AI BASED DIABETES PREDICTION SYSTEM

AU922321104040 : X.Simi Femina

PHASE : 1 Document Submission

project : DIABETES PREDITICTION

Objective:

The Objectives of this project is to develop a AI Model that accurately predicts the diabetes level based on a set of features such as insulin,BMI,blood pressure,glugose,age.

Title: AI-Based Diabetes Prediction System

Abstract:

Diabetes is a chronic medical condition that affects millions of people worldwide, leading to severe health complications if not managed properly. Early detection and prediction of diabetes risk are crucial for timely intervention and improved patient outcomes. This abstract presents an AI-based Diabetes Prediction System (DPS) comprising distinct modules to facilitate accurate and timely diabetes risk assessment.

Data Collection and Preprocessing

The first module of the DPS focuses on collecting and preprocessing data from various sources, including electronic health records, wearable devices, and patientreported information. Data preprocessing techniques such as data cleaning, normalization, and feature extraction are employed to ensure the quality and relevance of input data.

Program for diabetes prediction:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt import seaborn as sns
```

```
In [2]:
dataset=pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")
```

```
In [3]:
dataset.head() Out[3]:
```

	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
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
4	0	137	40	35	168	43.1	2.288	33	1

```
In [4]: dataset.shape
```

```
Out[4]:
(768, 9)
```

```
In [5]:
#check if null value is present
dataset.isnull().values.any()
```

```
Out[5]: False
```

```
In [6]: dataset.info()
```

```
<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
```

```
In [7]:
dataset.describe() Out[7]:
```

	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
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

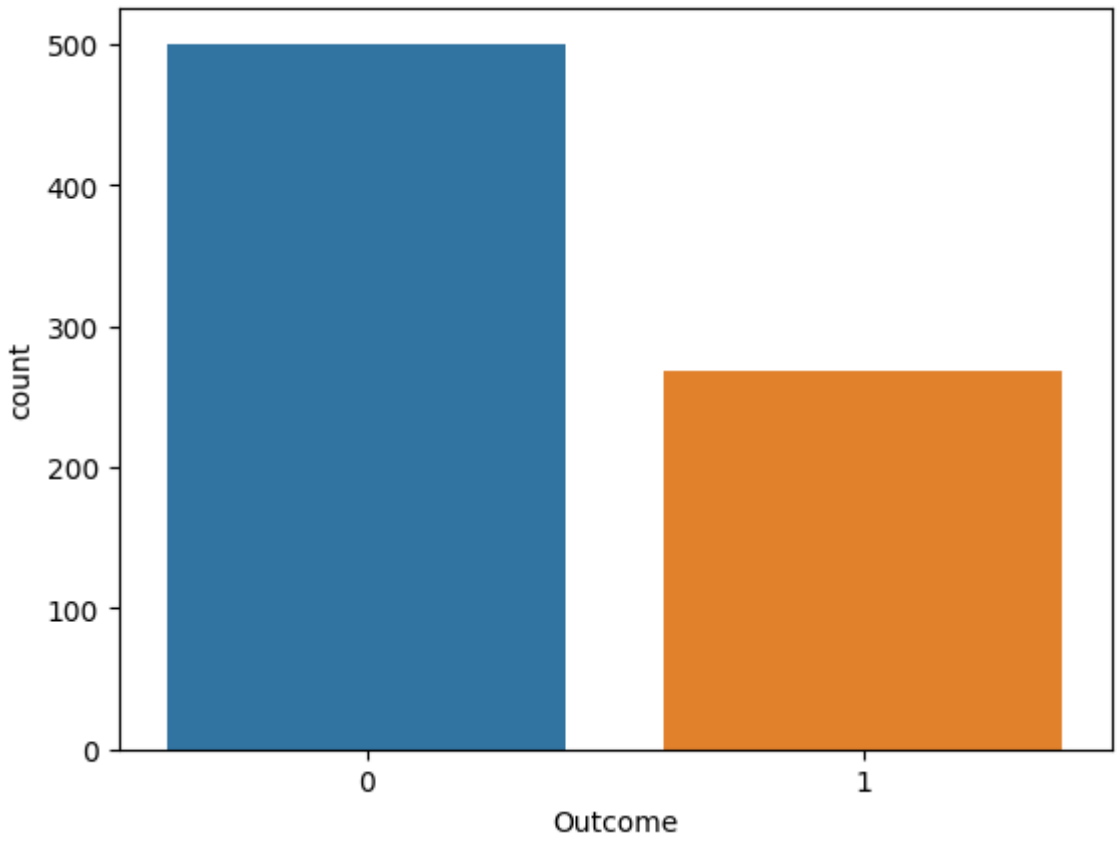
Max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [8]:
```
dataset.isnull().sum()
```

Out[8]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0 dtype:
int64
```

In [9]:
```
#data visualization sns.countplot(x =
'Outcome',data = dataset)
```

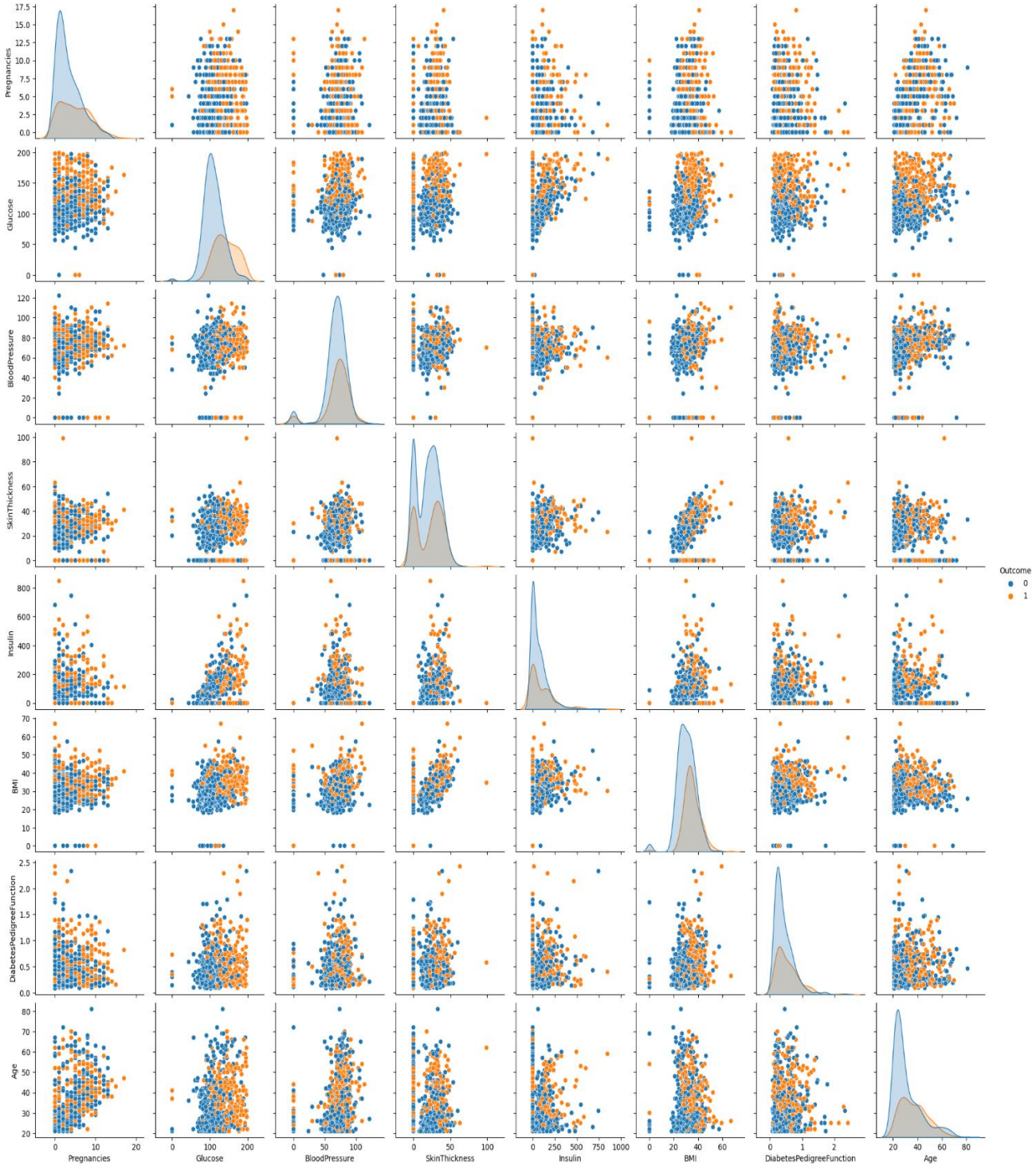
Out[9]:
```
<Axes: xlabel='Outcome', ylabel='count'>
```



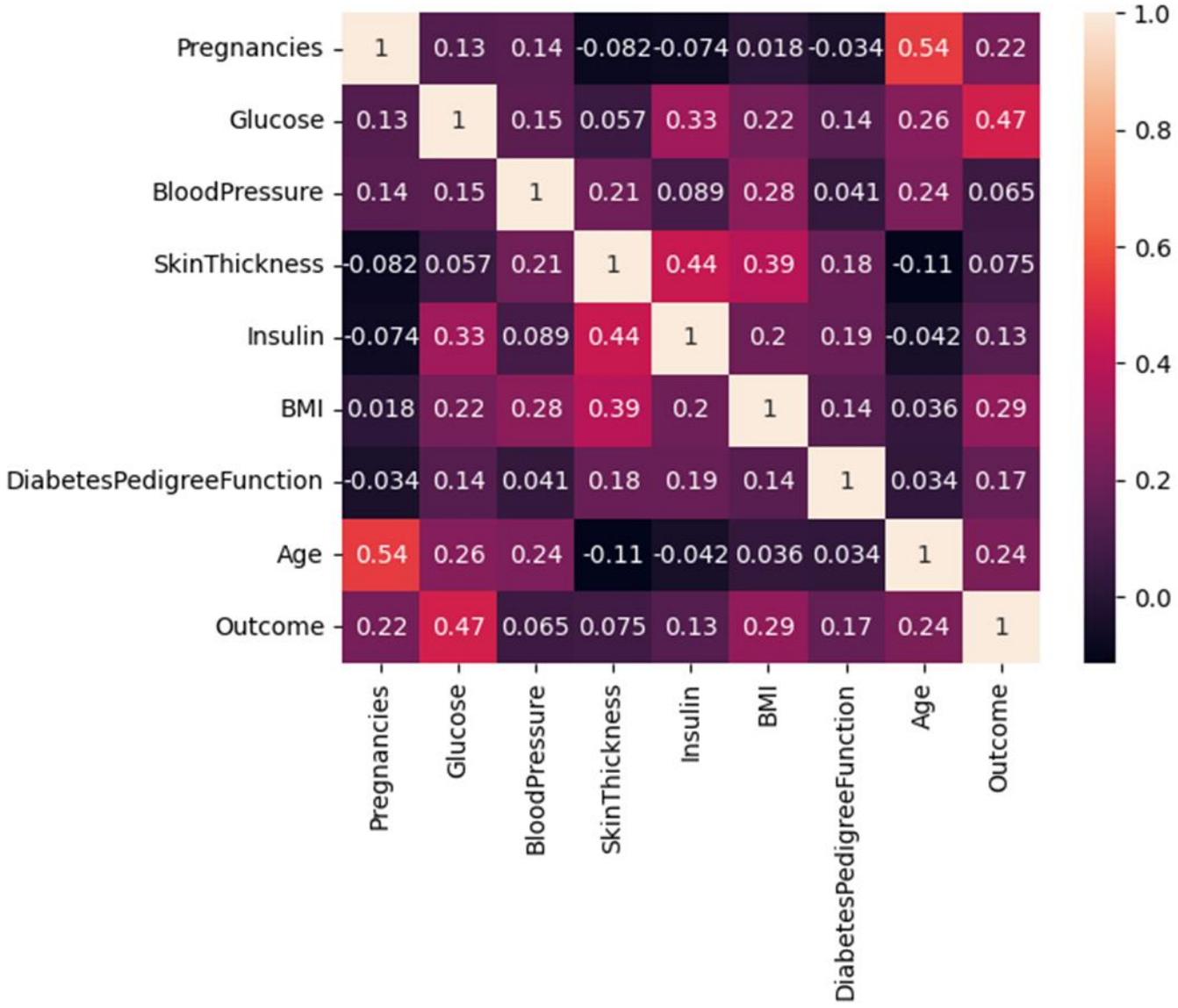
In [10]:

```
# Pairplot
sns.pairplot(data = dataset, hue = 'Outcome') plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning:
The figure layout has changed to tight   self._figure.tight_layout(*args,
**kwargs)  ____
```



```
In [11]:
# Heatmap
sns.heatmap(dataset.corr(), annot = True) plt.show()
```



In [12]:

```
# Replacing zero values with NaN dataset_new
= dataset
dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = dat
aset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].replace(
0, np.NaN)
```

In [13]:

```
# Count of NaN dataset_new.isnull().sum()
```

Out[13]:

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0 dtype:
int64
```

In [14]:

```
# Replacing NaN with mean values
dataset_new["Glucose"].fillna(dataset_new["Glucose"].mean(), inplace = True)
dataset_new["BloodPressure"].fillna(dataset_new["BloodPressure"].mean(), inplace =
True)
dataset_new["SkinThickness"].fillna(dataset_new["SkinThickness"].mean(), inplace =
True)
dataset_new["Insulin"].fillna(dataset_new["Insulin"].mean(), inplace = True)
dataset_new["BMI"].fillna(dataset_new["BMI"].mean(), inplace = True)
```

In [15]:

```
dataset_new.isnull().sum()
```

Out[15]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64 In [16]:
#Logistic regression y =
dataset_new['Outcome']
X = dataset_new.drop('Outcome', axis=1)
```

In [17]:

```
# Splitting X and Y
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.20, random
_state = 42, stratify = dataset_new['Outcome'] )
```

In [18]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression() model.fit(X_train,
Y_train) y_predict = model.predict(X_test)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html Please
also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
```

In [19]: y_predict

Out[19]:

```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 0, 1,        0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0])
```

In [20]:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_predict) cm
```

Out[20]:

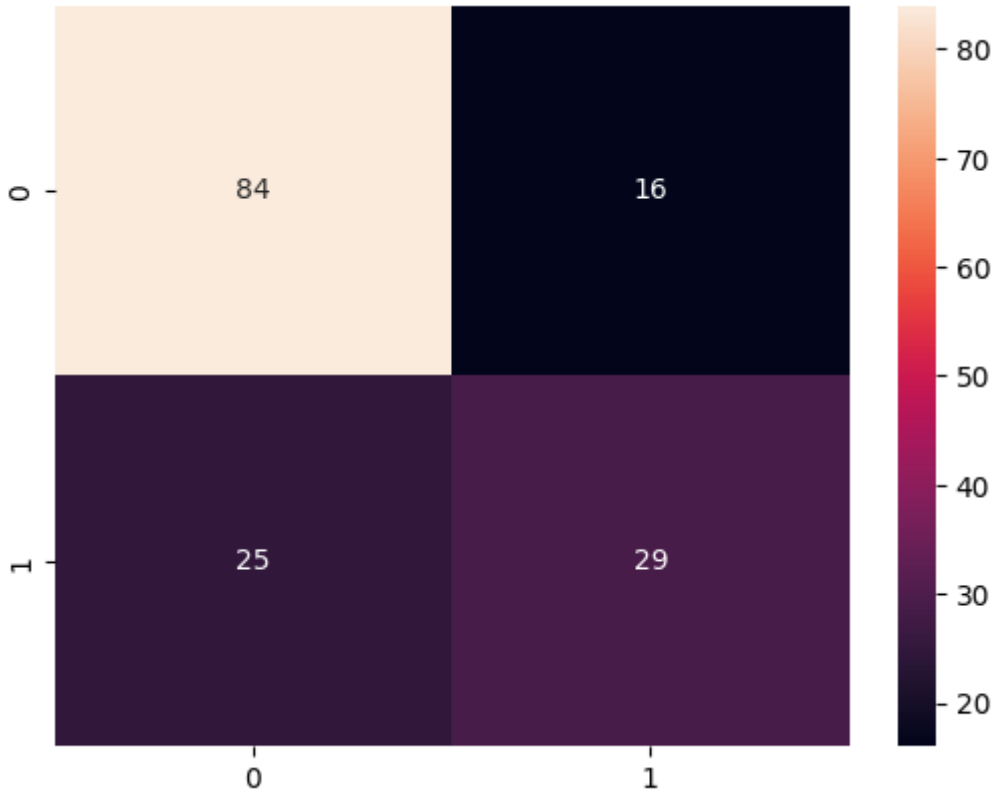
```
array([[84, 16],
       [25, 29]])
```

In [21]:

```
# Heatmap of Confusion matrix sns.heatmap(pd.DataFrame(cm),
annot=True)
```

Out[21]:

```
<Axes: >
```



In [22]:

```
from sklearn.metrics import accuracy_score
```

In [23]:

```
accuracy =accuracy_score(Y_test,
y_predict) accuracy
```

Out[23]:

```
0.7337662337662337
```

In [24]:

```
#Example: Let's check whether the person have diabetes or not using some random va
lues
y_predict = model.predict([[1,148,72,35,79.799,33.6,0.627,50]])
print(y_predict) if y_predict==1:
    print("Diabetic")
else:     print("Non
Diabetic")
```

```
[1]
Diabetic
/opt/conda/lib/python3.10/site-packages/sklearn/base.py:439: UserWarning: X do
es not have valid feature names, but LogisticRegression was fitted with featur
e names   warnings.warn(
```

Feature Selection and Engineering

This module aims to identify the most relevant features for diabetes prediction using advanced feature selection and engineering techniques. Machine learning algorithms are leveraged to reduce dimensionality and enhance the predictive power of the model.

Model Development

The core of the DPS lies in its machine learning and deep learning models. Various algorithms and neural network architectures are explored to develop accurate and robust prediction models. These models are trained on historical patient data, taking into account the selected features and target variables, to learn complex patterns and relationships.

Evaluation and Validation

The performance of the developed prediction models is rigorously evaluated using cross-validation, statistical metrics, and real-world patient data. Metrics such as accuracy, sensitivity, specificity, and AUC-ROC are used to assess the models' effectiveness in diabetes risk prediction.

User Interface and Accessibility

To make the system user-friendly, a user interface (UI) is designed to allow healthcare professionals and individuals to interact with the AI-based DPS easily. The UI provides a platform for entering patient data, viewing predictions, and accessing personalized recommendations.

Decision Support and Recommendations

The DPS not only predicts diabetes risk but also provides actionable recommendations to mitigate the risk. These recommendations may include lifestyle modifications, dietary changes, exercise routines, and regular check-ups. The system customizes these recommendations based on individual patient profiles.

Continuous Learning and Updates

The DPS is designed for continuous improvement. It utilizes feedback from healthcare professionals and patients to enhance its prediction accuracy and recommendation effectiveness over time. Regular updates and model retraining ensure that the system remains relevant and up-to-date.

CONCLUSION

In conclusion, the AI-based Diabetes Prediction System presented in this abstract integrates multiple modules to enable accurate diabetes risk assessment, userfriendly interaction, and personalized recommendations. By harnessing the power of artificial intelligence, this system contributes to early diabetes detection and improved patient care, ultimately promoting better health outcomes for individuals at risk of diabetes.