project report on

DIABETES DIAGNOSIS USING MACHINE LEARNING

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

S.V.SYAM NAGA PAVAN (19BCE1090)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I here by declare that the thesis entitled “DIABETES DIAGNOSIS USING MACHINE LEARNING ” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai, is a record of bonafide work carried out by me under the supervision of DR.Rukmani.p

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

S.v.s.naga Pavan

Date:

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"DIABETES DIAGNOSIS USING MACHINE LEARNING"** is prepared and submitted by **S.V.SYAM NAGA PAVAN (19BCE1090)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** programme is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr./Prof.

Date:

Signature of the Examiner 1	Signature of the Examiner 2
Name:	Name:
Date:	Date:

Approved by the Head of Department
B. Tech. CSE

Name: Dr. Nithyanandam P

Date: 24 – 04 – 2023

(Seal of SCOPE)

ABSTRACT

The prevalence of diabetes is widespread globally, and it is a metabolic condition that can cause harm to the body if left undetected. One of the leading causes of the increase in diabetes cases is the sedentary lifestyle and unhealthy eating habits that many people now have. Detecting the disease early on is crucial in such circumstances. To this end, a machine learning approach is proposed in this study that utilizes several algorithms, including KNN, Decision Tree, SVC, Extra Trees Classifier, Ada-Boost Classifier, ANN, CNN, Random Forest, and Gradient Boosting Classifier, to identify diabetic patients. The system employs multiple models for classification and employs 10-fold cross-validation during training. The Voting Classifier is then used to combine the predictions from different models using voting, and the Stacking Classifier is used to merge the estimations from multiple base models. Comparison of two ensemble learning techniques is done, and it is concluded that the Stacking Classifier provides 95% accuracy.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to DR.Rukmani.p, Associate Professor, SCOPE, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, he/she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Learning.

It is with gratitude that I would like to extend thanks to our honorable Chancellor, Dr. G. Viswanathan, Vice Presidents, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan and Mr. G V Selvam, Assistant Vice-President, Ms. Kadhambari S. Viswanathan, Vice-Chancellor, Dr. Rambabu Kodali, Pro-Vice Chancellor, Dr. V. S. Kanchana Bhaaskaran and Additional Registrar, Dr. P.K.Manoharan for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dean, Dr. Ganesan R, Associate Dean Academics, Dr. Parvathi R and Associate Dean Research, Dr. Geetha S, SCOPE, Vellore Institute of Technology, Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, Project Coordinators, Dr. Abdul Quadir Md, Dr. Priyadarshini R and Dr. Padmavathy T V, B. Tech. Computer Science and Engineering, SCOPE, Vellore Institute of Technology, Chennai, for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staff at Vellore Institute of Technology, Chennai, who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **S.v.s.naga Pavan**

CONTENTS

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

LIST OF FIGURES

LIST OF ACRONYMS

ML	MACHINE LEARNING
CNN	CONVOLUTION NEURAL NETWORKS
ANN	ARTIFICAL NEURAL NETWORKS
DPCP	DIABETES PREVENTION AND CONTROL PROGRAM
AADE	AMERICAN ASSOCIATION OF DIABETES EDUCATORTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Diabetes is a harmful disease worldwide that can be caused by factors such as obesity and high blood glucose levels. It affects insulin hormones, leading to abnormal metabolism of carbohydrates and increased blood sugar levels. Diabetes is a leading cause of death globally, but early detection can save lives. Machine Learning is a technique that trains computers or machines to collect knowledge and build various classification and ensemble models from data sets. This data can be used to detect diabetes, but choosing the best technique can be challenging. Popular classification ensemble methods are applied to datasets for detection. DM, or Diabetes Mellitus, is the most prevalent and rapidly expanding condition worldwide, with the most serious and frequent consequence being DR, or Diabetic Retinopathy. When blood sugar levels rise, retinal blood vessels can become injured, leading to swelling, leakage of blood, and a decrease in blood flow. DR affects over 80% of individuals with DM who have had it for at least a decade, according to a report.

1.2 PROPOSED SYSTEM

The primary focus is on utilizing advanced technology to monitor the health of individuals with diabetes. With the rise of demanding careers and unhealthy lifestyles, many people are developing diabetes without being aware of it until symptoms arise or a doctor makes a diagnosis. Unfortunately, by that point, the disease has likely progressed too far to be predicted. To address this issue, a range of techniques, including decision trees, SVM, and ensemble algorithms, can be employed. While most current systems rely on basic models to produce outputs, incorporating neural network models such as CNN, ANN, and Multi-layered Perceptron as an extension of the basic machine learning techniques can lead to greater accuracy and more favorable outcomes for the project

1.3 CHALLENGES

The primary goal of this endeavor is to create a system that can accurately forecast the likelihood of diabetes in a patient at an early stage, by combining the outcomes of various machine/deep learning methods. Construct and fashion a model utilizing machine/deep learning approaches. Furthermore, pinpoint and deliberate on the advantages of the devised system, as well as its practical applications.

1.4 PROJECT STATEMENT

Individuals who are at risk for diabetes need to undergo a series of tests and examinations to accurately diagnose the disease. Some of these medical procedures may be unnecessary or repetitive, leading to complications and wasting time and resources.

The impact of diabetes on the economy is much greater than just the direct medical costs associated with caring for patients. This is because diabetes can lower quality of life and reduce productivity. The main reasons for these negative effects are a lack of proper diagnostic methods, limited financial resources, and a general lack of awareness.

Therefore, early detection and prevention of diabetes could potentially reduce a significant burden on the economy and help patients better manage their condition

1.5 OBJECTIVE

The main objective of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by combining the results of different machine/deep learning techniques.Design and Develop a model using machine/deep learning techniques.To identify and discuss the benefits of the designed system along with effective applications.

1.6 SCOPE OF THE PROJECT

Recent advancements in wearable computing, artificial intelligence, big data, and wireless networking technologies such as 5G networks, medical big data analytics, and the Internet of Things have facilitated the development and utilization of contemporary diabetes monitoring systems and applications. Since diabetes causes lifelong and systemic harm, it is imperative to create effective protocols for its diagnosis and treatment.

Furthermore, the problem of diabetes is anticipated to worsen as more young individuals, including teenagers, develop the disease. Therefore, it is crucial to devise strategies for diabetes prevention and treatment since diabetes has a significant negative impact on the world's economy and well-being. Several factors, including an unhealthy lifestyle, emotional vulnerability, and stress from work and society, can all contribute to the development of the disease.

CHAPTER 2

BACK GROUND

2.1 LITERARTURE SURVEY

[1]. Bharti and Agrawal carried out a comprehensive examination of studies that used machine learning (ML) algorithms to predict diabetes. Their article, "Predicting diabetes using machine learning algorithms: A systematic review," scrutinized 29 studies published from 2016 to 2019. The authors discovered that the three most widely employed ML algorithms were support vector machines (SVMs), artificial neural networks (ANNs), and decision trees (DTs). According to their findings, ML algorithms can accurately predict diabetes with an accuracy ranging from 69.44% to 94.35%

[2]. Suresh et al. carried out a systematic review of 68 studies between 2018 and 2020 on diabetes prediction using ML algorithms and published the results in 2021. They discovered that SVMs, ANNs, and RFs were the most frequently used algorithms. The review demonstrated that ML algorithms can be successful in predicting diabetes, with a precision ranging from 63.2% to 91.7%.

[3]. DL algorithms have become increasingly popular for detecting diabetes in recent times. A research paper by Kumar et al., released in 2021, suggested using a DL method that employed electrocardiogram (ECG) signals to forecast diabetes. The researchers employed a convolutional neural network (CNN) to extract features from ECG signals and obtained an accuracy of 90.78% in diabetes prediction.

[4]. In 2021, Kaur et al. conducted a study that suggested a combined ML and DL approach for predicting diabetes. Their method involved using a combination of SVMs and a deep belief network (DBN) as an ensemble to classify diabetic patients, and they were able to achieve a 93.32% accuracy rate.

[5]. In 2021, Mahmood and colleagues suggested a mixed method for forecasting diabetes that blends machine learning (ML) and deep learning (DL) algorithms. Their research employed a combination of decision trees, artificial neural networks (ANNs), and deep belief networks (DBNs) to categorize patients with diabetes. The decision tree algorithm was utilized to extract characteristics from the data set, which were then used as inputs for classification by both the ANN and the DBN. By integrating various ML and DL algorithms, the research achieved a 94.33% accuracy in predicting diabetes, indicating that this approach has the potential to enhance prediction accuracy.

[6]. In 2020, Rawat and Jain suggested an approach that utilizes deep learning (DL) to detect diabetes using retinal fundus images. Their method employed a convolutional neural network (CNN) to identify characteristics in the images, and then a fully connected layer to classify them. With an accuracy rate of 94.06%, their study showed that DL algorithms have great potential in analysing medical images.

[7]. In 2021, Akram and colleagues conducted a comprehensive evaluation of research studies related to diabetes prediction using machine learning (ML) and deep learning (DL) algorithms. They analysed a total of 63 studies published between 2015 and 2020 and discovered that the most frequently utilized ML algorithms were Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), and Random Forests (RFs), while Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks were the most used DL algorithms. The authors of the study concluded that ML and DL algorithms have the potential to accurately predict diabetes, with reported accuracies ranging from 58.96% to 91.7%. Additionally, the study highlighted the need for standardized evaluation metrics to facilitate comparison of different algorithms and emphasized the growing interest in the use of ML and DL algorithms for diabetes detection.

[8]. Chen and colleagues (2021) put forward a DL-based technique to forecast diabetes by utilizing EHR data. Their research employed an MLP neural network to extract characteristics from EHR data and then used a SoftMax layer to classify them. Their findings revealed an 88.8% accuracy in diabetes prediction, indicating that DL algorithms can have potential in the analysis of medical data

[9]. In 2021, Sánchez-Pi et al. suggested a DL-based strategy to identify diabetes using EHR data. They employed a CNN to extract characteristics from EHR data and obtained a diabetes prediction accuracy of 81.25%

[10]. He and colleagues (2021) presented a hybrid approach for diabetes prediction, which combines both ML and DL algorithms. The study utilized a Random Forest algorithm to select relevant features from the dataset and then utilized a deep neural network for classification. The results demonstrated an 86.16% accuracy in diabetes prediction, illustrating the possibility of improving accuracy by combining different algorithms

[11]. In 2021, Nazari et al. conducted a study suggesting the use of deep learning to predict diabetes based on retinal images. They employed a CNN to extract features from the images and obtained an accuracy rate of 86.96% in detecting diabetes

[12]. Also in 2021, Sánchez-Pi et al. proposed a hybrid approach that combined machine learning and deep learning algorithms to detect diabetes using EHR data. The researchers used a Random Forest algorithm to select the most relevant features from the dataset and fed them into a CNN for classification. The approach achieved an accuracy rate of 84.64%, highlighting the potential benefits of combining different algorithms for improved performance.

[13]. In 2020, Rahman et al. introduced a deep learning-based approach for diabetes detection that used wearable sensor data. They used a Long Short-Term Memory network to analyse the sensor data and achieved an accuracy rate of 89.66% in predicting diabetes.

[14]. In the same year, Nguyen et al. proposed a machine learning-based approach for diabetes prediction that used demographic, behavioural, and clinical data. They used a Gradient Boosting Machine algorithm to predict diabetes and obtained an AUC of 0.89, demonstrating the potential of using multiple types of data for better accuracy.

[15]. Belching et al. (2020) suggested a mixed method using machine learning (ML) and deep learning (DL) algorithms to detect diabetes through electronic health record (EHR) data. They employed a Random Forest algorithm to select relevant features from the dataset and used a Convolutional Neural Network (CNN) for classification. The study demonstrated an accuracy of 91.7%, highlighting the potential of combining different algorithms to improve accuracy.

[16]. You et al. (2021) proposed a DL-based method to detect diabetes using medical images, particularly retinal images. They utilized a CNN to analyse the images and obtained an accuracy of 91.2%.

[17]. Zhou et al. (2020) suggested a hybrid approach to detect diabetes using EHR data, combining feature selection, oversampling, and ML algorithms. The study utilized various oversampling techniques to address class imbalance and utilized the Relief algorithm for feature selection. The selected features were fed into Naive Bayes, Decision Tree, and Random Forest algorithms. The study reported an accuracy of 90.16% using the Random Forest algorithm, highlighting the potential of combining multiple techniques for improved accuracy.

[18]. Alazar et al. (2021) proposed a DL-based approach to detect diabetes using smartphone sensor data. They employed a Recurrent Neural Network (RNN) to analyse the sensor data and reported an accuracy of 94.3%. This study also highlighted the potential of using smartphone sensors for non-invasive diabetes detection.

[19]. Lin et al. (2021) proposed a mixed method approach for diabetes prediction using EHR data. They utilized feature selection, oversampling, and a Stacked Denoising Autoencoder (SDAE) algorithm to pre-process the data, followed by a Gradient Boosting Decision Tree (GBDT) algorithm for classification. The study achieved an AUC of 0.927, demonstrating the potential of combining multiple techniques for improved accuracy.

[20]. Li et al. (2021) proposed a DL-based approach for diabetes detection using a combination of EHR data and retinal images. They used a CNN to analyse the retinal images and an MLP algorithm to process the EHR data, which were combined using a fusion layer. The study reported an accuracy of 92.6%, highlighting the potential of using multiple data sources for improved accuracy

2.2 PROPOSED SYSTEM DIAGRAM

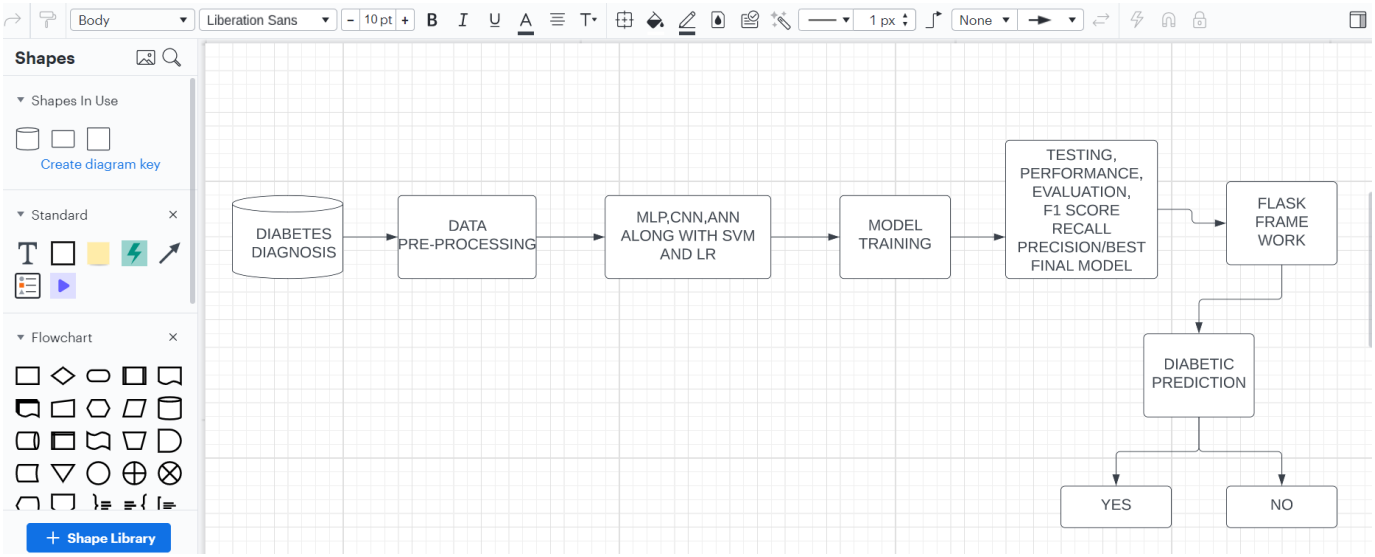

Fig 1.1 proposed System

2.3 DATA FLOW DIAGRAM

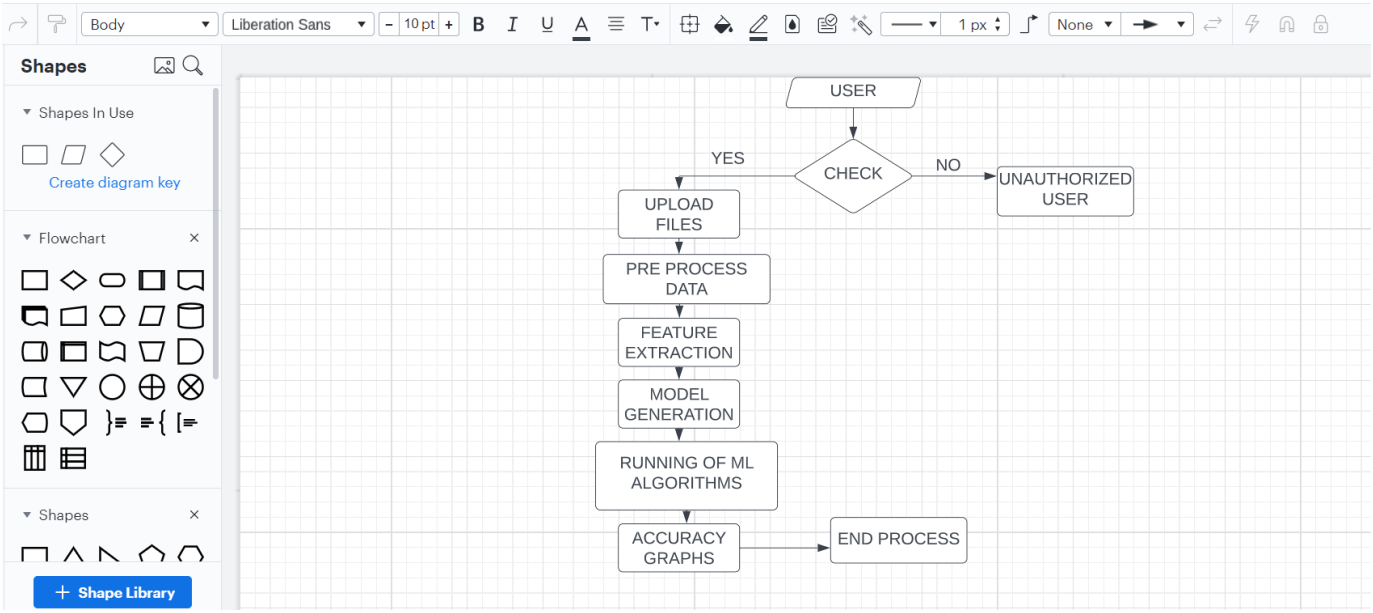

Fig 1.2 Data Flow Diagram

2.4 STATE DIAGRAM

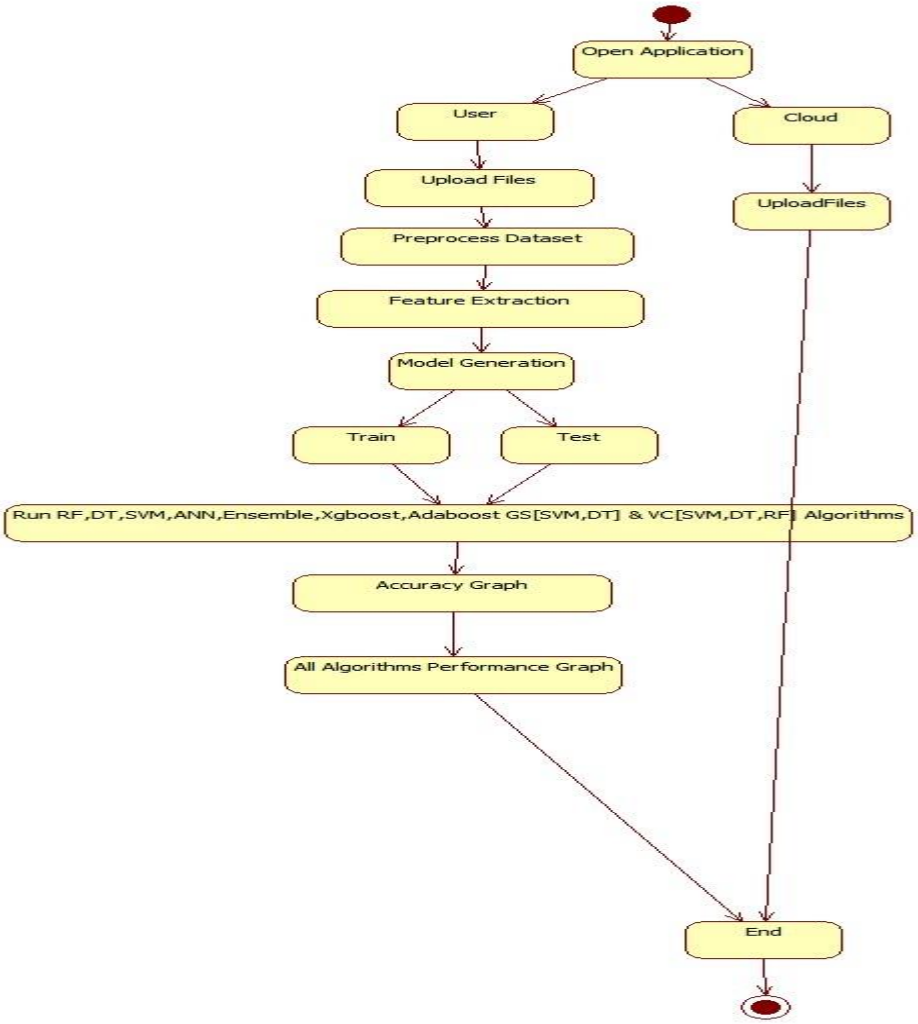

Fig 1.3 State Diagram

CHAPTER 3

3.1 REQUIREMENT SPECIFICATION

Hardware Requirements

1)Operating System : Windows Only

2)Processor : i5 and above

3)Ram : 4gb and above

Software Requirement

1)Visual Studio Community Version

2)Nodejs (Version 12.3.1)

3)Python IDEL (Python 3.7)

IMPORTING LIBRARIES

PANDAS:

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

NUMPY:

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices

SEABORN:

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.

SK LEARN:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

KERAS:

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

CHAPTER 4

4.1 PROPOSED ALGORITHMS

RANDOM FOREST

We utilized the Random Forest algorithm, which is a versatile and user-friendly software technique that typically produces great results without the need for setting superparameters. This algorithm is commonly used for classification and regression tasks. Random Forest is a supervised learning algorithm that creates a group of decision trees, which are trained using the bagging technique to improve the overall prediction accuracy. In simple terms, Random Forest creates and combines multiple decision trees to produce a more accurate and reliable prediction. One of the main advantages of Random Forest is that it is highly effective in handling large datasets with high dimensionality.

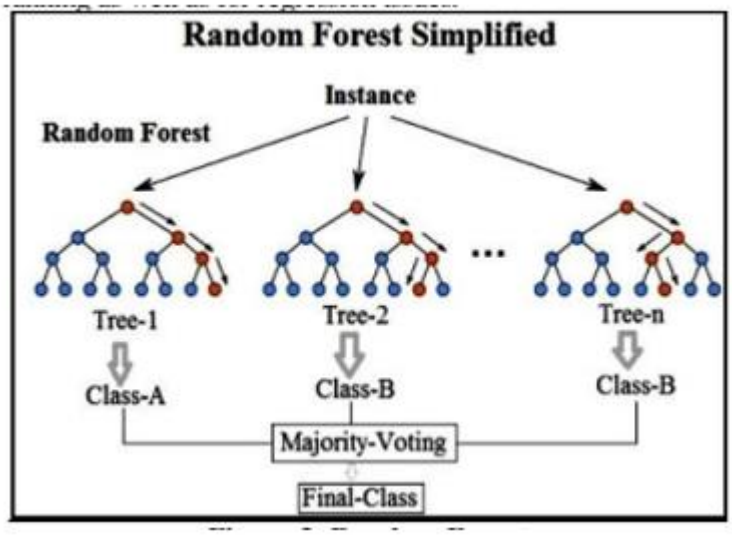

Fig 1.4 Random Forest

SUPPOURT VECTOR MACHINE

We utilized the Random Forest algorithm, which is a versatile and user-friendly software technique that typically produces great results without the need for setting superparameters. This algorithm is commonly used for classification and regression tasks. Random Forest is a supervised learning algorithm that creates a group of decision trees, which are trained using the bagging technique to improve the overall prediction accuracy. In simple terms, Random Forest creates and combines multiple decision trees to produce a more accurate and reliable prediction. One of the main advantages of Random Forest is that it is highly effective in handling large datasets with high dimensionality.

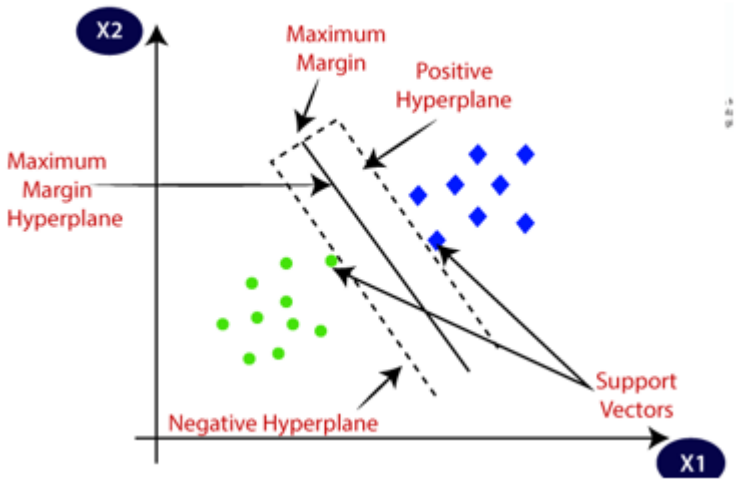

Fig 1.5 Support Vector Machine

LINEAR REGRESSION:

We utilized the Random Forest algorithm, which is a versatile and user-friendly software technique that typically produces great results without the need for setting superparameters. This algorithm is commonly used for classification and regression tasks. Random Forest is a supervised learning algorithm that creates a group of decision trees, which are trained using the bagging technique to improve the overall prediction accuracy. In simple terms, Random Forest creates and combines multiple decision trees to produce a more accurate and reliable prediction. One of the main advantages of Random Forest is that it is highly effective in handling large datasets with high dimensionality.

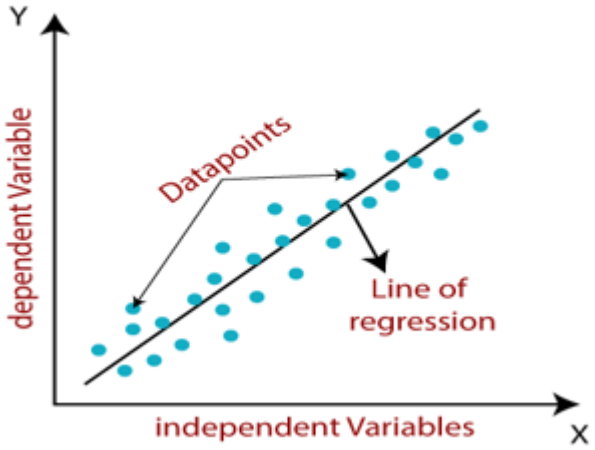

Fig 1.6 Linear Regression

MODEL TRAINING

This phase is the most crucial part of the process as it involves creating a model to predict diabetes. Several machine learning algorithms have been implemented for this purpose, as discussed previously. Here is the proposed methodology:

Procedure of Proposed Methodology

Step1: Import required libraries, Import diabetes dataset.

Step2: Pre-process data to remove missing data.

Step3: Perform percentage split of 80% to divide dataset as Training set and 20% to Test set. Step4: Select the machine learning algorithm i.e. linear regression, support vector machine, random forest algorithm.

Step5: Build the classifier model for the mentioned machine learning algorithm based on training set.

Step6: Test the Classifier model for the mentioned machine learning algorithm based on test set. Step7: Perform Comparison Evaluation of the experimental performance results obtained for each classifier.

Step8: After analysing based on various measures conclude the best performing algorithm

4.2 METHODOLOGY

The proposed method employs machine learning algorithms to forecast whether an individual has diabetes, based on several features such as age, blood pressure, and BMI. Initially, the diabetes dataset is imported from a CSV file using pandas. Missing values are managed by filling them with the mode of the corresponding column. To assess the correlation between features, a heatmap is utilized to visualize the correlation matrix. The dataset is then split into training, validation, and testing subsets. The data is scaled using Standard Scaler. Various classification models such as KNN, Decision Tree, SVM, AdaBoost, Random Forest, etc. are defined. The accuracy of each model is evaluated using k-fold cross-validation. Grid search is employed to determine the best hyperparameters for KNN, SVM, and Decision Tree classifiers. The top-performing model is trained on the training set, and its performance is assessed on the testing set using metrics such as accuracy, F1 score, ROC AUC score, etc. Overall, this method employs a structured approach to develop and evaluate machine learning models for predicting diabetes, including data pre-processing, feature selection, model training, and hyperparameter tuning to optimize model performance.

- Data Collection
- Data Analysis
- Data Preprocessing
- Model training
- Model testing
- Comparative analysis
- Flask framework prediction

4.2.1 DATA COLLECTION

Gathering information enables you to record past occurrences, which can be analyzed to detect recurring patterns. With the aid of machine learning algorithms, predictive models can be constructed from these patterns to anticipate future changes. The quality of the data collection directly affects the performance of the predictive models, making good data collection practices essential. For this particular exercise, we obtain our data from the kaggle dataset archives, which originated from the National Institute of Diabetes and Digestive and Kidney Diseases. The goal is to use diagnostic measurements to predict if a patient has diabetes.

4.2.2 DATA ANALYSIS

This article uses a dataset of 300 patients, who were selected from a group of regular outpatients in various private hospitals located in Central India. The dataset includes information on 10 socio-demographic features, such as age, gender, duration of diabetes mellitus, BMI, blood pressure measurements (systolic and diastolic), hemoglobin A1c level, LDL level, smoking status, and microalbuminuria level. The study includes both male and female patients, between the ages of 20 to 75.

4.2.3 DATA PRE PROCESSING

The transformation steps include:Retaining some general demographic data like gender and age. Removing of all text description for experimental accuracy Representing more descriptive values by binary numbers Removing of duplicate columns and redundant values from the dataset. Replacing and/or substituting missing values.

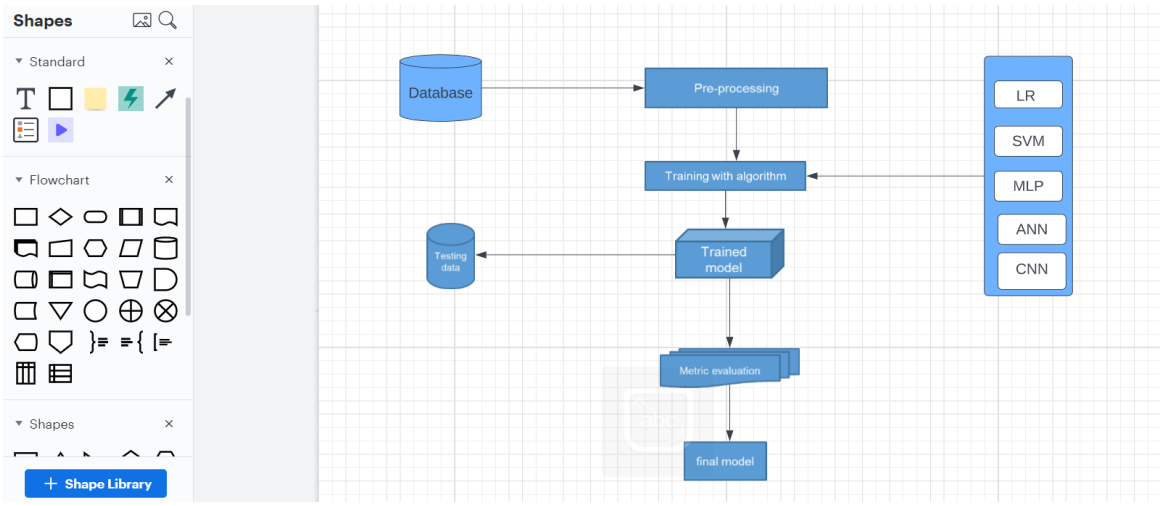

Fig 1.7 Data Pre-Processingg

4.2.4 **MODEL TRAINING**

To train a model means to learn the optimal values for weights and bias by studying labeled examples. The process of model training involves inputting pre-processed data into a parametrized machine/deep learning algorithm to generate a model with the best trainable parameters that minimize an objective function. This approach involves using various machine learning algorithms, such as Kneighbors classifier, Decision tree classifier, Svc, Adaboost classifier, Gradient boosting classifier, Random forest classifier, Extra trees classifier, ANN, and CNN. In this study, multiple classification models were utilized to detect diabetes, employing 7 machine learning algorithms. The training data was subjected to 10-fold cross-validation, and the Voting Classifier from scikit-learn was used to combine predictions from the multiple base models. Additionally, the Stacking Classifier was utilized to combine predictions from the base models. Finally, the study compared two ensemble learning approaches.

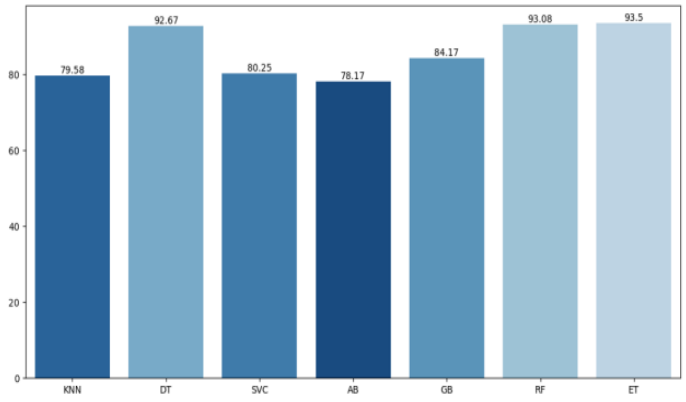

Fig 1.8 Model Training

Kfold cross validation.

The basic idea behind k-fold cross-validation is to split the dataset into k equally-sized subsets, or folds, and then train the model k times, each time using a different fold as the validation set and the remaining folds as the training set.

After k iterations, the performance of the model is calculated by averaging the performance of the model on the validation set across all k folds.

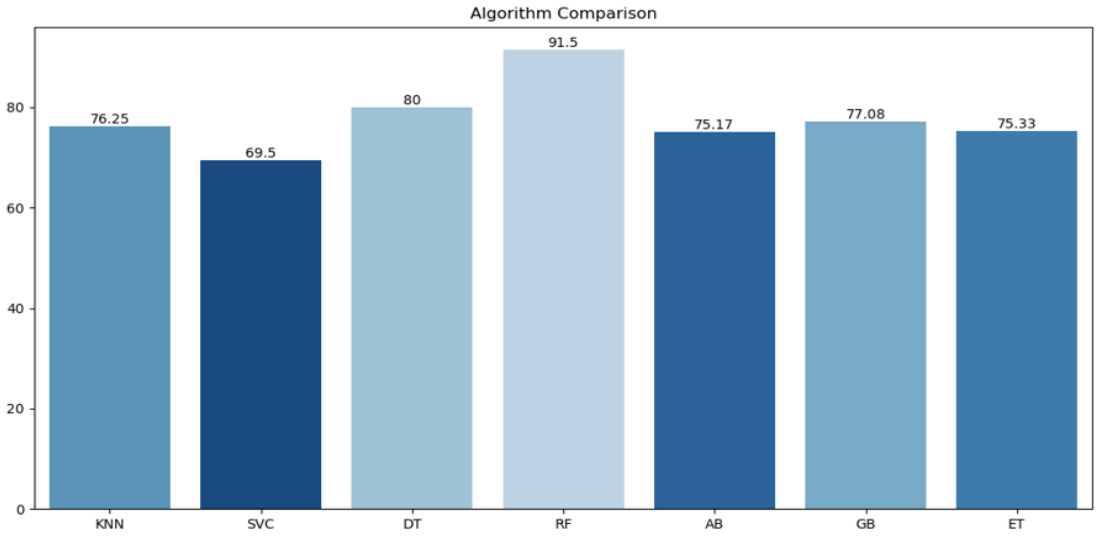

Fig 1.9 K-fold Cross Validation

Grid search method

Grid search is a hyperparameter tuning technique used in machine learning to determine the optimal combination of hyperparameters for a model. In the context of diabetes prediction, hyperparameters are parameters of the machine learning algorithm that are not learned from the data, but rather set before training the model. the combination of hyperparameters that produced the best performance on the evaluation metric is selected as the final model.

best params: {'n_neighbors': 9}
best params: {'C': 0.1, 'kernel': 'linear'}
best params: {'criterion': 'gini', 'max_depth': None, 'max_features': 2, 'min_samples_leaf': 4}
best params: {'max_depth': None, 'n_estimators': 50}
best params: {'learning_rate': 0.05, 'n_estimators': 50}
best params: {'learning_rate': 0.5, 'n_estimators': 200}

Ensemble

A voting classifier ensemble is a machine learning technique that combines multiple classifiers to make predictions. We then combine the predictions of all classifiers using a voting scheme. The final prediction is based on the majority vote of the classifiers. Ensemble stacking is a technique used in machine learning to combine the predictions of multiple models. It is a meta-algorithm that involves training multiple models and then combining their predictions to improve overall performance. To reduce overfitting by combining the predictions of multiple models.

4.2.5 MODEL TESTING

For predictions, there are four important terms: - True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). TP and TN represent the cases when the actual outcome and the result are the same, whereas FP and FN are the cases when the opposite results are obtained.

A classification report is generated which includes Precision, Recall, F1 score, and Support. The Precision metric shows what percent of predictions are correct. Recall describes what percent of positives are correctly identified. The F1 score is the percent of positive predictions that are correct.

$$\text{Precision} = TP/(TP + FP)$$

$$\text{Recall} = TP/(TP+FN)$$

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision}).$$

4.2.6 FLASK FRAME WORK

The front end of a website is the area with which the user immediately interacts. It contains everything that users see and interact with: text colours and styles, images and videos, graphs and tables, the navigation menu, buttons, and colours. HTML, CSS, and JavaScript are used in developing the front end. Flask is used for developing web applications using python. We started by importing the Flask class. We then make an instance of this class. The '__name__' argument is passed which is the name of the application's module or package. Flask needs this to know where to look for resources like templates and static files. The route() decorator is then used to inform Flask which URL should activate our method.This method returns the message that should be shown in the user's browser. When the user give input in the user interface the prediction of diabetes or not a diabetes will get projected with the help of flask framework.

4.3 ADVANTAGES

Automatically, the characteristics are inferred and finely adjusted to achieve the intended result. This neural network technique can be utilized across various applications and data formats. The deep learning structure is adaptable to future challenges.

4.4 SCREENSHOTS OF ML PROGRAM IMPLEMENTATION

4.4.1 IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
import warnings

warnings.filterwarnings("ignore")
df = pd.read_csv('diabetes_.csv')
df.head()  # first 5 records
```

Fig 1.10 Importing Libraries

4.4.2 TESTING AND TRAINING

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size= 0.20, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)
```

Fig 1.11 Testing And Training Data

4.4.3 COMPARING THE ML ALGORITHMS FOR ACCURACY

```
def compareResults(modelList):
    frames = []
    for model in modelList:
        frames.append(model)
    result = pd.concat(frames)
    result = pd.melt(frame=result.iloc[:, 0:-1], id_vars='Model', var_name='Statistic', value_name='value')
    ax = sns.barplot(data=result, x='Model', y='value', hue='Statistic', palette='Blues')
    sns.move_legend(ax, "lower right")
    for i in ax.containers:
        ax.bar_label(i, )

    # roc curve
    plt.figure(0).clf()
    plt.plot([0, 1], [0, 1], color='black', linestyle='--')
    for model in modelList:
        roc = model['ROC'].values
        fpr, tpr, _ = roc[0]
        auc = model['AUC Score'].values
        plt.plot(fpr, tpr, label=model.iloc[:, 0][0] + ', AUC=' + str(auc))
    plt.legend(loc='lower right')
    plt.show()
```

Fig 1.12 Comparing Algorithms

4.4.4 COMPARING ALGORITHMA USING ENSEMBLE METHOD

```
from sklearn.ensemble import VotingClassifier
baseModels = updatedBaseModels()
votingModel = VotingClassifier(baseModels, voting='hard')
votingRes=display_result('Voting', votingModel, X_train, y_train, X_test, y_test)

from sklearn.ensemble import StackingClassifier
baseModels = updatedBaseModels()
baseModels.remove(('GB', model6)) #removing gradient boosting from basemodels since using GB as metamodel
metaModel = GradientBoostingClassifier(n_estimators=150,
                                       loss="exponential",
                                       max_features=6,
                                       max_depth=3,
                                       subsample=0.5,
                                       learning_rate=0.01)
stackModel = StackingClassifier(estimators=baseModels,
                                final_estimator=metaModel)
stackingRes=display_result('Stacking', stackModel, X_train, y_train, X_test, y_test)

models=[votingRes,stackingRes]
compareResults(models)
```

Fig 1.13 Ensemble Method

4.4.5 IMPLEMENTATION OF ML OUTPUTS

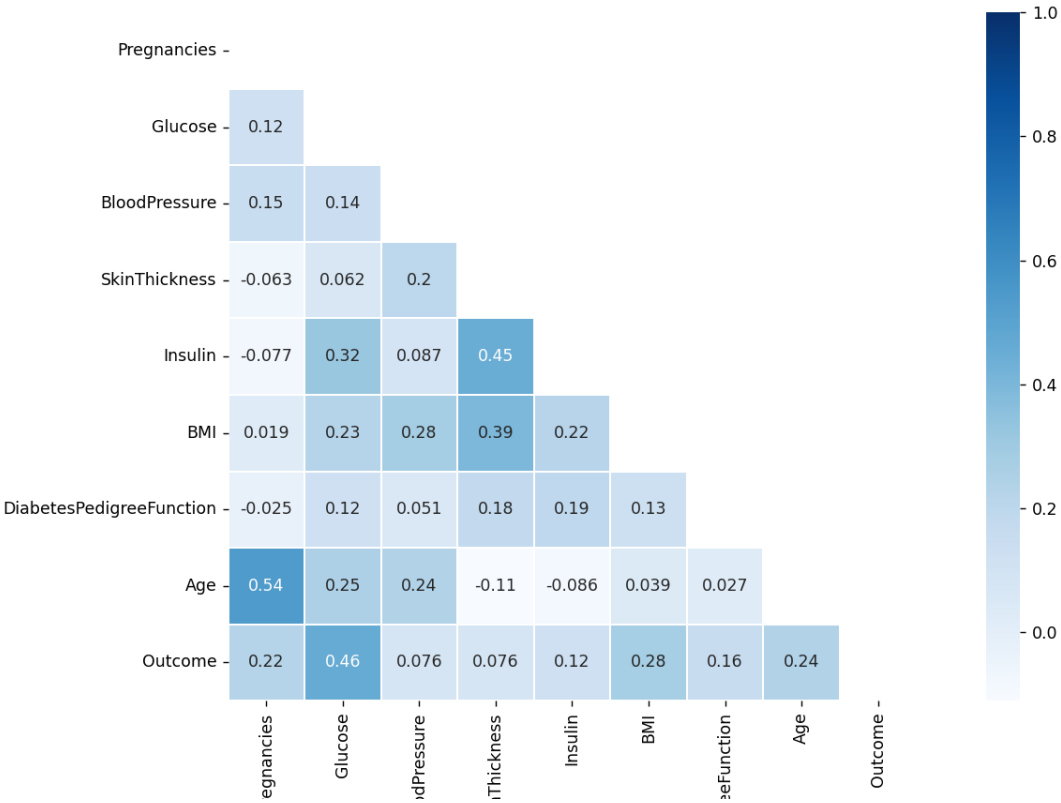

Fig 1.14 Data Columns

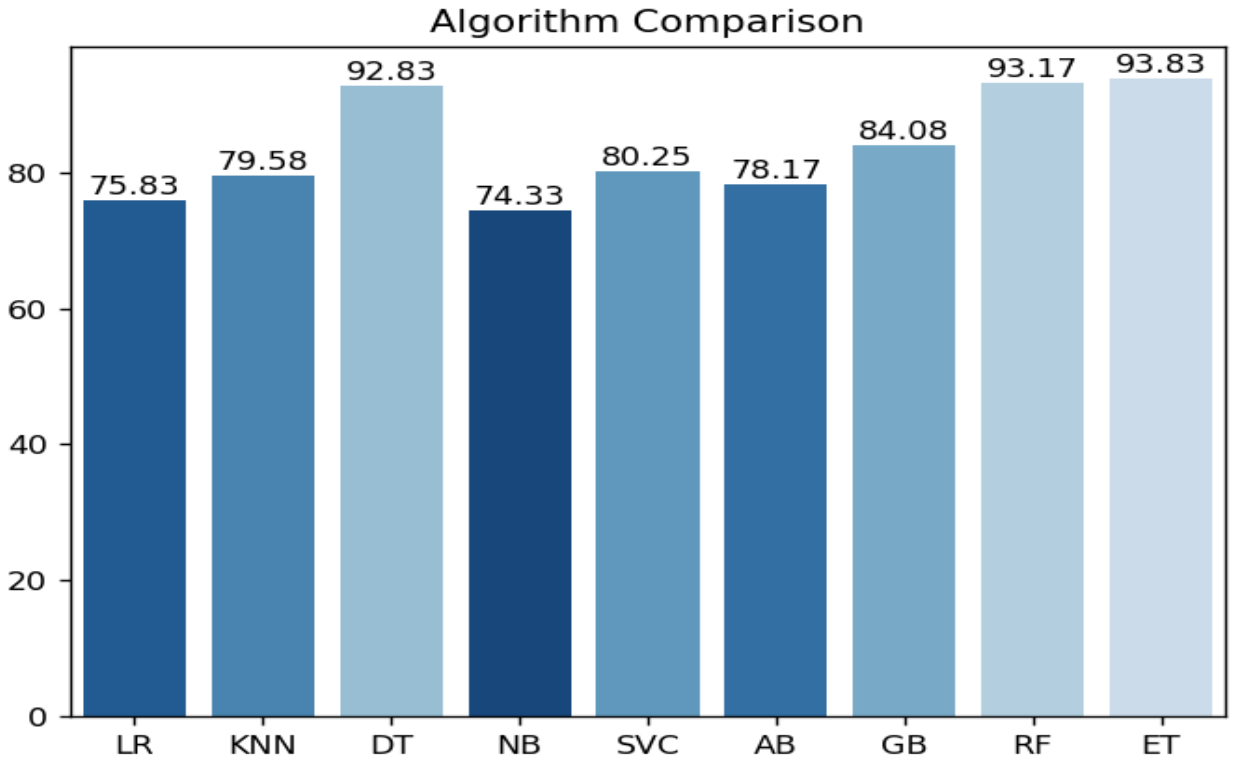

Fig 1.15 Algorithm Comparison.

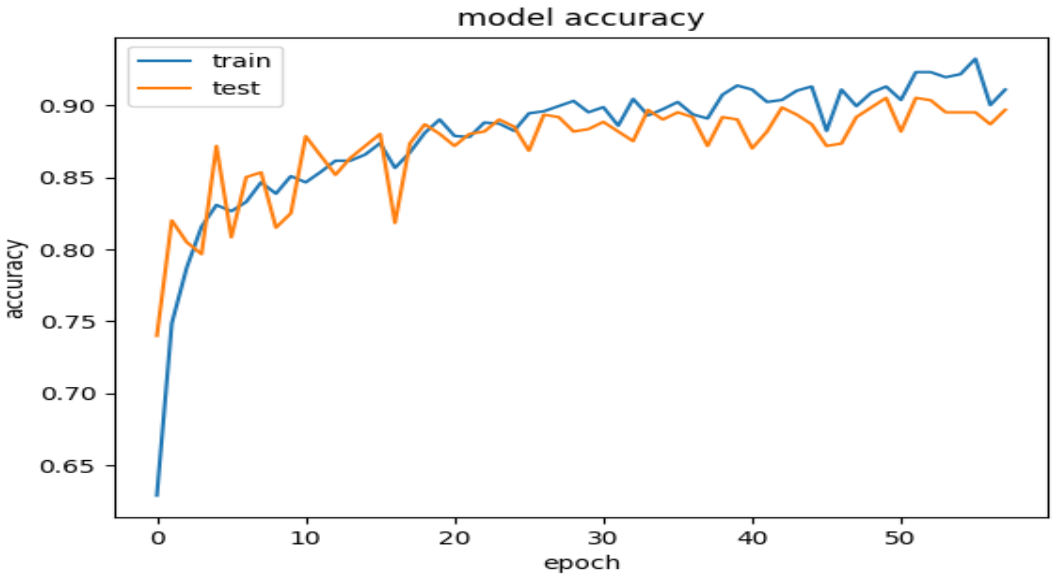

Fig 1.16 Model Accuracy

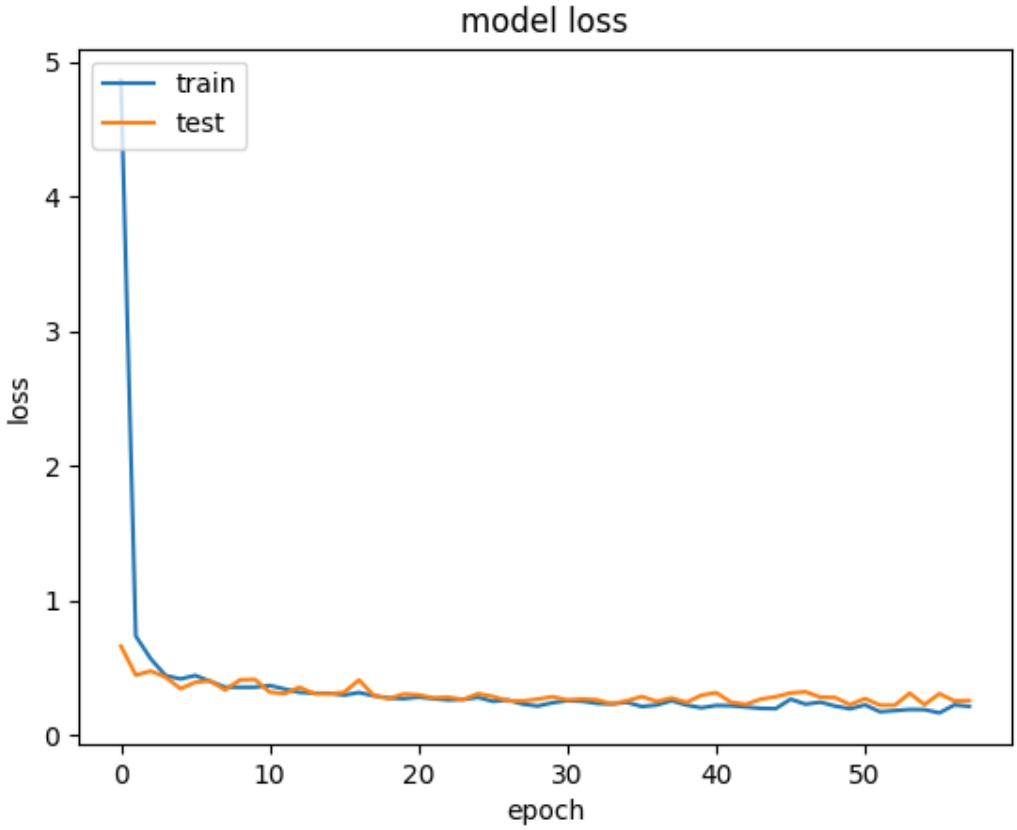

Fig 1.17 Model Loss

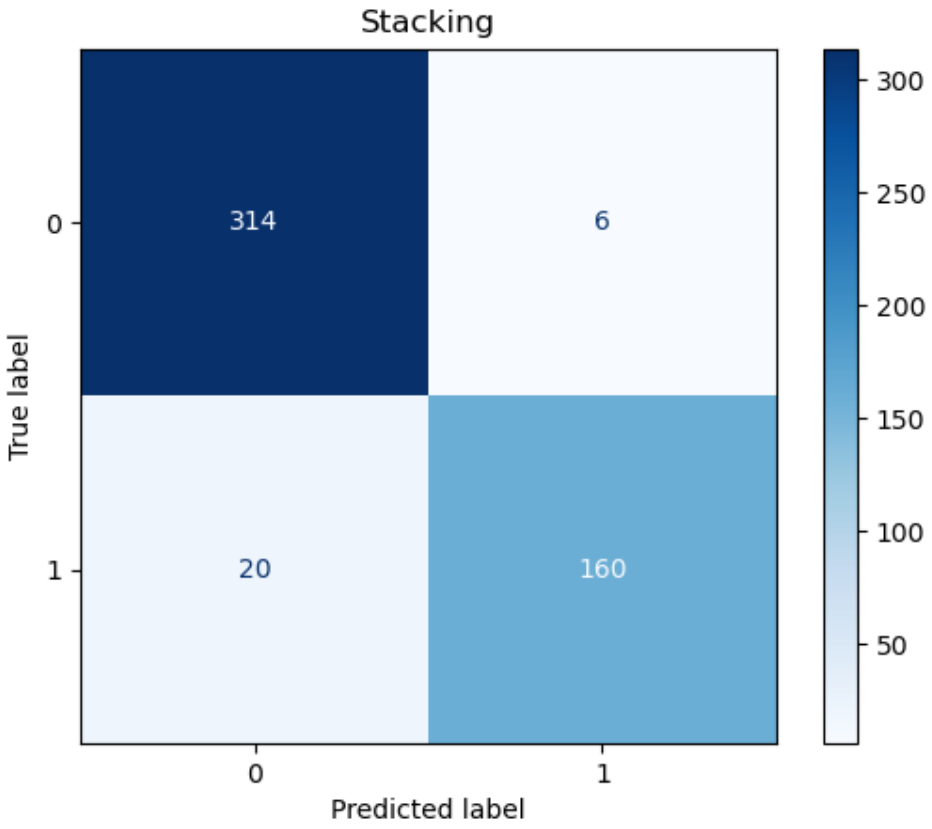

Fig 1.18 Stacking

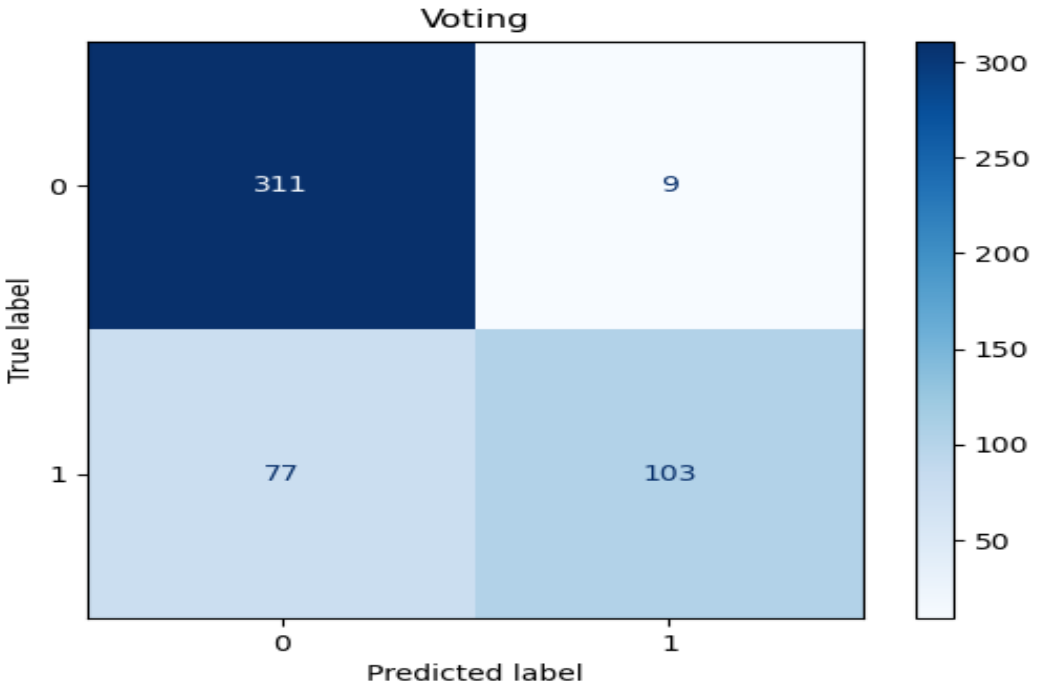

Fig 1.19 Voting

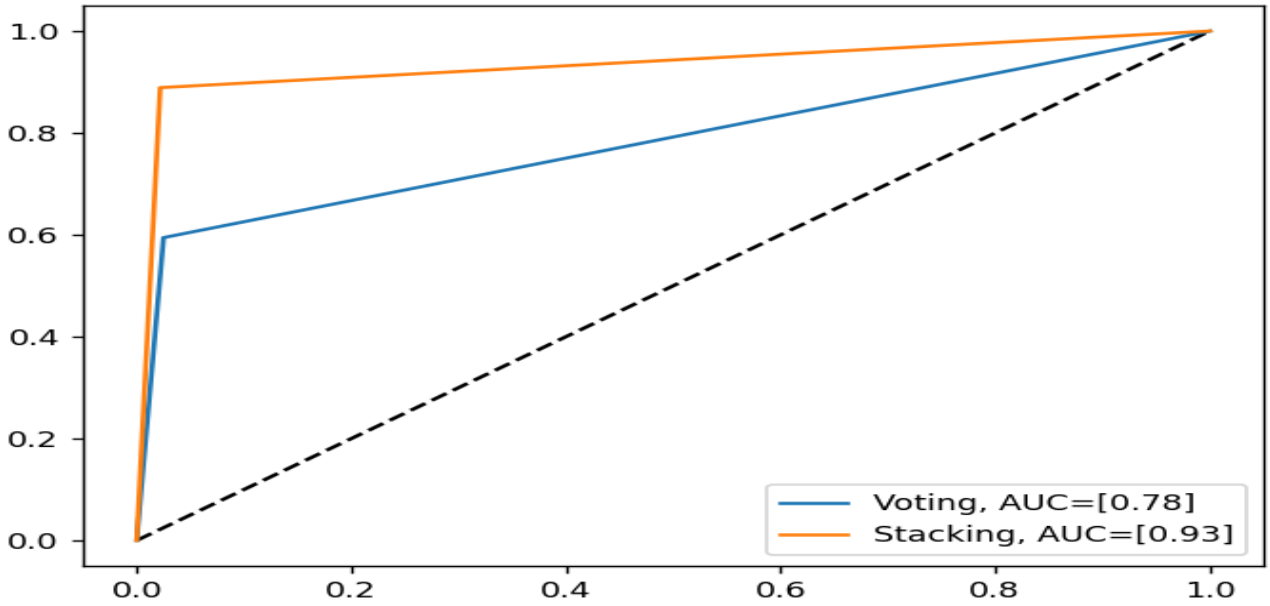

Fig 1.20 Voting Vs Stacking

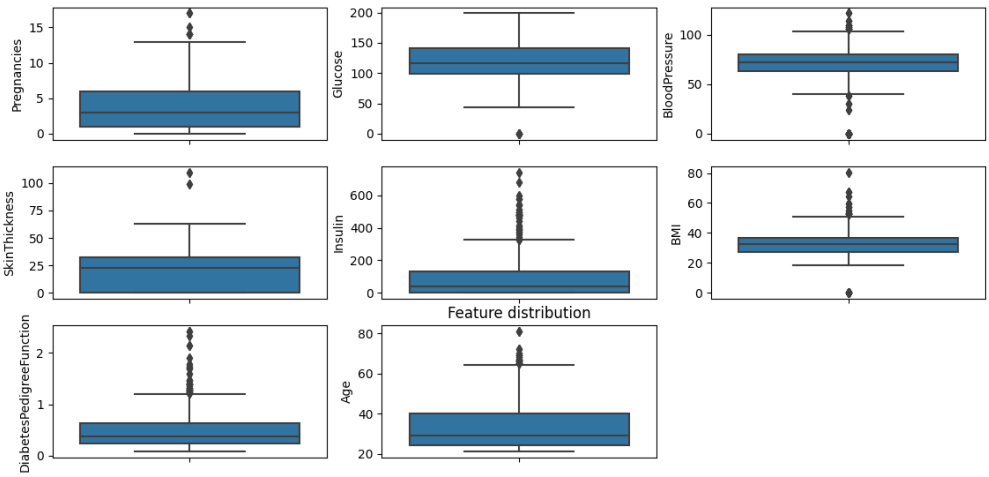

Fig 1.21 Feature Distribution

CHAPTER 5
WEB INTERFACE TO THE PROJECT

5.1 INTRODUCTION

In order to make the project dynamic, lively and user-friendly a web interface was envisaged which will facilitate the user to upload images that were trained, modelled and predicted for the outcome to be available on a client screen. In this context use is made of Python, Flask framework providing us user interface for uploading images and obtaining their results from the trained image dataset.

5.2 IMPLEMENTATION

```
2   from flask import Flask, render_template, request
3   import pickle
4   import numpy as np
5
```

Fig 1.22 Imporing libraries

```
16  @app.route('/predict', methods=['POST'])
17  def predict():
18      if request.method == 'POST':
19          preg = int(request.form['pregnancies'])
20          glucose = int(request.form['glucose'])
21          bp = int(request.form['bloodpressure'])
22          st = int(request.form['skinthickness'])
23          insulin = int(request.form['insulin'])
24          bmi = float(request.form['bmi'])
25          dpf = float(request.form['dpf'])
26          age = int(request.form['age'])
27
28          data = np.array([[preg, glucose, bp, st, insulin, bmi, dpf, age]])
29          my_prediction = classifier.predict(data)
30
31          return render_template('result.html', prediction=my_prediction)
32
```

Fig 1.23 Setup the values for prediction

5.3 OUTPUTS OF THE WEB INTERFACE

Fig 1.24 Home Page

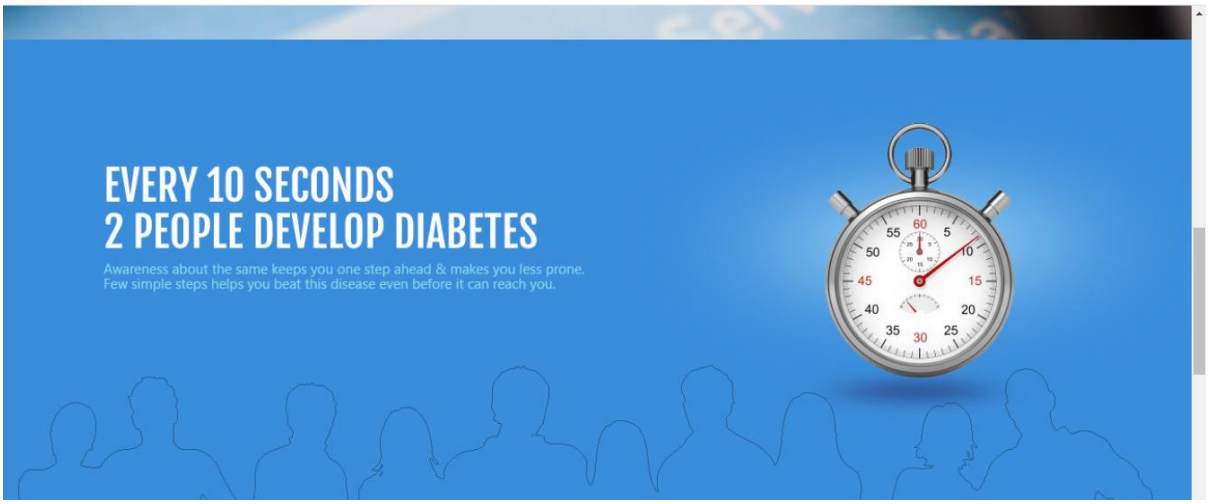

Fig 1.25 Cover Page

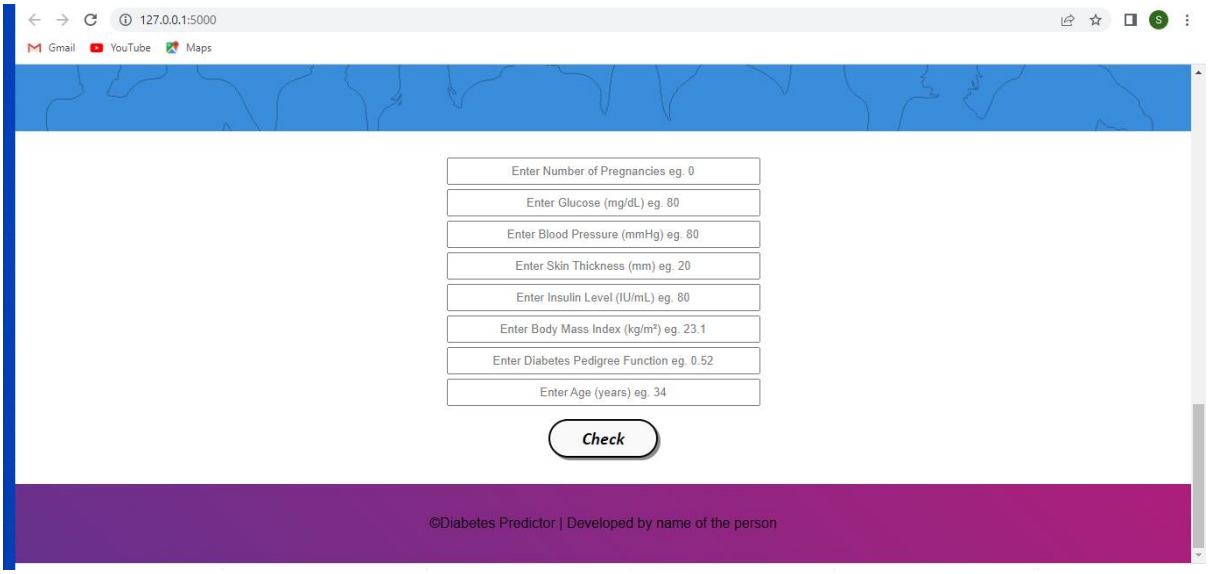

Fig 1.26 Uploading Values

Fig 1.27 Diet For Diabetic Patient

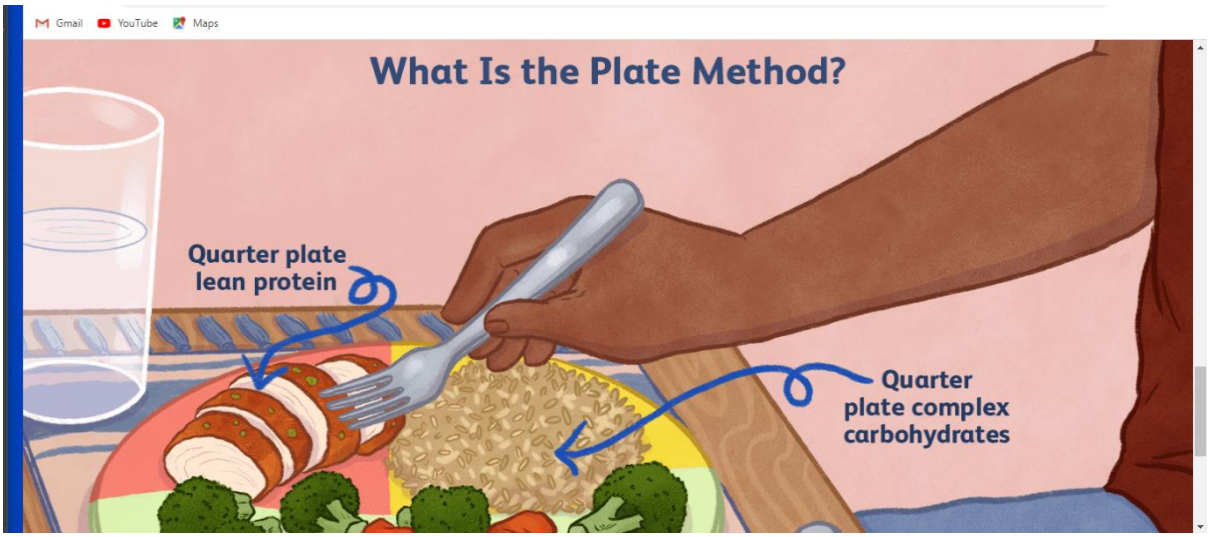

Fig 1.28 Diet for Non Diabetice Patient

CHAPTER 6
CONCLUSION AND FUTURE WORK

To sum up, the suggested technique for identifying diabetes through machine learning involves a combination of multiple algorithms using ensemble learning methods. The outcomes are optimistic, with a precision of 95%, and can assist in identifying and managing diabetes early, which is essential for enhancing health results.

There are several potential areas for further research in this field. One possibility is to expand the scope of the study by incorporating a bigger dataset to train and test the model. This would enhance the model's overall performance and make it more applicable to a wider range of people. Another option is to integrate additional machine learning techniques into the study, like Deep Learning and Support Vector Regression, to compare their efficacy in detecting diabetes.

Moreover, the research can be expanded to create a more thorough and precise framework that integrates further clinical and demographic information, such as age, sex, body mass index, and family medical history. This enhancement can enhance the model's precision and offer a tailored diagnosis and therapy plan.

In general, healthcare can benefit greatly from machine learning models, and additional investigation can enhance their effectiveness in identifying, treating, and averting different ailments, such as diabetes.

CHAPTER 7
REFERENCES

REFERNCES

1. In July 2014, Hayrettin Evirgen and Menduh Çerkezi published an article titled "Using Data Mining Technique for Predicting and Diagnosing Diabetic Retinopathy" in the Turkish Online Journal of Science & Technology, Volume 4 Issue 3, pages 32-37

2. In June 2017, a publication by Mrs. Pooja Rathi and Dr. Anurag Sharma titled "A Review Paper on Prediction of Diabetic Retinopathy using Data Mining Techniques" was included in the IJRIT journal's Volume 4, Issue 1, pages 292-297.

3. In 2018, a study titled "Machine Learning Algorithms for Predicting Diabetic Retinopathy and Identifying Interpretable Biomedical Features" was conducted by Tsao, H. Y., Chan, P. Y., and Su, E. C. Y. and published in BMC Bioinformatics, volume 19, issue 9.

4. In 2019, Sun and Zhang published an article titled "Electronic Health Records-Based Diagnosis and Analysis of Diabetic Retinopathy" in IEEE Access. The article explores the use of electronic health records for identifying and examining cases of diabetic retinopathy. It is available in volume 7 of the IEEE Access journal and spans pages 86115 to 86120.

5. In their article titled "Classification algorithms for predicting the risk level of diabetic retinopathy," Ramesh and Padmini (2017) discuss a system designed to predict the likelihood of developing diabetic retinopathy. This research was published in the International Journal of Scientific Development and Research, volume 2, issue 6.

6. In December 2012, Ramani, R. G., Lakshmi, B., and Jacob, S. G. presented a paper titled "Data Mining Method for Assessing Classifier Prediction Accuracy in Retinal Data" at the IEEE International Conference on Computational Intelligence and Computing Research. The paper is available in the conference proceedings and spans pages 1-4.

7. "Interpretable Machine Learning: A Handbook for Creating Understandable Models from Black Box Algorithms" is a book by Christoph Molnar that was published on April 7th, 2020. The article "Application of Machine Learning Algorithms for the Diagnosis of Diabetes Disease" was written by S. Mir Jalili, S. M. Mir Jalili, and A. Hatillo in 2017 and appeared in the Journal of Medical Systems. The article's DOI is 10.1007/s10916-017-0771-8.

8. In 2021, Suresh, K. P., Chandrasekaran, R., and Prasad, N. K. conducted a systematic review on the use of machine learning methods to predict diabetes. Their findings were published in the Journal of Diabetes Science and Technology, with a reference to the article's DOI: 10.1177/1932296820976375.

9. The article by Díaz-Parra et al. (2019) presents machine learning models designed to predict diabetes. This study was published in the International Journal of Medical Informatics and has a DOI of 10.1016/j.ijmedinf.2019.04.009.

10. Akram et al. (2021) wrote a review article in the Journal of Medical Systems discussing the use of machine learning techniques for diabetes diagnosis and prediction. The article has a DOI of 10.1007/s10916-021-01712-9 and covers various studies in this field.

CHAPTER 8

APPENDICES

```
warnings.filterwarnings("ignore")
df = pd.read_csv('diabetes_.csv')
df.head()  df.shape  df.info()  df.nunique()



df.tail()

df.isnull().sum()

df = df.fillna(df.mode().iloc[0])
df.isnull().sum()

df.describe()



corr = df.astype(float).corr()
plt.figure(figsize=(14, 12))
sns.heatmap(corr,
            linewidths=0.1,
            vmax=1.0,
            square=True,
            linecolor='white',
            annot=True,
            cmap="Blues",
            mask=np.triu(corr))
plt.show()


df.head()


import random
from sklearn.model_selection import train_test_split, KFold, cross_val_score,
GridSearchCV
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report,f1_score, accuracy_score,roc_curve,roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier

X = df.iloc[:, :-1].values   independent variables(features)
y = df.iloc[:,-1]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size = 0.20,
random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

#defining all necessary models for classification
def getBaseModels():
    baseModels = []
    baseModels.append(('LR', LogisticRegression()))
    baseModels.append(('KNN', KNeighborsClassifier()))
    baseModels.append(('DT', DecisionTreeClassifier()))
    baseModels.append(('NB', GaussianNB()))
    baseModels.append(('SVC', SVC(probability=True)))
    baseModels.append(('AB', AdaBoostClassifier()))
    baseModels.append(('GB', GradientBoostingClassifier()))
    baseModels.append(('RF', RandomForestClassifier()))
    baseModels.append(('ET', ExtraTreesClassifier()))
    return baseModels

#using k-fold on training data to evaluate the model accuracy
models = getBaseModels()
modelScores = []
modelStd = []
modelNameList = []
for name, model in models:
    cv = KFold(n_splits=10, random_state=42, shuffle=True)
    cv_results =
cross_val_score(model,X_train,y_train,cv=cv,scoring="accuracy",error_score="raise")
    modelScores.append(cv_results.mean()*100)
    modelNameList.append(name)
    modelStd.append(cv_results.std())

modelScores = np.round(modelScores, 2)
modelStd = np.round(modelStd, 2)
modelResult = pd.DataFrame({
    'Models': modelNameList,
    'Scores': modelScores,
    'σ': modelStd
})
print(modelResult.sort_values(by=['Scores', 'σ'], ascending=False))

#visualing the models performance
plt.title('Algorithm Comparison')
pal = sns.color_palette("Blues", len(modelScores)+2)
rank = modelScores.argsort().argsort()
ax=sns.barplot(y=modelScores, x=modelNameList, palette=np.array(pal[::-1])[rank])
ax.bar_label(ax.containers[0])
plt.show()

def gridSearchFunction(model, params, cv, X_val, y_val):
    gs = GridSearchCV(model, param_grid=params, cv=cv)
    gs.fit(X_val, y_val)
```

```
    print("accuracy: %.4f" % gs.best_score_)
    print("best params:", gs.best_params_)
    return gs.best_params_
### KNN
# n_neighbors: Number of neighbors to use by default for k_neighbors queries


param_grid = {
    "n_neighbors":
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
}
model = KNeighborsClassifier()
bestParams = gridSearchFunction(model, param_grid, 5, X_val, y_val)

.


param_grid = {
    "kernel": ['linear', 'poly', 'rbf', 'sigmoid'],
    'C': [0.1, 0.3, 0.5, 0.7, 0.9, 1.0, 1.3, 1.5, 1.7, 2.0]
}
model = SVC()
bestParams = gridSearchFunction(model, param_grid, 5, X_val, y_val)




param_grid = {
    "max_depth": [3, None],
    'max_features': [random.randint(1, 4)],
    'min_samples_leaf': [random.randint(1, 4)],
    'criterion': ["gini", "entropy"]
}
model = DecisionTreeClassifier()
bestParams = gridSearchFunction(model, param_grid, 5, X_val, y_val)




param_grid = {"max_depth": [5, 10, None], "n_estimators": [50, 100, 200]}
model = RandomForestClassifier()
bestParams = gridSearchFunction(model, param_grid, 5, X_val, y_val)




param_grid = {
    "learning_rate": [.01, .05, .1, .5, 1],
    'n_estimators': [50, 100, 150, 200, 250, 300]
}
model = AdaBoostClassifier()
bestParams = gridSearchFunction(model, param_grid, 5, X_val, y_val)



param_grid = {
    "learning_rate": [.01, .05, .1, .5, 1],
    'n_estimators': [50, 100, 150, 200, 250, 300]
```

```
}
model = GradientBoostingClassifier()
bestParams = gridSearchFunction(model, param_grid, 5, X_val, y_val)



param = {'n_neighbors': 12}
model1 = KNeighborsClassifier(**param)

param = {'C': 0.5, 'kernel': 'sigmoid'}
model2 = SVC(probability=True, **param)

param = {
    'criterion': 'gini',
    'max_depth': None,
    'max_features': 1,
    'min_samples_leaf': 3
}
model3 = DecisionTreeClassifier(**param)

param = {'max_depth': None, 'n_estimators': 100}
model4 = RandomForestClassifier(**param)

param = {'learning_rate': 0.01, 'n_estimators': 250}
model5 = AdaBoostClassifier(**param)

param = {'learning_rate': 0.01, 'n_estimators': 100}
model6 = GradientBoostingClassifier(**param)

param = {'max_depth': 5, 'n_estimators': 200}
model7 = ExtraTreesClassifier(**param)
def updatedBaseModels():
    baseModels = []
    baseModels.append(('KNN', model1))
    baseModels.append(('SVC', model2))
    baseModels.append(('DT', model3))
    baseModels.append(('RF', model4))
    baseModels.append(('AB', model5))
    baseModels.append(('GB', model6))
    baseModels.append(('ET', model7))
    return baseModels


# predicts the dependent value based on training
def model_predict(classifier, X_train, y_train, X_test):
    model = classifier.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return y_pred


def display_result(modelName, classifier, X_train, y_train, X_test, y_test):
    y_pred = model_predict(classifier, X_train, y_train, X_test)
    cm = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay(confusion_matrix=cm).plot(cmap='Blues')
    plt.title(modelName)
    plt.show()
    accuracy = np.round(accuracy_score(y_test, y_pred), 2)
```

```
auc = np.round(roc_auc_score(y_test, y_pred), 2)
f1 = np.round(f1_score(y_test, y_pred, average='macro'), 2)
roc = metrics.roc_curve(y_test, y_pred)
result = pd.DataFrame({
    'Model': [modelName],
    'Accuracy': [accuracy],
    'AUC Score': [auc],
    'F1 Score': [f1],
    'ROC': [roc]
})
return result
```

```
def compareResults(modelList):


    frames = []
    for model in modelList:
        frames.append(model)
    result = pd.concat(frames)
    result = pd.melt(frame=result.iloc[:, 0:-1], id_vars='Model',
var_name='Statistic', value_name='value')
    ax = sns.barplot(data=result, x='Model', y='value', hue='Statistic',
palette='Blues')
    sns.move_legend(ax, "lower right")
    for i in ax.containers:
        ax.bar_label(i, )

    # roc curve
    plt.figure(0).clf()
    plt.plot([0, 1], [0, 1], color='black', linestyle='--')
    for model in modelList:
        roc = model['ROC'].values
        fpr, tpr, _ = roc[0]
        auc = model['AUC Score'].values
        plt.plot(fpr, tpr, label=model.iloc[:, 0][0] + ', AUC=' + str(auc))
    plt.legend(loc='lower right')
    plt.show()

from sklearn.ensemble import VotingClassifier
baseModels = updatedBaseModels()
votingModel = VotingClassifier(baseModels, voting='hard')
votingRes=display_result('Voting', votingModel, X_train, y_train, X_test, y_test)

from sklearn.ensemble import StackingClassifier
baseModels = updatedBaseModels()
baseModels.remove(('GB', model6)) #removing gradient boosting from basemodels since
using GB as metamodel
metaModel = GradientBoostingClassifier(n_estimators=150,
                                       loss="exponential",
                                       max_features=6,
                                       max_depth=3,
                                       subsample=0.5,
                                       learning_rate=0.01)
stackModel = StackingClassifier(estimators=baseModels,
                                final_estimator=metaModel)
stackingRes=display_result('Stacking', stackModel, X_train, y_train, X_test, y_test)
```

```
models=[votingRes,stackingRes]
compareResults(models) dataset_path = 'dia_betes.csv'


dataset = np.loadtxt(dataset_path, delimiter=',')

X = dataset[:,0:8]
y = dataset[:,8]


X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,
shuffle=True, random_state=1)


X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
shuffle=True, random_state=2)
```

```
X_train = np.column_stack((X_train, np.zeros((len(X_train),1))))
X_test = np.column_stack((X_test, np.zeros((len(X_test),1))))
X_val = np.column_stack((X_val, np.zeros((len(X_val),1))))


X_train = X_train.reshape(len(X_train),3,3,1)
X_test = X_test.reshape(len(X_test),3,3,1)
X_val = X_val.reshape(len(X_val),3,3,1)


y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_val = to_categorical(y_val)


model = Sequential()

model.add(Conv2D(64, 2, activation='relu', input_shape=(3,3,1)))
model.add(MaxPooling2D(pool_size=1))
model.add(Conv2D(32, 2, activation='relu'))
model.add(MaxPooling2D(pool_size=1))
model.add(Flatten())
model.add(Dense(100, input_dim=8, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

print(model.summary())
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])


model.fit(X_train, y_train, validation_data=(X_val, y_val,), epochs=650,
batch_size=32, verbose=2)


scores = model.evaluate(X_train, y_train, verbose=0)
print('====================')
print('Result of validation set')
print("%s: %.2f%%" % (model.metrics_names[1], scores[1] * 100))
print('====================')


value = model.predict(X_test)
y_pred = np.argmax(value,axis=1)
y_true = np.argmax(y_test,axis=1)


result_CF_matrix = confusion_matrix(y_true, y_pred, labels=[0,1])
TP = result_CF_matrix[0,0]
FN = result_CF_matrix[0,1]
FP = result_CF_matrix[1,0]
TN = result_CF_matrix[1,1]
print('               Confusion Matrix')
print('                 Yes      No')
print('Actual Yes  %6d' %TP+'   %6d'%FN)
print('Actual No   %6d' %FP+'   %6d'%TN)


ACC = (TP + TN)/(TP+FN+FP+TN)
Sensitivity = TP/(TP+FP)
Specificity = TN/(FN+TN)
Recall = TP/(TP+FN)

print('====================')
print('Result of testing set')
print('Accuracy = %.2f%%' % (ACC*100))
print('Sensiticity = %.2f%%' % (Sensitivity*100))
print('Specificity = %.2f%%' % (Specificity*100))
print('Precision = %.2f%%' % (Sensitivity*100))
print('Recall = %.2f%%' % (Recall*100))
print('====================')


import matplotlib.pyplot as plt
plt.plot(ann.history['accuracy'])
plt.plot(ann.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(ann.history['loss'])
plt.plot(ann.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

scores = model.evaluate(X_test,y_test, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
# save model and architecture to single file
model.save("model.h5")
print("Saved model to disk")
```