Deep Neural Networks

Project Report

DIABETES PREDICTION SYSTEM

Introduction : Diabetes is a chronic health condition characterized by high blood sugar levels that can lead to a range of complications if left untreated. The early detection of diabetes can prevent or delay the onset of these complications. To this end, various predictive models and algorithms have been developed to predict the risk of diabetes in individuals. In this report, we will discuss the diabetes prediction system, which is designed to predict the risk of diabetes using machine learning algorithms.

Overview : The diabetes prediction system is a machine learning-based system that predicts the risk of diabetes in individuals based on their medical history and other risk factors. The system uses various machine learning algorithms to analyze the input data and generate a prediction.

Data Collection : The first step in developing a diabetes prediction system is to collect data. The data collected typically includes medical history, family history, lifestyle factors, and other risk factors that may contribute to the development of diabetes. The data is collected from electronic health records, surveys, and other sources.

Feature Selection : Once the data is collected, the next step is to select the relevant features or risk factors that will be used to predict the risk of diabetes. The features selected may vary depending on the study and the available data. Some common features that are used in diabetes prediction models include age, body mass index (BMI), blood pressure, cholesterol levels, and fasting glucose levels.

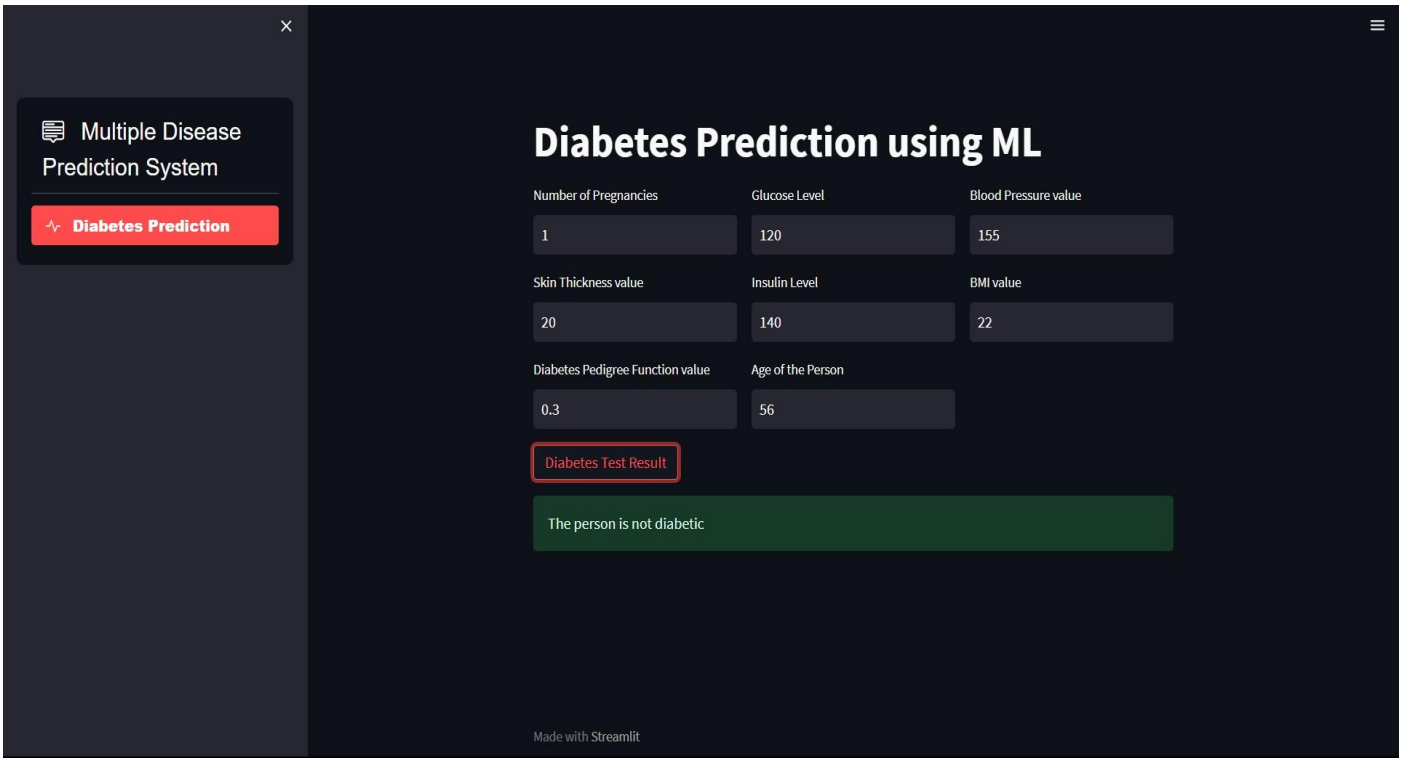
Model Development : Once the features are selected, the next step is to develop a predictive model. There are various machine learning algorithms that can be used for diabetes prediction, including logistic regression, decision trees, and neural networks. The selected algorithm is trained on the available data, and the model's performance is evaluated using various metrics such as accuracy, precision, recall, and F1 score.

Deployment : Once the model is developed and evaluated, it is deployed in a clinical setting or a web-based application, where it can be used to predict the risk of diabetes in individuals. The user inputs their medical history and other risk factors, and the system generates a prediction.

Brief : This report provides an overview of the diabetes prediction system, which is a machine learning-based system designed to predict the risk of diabetes in individuals. The report covers the various stages involved in developing a diabetes prediction system, including data collection, feature selection, model development, and deployment. The report highlights the importance of selecting relevant features and collecting high-quality data for developing an accurate predictive model. The diabetes prediction system can be a useful tool for identifying individuals at high risk of developing diabetes and enabling early intervention to prevent or delay the onset of complications.

Project Overview :

GUI and Code :




```python
import pickle
import streamlit as st
from streamlit_option_menu import option_menu


# loading the saved models

diabetes_model = pickle.load(open("C:\\Users\\win10\\OneDrive\\Desktop\\DNN Project\\saved models\\diabetes_model.sav", "rb"))

# heart_disease_model = pickle.load(open("C:\\Users\\win10\\OneDrive\\Desktop\\DNN Project\\saved models\\heart_disease_model.sav","rb"))

# parkinsons_model = pickle.load(open("C:\\Users\\win10\\OneDrive\\Desktop\\DNN Project\\saved models\\parkinsons_model.sav", "rb"))



# sidebar for navigation
with st.sidebar:

    selected = option_menu("Multiple Disease Prediction System",
                           ["Diabetes Prediction"],
                           icons=["activity"],
                           default_index=0)


# Diabetes Prediction Page
if (selected == "Diabetes Prediction"):

    # page title
    st.title("Diabetes Prediction using ML")


    # getting the input data from the user
    col1, col2, col3 = st.columns(3)

    with col1:
        Pregnancies = st.text_input("Number of Pregnancies")

    with col2:
        Glucose = st.text_input("Glucose Level")
```

```
with col2:
    Glucose = st.text_input("Glucose Level")

with col3:
    BloodPressure = st.text_input("Blood Pressure value")

with col1:
    SkinThickness = st.text_input("Skin Thickness value")

with col2:
    Insulin = st.text_input("Insulin Level")

with col3:
    BMI = st.text_input("BMI value")

with col1:
    DiabetesPedigreeFunction = st.text_input("Diabetes Pedigree Function value")

with col2:
    Age = st.text_input("Age of the Person")


# code for Prediction
diab_diagnosis = ""

# creating a button for Prediction

if st.button("Diabetes Test Result"):
    diab_prediction = diabetes_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness,
                                               Insulin, BMI, DiabetesPedigreeFunction, Age]])

    if (diab_prediction[0] == 1):
      diab_diagnosis = "The person is diabetic"
    else:
      diab_diagnosis = "The person is not diabetic"

st.success(diab_diagnosis)
```

The Dataset :

	A	B	C	D	E	F	G	H	I	J
1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	
2	6	148	72	35	0	33.6	0.627	50	1	
3	1	85	66	29	0	26.6	0.351	31	0	
4	8	183	64	0	0	23.3	0.672	32	1	
5	1	89	66	23	94	28.1	0.167	21	0	
6	0	137	40	35	168	43.1	2.288	33	1	
7	5	116	74	0	0	25.6	0.201	30	0	
8	3	78	50	32	88	31	0.248	26	1	
9	10	115	0	0	0	35.3	0.134	29	0	
10	2	197	70	45	543	30.5	0.158	53	1	
11	8	125	96	0	0	0	0.232	54	1	
12	4	110	92	0	0	37.6	0.191	30	0	
13	10	168	74	0	0	38	0.537	34	1	
14	10	139	80	0	0	27.1	1.441	57	0	
15	1	189	60	23	846	30.1	0.398	59	1	
16	5	166	72	19	175	25.8	0.587	51	1	
17	7	100	0	0	0	30	0.484	32	1	
18	0	118	84	47	230	45.8	0.551	31	1	
19	7	107	74	0	0	29.6	0.254	31	1	
20	1	103	30	38	83	43.3	0.183	33	0	
21	1	115	70	30	96	34.6	0.529	32	1	
22	3	126	88	41	235	39.3	0.704	27	0	
23	8	99	84	0	0	35.4	0.388	50	0	
24	7	196	90	0	0	39.8	0.451	41	1	
25	9	119	80	35	0	29	0.263	29	1	
26	11	143	94	33	146	36.6	0.254	51	1	
27	10	125	70	26	115	31.1	0.205	41	1	
28	7	147	76	0	0	39.4	0.257	43	1	
29	1	97	66	15	140	23.2	0.487	22	0	
30	13	145	82	19	110	22.2	0.245	57	0	

diabetes

Diabetes Using NN code :

+ Code + Markdown | Run All | Clear All Outputs | Outline ··· Python 3.11.1

```
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
```

[28] Python

```
diabetes = pd.read_csv("C:\\Users\\win10\\OneDrive\\Desktop\\DNN Project\\datasets\\diabetes.csv")
diabetes
```

[22] Python

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

```
diabetes.shape
```

[42] Pyth

```
(768, 9)
```

```
diabetes['Outcome'].value_counts()
```

[43] Pyth

```
0    500
1    268
Name: Outcome, dtype: int64
```

```
diabetes.groupby('Outcome').mean()
```

[44] Pyth

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

```python
# Split the dataset into training, validation, and test sets
train_data = diabetes.sample(frac=0.8, random_state=0)
val_data = diabetes.drop(train_data.index).sample(frac=0.5, random_state=0)
test_data = diabetes.drop(train_data.index).drop(val_data.index)

# Split features and labels
train_labels = train_data.pop('Outcome')
val_labels = val_data.pop('Outcome')
test_labels = test_data.pop('Outcome')
```

[6] Python

```python
# Define the logistic regression model
model = keras.Sequential([
    keras.layers.Dense(units=1, input_shape=[8], activation='sigmoid')
])
```

[7] Python

```python
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

[8] Python

```python
# Training the model
model.fit(train_data, train_labels, epochs=256, validation_data=(val_data, val_labels),verbose=1)
```

[23] Python

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/256
20/20 [==============================] - 0s 12ms/step - loss: 0.5532 - accuracy: 0.7248 - val_loss: 0.5651 - val_accuracy: 0.6883
Epoch 2/256
20/20 [==============================] - 0s 8ms/step - loss: 0.5520 - accuracy: 0.7362 - val_loss: 0.5757 - val_accuracy: 0.7013
Epoch 3/256
20/20 [==============================] - 0s 6ms/step - loss: 0.5523 - accuracy: 0.7410 - val_loss: 0.5862 - val_accuracy: 0.7013
Epoch 4/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5632 - accuracy: 0.7248 - val_loss: 0.5740 - val_accuracy: 0.6753
Epoch 5/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5511 - accuracy: 0.7248 - val_loss: 0.5656 - val_accuracy: 0.7143
Epoch 6/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5552 - accuracy: 0.7166 - val_loss: 0.5898 - val_accuracy: 0.6753
Epoch 7/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5651 - accuracy: 0.7345 - val_loss: 0.5721 - val_accuracy: 0.6753
Epoch 8/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5537 - accuracy: 0.7296 - val_loss: 0.5721 - val_accuracy: 0.7143
Epoch 9/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5544 - accuracy: 0.7362 - val_loss: 0.5749 - val_accuracy: 0.7013
Epoch 10/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5537 - accuracy: 0.7215 - val_loss: 0.5687 - val_accuracy: 0.7143
Epoch 11/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5551 - accuracy: 0.7280 - val_loss: 0.5866 - val_accuracy: 0.6883
Epoch 12/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5621 - accuracy: 0.7231 - val_loss: 0.5847 - val_accuracy: 0.7143
Epoch 13/256
...
Epoch 57/256
20/20 [==============================] - 0s 6ms/step - loss: 0.5495 - accuracy: 0.7296 - val_loss: 0.5754 - val_accuracy: 0.6753
Epoch 58/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5569 - accuracy: 0.7345 - val_loss: 0.5793 - val_accuracy: 0.7143
Output exceeds the size limit. Open the full output data in a text editor
Epoch 59/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5439 - accuracy: 0.7476 - val_loss: 0.5571 - val_accuracy: 0.6883
Epoch 60/256
20/20 [==============================] - 0s 7ms/step - loss: 0.5466 - accuracy: 0.7248 - val_loss: 0.5640 - val_accuracy: 0.7013
```

```
# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(test_data, test_labels)
print('Test accuracy:', test_acc)
print('Test Loss:',test_loss)
```

[26]

```
3/3 [==============================] - 0s 8ms/step - loss: 0.5202 - accuracy: 0.7532
Test accuracy: 0.7532467246055603
Test Loss: 0.5201760530471802
```

```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] < 0.5):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

[36]

```
1/1 [==============================] - 0s 63ms/step
[[0.52745056]]
The person is diabetic
```

```
import pickle
```

[37]

```
filename = 'diabetes_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

[39]

```
loaded_model = pickle.load(open('diabetes_model.sav', 'rb'))
```

[40]

```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = loaded_model.predict(input_data_reshaped)
print(prediction)

if (prediction[0] < 0.5):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

[41]

```
1/1 [==============================] - 0s 94ms/step
[[0.52745056]]
The person is diabetic
```

Conclusion : The diabetes prediction system is an important tool for predicting the risk of diabetes in individuals. It can help identify individuals who are at high risk of developing diabetes and enable early intervention to prevent or delay the onset of complications. However, like all machine learning models, the accuracy of the diabetes prediction system depends on the quality and quantity of the input data. Therefore, it is essential to collect high-quality data and select the relevant features for developing an accurate predictive model.