AI BASED DIABETES PREDICTION SYSTEM

TEAM MEMBER

NAME: S.HARIPRABU

REGNO: 912421106303

Gmail id: hariprabu1734@gmail.com

NM ID: aut2291240022

PHASE 2 Submission Document

Introduction

- Diabetes is a chronic disease that directly affects the pancreas, and the body is incapable of producing insulin.It is mainly responsible for maintaining the blood glucose level.
- Many factors, such as excessive body weight, physical inactivity, high blood pressure, and abnormal cholesterol level, can cause a person get affected by diabetes.
- To use machine learning classification methods, that is, decision tree, SVM, Random Forest, Logistic Regression, KNN, and various ensemble techniques, to determine which algorithm produces the best prediction results.
- In this paper, we have employed machine learning and explainable AI techniques to detect diabetes.

Content for Project Phase 2

Consider exploring innovative techniques such as ensemble methods and deep learning architectures to improve the prediction system's accuracy and robustness.

Data source

AI-powered diabetes prediction system that leverages machine learning algorithms to analyze medical data and predict the likelihood of an individual developing diabetes, providing early risk assessment and personalized preventive measures.

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

Pregnanci	Glucose	BloodPres	SkinThickr	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
10	125	70	26	115	31.1	0.205	41	1
7	147	76	0	0	39.4	0.257	43	1
1	97	66	15	140	23.2	0.487	22	0
13	145	82	19	110	22.2	0.245	57	0
5	117	92	0	0	34.1	0.337	38	0
5	109	75	26	0	36	0.546	60	0
3	158	76	36	245	31.6	0.851	28	1
3	88	58	11	54	24.8	0.267	22	0
6	92	92	0	0	19.9	0.188	28	0

Data Collection:

We need a dataset containing medical features such as glucose levels, blood pressure, BMI, etc., along with information about whether the individual has diabetes or not.

Data Preprocessing:

The medical data needs to be cleaned, normalized, and prepared for training machine learning models.

Feature Selection:

We will select relevant features that can impact diabetes risk prediction.

Model Selection:

We can experiment with various machine learning algorithms like Logistic Regression, Random Forest, and Gradient Boosting.

Evaluation:

We will evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and ROC-AUC.

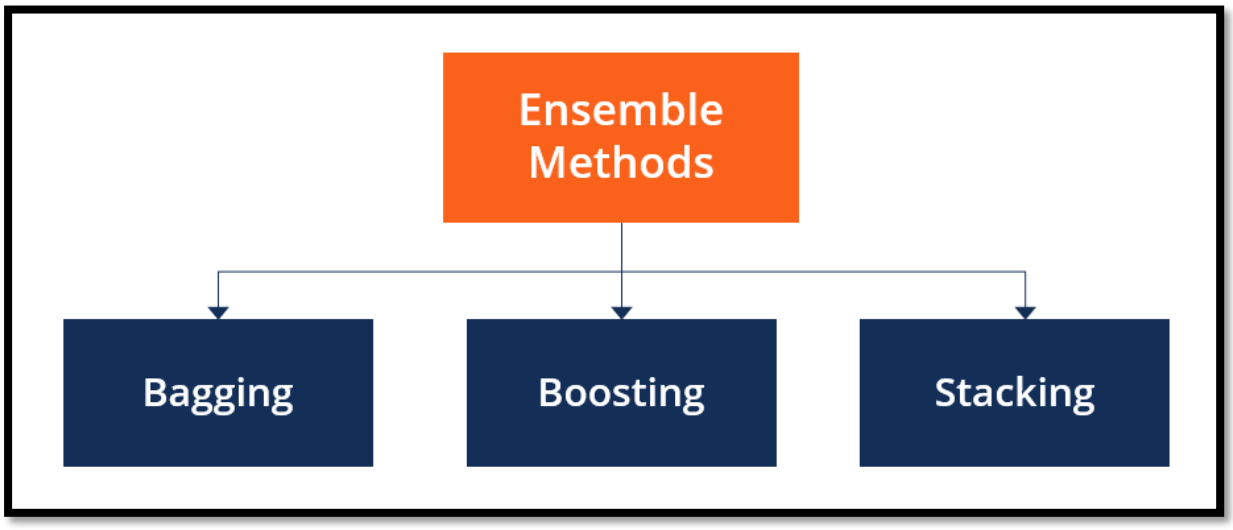
Iterative Improvement:

We will fine-tune the model parameters and explore techniques like feature engineering to enhance prediction accuracy.

Ensemble methods to improve the prediction system's accuracy

Ensemble methods are a type of machine learning technique that involve combining multiple models to improve the accuracy and robustness of predictions. Ensemble methods are typically used in situations where a single model may not be sufficient or where different models may have complementary strengths.

Main Types of Ensemble Methods



1. Bagging

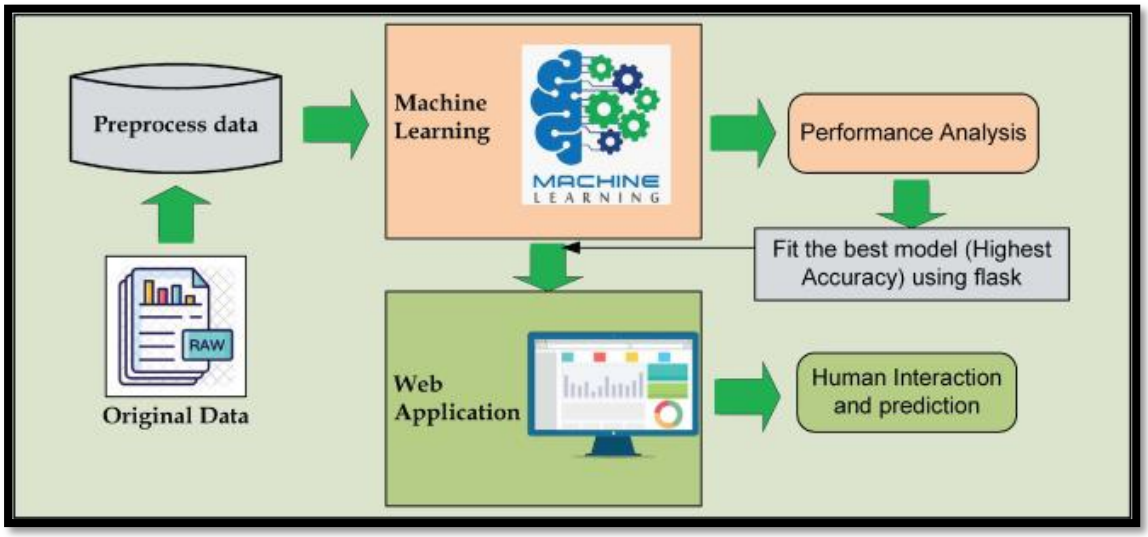
Bagging, the short form for bootstrap aggregating, is mainly applied in classification and regression. It increases the accuracy of models through decision trees, which reduces variance to a large extent. The reduction of variance increases accuracy, eliminating overfitting, which is a challenge to many predictive models.

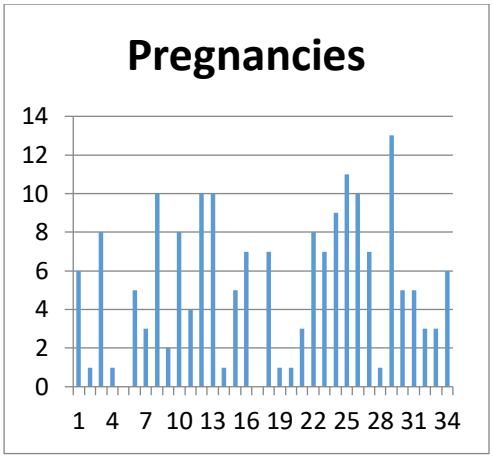
2. Boosting

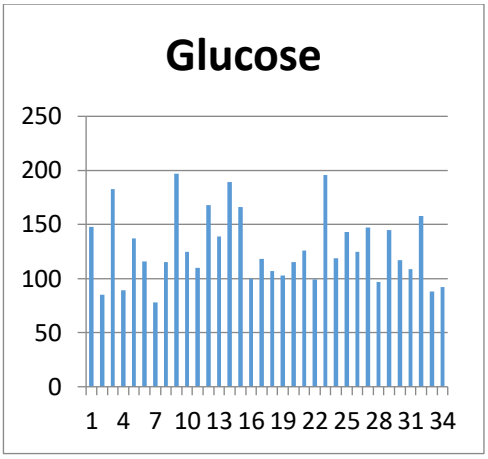
Boosting is an ensemble technique that learns from previous predictor mistakes to make better predictions in the future. The technique combines several weak base learners to form one strong learner, thus significantly improving the predictability of models. Boosting works by arranging weak learners in a sequence, such that weak learners learn from the next learner in the sequence to create better predictive models.

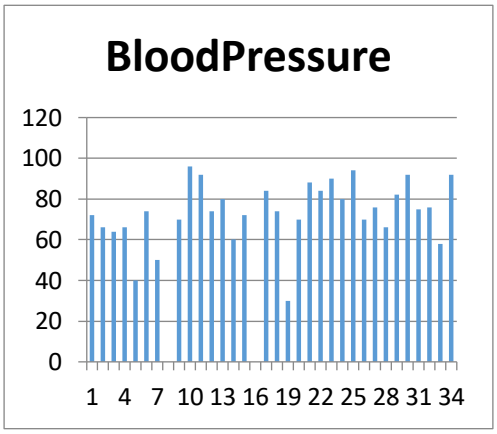
3. Stacking

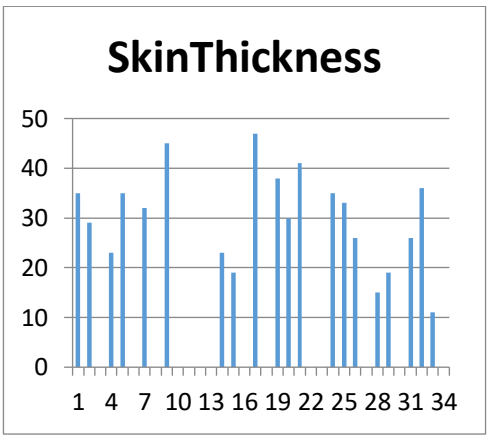
Stacking, another ensemble method, is often referred to as stacked generalization. This technique works by allowing a training algorithm to ensemble several other similar learning algorithm predictions. Stacking has been successfully implemented in regression, density estimations, distance learning, and classifications.

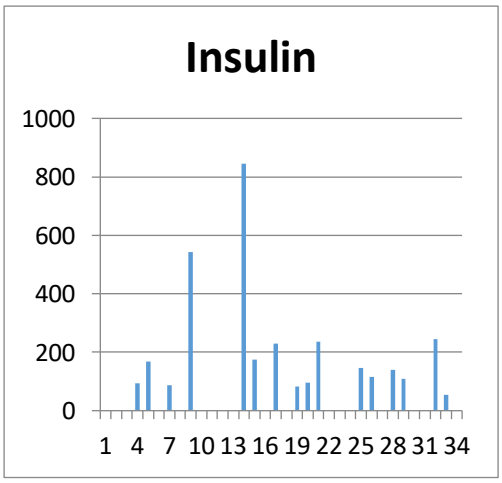


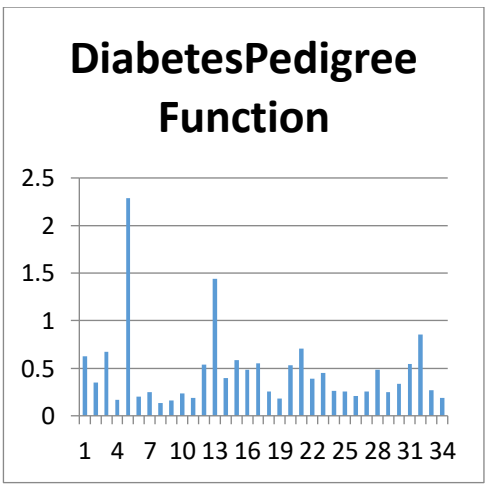

Pregnancies
14
12
10
8
6
4
2
0
1 4 7 10 13 16 19 22 25 28 31 34


Glucose
250
200
150
100
50
0
1 4 7 10 13 16 19 22 25 28 31 34


BloodPressure
120
100
80
60
40
20
0
1 4 7 10 13 16 19 22 25 28 31 34


SkinThickness
50
40
30
20
10
0
1 4 7 10 13 16 19 22 25 28 31 34


Insulin
1000
800
600
400
200
0
1 4 7 10 13 16 19 22 25 28 31 34


DiabetesPedigree Function
2.5
2
1.5
1
0.5
0
1 4 7 10 13 16 19 22 25 28 31 34

Program

AI Based Diabetes Prediction System

```
# Import libraries
import numpy as np # for linear algebra
import pandas as pd # for data processing, CSV file I/O (e.g. pd.read_cs
v)
import seaborn as sns # for data visualization
import matplotlib.pyplot as plt # to plot data visualization charts
from collections import Counter
import os

# Modeling Libraries
from sklearn.metrics import confusion_matrix, accuracy_score, precisio
n_score
from sklearn.preprocessing import QuantileTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClass
ifier, GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score,
StratifiedKFold, learning_curve, train_test_split
from sklearn.svm import SVC
```

Importing the Dataset

```
# Importing the dataset from Kaggle
data = pd.read_csv("../input/pima-indians-diabetes-
database/diabetes.csv")

# First step is getting familiar with the structure of the dataset
data.info()
```

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
```

Code

```
# Showing the top 5 rows of the dataset
data.head()
```

Output

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35	30.5	33.6	0.627	50	1
1	1	85.0	66.0	29	30.5	26.6	0.351	31	0
2	8	183.0	64.0	23	30.5	23.3	0.672	32	1
3	1	89.0	66.0	23	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35	168.0	43.1	2.288	33	1

Filling the Missing Values Code

```
# Exploring the missing values in the diabetes dataset
data.isnull().sum()
```

Output

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Code

```
# Replacing 0 values with the mean of that column

# Replacing 0 values of Glucose

data['Glucose'] = data['Glucose'].replace(0, data['Glucose'].median())

# Filling 0 values of Blood Pressure

data['BloodPressure'] = data['BloodPressure'].replace(0, data['BloodPress
ure'].median())

# Replacing 0 values in BMI

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

# Replacing the missing values of Insulin and SkinThickness

data['SkinThickness'] = data['SkinThickness'].replace(0, data['SkinThick
ness'].mean())

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

data.head()
```

Output

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35.000000	79.799479	33.6	0.627	50	1
1	1	85	66	29.000000	79.799479	26.6	0.351	31	0
2	8	183	64	20.536458	79.799479	23.3	0.672	32	1
3	1	89	66	23.000000	94.000000	28.1	0.167	21	0
4	0	137	40	35.000000	168.000000	43.1	2.288	33	1

Code

Reviewing the dataset statistics

data.describe()

Output

	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	121.656250	72.386719	26.606479	118.660163	32.450805	0.471876	33.240885	0.348958
std	3.369578	30.438286	12.096642	9.631241	93.080358	6.875374	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	79.799479	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

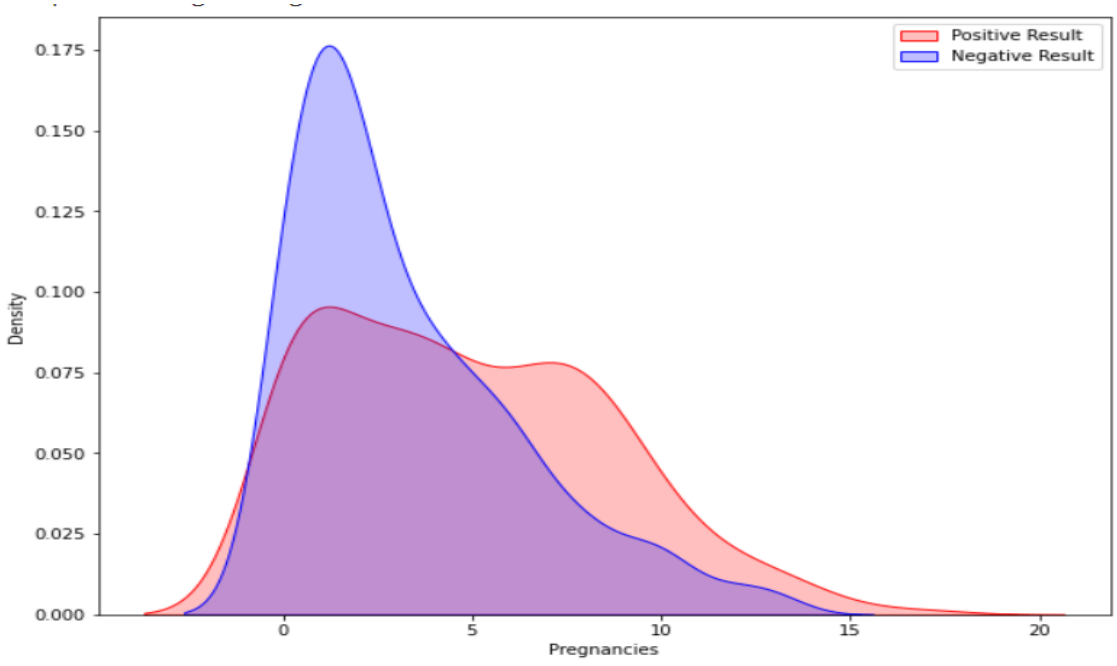
Pregnancy Code

Exploring Pregnancy and target variables together

plt.figure(figsize = (10, 8))

```
# Plotting density function graph of the pregnancies and the target variab
le

kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 1], color = "
Red", shade = True)

kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 0], ax = kde,
 color = "Blue", shade= True)

kde.set_xlabel("Pregnancies")

kde.set_ylabel("Density")

kde.legend(["Positive Result", "Negative Result"])
```

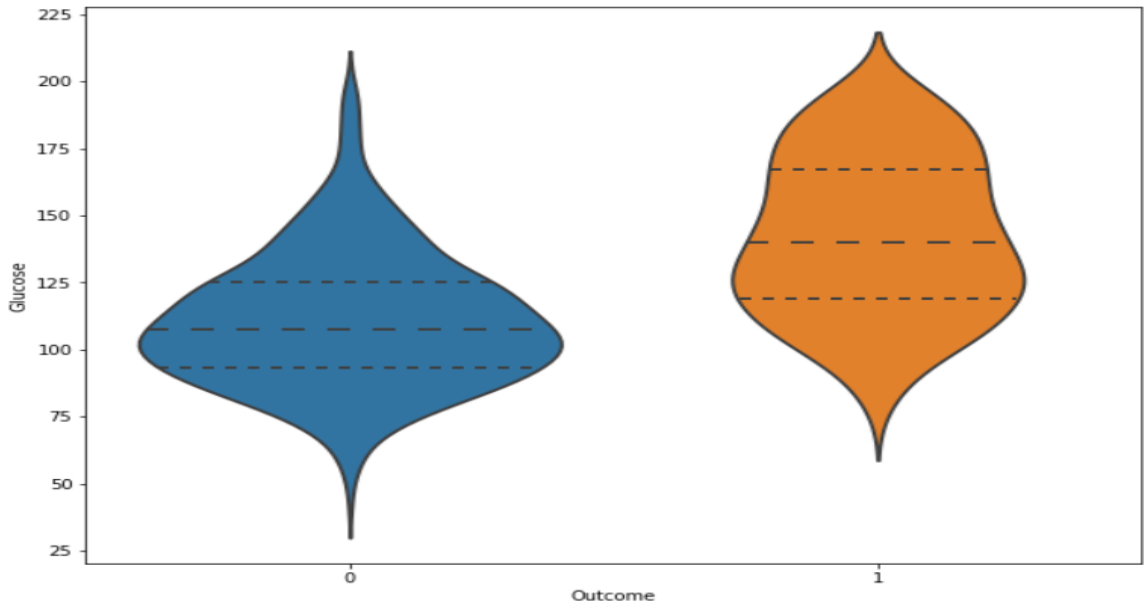
Output



Glucose code

```
# Exploring the Glucose and the Target variables together

plt.figure(figsize = (10, 8))

sns.violinplot(data = data, x = "Outcome", y = "Glucose",

        split = True, inner = "quart", linewidth = 2)
```

Output

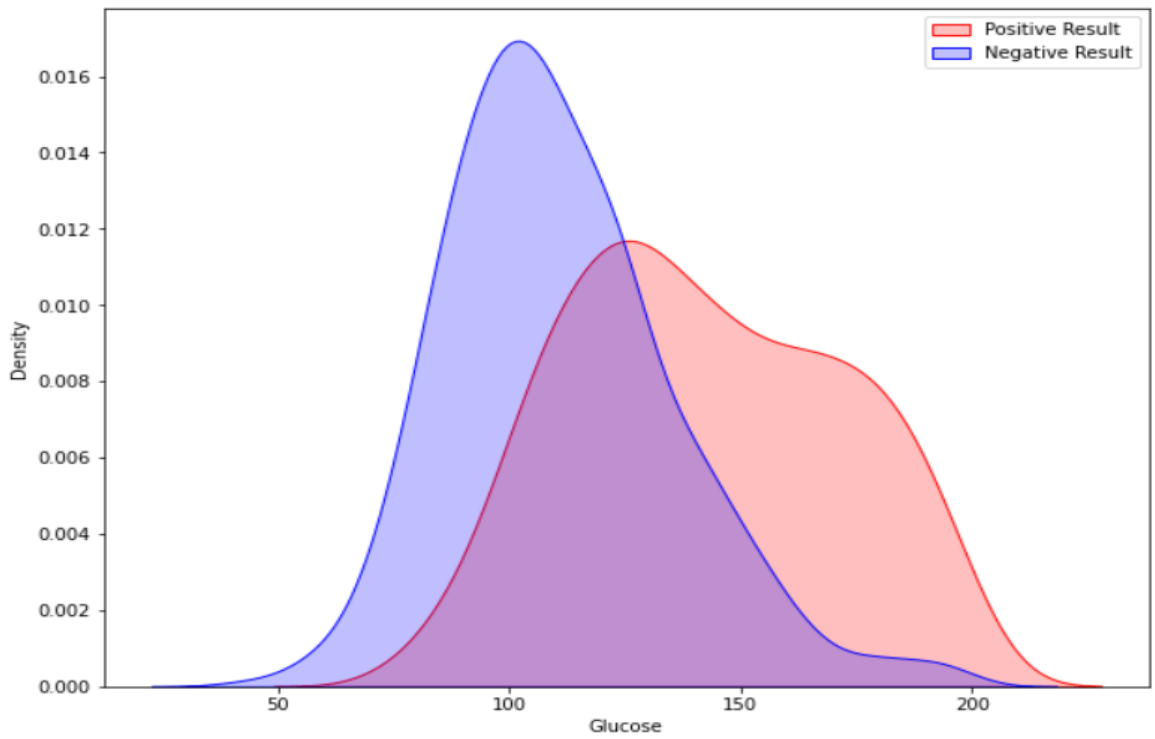


Code

```
# Exploring the density function plot of the Glucose levels

plt.figure(figsize = (10, 8))
```

```
kde = sns.kdeplot(data["Glucose"][data["Outcome"] == 1], color = "Red", shade = True)

kde = sns.kdeplot(data["Glucose"][data["Outcome"] == 0], ax = kde, color = "Blue", shade= True)

kde.set_xlabel("Glucose")

kde.set_ylabel("Density")

kde.legend(["Positive Result","Negative Result"])
```

Output



Future for diabetes prediction system

- Researchers are motivated to create a Machine Learning methodology that can predict diabetes in the future.
- Exploiting Machine Learning Algorithms (MLA) is essential if healthcare professionals are able to identify diseases more effectively.

Conclusions

- One of the risks during pregnancy is diabetes. It will have to be diagnosed to avoid problems.
- An increase in glucose levels is strongly correlated to a rise in diabetes.
- In this paper, an automatic diabetes prediction system using various machine learning approaches has been proposed. The open-source Pima Indian and a private dataset of female Bangladeshi patients have been used in this work.
- This research paper reported different performance metrics, that is, precision, recall, accuracy, F1 score, and AUC for various machine learning and ensemble techniques.
- There are some future scopes of this work, for example, we recommend getting additional private data with a larger cohort of patients to get better results.