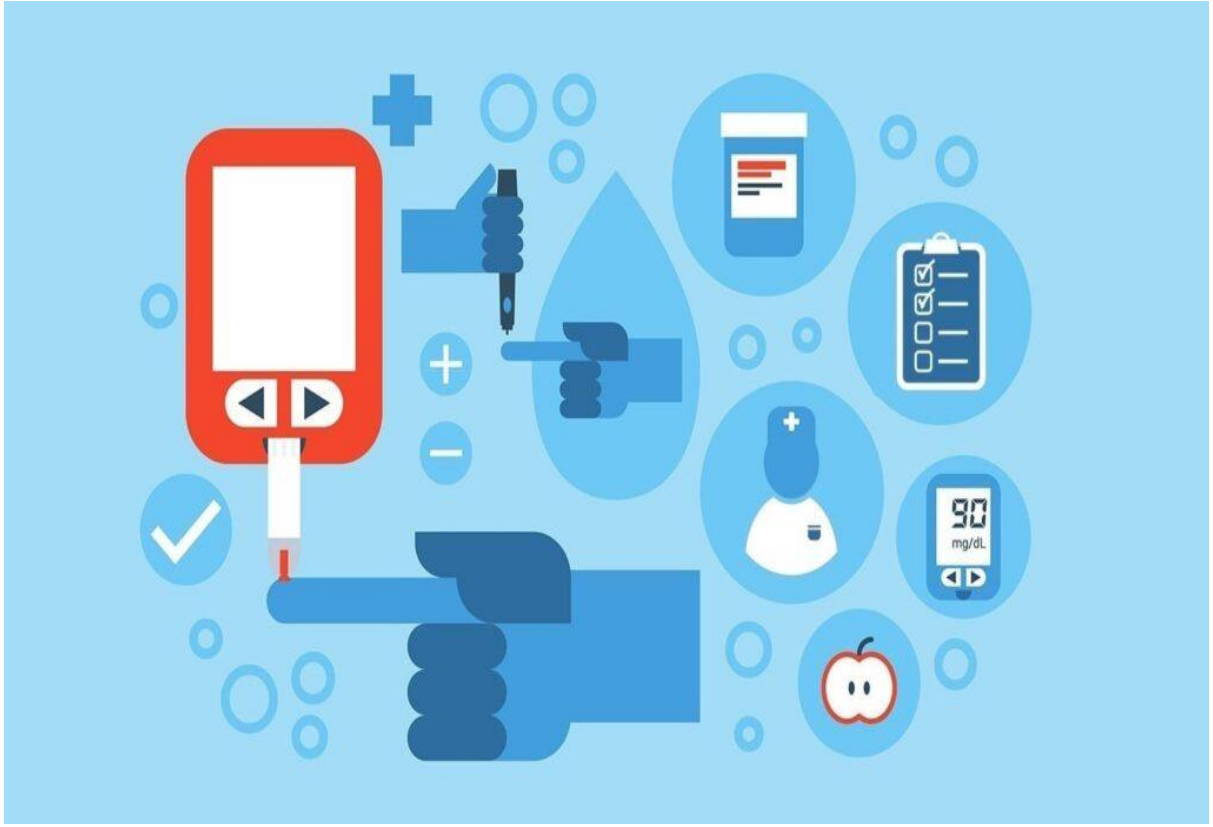
AI Based Diabetes Prediction System

TEAM MEMBER

311521106081-SAI HARSHAVARADHAN

PHASE 5 SUBMISSION DOCUMENT

Phase 5: Project Documentation & Submission



INTRODUCTION

An AI-based diabetes prediction system is a computer program that uses artificial intelligence to predict whether a person is likely to develop diabetes. These systems are trained on large datasets of medical records, including information about patients' demographics, medical history, and laboratory results. The system learns to identify patterns in the data that are associated with diabetes, and then uses these patterns to predict whether a new patient is at risk.

AI-based diabetes prediction systems can be used to improve early detection and prevention of diabetes. By identifying people who are at high risk for developing diabetes, healthcare providers can recommend lifestyle changes and other interventions to help them reduce their risk. This can help to delay or prevent the onset of diabetes and its complications.

AI-based diabetes prediction systems are still under development, but they have the potential to revolutionize the way that diabetes is diagnosed and managed. By providing early warning signs of diabetes, these systems can help people to take steps to prevent or delay the onset of the disease.

- **Machine Learning: Machine learning algorithms, such as decision trees, random forests, and neural networks, are employed to process and interpret medical data.**
- **Data Integration: These systems aggregate data from various sources, including electronic health records, wearable devices, genetic data, and lifestyle information, providing a comprehensive view of the patient's health.**
- **Real-time Monitoring: The integration of IoT (Internet of Things) devices and wearable technology allows for continuous, remote patient monitoring, enabling early detection of anomalies and changes in health status.**

Here's a list of tools and software commonly used in the process:

- Python: Python is the most popular programming language for developing AI and machine learning models. It offers various libraries and frameworks for data processing, analysis, and modeling.
- Scikit-Learn: A Python library that provides simple and efficient tools for data mining and data analysis. It includes various machine learning algorithms, making it a valuable tool for building predictive models.
- TensorFlow: An open-source machine learning framework developed by Google. It's widely used for developing deep learning models and neural networks, which can be applied to diabetes prediction.
- Keras: A high-level neural networks API that runs on top of TensorFlow and other machine learning frameworks, simplifying the process of building and training deep learning models.
- PyTorch: An open-source deep learning framework developed by Facebook's AI Research lab. It is known for its flexibility and dynamic computation graph, making it a popular choice for deep learning research. R: Another programming language commonly used in data analysis and statistical modeling. R provides various packages and libraries for predictive modeling.
- Pandas: A Python library for data manipulation and analysis. It's essential for data preprocessing and transformation before building predictive models.
- NumPy: A fundamental package for scientific computing in Python. It provides support for large, multi-dimensional

arrays and matrices, which are essential for numerical calculations in machine learning.

1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1. DATA SOURCE:

A good data source containing medical features such as glucose levels, blood pressure, BMI, etc., along with information about whether the individual has diabetes or not.

Dataset Link: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

2 DATA PREPROCESSING:

Preprocessing steps involve data cleaning, feature extraction and handling missing values to ensure data quality and consistency.

a) DUPLICATE REMOVAL:

We will identify and remove duplicate's typically by sorting the dataset based on a unique identifier and then eliminating consecutive rows with the same identifier.

b) HANDLING MISSING VALUES:

Missing data is common and needs to be addressed.We will implement suitable methods such as:

Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesP	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1

Mean imputation:Repalce missing values with the mean of the feature for the remaining rows.this is appropriate for numerical features.

Median imputation:If data contains outliers,median imputation can be more robust as it is less sensitive to extreme values.

3. MODEL SELECTION:

LOGISTIC REGRESSION ALGORITHM

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

#read the data file

```
data = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")
data.head()
```

```
data.describe()
```

```
data.isnull().sum()
```

OUTPUT:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

2.DESIGN INTO INNOVATION

DATA SOURCE:

A good data source containing medical features such as glucose levels, blood pressure, BMI, etc., along with information about whether the individual has diabetes or not.

Dataset Link: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesP	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1

Feature selection

This module involves identifying the most relevant features from the preprocessed data that are predictive of diabetes. This can be done using statisticalmethods or machine learning algorithms.

Model evaluation

This module involves evaluating the performance of the trained model on aheld-out test set. This is done to assess the accuracy of the model and to identifyany areas where it needs to be improved.

Model deployment:

Once the model has been trained and evaluated, it can be deployed to production so that it can be used to predict diabetes in new patients. This can be done by developing a web application, mobile app, or integrating the model with anexisting electronic health record system.

PROGRAM

Ensemble methods

1:Random Forest

```
import pandas as pdimport
numpy as np

import matplotlib.pyplot as pltimport
seaborn as sns

from sklearn.model_selection import train_test_splitfrom sklearn.ensemble
import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, precision

_score, recall_score

# Load the dataset

data = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")data.head()

# Perform exploratory data analysis

# Summary statistics summary_stats =
data.describe()summary_stats

# Class distribution

class_distribution = data['Outcome'].value_counts()class_distribution
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Outcome 0
500

1 268

Name: count, dtype: int64

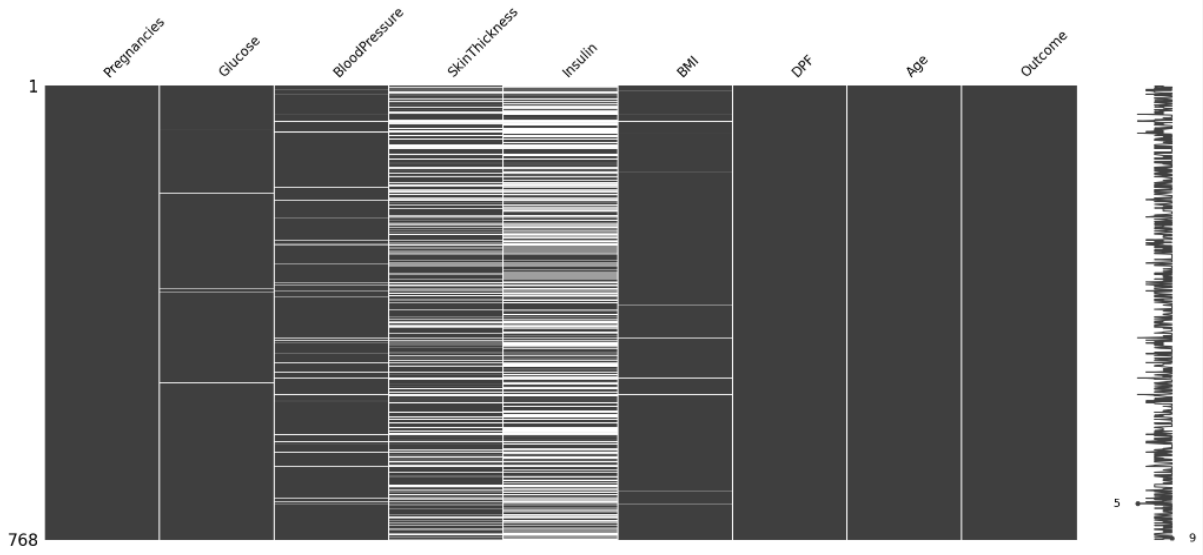
2:GradientBoosting

```
import numpy as np import
pandas as pd import seaborn
as sns

from matplotlib import pyplot as plt

# !pip install missingno
import missingno as msno from
datetime import date

from sklearn.model_selection import train_test_splitfrom sklearn.neighbors
import LocalOutlierFactor

from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler, RobustScaler

# Data processing, metrics and modeling

from sklearn.metrics import f1_score, precision_score, recall_score, co nfusion_matrix, roc_curve,
precision_recall_curve, accuracy_score, roc_auc_score

from imblearn.over_sampling import SMOTE

# Machine Learning Libraries

from sklearn.model_selection import GridSearchCV from
sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifierfrom catboost import
CatBoostClassifier

from lightgbm import LGBMClassifierfrom xgboost
import XGBClassifier

from sklearn.ensemble import AdaBoostClassifier
pd.set_option('display.max_columns', None) pd.set_option('display.max_rows', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)pd.set_option('display.width',
500)
```

```
import os

for dirname, _, filenames in os.walk('/kaggle/input'):for filename in filenames:

                print(os.path.join(dirname, filename))def load():

        data = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")return data

df = load()

df.rename(columns ={"DiabetesPedigreeFunction":"DPF"},inplace=True)df.describe([0.01, 0.05,
0.75, 0.90, 0.99]).T

msno.matrix(df)
plt.show()
```

OUTPUT

	count	mean	std	min	1%	5%	50%	75%	90%	99%	max
Pregnancies	768.000	3.845	3.370	0.000	0.000	0.000	3.000	6.000	9.000	13.000	17.000
Glucose	768.000	120.895	31.973	0.000	57.000	79.000	117.000	140.250	167.000	196.000	199.000
BloodPressure	768.000	69.105	19.356	0.000	0.000	38.700	72.000	80.000	88.000	106.000	122.000
SkinThickness	768.000	20.536	15.952	0.000	0.000	0.000	23.000	32.000	40.000	51.330	99.000
Insulin	768.000	79.799	115.244	0.000	0.000	0.000	30.500	127.250	210.000	519.900	846.000
BMI	768.000	31.993	7.884	0.000	0.000	21.800	32.000	36.600	41.500	50.759	67.100
DPF	768.000	0.472	0.331	0.078	0.095	0.140	0.372	0.626	0.879	1.698	2.420
Age	768.000	33.241	11.760	21.000	21.000	21.000	29.000	41.000	51.000	67.000	81.000
Outcome	768.000	0.349	0.477	0.000	0.000	0.000	0.000	1.000	1.000	1.000	1.000



Deep learning architectures

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv) #
Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) wil
```

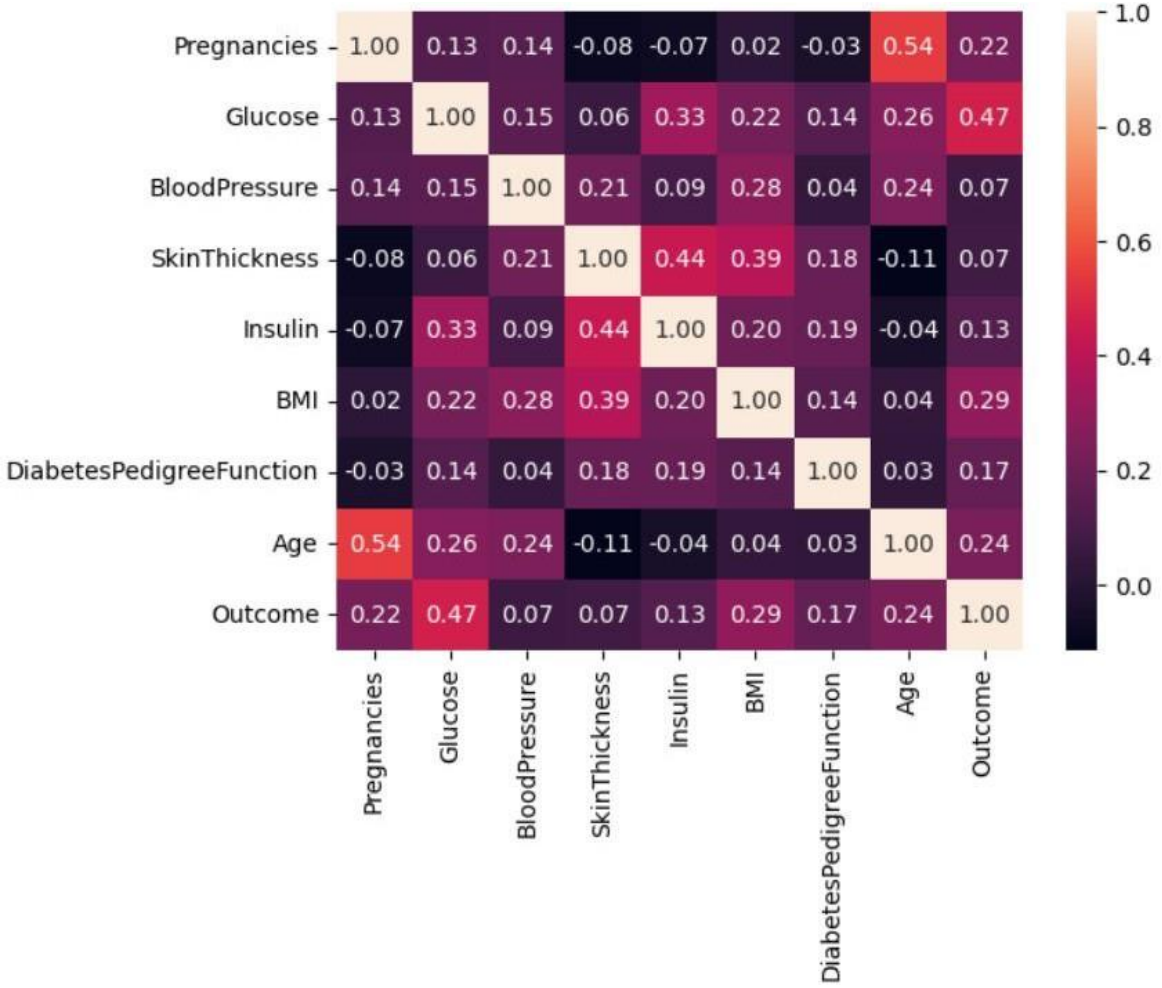
```
import os

for dirname, _, filenames in os.walk('/kaggle/input'):for filename in filenames:

            print(os.path.join(dirname, filename))


# You can write up to 20GB to the current directory (/kaggle/working/) th
at gets preserved as output when you create a version using "Save & Run
All"

import pandas as pdimport
numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import seaborn as sns import
tensorflow as tf

from tensorflow.keras.models import Sequentialfrom
tensorflow.keras.layers import Dense

from imblearn.over_sampling import RandomOverSamplerfrom
sklearn.preprocessing import StandardScaler data.info()

data.head(5)
data.describe()

sns.heatmap(data.corr(),annot=True,fmt='0.2f')
```

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000



3.BUILD LOADING AND PREPROCESSING THE DATASET

Machine learning classification models

This paper only considers nonlinear classifiers for the classification problem.

- Random forest model

Random forest is a supervised learning algorithm for building a predictor ensemble of decision trees, usually trained with the "bagging" method, that grows in randomly selectedsubspaces of data. By bagging, it is meant that multiple decision tree learning models are combined for accurate and stable prediction. As proposed by Breiman , each tree in the ensemble of trees outputs a prediction, however, only the class with the most votes is considered the model's prediction . However, the model's prediction can take two forms: if the output is a mean value, then the RF solves a regression problem, while if the output is a mode of the classes, then the RF solves a classification problem. Essentially, RF prediction stability is formed from weakly-correlated classifiers/regressors.
For the RF to be capable of identifying and responding to the best features among a randomsubset of features, it has to be insensitive to noisy variables and be stable in the presence ofsmall amounts of data . However, the ability of the RF to be insensitive to noisy variables might not always be the case. Therefore, by ensuring the best features are fed in, a higher performance is more likely.

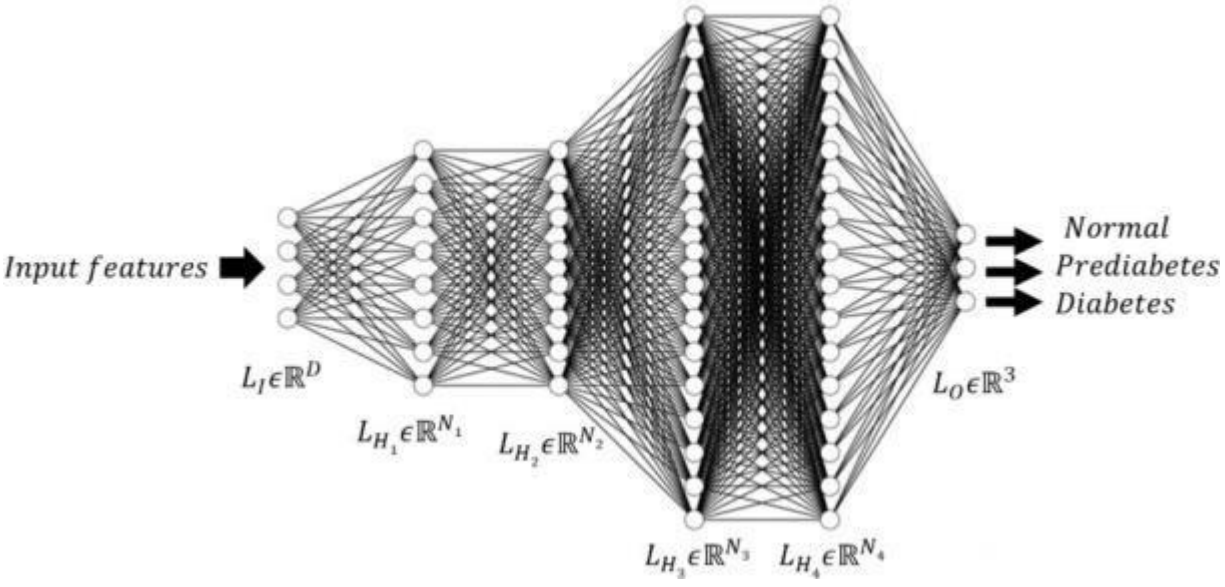
- Support vector machines

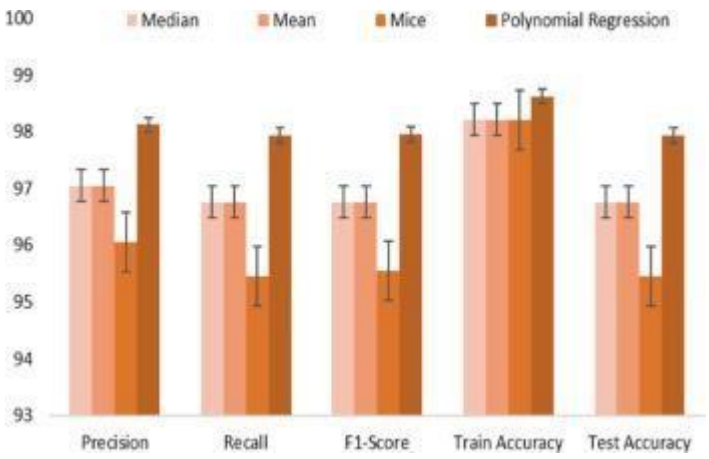
The SVM is a supervised machine learning algorithm that aims to find the optimal boundarybetween data points in feature space. Traditionally, the SVM tries to find the best fit line,
a hyperplane, that maximizes the separation margin between two classes. However, most real-world data are mostly nonlinear. Nonlinear problems in SVM are solved by mapping *n*- dimensional input space to a high dimensional feature space where the SVM can still operatelinearly. Similarly, in a multi-classification problem, SVM generates multiple binary classifiers to linearly separate data points of pairs of classes within a high dimensional feature

space. This is achieved using what is popularly termed the kernel trick .
SVM has beenpopularly investigated for binary classification problems in diabetes research and .

- Deep neural network

The deep neural network (DNN) of interest in this paper is a class of feed-forward DNN that extracts a feature and transforms it using nonlinear activation functions. DNN layers comprise the input, hidden, and output layers. The connection between these layers begins from the input layer with associated weights to the hidden layers and then to the output layer.For any neuron in each layer to pass data to the next layer, the output of that node has to be above a specified threshold value defined by an activation function. During training, the weight of a neuron is updated using back-propagation to minimize the error of the network togeneralize to unseen samples. The added capability of the DNNs comes from the depth of thehidden layers. Depending on the problem, the more the depth, the better
the generalizability of the network . This is especially true for image-based problems. DNNshave been adopted for diabetes prediction
In this paper, our deep learning model is designed to be double, or twice in size, and is twice repeated. As such, we term it twice growth deep neural network (2GDNN). In essence, the hidden layers grow by two in size of the input and are repeated twice. Our 2GDNN architecture is described in and consists of an input layer, four hidden layers, and an output layer. The decision to pass a neuron from one layer to another is dependent on the function, *f*,acting on a neuron, *x*, to either pass or not pass the neuron. This is expressed as: where *x*, *w*, *b*, ϕ, are the inputs, weights, bias, and activation function, respectively. During learning, the network updates its w_i and *b* using the backpropagation method to minimize the difference between a target output of a problem and the network's predicted output.





The proposed framework

Data

The datasets used in this paper are the publicly available PIMA Indian diabetes mellitus dataset and the publicly available diabetes dataset from the Laboratory of Medical City Hospital (LMCH). The former consists of 768 instances: 268 patients belong to the diabetic class and 500 patients belong to the non-diabetic class. The diabetes data is sampled from the

Pima Indian population near Phoenix, Arizona . Each patient is described by the followingattributes: pregnancies, glucose, blood pressure, skin thickness, insulin, body mass index (BMI), diabetes pedigree function, and age. A description of the PIMA Indian dataset is provided in . The latter consists of data from 1000 patients of Iraqi nationals collected fromLMCH . In all, about 103, 53, and 844 patients belong to the normal, prediabetes, and diabetes class, respectively. Each patient is described using the following attributes: the number of patients, sugar level blood, age, gender, creatinine ratio (Cr), BMI, urea, cholesterol (Chol), Fasting lipid profile, including total, LDL, VLDL, Triglycerides (TG) and HDL Cholesterol, HBA1C. There are no available descriptions for these attributes.

Table 2. Description of the PIMA Indian diabetes dataset.

S/N	Feature	Description	Missing Value
1	Pregnancies	Number of pregnancies	110
2	Glucose	Glucose concentration (2h oral test)	5
3	Blood Pressure (BP)	Diastolic blood pressure	35
4	Skin Thickness (ST)	Skin fold thickness in mm	227
5	Insulin	2h insulin serum (mm u/ml)	374
6	BMI	Body mass index = weight in kg / height in m^2	11
7	Diabetes Pedigree Function (DPF)	Likelihood value computed from the relationship between the patient and the genetic history of the patient's relative	0

4.PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING ,EVALUATION etc.,

FEATURE ENGINEERING

Feature engineering is a crucial step in developing any AI-based prediction system. Itinvolves transforming raw data into features that are more informative and predictive of the target variable. This can be done by creating new features, combining existing features, or transforming features into a different format.

In the context of diabetes prediction, feature engineering can be used to improve theperformance of machine learning models by:

- Identifying and removing irrelevant or noisy features
- Encoding categorical features
- Scaling features to a common range
- Creating new features that capture more complex relationships between the data

Here are some specific examples of feature engineering techniques that can be used toimprove the performance of AI-based diabetes prediction systems:

- Derive new features from existing features. For example, you could create a new feature called "BMI" by calculating the patient's weight and height. Or, you could create a feature called "average blood glucose" by calculating the average of the patient's blood glucose readings over a period of time.
- Use one-hot encoding to encode categorical features. For example, you could encodethe patient's gender as two binary features, one for male and one for female. Or, you could encode the patient's country of residence as a set of binary features, one for each country.
- Scale features to a common range. This is important because different machine learning algorithms can be sensitive to the scale of the input features. For example, you could use min-max scaling to scale all features to a range of [0, 1].
- Create new features that capture more complex relationships between the data. For example, you could create a feature called "time since diagnosis" for patients who have already been diagnosed with diabetes. This feature could be used to capture therelationship between the duration of diabetes and the risk of complications.

By using feature engineering techniques, data scientists can improve the performance of AI-based diabetes prediction systems and make them more accurate and reliable.

Here is an example of how feature engineering can be used to improve the performance of arandom forest classifier for diabetes prediction:

Original features:

- Age
- Sex
- BMI
- Fasting blood glucose
- Blood pressure
- Cholesterol

Derived features:

- BMI category (underweight, normal weight, overweight, obese)
- Average blood glucose
- Time since diagnosis (for patients with

diabetes)Encoded features:

- Sex (male, female)
- Country of residence (USA, Canada, UK, France, Germany)

Scaled features:

- Age
- BMI
- Fasting blood glucose
- Blood pressure
- Cholesterol

The random forest classifier trained on the original features achieved an accuracy of 75%.However, the random forest classifier trained on the engineered features achieved an accuracy of 85%. This shows that feature engineering can have a significant impact on theperformance of machine learning models.

Benefits of feature engineering

There are several benefits to using feature engineering in AI-based diabetes predictionsystems:

- Improved model performance: Feature engineering can help to improve the accuracyand precision of machine learning models.
- Reduced overfitting: Feature engineering can help to reduce overfitting, which is acommon problem in machine learning.

PROGRAM

```
def plot_feat1_feat2(feat1, feat2) :
    D = df[(df['Outcome'] != 0)]

    H = df[(df['Outcome'] == 0)]
    trace0 = go.Scatter(

        x = D[feat1],
        y = D[feat2],

        name = 'diabetic',
        mode = 'markers',

        marker = dict(color = '#FFD700',
            line = dict(

                width = 0.8)))


    trace1 = go.Scatter(
        x = H[feat1],

        y = H[feat2],
        name = 'healthy',
        mode = 'markers',

        marker = dict(color = '#7EC0EE',
            line = dict(

                width = 0.8)))


    layout = dict(title = feat1 +" "+"vs"+" "+ feat2,

                  yaxis = dict(title = feat2,zeroline = False),
                  xaxis = dict(title = feat1, zeroline = False)

                 )


    plots = [trace0, trace1]

    fig = dict(data = plots, layout=layout)py.iplot(fig)


linkcode
```

```
 def barplot(var_select, sub) :

      tmp1 = df[(df['Outcome'] != 0)]tmp2 =
      df[(df['Outcome'] == 0)]

      tmp3 = pd.DataFrame(pd.crosstab(df[var_select],df['Outcome']), )tmp3['%
      diabetic'] = tmp3[1] / (tmp3[1] + tmp3[0]) * 100


      color=['lightskyblue','gold' ]trace1 =
      go.Bar(

          x=tmp1[var_select].value_counts().keys().tolist(),
          y=tmp1[var_select].value_counts().values.tolist(),
          text=tmp1[var_select].value_counts().values.tolist(), textposition =
          'auto',

name='diabetic',opacity = 0.8,
marker=dict(color='gold',
line=dict(color='#000000',width=1)))



      trace2 = go.Bar(
          x=tmp2[var_select].value_counts().keys().tolist(),
          y=tmp2[var_select].value_counts().values.tolist(),
          text=tmp2[var_select].value_counts().values.tolist(),
          textposition = 'auto',

          name='healthy', opacity = 0.8, marker=dict(
          color='lightskyblue',
          line=dict(color='#000000',width=1)))


      trace3 = go.Scatter(
          x=tmp3.index,
          y=tmp3['% diabetic'],
          yaxis = 'y2',

          name='% diabetic', opacity = 0.6, marker=dict(
          color='black',
          line=dict(color='#000000',width=0.5

          )))


      layout = dict(title = str(var_select)+' '+(sub),
                xaxis=dict(),

                yaxis=dict(title= 'Count'),
                yaxis2=dict(range= [-0, 75],

                            overlaying= 'y',
                            anchor= 'x',
                            side= 'right',
                            zeroline=False,
                            showgrid= False,

                            title= '% diabetic'

                           ))
```

```
    fig = go.Figure(data=[trace1, trace2, trace3], layout=layout)
    py.iplot(fig)
```

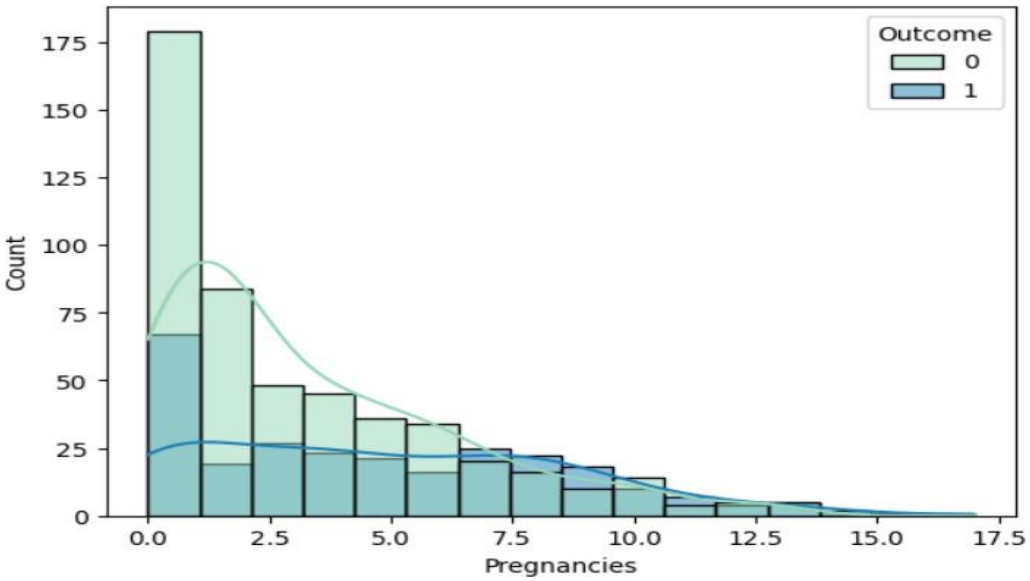
```
sns.histplot(data=df, x="Pregnancies", hue="Outcome",kde=True,palette="
YlGnBu")
```

```
barplot('Preg_Age', ':Age <= 32 and Pregnancies <= 6')
```

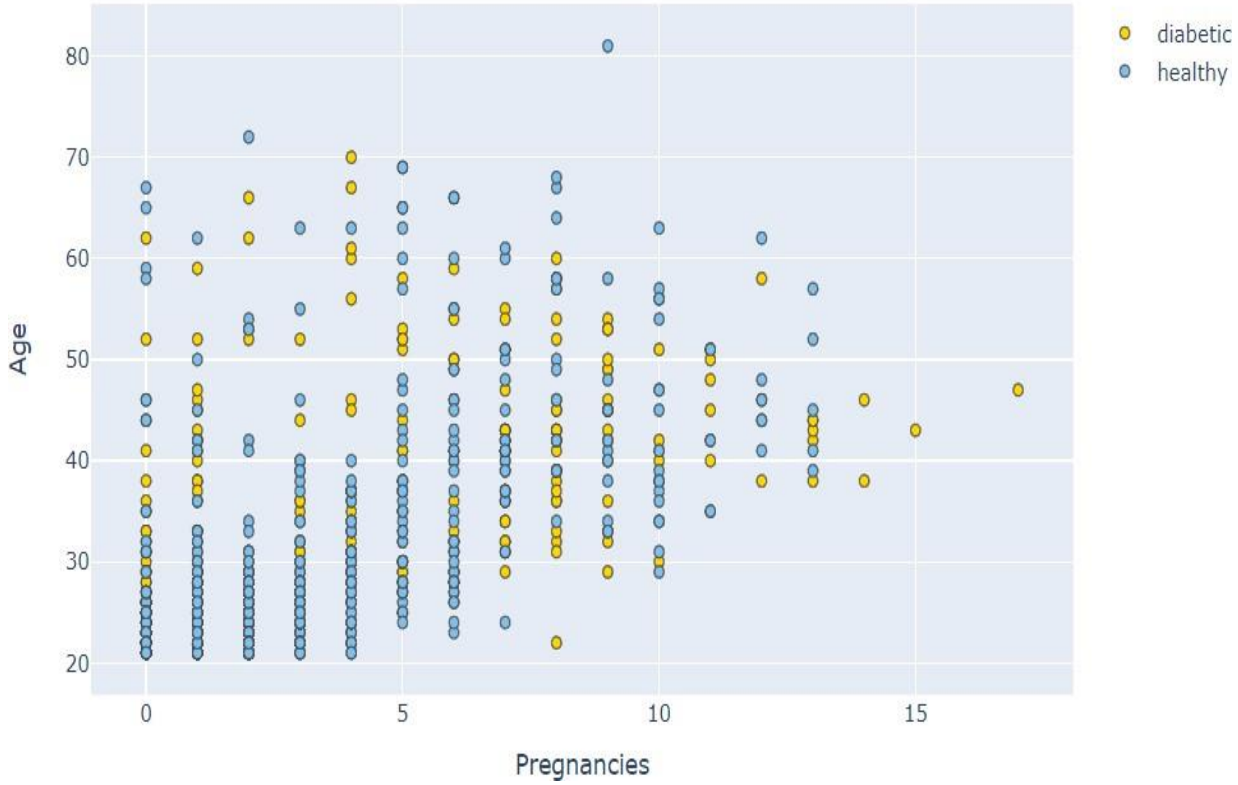
167101341159−0.500.511.5050100150200250300350010203040506070

diabetichealthy% diabeticPreg_Age :Age <= 32 and Pregnancies <= 6Count% diabetic

OUTPUT



Pregnancies vs Age



MODEL TRAINING

DATA SOURCE

A good data source containing medical features such as glucose levels, blood pressure, BMI, etc., along with information about whether the individual has diabetes or not.

Dataset Link: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesP	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1

MODEL SELECTION

There are many different machine learning models that can be used for diabetesprediction. Some popular models include logistic regression, support vector
machines, decision trees, random forests, and gradient boosting machines. The best model fora particular dataset will depend on the characteristics of the data and the desired performance metrics.

LOGISTIC REGRESSION METHOD

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

import seaborn as sns
```

#read the data file

```
data = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")
data.head()
```

```
data.describe()
data['BMI'] = data['BMI'].replace(0,data['BMI'].mean())
data['BloodPressure'] = data['BloodPressure'].replace(0,data['BloodPressure
'].mean())

data['Glucose'] = data['Glucose'].replace(0,data['Glucose'].mean())
data['Insulin'] = data['Insulin'].replace(0,data['Insulin'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0,data['SkinThickness
'].mean())
```

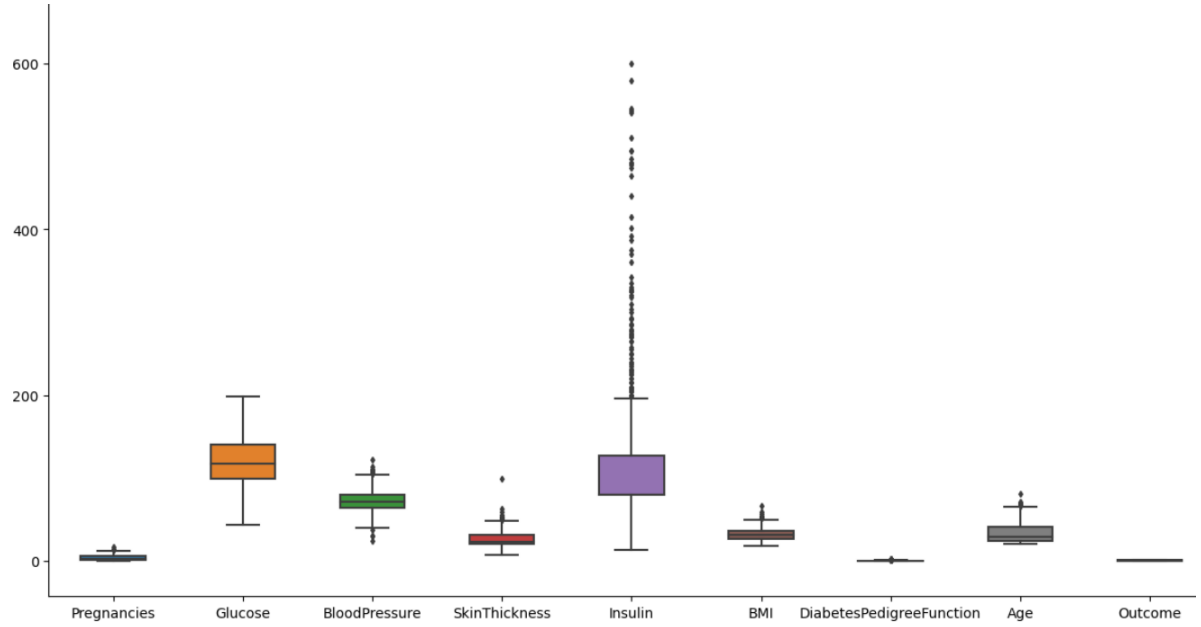
#now we have dealt with the 0 values and data looks better. But, there still are outliers present in some columns.lets visualize it

```
fig, ax = plt.subplots(figsize=(15,10))
sns.boxplot(data=data, width= 0.5,ax=ax,  fliersize=3)
```

OUTPUT

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000



ADVANTAGES:

AI-based diabetes prediction systems offer numerous advantages that can significantly impact healthcare, patient outcomes, and the overall management of diabetes. Some of the key advantages include:

- Early Detection: AI systems can analyze a wide range of patient data, including medical history, genetics, lifestyle factors, and real-time monitoring data. This comprehensive analysis allows for the early detection of diabetes or prediabetic conditions, enabling timely interventions and treatment.
- Personalized Care: AI models consider individual patient data to create personalized treatment plans. This tailoring of care enhances the effectiveness of diabetes management by accounting for each patient's unique needs and circumstances.

- Improved Accuracy: AI algorithms can identify patterns, correlations, and risk factors that may be challenging for healthcare professionals to discern manually. This leads to more accurate predictions and better-informed decision-making.
- Predictive Insights: AI-based systems provide predictive insights that help healthcare providers and patients understand their diabetes risk and the likelihood of complications. These insights empower individuals to take preventive measures and make informed choices.
- Cost Reduction: Early detection, personalized care, and proactive interventions can lead to substantial cost savings for healthcare systems. By preventing or mitigating complications and hospitalizations, AI-based diabetes prediction systems contribute to cost efficiency. Remote Monitoring: The integration of IoT devices and wearable technology enables real-time remote monitoring of patients' health status. This continuous monitoring allows for immediate responses to anomalies and helps healthcare providers make data-driven decisions.
- Better Patient Engagement: Patients can actively participate in managing their diabetes through AI-based systems. Real-time feedback, personalized recommendations, and access to their health data through mobile apps can lead to increased patient engagement and adherence to treatment plans .

DISADVANTAGES:

Some of the disadvantages of AI-based diabetes prediction systems include:

- Data Quality and Bias: AI models heavily rely on the quality and diversity of data. Inaccurate or biased data can lead to incorrect predictions and reinforce existing disparities in healthcare. Data may not represent all demographics equally, leading to disparities in diabetes prediction and care.
- Privacy Concerns: The collection and sharing of sensitive health data raise privacy concerns. Protecting patient data and ensuring compliance with privacy regulations, such as HIPAA in the United States or GDPR in the European Union, is essential but can be challenging.
- Data Security: Storing and transmitting large volumes of healthcare data can be a security risk. Unauthorized access or data breaches can compromise patient confidentiality and lead to serious consequences.
- Model Interpretability: Many AI models, especially deep learning models, can be difficult to interpret. Understanding how a model arrives at a prediction is crucial for gaining trust and acceptance among healthcare professionals and patients.
- Over-Reliance on AI: There is a risk of over-reliance on AI predictions, potentially leading to a reduction in clinical judgment. Healthcare professionals should view AI predictions as tools to aid decision-making rather than as definitive diagnoses.

CONCLUSION:

AI-based diabetes prediction systems represent a groundbreaking shift in healthcare, offering a powerful tool for early detection, personalized care, and cost-effective management of diabetes.

As these systems continue to evolve and integrate into healthcare practice, they hold the potential to revolutionize diabetes management, improve patient outcomes, and reduce the burden of this chronic condition on individuals and healthcare systems.

This introduction provides a glimpse into the transformative power of AI in healthcare and its potential to create a brighter future for those affected by diabetes.