Create a diabetes prediction in python

Phase 3: Development part-1

Objective:

In this phase, I am focusing on building my diabetes by loading and preprocessing the dataset with the help of machine learning techniques.

Explanation of Data Loading Code in Steps:

Step 1: **Data collection**

The very first step is to choose the dataset for our model. We can get a lot of different datasets from Kaggle. You just need to sign in to Kaggle and search for any dataset you need for the project. The Diabetes dataset required for our model can be downloaded here.

Step 2: **Exploring the Data**

Now we have to set the development environment to build our project. For this project, we are going to build this Diabetes prediction using Machine Learning in Google Colab. You can also use Jupyter Notebook.

Step 3: **Splitting the data**

The next step in the building of the Machine learning model is splitting the data into training and testing sets. The training and testing data should be split in a ratio of 3:1 for better prediction results.

Step 4: **Training the model**

The next step is to build and train our model. We are going to use a Support vector classifier algorithm to build our model.

Step 5: **Evaluating the model**

Evaluating the model using python code.

Project code for Data Preprocessing:

```
import numpy as np

import pickle

import streamlit as st


# Load the saved model

loaded_model = pickle.load(open('C:/Users/ELCOT/Downloads/trained_model.sav', 'rb'))


# Create a function for Prediction

def diabetes_prediction(input_data):


    # Change 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):
      return 'The person is not diabetic'
    else:
      return 'The person is diabetic'

def main():

    # Give a title
    st.title('Diabetes Prediction Web App')

    # To get the input data from the user
    Pregnancies = st.text_input('Number of Pregnancies')
    Glucose = st.text_input('Glucose Level')
    BloodPressure = st.text_input('Blood Pressure value')
    SkinThickness = st.text_input('Skin Thickness value')
    Insulin = st.text_input('Insulin Level')
    BMI = st.text_input('BMI value')
    DiabetesPedigreeFunction = st.text_input('Diabetes Pedigree Function value')
    Age = st.text_input('Age of the Person')


    # Code for Prediction
    diagnosis = ''


    # Create a button for Prediction


    if st.button('Diabetes Test Result'):
        diagnosis = diabetes_prediction([Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age])


    st.success(diagnosis)


if __name__ == '__main__':
```

```
main()
```

Explanation of Data preprocessing code in Steps:

Step 1: Missing Observation Analysis

We saw on df.head() that some features contain 0, it doesn't make sense here and this indicates missing value Below we replace 0 value by NaN:

Step 2: Outlier Observation Analysis

```
Q1 = df[feature].quantile(0.25)
Q3 = df[feature].quantile(0.75)
IQR = Q3-Q1
lower = Q1- 1.5*IQR
upper = Q3 + 1.5*IQR

if df[(df[feature] > upper)].any(axis=None):
    print(feature,"yes")
else:
    print(feature, "no")
```

Step 3: Local Outlier Factor (LOF)

```
from sklearn.neighbors import LocalOutlierFactor
lof =LocalOutlierFactor(n_neighbors= 10)
lof.fit_predict(df)
```