Kumar N M, Eswari, Sampath P, and Lavanya S (2015) utilized Hadoop and MapReduce to analyze diabetic data, predicting the type of diabetes and associated risks. The system they developed is cost-effective for healthcare organizations that want to implement a Hadoop-based solution.[1] Aiswarya Iyer (2015) employed classification techniques to identify hidden patterns in a diabetes dataset. The study compared the performance of Naïve Bayes and Decision Trees algorithms and showed the effectiveness of both approaches.[2] K. Rajesh and V. Sangeetha (2012) used classification technique. They used C4.5 decision tree algorithm to find hidden patterns from the dataset for classifying efficiently.[3] B.M. Patil, R.C. Joshi and Durga Toshniwal (2010) proposed Hybrid Prediction Model which includes Simple K-means clustering algorithm, followed by application of classification algorithm to the result obtained from clustering algorithm. In order to build classifiers C4.5 decision tree algorithm is used.[4]

**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1 IMPLEMENTED METHODS/FUNCTIONS**

We have used different Machine learning algorithms in this project. Details of each machine learning model will be given below:

1. Random Forest:

Random forest is a machine learning algorithm used for classification and regression analysis. It is an ensemble learning method that combines multiple decision trees to create a more robust and accurate model. In a random forest model, each decision tree is created using a subset of the available features and a random sample of the training data. This helps to reduce overfitting and improve the model's generalization ability. During prediction, each decision tree in the forest independently generates a prediction, and the final output is determined by a majority vote (in classification) or an average (in regression) of all the individual tree predictions.

1. Decision Tree:

Decision tree is a machine learning algorithm used for classification and regression analysis. It is a type of supervised learning algorithm that is used to model decisions or decision-making processes by creating a tree-like model of decisions and their possible consequences. In a decision tree model, the data is split into different subsets based on the values of different input features. The algorithm iteratively splits the data into different subsets by selecting the feature that provides the most information gain or reduction in entropy. This process continues until the subsets are pure or no further improvements can be made.

1. KNN:

KNN stands for "k-nearest neighbors" and is a machine learning algorithm used for both classification and regression. It is a type of instance-based learning, which means that the model is trained on the specific examples in the training dataset rather than by fitting a parametric function. In a KNN model, the algorithm searches for the k nearest neighbors to the new input data point in the training dataset based on some distance metric (such as Euclidean distance). The output of the model is then determined by the most common class (in classification) or the average of the k-nearest neighbors (in regression).

1. Extra tree classifier:

Extra Trees (or Extremely Randomized Trees) is a machine learning algorithm used for classification and regression. It is a variant of the random forest algorithm that uses an ensemble of decision trees to make predictions. In an Extra Trees model, each decision tree is constructed using a random subset of the input features and a random subset of the training data. In addition, instead of finding the best split point for each feature based on the data, the algorithm randomly selects a split point for each feature. This leads to a larger number of possible splits and helps to increase the diversity of the decision trees in the ensemble.

1. Support Vector Classifier:

Support Vector Classifier (SVC) is a machine learning algorithm used for classification. It is a type of supervised learning algorithm that separates the classes of data points by finding the optimal hyperplane that maximizes the margin between the classes. In an SVC model, the algorithm first maps the input data into a high-dimensional feature space using a kernel function. It then tries to find the optimal hyperplane that maximizes the margin between the two classes in this higher-dimensional space. The support vectors are the data points closest to the decision boundary and play a crucial role in defining the optimal hyperplane.

1. Logistic Regression:

Where the goal is to predict the probability of a binary outcome (i.e., 0 or 1). It is a supervised learning algorithm that models the probability of the outcome as a function of the input features. In Logistic Regression, the algorithm learns a set of weights for each input feature that determines the contribution of that feature to the final prediction. The output of the algorithm is a probability score between 0 and 1, which is then mapped to a binary outcome using a threshold value.

**3.2 SOFTWARE USED**

We have used **JUPYTER notebook** for our project. Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. It is a popular tool used by data scientists, researchers, and developers to work with and share code in a collaborative and interactive manner.

**3.3 CODE**

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split,cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, f1\_score, precision\_score,confusion\_matrix, recall\_score, roc\_auc\_score

from xgboost import XGBClassifier

from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier

from sklearn.svm import SVC

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from IPython.display import Image

#reading the dataset

df = pd.read\_csv('diabetes\_data\_upload.csv')

#showing first few rows of the dataset

df.head()

#checking missing values per feature

df.isna().sum()

# plotting to create pie chart and bar plot distribution of target variable

plt.figure(figsize=(14,7))

plt.subplot(121)

df["class"].value\_counts().plot.pie(autopct = "%1.0f%%",colors = sns.color\_palette("prism",7),startangle = 60,labels=["Positive","Negative"],

wedgeprops={"linewidth":2,"edgecolor":"k"},explode=[.1,0],shadow =True)

plt.title("Distribution of Target variable")

plt.subplot(122)

ax = df["class"].value\_counts().plot(kind="barh")

for i,j in enumerate(df["class"].value\_counts().values):

ax.text(.7,i,j,weight = "bold",fontsize=20)

plt.title("Count of Target variable")

plt.show()

#plotting barchart for distribution

sns.countplot(x='Gender', hue='class', data=df)

#plotting target variable wrt Gender variable

plot\_criteria= ['Gender', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='Polyuria', hue='class', data=df)

#plotting target variable wrt Polyuria variable

plot\_criteria= ['Polyuria', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='Polydipsia', hue='class', data=df)

#plotting target variable wrt Polydispia variable

plot\_criteria= ['Polydipsia', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='sudden weight loss', hue='class', data=df)

#plotting target variable wrt sudden weight loss variable

plot\_criteria= ['sudden weight loss', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='weakness', hue='class', data=df)

#Distribution of Weakness

plot\_criteria= ['weakness', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='Polyphagia', hue='class', data=df)

#Distribution of Polyphagia

plot\_criteria= ['Polyphagia', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='Genital thrush', hue='class', data=df)

#Distribution of genital thrush

plot\_criteria= ['Genital thrush', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='visual blurring', hue='class', data=df)

#Distribution of visual blurring

plot\_criteria= ['visual blurring', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='delayed healing', hue='class', data=df)

#Distribution of Delayed Healing

plot\_criteria= ['delayed healing', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

#plotting barchart for distribution

sns.countplot(x='partial paresis', hue='class', data=df)

#Distribution of Partial Paresis

plot\_criteria= ['partial paresis', 'class']

cm = sns.light\_palette("red", as\_cmap=True)

(round(pd.crosstab(df[plot\_criteria[0]], df[plot\_criteria[1]], normalize='columns') \* 100,2)).style.background\_gradient(cmap = cm)

# transforming target column from string to numeric format

df['class'] = df['class'].apply(lambda x: 0 if x=='Negative' else 1)

# creating feature and target variable

X= df.drop(['class'],axis=1)

y=df['class']

#creating a list of object datatypes

objList = X.select\_dtypes(include = "object").columns

#Label Encoding for object to numeric conversion

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for feat in objList:

X[feat] = le.fit\_transform(X[feat].astype(str))

print (X.info())

X.head

# calculating correlation of all the input variables with target variable

X.corrwith(y)

X.corrwith(y).plot.bar(

figsize=(16,6),title = "Correlation with Diabetes",fontsize=15,rot=90,grid=True)

# train test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2,stratify=y, random\_state = 1234)

## checking distribution of target variable in train test split

print('Distribution of target variable in training set')

print(y\_train.value\_counts())

print('Distribution of target variable in test set')

print(y\_test.value\_counts())

# instantiating minmax scaling object

minmax = MinMaxScaler()

#apply minmax scaling on Age feature

X\_train[['Age']] = minmax.fit\_transform(X\_train[['Age']])

X\_test[['Age']] = minmax.transform(X\_test[['Age']])

X\_train.head()

import warnings

warnings.filterwarnings('ignore')

# data wrangling & pre-processing

import pandas as pd

import numpy as np

# data visualization

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

#model validation

from sklearn.metrics import log\_loss,roc\_auc\_score,precision\_score,f1\_score,recall\_score,roc\_curve,auc

from sklearn.metrics import classification\_report, confusion\_matrix,accuracy\_score,fbeta\_score,matthews\_corrcoef

from sklearn import metrics

# cross validation

from sklearn.model\_selection import StratifiedKFold

# machine learning algorithms

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier,VotingClassifier,AdaBoostClassifier,GradientBoostingClassifier,RandomForestClassifier,ExtraTreesClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import SGDClassifier

from sklearn.svm import SVC

import xgboost as xgb

from scipy import stats

from sklearn import model\_selection

from sklearn.model\_selection import cross\_val\_score

import xgboost as xgb

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

# function initializing baseline machine learning models

def GetBasedModel():

basedModels = []

basedModels.append(('LR\_L2' , LogisticRegression(penalty='l2')))

basedModels.append(('KNN7' , KNeighborsClassifier(7)))

basedModels.append(('KNN5' , KNeighborsClassifier(5)))

basedModels.append(('KNN9' , KNeighborsClassifier(9)))

basedModels.append(('KNN11' , KNeighborsClassifier(11)))

basedModels.append(('CART' , DecisionTreeClassifier()))

basedModels.append(('SVM Linear' , SVC(kernel='linear',gamma='auto',probability=True)))

basedModels.append(('SVM RBF' , SVC(kernel='rbf',gamma='auto',probability=True)))

basedModels.append(('RF\_Ent100' , RandomForestClassifier(criterion='entropy',n\_estimators=100)))

basedModels.append(('RF\_Gini100' , RandomForestClassifier(criterion='gini',n\_estimators=100)))

basedModels.append(('ET100' , ExtraTreesClassifier(n\_estimators= 100)))

basedModels.append(('ET500' , ExtraTreesClassifier(n\_estimators= 500)))

basedModels.append(('ET1000' , ExtraTreesClassifier(n\_estimators= 1000)))

return basedModels

# function for performing 10-fold cross validation of all the baseline models

def BasedLine2(X\_train, y\_train,models):

# Test options and evaluation metric

from sklearn.model\_selection import StratifiedKFold

skfolds=StratifiedKFold(n\_splits=3, shuffle=True, random\_state=42)

num\_folds = 10

scoring = 'accuracy'

seed = 7

results = []

names = []

for name, model in models:

kfold = model\_selection.KFold(n\_splits=10, shuffle=True,random\_state=seed)

cv\_results = model\_selection.cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring=scoring)

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

print(msg)

return results,msg

models = GetBasedModel()

name,results = BasedLine2(X\_train, y\_train,models)

logi = LogisticRegression(random\_state = 0, penalty = 'l2')

logi.fit(X\_train, y\_train)

y\_pred\_logi= logi.predict(X\_test)

rf\_ent = RandomForestClassifier(criterion='entropy',n\_estimators=100)

rf\_ent.fit(X\_train, y\_train)

y\_pred\_rfe = rf\_ent.predict(X\_test)

rf\_gini = RandomForestClassifier(criterion='gini',n\_estimators=100)

rf\_gini.fit(X\_train, y\_train)

y\_pred\_gini = rf\_gini.predict(X\_test)

knn = KNeighborsClassifier(9)

knn.fit(X\_train,y\_train)

y\_pred\_knn = knn.predict(X\_test)

et\_100 = ExtraTreesClassifier(n\_estimators= 100)

et\_100.fit(X\_train,y\_train)

y\_pred\_et\_100 = et\_100.predict(X\_test)

svc = SVC(kernel='linear',gamma='auto',probability=True)

svc.fit(X\_train,y\_train)

y\_pred\_svc1 = svc.predict(X\_test)

svc = SVC(kernel='rbf',gamma='auto',probability=True)

svc.fit(X\_train,y\_train)

y\_pred\_svc2 = svc.predict(X\_test)

decc = DecisionTreeClassifier()

decc.fit(X\_train,y\_train)

y\_pred\_decc = decc.predict(X\_test)

CM=confusion\_matrix(y\_test,y\_pred\_rfe)

sns.heatmap(CM, annot=True)

TN = CM[0][0]

FN = CM[1][0]

TP = CM[1][1]

FP = CM[0][1]

specificity = TN/(TN+FP)

loss\_log = log\_loss(y\_test, y\_pred\_rfe)

acc= accuracy\_score(y\_test, y\_pred\_rfe)

roc=roc\_auc\_score(y\_test, y\_pred\_rfe)

prec = precision\_score(y\_test, y\_pred\_rfe)

rec = recall\_score(y\_test, y\_pred\_rfe)

f1 = f1\_score(y\_test, y\_pred\_rfe)

mathew = matthews\_corrcoef(y\_test, y\_pred\_rfe)

model\_results =pd.DataFrame([['Random Forest',acc, prec,rec,specificity, f1,roc, loss\_log,mathew]],

columns = ['Model', 'Accuracy','Precision', 'Sensitivity','Specificity', 'F1 Score','ROC','Log\_Loss','mathew\_corrcoef'])

model\_results

data = { 'DEC':y\_pred\_decc,

'KNN': y\_pred\_knn,

'EXtra tree classifier': y\_pred\_et\_100,

'SVC2':y\_pred\_svc2,

'SVC': y\_pred\_svc1,

'LOGI':y\_pred\_logi,

'RF gini':y\_pred\_gini

}

models = pd.DataFrame(data)

for column in models:

CM=confusion\_matrix(y\_test,models[column])

TN = CM[0][0]

FN = CM[1][0]

TP = CM[1][1]

FP = CM[0][1]

specificity = TN/(TN+FP)

loss\_log = log\_loss(y\_test, models[column])

acc= accuracy\_score(y\_test, models[column])

roc=roc\_auc\_score(y\_test, models[column])

prec = precision\_score(y\_test, models[column])

rec = recall\_score(y\_test, models[column])

f1 = f1\_score(y\_test, models[column])

mathew = matthews\_corrcoef(y\_test, models[column])

results =pd.DataFrame([[column,acc, prec,rec,specificity, f1,roc, loss\_log,mathew]],

columns = ['Model', 'Accuracy','Precision', 'Sensitivity','Specificity', 'F1 Score','ROC','Log\_Loss','mathew\_corrcoef'])

model\_results = model\_results.append(results, ignore\_index = True)

model\_results

CM1=confusion\_matrix(y\_test,y\_pred\_rfe)

CM2=confusion\_matrix(y\_test,y\_pred\_decc)

CM3=confusion\_matrix(y\_test,y\_pred\_svc1)

CM4=confusion\_matrix(y\_test,y\_pred\_svc2)

CM5=confusion\_matrix(y\_test,y\_pred\_et\_100)

CM6=confusion\_matrix(y\_test,y\_pred\_logi)

CM7=confusion\_matrix(y\_test,y\_pred\_gini)

CM1

sns.heatmap(CM1, annot=True)

CM2

sns.heatmap(CM2, annot=True)

CM3

sns.heatmap(CM3, annot=True)

CM4

sns.heatmap(CM4, annot=True)

CM5

sns.heatmap(CM5, annot=True)

CM6

sns.heatmap(CM6, annot=True)

CM7

sns.heatmap(CM7, annot=True)

new\_data = pd.DataFrame({

'Age':40,

'Gender':1,

'Polyuria':0,

'Polydipsia':1,

'sudden weight loss':0,

'weakness':1,

'Polyphagia':0,

'Genital thrush':0,

'visual blurring':0,

'Itching':1,

'Irritability':0,

'delayed healing':1,

'partial paresis':0,

'muscle stiffness':1,

'Alopecia':1,

'Obesity':1

},index=[0])

import joblib

joblib.dump(rf\_ent,'model\_joblib\_diabetes')

model = joblib.load('model\_joblib\_diabetes')

model.predict(new\_data)

from tkinter import \*

import joblib

def show\_entry\_fields():

p1=int(e1.get())

p2=int(e2.get())

p3=int(e3.get())

p4=int(e4.get())

p5=int(e5.get())

p6=int(e6.get())

p7=int(e7.get())

p8=int(e8.get())

p9=int(e9.get())

p10=int(e10.get())

p11=int(e11.get())

p12=int(e12.get())

p13=int(e13.get())

p14=int(e14.get())

p15=int(e15.get())

p16=int(e16.get())

model = joblib.load('model\_joblib\_diabetes')

result=model.predict([[p1,p2,p3,p4,p5,p6,p7,p8,p8,p10,p11,p12,p13,p14,p15,p16]])

if result == 0:

Label(master, text="No Diabetes").grid(row=31)

else:

Label(master, text="Diabetes").grid(row=31)

master = Tk()

master.title("Diabetes Prediction System")

label = Label(master, text = "Diabetes Prediction System", bg = "black", fg = "white").\

grid(row=0,columnspan=2)

Label(master, text="Enter Your Age").grid(row=1)

Label(master, text="Male Or Female [1/0]").grid(row=2)

Label(master, text="Enter Value of Polyuria").grid(row=3)

Label(master, text="Enter Value of Polydipsia").grid(row=4)

Label(master, text="Enter Value of sudden weakness").grid(row=5)

Label(master, text="Enter Value of weakness").grid(row=6)

Label(master, text="Enter Value of polyphagia").grid(row=7)

Label(master, text="Enter Value of genital thrush").grid(row=8)

Label(master, text="Enter Value of visual blurring").grid(row=9)

Label(master, text="Enter Value of itching").grid(row=10)

Label(master, text="Enter Value of irritability").grid(row=11)

Label(master, text="Enter Value of delayed healing").grid(row=12)

Label(master, text="Enter Value of partial paresis").grid(row=13)

Label(master, text="Enter Value of muscle stiffness").grid(row=14)

Label(master, text="Enter Value of Alopecia").grid(row=15)

Label(master, text="Enter Value of Obesity").grid(row=16)

e1 = Entry(master)

e2 = Entry(master)

e3 = Entry(master)

e4 = Entry(master)

e5 = Entry(master)

e6 = Entry(master)

e7 = Entry(master)

e8 = Entry(master)

e9 = Entry(master)

e10 = Entry(master)

e11 = Entry(master)

e12 = Entry(master)

e13 = Entry(master)

e14 = Entry(master)

e15 = Entry(master)

e16 = Entry(master)

e1.grid(row=1, column=1)

e2.grid(row=2, column=1)

e3.grid(row=3, column=1)

e4.grid(row=4, column=1)

e5.grid(row=5, column=1)

e6.grid(row=6, column=1)

e7.grid(row=7, column=1)

e8.grid(row=8, column=1)

e9.grid(row=9, column=1)

e10.grid(row=10, column=1)

e11.grid(row=11, column=1)

e12.grid(row=12, column=1)

e13.grid(row=13, column=1)

e14.grid(row=14, column=1)

e15.grid(row=15, column=1)

e16.grid(row=16, column=1)

Button(master, text='Predict', command=show\_entry\_fields).grid()

mainloop()

**CHAPTER 4**

**4.1 MAIN RESULTS AND INFERENCES**

****

Figure 1: Reading Diabetes Dataset

Here we are reading the diabetes dataset and displaying the first 5 rows of the dataset in jupyter notebook.

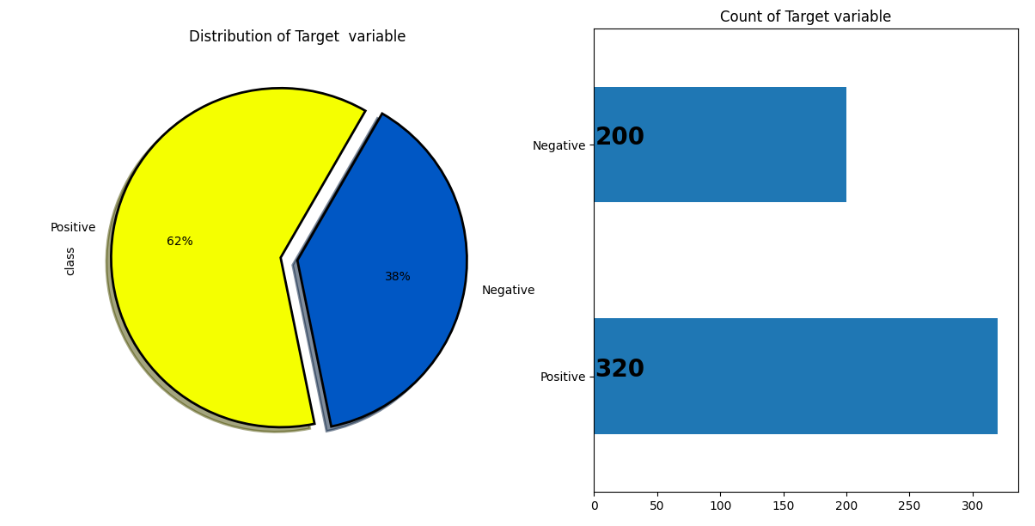
****

Figure 2: Distribution of patients with and without diabetes, in the training and testing dataset

Here we can see there are 62 percent of people with diabetes and 38 percent of people are normal. Number of patients with diabetes in the dataset are 320 and normal patients are 200.

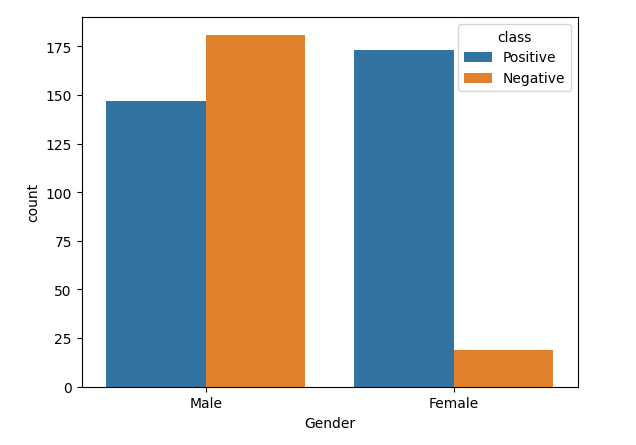
****

Figure 3: Distribution of gender with and without diabetes in the training and testing dataset

Here we are checking the distribution of male and female with and without diabetes.

****

Figure 4 : Creating heatmap to visualize the relationship between the features and target variable.

Overall, we are generating a heatmap that allows us to quickly visualize the distribution of classes across different features in the dataset.

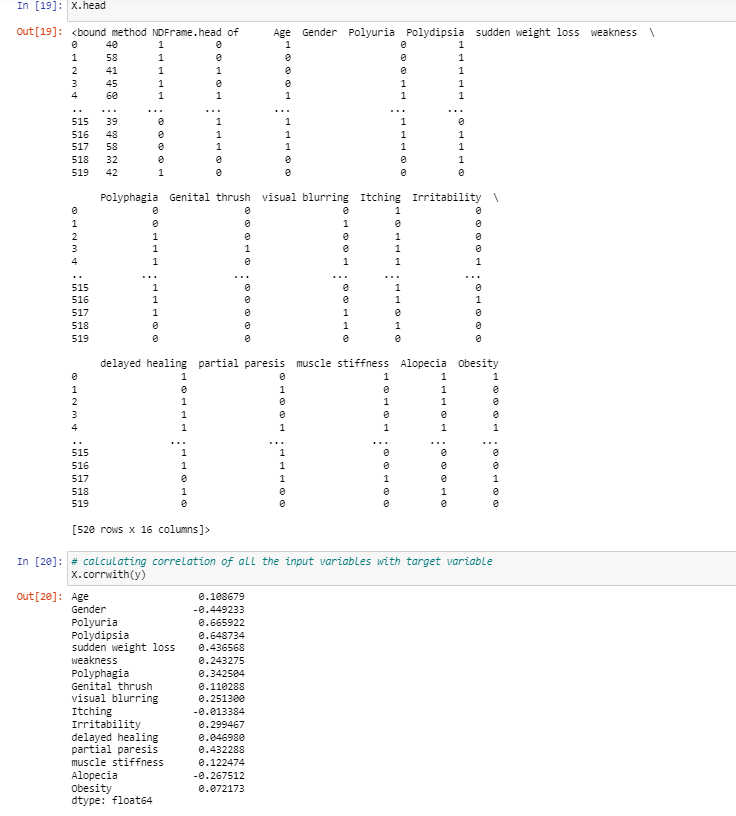
****

Figure 5: Correlation value of each feature with Target Variable

Here we are finding the correlation value of each feature with target variable.

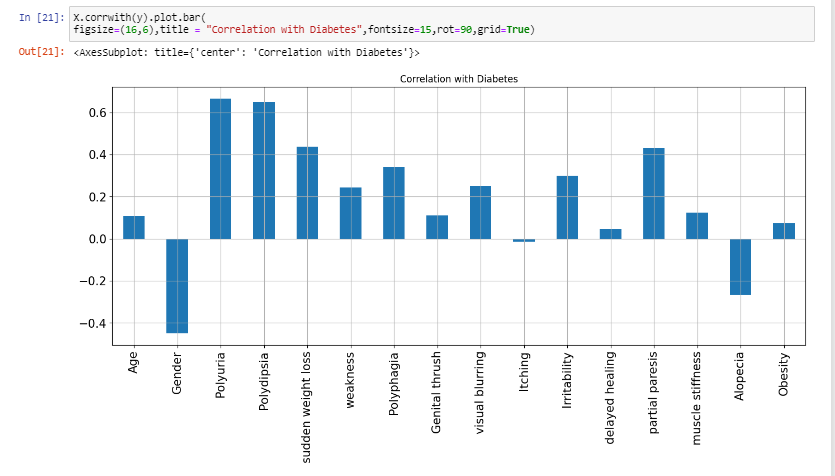
****

Figure 6: Correlation graph of each feature with Target Variable

Here we are visualizing the correlation graph of each feature with target variable.

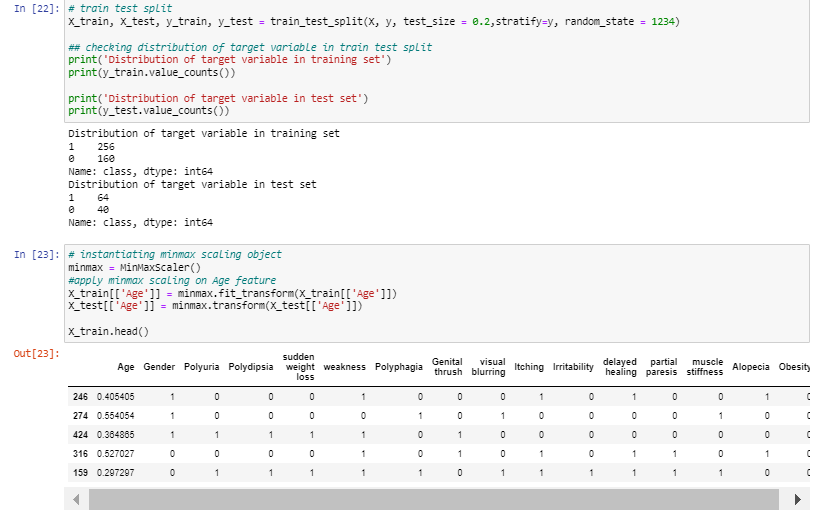
****

Figure 7: Splitting the data and performing minmax scaling.

Here we have done train test split of 80:20 and performed minmax scaling on features that require scaling.

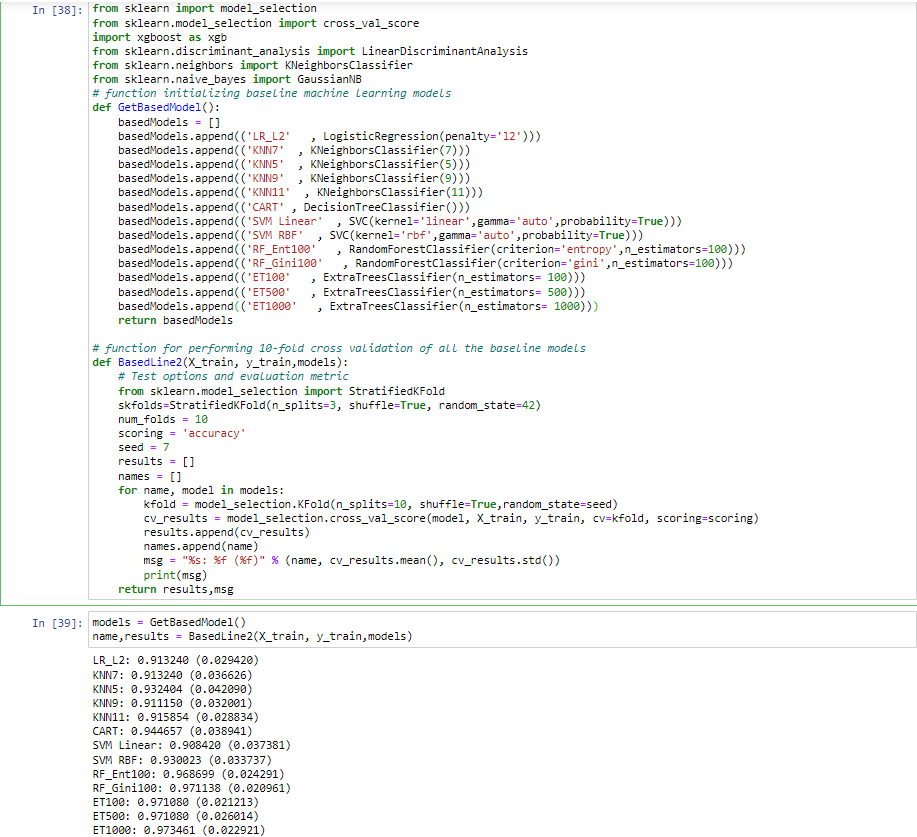
****

Figure 8: Cross Validation Scores of different machine learning algorithms

In this step, we built different baseline models like Logistic Regression, KNN with 5,7,9,11 neighbors , Decision Tree, SVM with different kernels like Linear and RBF, Random Forest , Extra Tree Classifier and performed 10-fold cross validation.

Here we have found that from cross validation score that Extra Tree classifier with 1000 trees has 97.3461 accuracy followed by Random Forest , Extra Tree Classifier with accuracies of 97.1138 and 97.1080 respectively.

****

Figure 9: Calling different Machine Learning Algorithms.

Here we are calling the machine learning algorithms and fitting them and storing the prediction of each algorithm in a new variable.

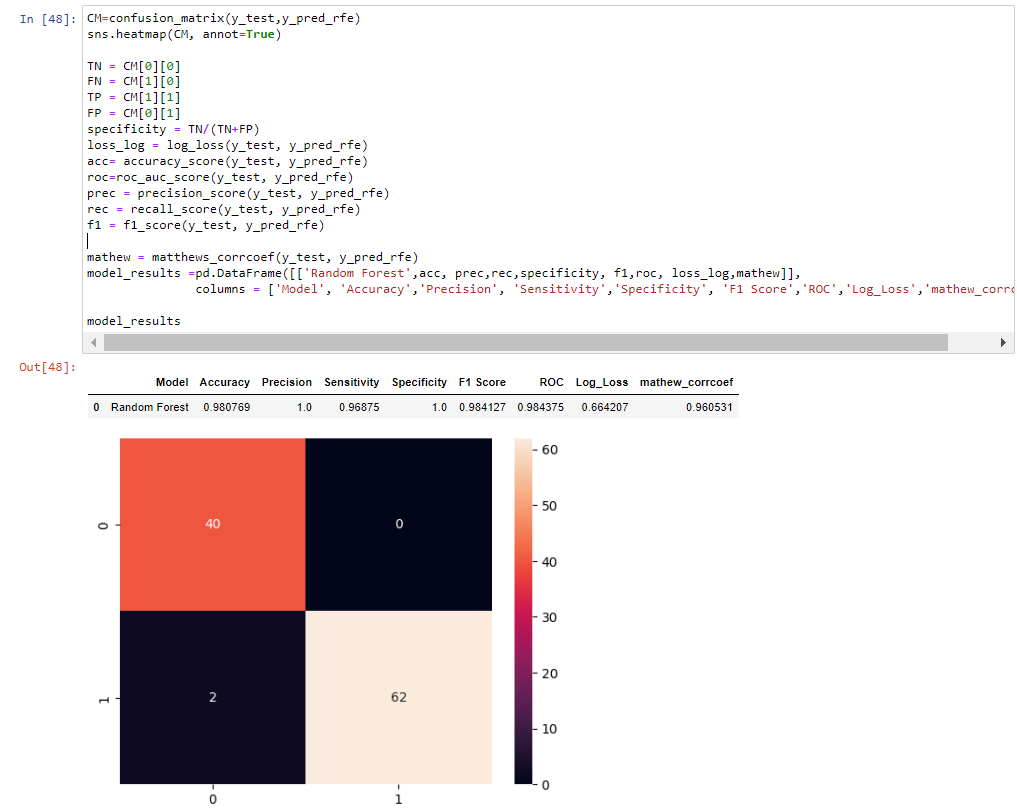
****

Figure 10: Confusion matrix with Heat Map of Random Forest

Here we are plotting the confusion matrix with heat map of random forest algorithm along with that we are also calculating different performance metrics like accuracy, precision, sensitivity, f1 score, ROC, log loss and Mathew correlation coefficient.

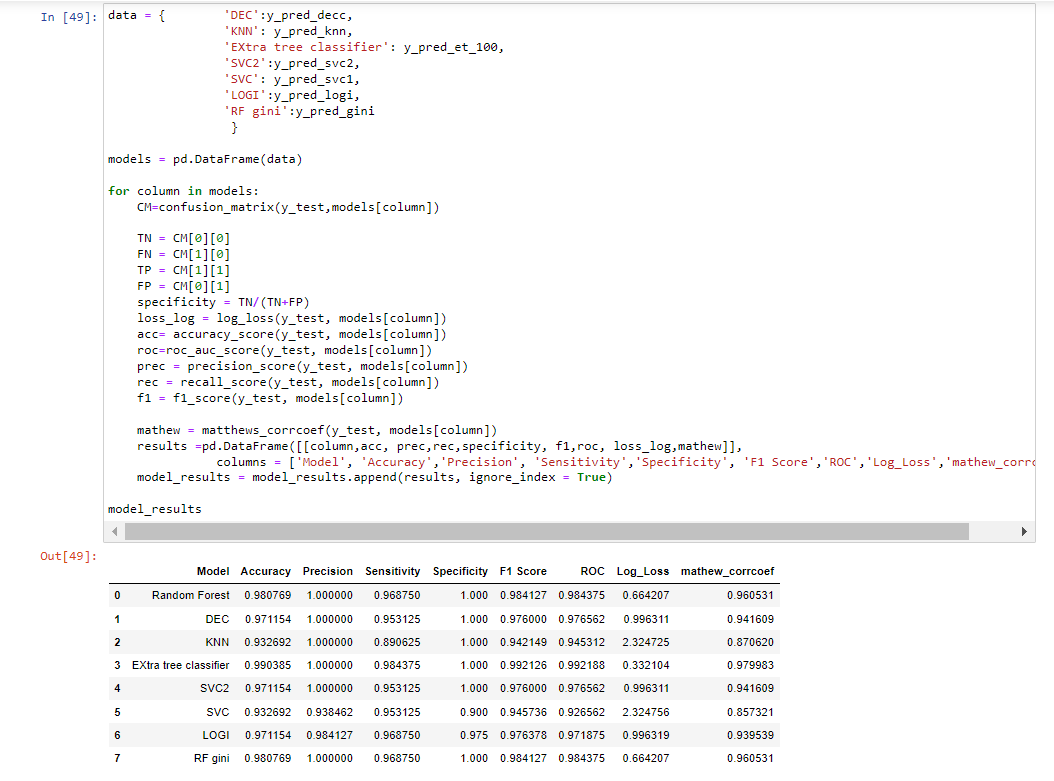
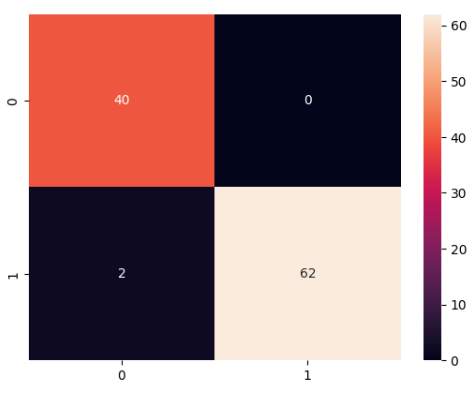
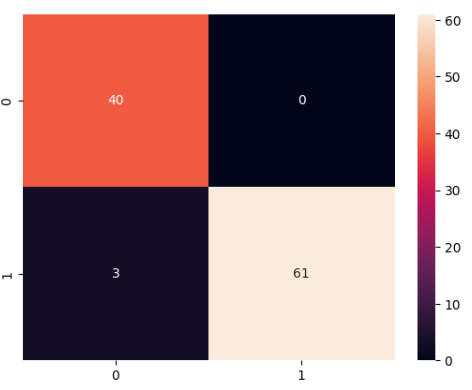
****

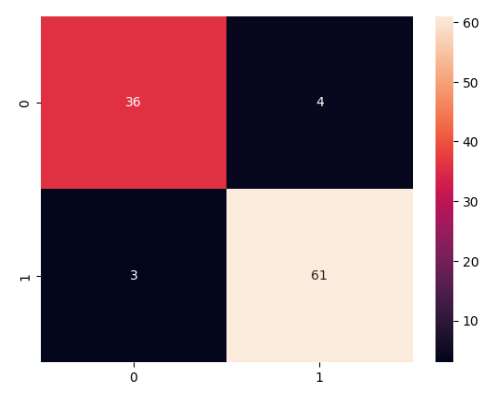
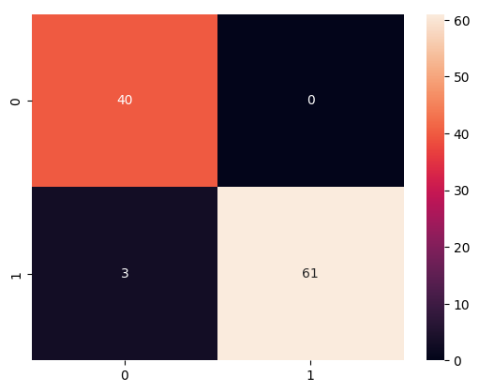
Figure 11: Accuracies of different models with performance metrics values.

Here we calculating different performance metrics like accuracy, precision, sensitivity, f1 score, ROC, log loss and Mathew correlation coefficient of different machine learning algorithms and we can see Extra tree classifier has highest accuracy.

**DECISION TREE**

**RANDOM FOREST**

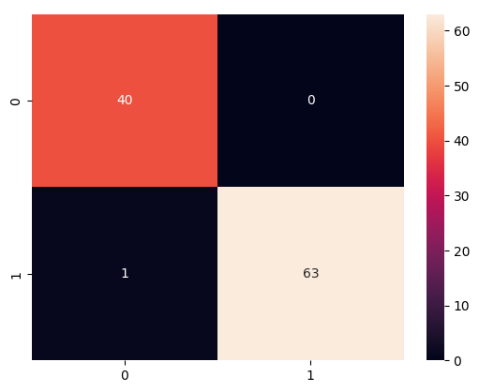
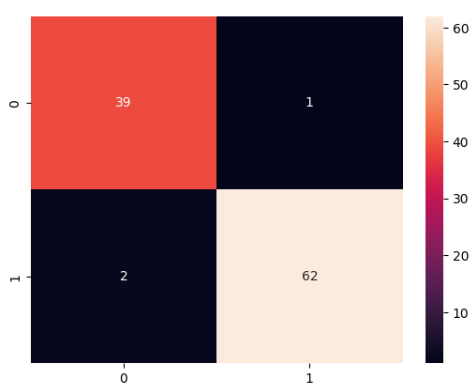
****

****

**LOGISTIC REGRESSION**

**SVM WITH RBF KERNEL**

**SVM WITH LINEAR KERNEL**

****

**EXTRA TREE CLASSIFIER**

Figure 12:Plotting confusion matrix of each machine learning algorithm wrt target variable

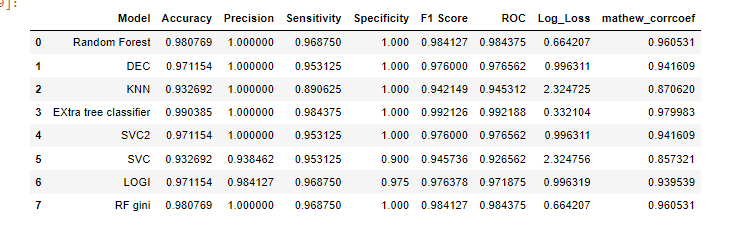
****

Figure 13:Accuracies of different machine learning algorithms

Here are the accuracy scores and performance metrics of different algorithms.



Figure 14:Prediciton of Random data

Here we are predicting if a person has diabetes or not by the data given by user. Array[1] represents that the person has diabetes, we have predicted using random forest algorithm.

**Prediction with Tkinter GUI**

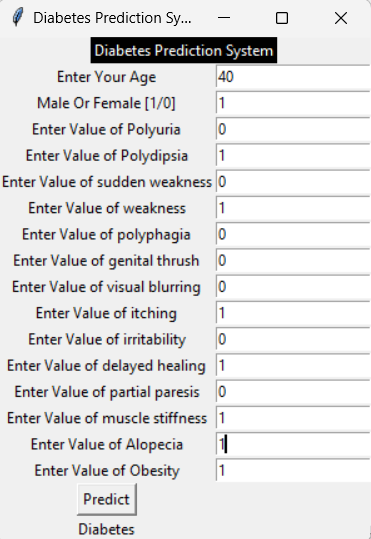
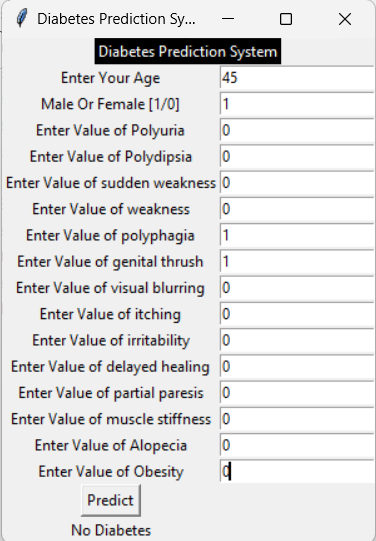
****

Figure 15:Prediciton using tkinter GUI

Here we are predicting if a person has diabetes or not by the data given by user using the GUI. We have predicted using random forest algorithm.

**CHAPTER 5**

**5.1 CONCLUSION**

All the seven machine learning methods accuracies are compared based on which one prediction model is generated. Hence, the aim is to use various evaluation metrics like confusion matrix, accuracy, precision, recall, and f1- score which predicts the disease efficiently. Comparing all seven the extra tree classifier gives the highest accuracy of 99.03% and then Random forest entropy and gini has also accuracy of 98.07% but sensitivity less than extra tree classifier. We attempted to study several machine learning algorithms in order to predict whether or not a specific person will develop diabetes based on various individual characteristics and indicators. Our report's main focus was on examining the accuracy and delving into the causes of the variations in different algorithms. We used one dataset for diabetes illnesses, totaling 520 occurrences, and utilized a percent split to separate the data into training and testing datasets. To test the accuracy, we used distinct algorithms and 16 different attributes. At the conclusion of the implementation phase, we found that the Extra tree classifier provided the highest accuracy level in our dataset, 99.03 percent, and the KNN performed the least accurately, 93.26 percent. Various algorithms may function more effectively for other situations and datasets, but in our case, we have found this result. Additionally, if we increase the amount of training data, we might be able to obtain results that are more accurate, but processing time would be longer, and the system would be slower than it is currently since it would have to deal with more data and be more complex. In this manner, taking these probable factors into account, we made this better decision.

**5.2 RECOMMNDATIONS FOR FUTURE WORK**

With the aid of contemporary technologies like machine learning, fuzzy logics, and others, comparable prediction systems can be created for a number of other deadly or chronic conditions including cancer, diabetes, etc. Massive amounts of data from all users globally can be stored using big data technologies like Hadoop, and technologies like cloud computing can be used to manage the data or reports of the user. Further this work can be extended to find how likely nondiabetic people can have diabetes in next few years

**REFERENCES**

[1] N.M. Saravana kumar, T. Eswari, P. Sampath, S. Lavanya,(2015). Predictive Methodology for Diabetic Data Analysis in Big Data. *Procedia Computer Science*, *50*, 203-208. <https://doi.org/10.1016/j.procs.2015.04.069>

[2] Iyer, Aiswarya & Jeyalatha, s & Sumbaly, Ronak. (2015). Diagnosis of Diabetes Using Classification Mining Techniques. International Journal of Data Mining & Knowledge Management Process. 5. 1-14. 10.5121/ijdkp.2015.5101.

[3] K. Rajesh and V. Sangeetha, “Application of Data Mining Methods and Techniques for Diabetes Diagnosis”, International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 3, September 2012

[4] B. M. Patil, R. C. Joshi and D. Toshniwal, "Association Rule for Classification of Type-2 Diabetic Patients," 2010 Second International Conference on Machine Learning and Computing, Bangalore, India, 2010, pp. 330-334, doi: 10.1109/ICMLC.2010.67.

[5]https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.#