AI\_Phase4

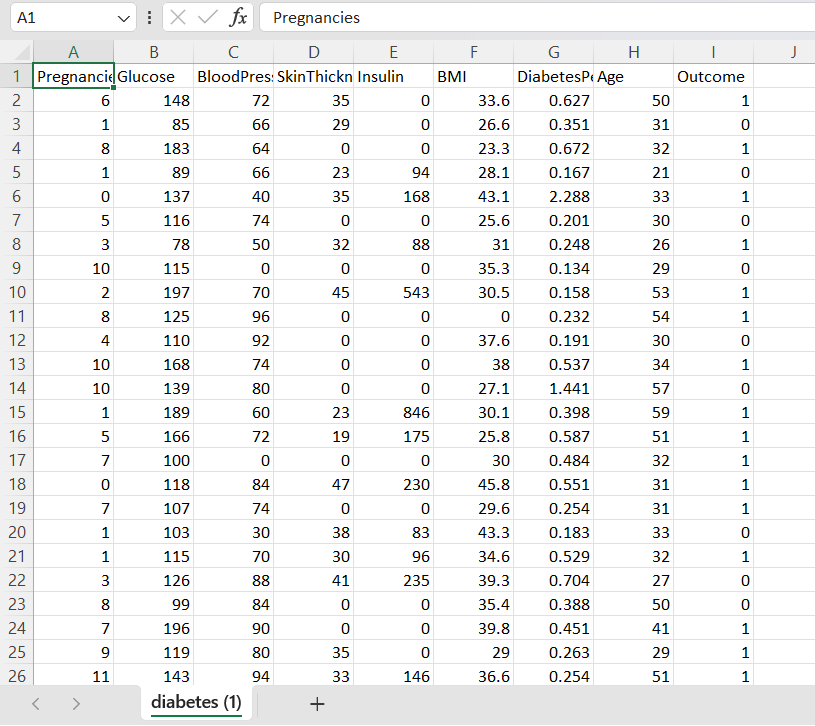
AI- Based diabetes

Prediction system

Development part 2

Document submission

DATASET



To build an AI-based diabetes prediction system, we can use a machine learning algorithm such as Logistic Regression or Random Forest.

Here is a step-by-step guide to building and evaluating the performance of the model:

1: Import Libraries and Load Dataset

python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

# Load the diabetes dataset (replace 'diabetes.csv' with your dataset filename)

data = pd.read\_csv('diabetes.csv')

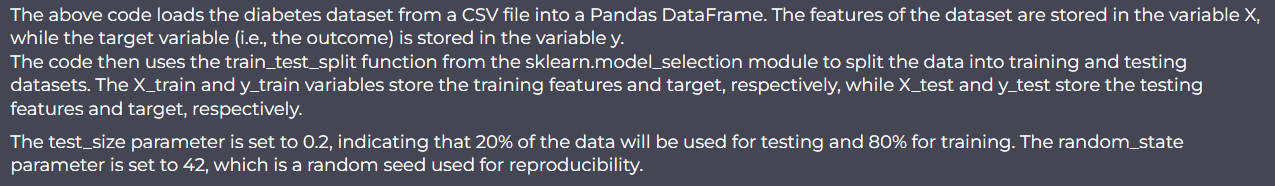
# Split the data into features (X) and target (y)

X = data.drop('Outcome', axis=1)

y = data['Outcome']

# Split the data into training and testing datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)



2: Train and Evaluate the Model

python

from sklearn.linear\_model import LogisticRegression

# Create an instance of the Logistic Regression model

model = LogisticRegression()

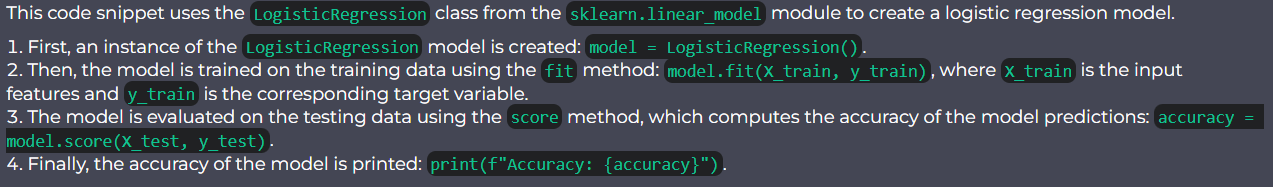
# Train the model on the training data

model.fit(X\_train, y\_train)

# Evaluate the model on the testing data

accuracy = model.score(X\_test, y\_test)

print(f"Accuracy: {accuracy}")



3: Perform Analysis

Depending on the specific requirements and analysis needed, you can incorporate different additional steps. For example, you could perform feature engineering, hyperparameter tuning, or data preprocessing techniques.

Here is an example of hyperparameter tuning using GridSearchCV:

python

from sklearn.model\_selection import GridSearchCV

# Create a parameter grid for hyperparameter tuning

param\_grid = {'C': [0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}

# Create an instance of the Logistic Regression model

model = LogisticRegression()

# Create a GridSearchCV object

grid\_search = GridSearchCV(model, param\_grid, cv=5)

# Train the model on the training data using grid search

grid\_search.fit(X\_train, y\_train)

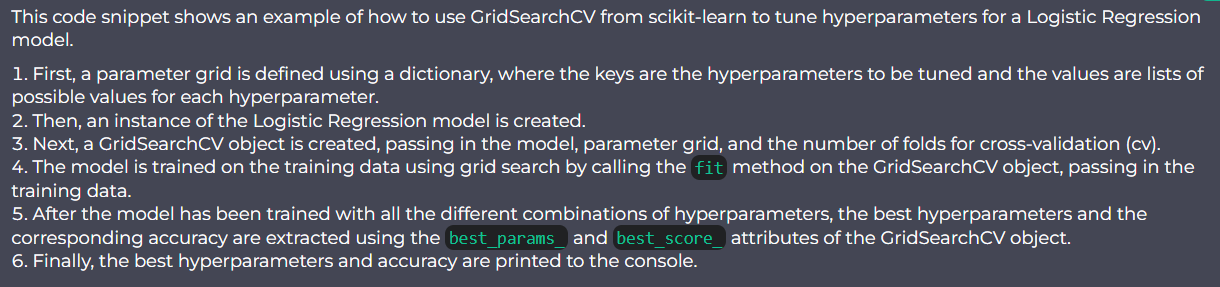
# Get the best hyperparameters and model accuracy

best\_params = grid\_search.best\_params\_

best\_accuracy = grid\_search.best\_score\_

print(f"Best Parameters: {best\_params}")

print(f"Best Accuracy: {best\_accuracy}")



4: Further Evaluation and Interpretation

You can also perform further evaluation and interpretation of the model's performance, such as calculating precision, recall, and F1-score, or generating a confusion matrix.

python

from sklearn.metrics import classification\_report, confusion\_matrix

# Evaluate the model on the testing data

y\_pred = model.predict(X\_test)

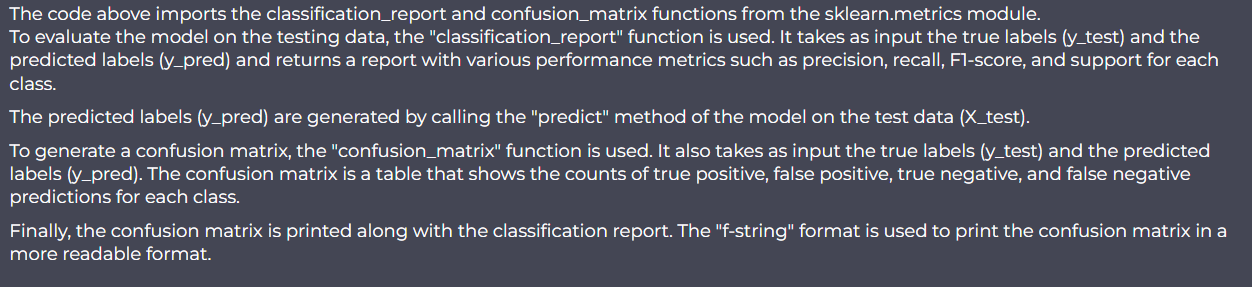
print(classification\_report(y\_test, y\_pred))

# Generate a confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(f"Confusion Matrix:

{cm}")



This is a basic outline for building an AI-based diabetes prediction system. Depending on the specific dataset and requirements, you may need to modify the steps accordingly.