HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

_____ o0o _____

BÁO CÁO THỰC TẬP CƠ SỞ

Bài tập số 1

NHÓM 30

Trần Đức Trung MSSV: B22DCCN875

Giảng viên hướng dẫn: PGS. TS. Trần Đình Quế

HÀ NỘI, 03/2025

MỤC LỤC

CHƯƠNG 1. VIẾT TÀI LIỆU

1.1 Kỹ thuật Cleaning data

Trong học máy (Machine learning), việc làm sạch dữ liệu (Cleaning Data) là một bước quan trọng để đảm bảo chất lượng dữ liệu đầu vào, từ đó cải thiện hiệu suất của mô hình. Dữ liệu "bẩn" (chứa lỗi, thiếu sót, hoặc không nhất quán) có thể dẫn đến kết quả dự đoán sai lệch hoặc mô hình không hoạt động tốt.

1.1.1 Hiểu dữ liệu ban đầu

Trước khi làm sạch, cần phân tích dữ liệu để hiểu cấu trúc, định dạng và các vấn đề tiềm ẩn:

- Xem xét tổng quan: Sử dụng các công cụ như pandas trong Python để kiểm tra dữ liệu (df.head(), df.info(), df.describe()).
- Xác định các vấn đề: Tìm kiếm giá trị thiếu, giá trị ngoại lai (outliers), dữ liệu trùng lặp, định dạng không nhất quán, hoặc nhiễu.
- Xác định kiểu dữ liệu: Đảm bảo các cột có kiểu dữ liệu phù hợp (ví dụ: số, chuỗi, ngày tháng).

1.1.2 Xử lý giá trị null (Missing values)

Giá trị null là vấn đề phổ biến trong dữ liệu thực tế.Việc đầu tiên cần làm là kiểm tra xem có giá trị null (Missing Values) nào trong bộ dữ liệu hay không. Hàm isnull().sum() trong Pandas được sử dụng để thống kê số lượng giá trị null trong mỗi cột. Các phương pháp xử lý bao gồm:

- Loại bỏ giá trị null:
  - Nếu tỷ lệ giá trị null nhỏ (dưới 5-10
  - Trong Azure Machine Learning Studio (MAML), module "Clean Missing Data" có thể được sử dụng để loại bỏ các hàng chứa giá trị null.
  - Chỉ áp dụng khi việc xóa không làm mất thông tin quan trọng.
- Thay thế giá trị null:
  - Giá trị trung bình/trung vị/mốt: Thay bằng giá trị trung bình (mean), trung vị (median) hoặc mốt (mode) của cột, dùng df.fillna(df.mean()).
  - Dựa trên ngữ cảnh: Sử dụng giá trị từ các hàng gần kề hoặc dựa trên mối quan hệ với các cột khác.
  - Sử dụng mô hình dự đoán: Dùng các thuật toán như KNN (KNeigh-

borsClassifier), hoặc hồi quy để dự đoán giá trị thiếu.

- Đánh dấu giá trị thiếu: Thêm cột chỉ báo (indicator column) để đánh dấu đâu là giá trị được điền, nếu cần giữ thông tin gốc.

1.1.3 Kiểm tra và xử lý giá trị ngoại lệ (Outliers)

Giá trị ngoại lệ là các điểm dữ liệu khác biệt đáng kể so với các điểm dữ liệu còn lại. Giá trị ngoại lệ có thể làm sai lệch mô hình học máy, do đó cần phải loại bỏ chúng trước khi huấn luyện mô hình. Cách xử lý:

- Phát hiện outliers:
  - Sử dụng quy tắc IQR (Interquartile Range): Giá trị nằm ngoài khoảng [Q1 - 1.5*IQR, Q3 + 1.5*IQR] được coi là ngoại lai.
  - Dùng Z-score: Giá trị có Z-score vượt quá ngưỡng (thường là 3) là ngoại lai.
  - Trực quan hóa bằng boxplot hoặc scatter plot.
- Xử lý:
  - Xóa: Nếu outliers là lỗi nhập liệu hoặc không đại diện, xóa chúng.
  - Giới hạn (Capping): Thay thế outliers bằng giá trị ngưỡng tối đa/tối thiểu.
  - Giữ nguyên: Nếu outliers mang thông tin quan trọng (ví dụ: trong tài chính hoặc y học), giữ lại và chọn mô hình ít nhạy cảm với outliers (như Random Forest).

1.1.4 Xử lý dữ liệu trùng lặp

Xử lý dữ liệu trùng lặp

- Phát hiện: Dùng df.duplicated() trong pandas để tìm các hàng giống nhau.
- Xử lý: Xóa các bản sao bằng df.drop_duplicates(), nhưng cần xem xét liệu sự trùng lặp có ý nghĩa (ví dụ: trong dữ liệu giao dịch).

1.1.5 Chuẩn hóa định dạng dữ liệu (Data normalization)

- Nếu có các cột có giá trị khác nhau quá lớn, cần phải chuẩn hóa chúng để đảm bảo rằng không có cột nào có ảnh hưởng quá lớn đến mô hình
- Chuyển đổi cột sang kiểu phù hợp (ví dụ: chuỗi thành số bằng pd.to_numeric(), hoặc ngày tháng bằng pd.to_datetime()).
- Xử lý văn bản:
  - Chuyển đổi về cùng chữ cái (lowercase hoặc uppercase): df['column'].str.lower().

  - Loại bỏ khoảng trắng thừa: df['column'].str.strip().
  - Xóa ký tự đặc biệt: Dùng regex hoặc str.replace().

- Đơn vị đo lường: Chuyển đổi về cùng đơn vị (ví dụ: từ km sang m).

1.1.6 Xử lý dữ liệu không nhất quán (Inconsistencies)

Dữ liệu không nhất quán thường xuất hiện trong các cột phân loại (categorical):

- Sửa lỗi nhập liệu: Ví dụ, "VN" và "Việt Nam" nên được gộp thành một giá trị duy nhất.
- Chuẩn hóa danh mục: Dùng df['column'].replace('old_value': 'new_value') để thay thế.
- Kiểm tra logic: Đảm bảo dữ liệu phù hợp với thực tế (ví dụ: tuổi không âm).

1.1.7 Chuyển đổi kiểu dữ liệu (Data type conversion)

Đảm bảo rằng các cột có kiểu dữ liệu phù hợp. Ví dụ, các cột biểu diễn các category nên được chuyển đổi thành kiểu categorical. Trong Azure Machine Learning Studio (MAML), module "Edit Metadata" có thể được sử dụng để chuyển đổi kiểu dữ liệu của các cột

1.2 Các kỹ thuật Machine Learning trình bày trong Chapter 12:

1.2.1 Hồi quy Logistic (Logistic Regression)

- Là một thuật toán phân loại tuyến tính, được sử dụng để dự đoán xác suất của một biến mục tiêu nhị phân.
- Trong chương 12, nó được sử dụng để ước tính khả năng một người mắc bệnh tiểu đường
- Không giống như hồi quy tuyến tính, đầu ra của hồi quy logistic nằm trong khoảng từ 0 đến 1
- Để đánh giá thuật toán, người ta sử dụng 10-fold cross-validation để thu được điểm số trung bình

1.2.2 K-Nearest Neighbors (KNN)

- Là một thuật toán phân loại dựa trên khoảng cách.
- Dự đoán lớp của một điểm dữ liệu mới dựa trên lớp của K điểm dữ liệu gần nhất.
- KNN so sánh khoảng cách của một thể hiện truy vấn với các mẫu huấn luyện khác và chọn K-Nearest Neighbors.
- Các neighbors gần nhất chiếm đa số sẽ là dự đoán cho thể hiện truy vấn.

- Trong chương 12, KNN được sử dụng để phân loại xem một người có mắc bệnh tiểu đường hay không.
- Để có được độ chính xác tốt nhất, cần thử các giá trị K khác nhau để có được K tối ưu.

1.2.3 Support Vector Machines (SVM)

- Là một thuật toán phân loại mạnh mẽ có thể xử lý các bài toán phân loại tuyến tính và phi tuyến tính bằng cách sử dụng các kernel khác nhau.
- Ý tưởng chính đằng sau SVM là vẽ một đường thẳng giữa hai hoặc nhiều lớp theo cách tốt nhất có thể.
- Có thể được sử dụng cho các vấn đề phân loại, và chương này đề cập đến SVM với kernel tuyến tính (linear) và kernel RBF (Radial Basis Function).
- SVM tìm kiếm sự phân tách tối đa.
- SVM có thể được sử dụng để phân loại các bệnh nhân tiểu đường và không tiểu đường.
- C parameter là tham số điều chỉnh trong SVM. Giá trị C cao sẽ cố gắng phân loại tất cả các điểm một cách chính xác, có thể dẫn đến biên hẹp hơn. Giá trị C thấp nhằm mục đích có biên rộng nhất, nhưng có thể phân loại sai một số điểm.
- Gamma xác định phạm vi ảnh hưởng của một mẫu huấn luyện duy nhất. Gamma thấp có nghĩa là mọi điểm đều có tầm ảnh hưởng xa. Gamma cao có nghĩa là các điểm gần ranh giới có ảnh hưởng lớn hơn.

Các bước thực hiện chung:

- Chọn thuật toán tốt nhất: Sử dụng cross-validation để đánh giá hiệu suất của các thuật toán khác nhau và chọn ra thuật toán tốt nhất cho bộ dữ liệu. Trong ví dụ này, KNN với k = 19 là thuật toán tốt nhất.
- Huấn luyện mô hình: Sử dụng toàn bộ bộ dữ liệu để huấn luyện mô hình với thuật toán đã chọn.
- Lưu mô hình: Lưu mô hình đã huấn luyện vào đĩa để sử dụng sau này. Thư viện pickle trong Python được sử dụng để lưu và tải mô hình.
- Triển khai mô hình: Triển khai mô hình đã lưu dưới dạng một dịch vụ REST bằng framework Flask.
- Xây dựng ứng dụng khách hàng: Xây dựng một ứng dụng khách hàng (ví dụ: ứng dụng dòng lệnh) để tương tác với dịch vụ REST và đưa ra dự đoán.

CHƯƠNG 2. CÀI ĐẶT VÀ CHẠY CASE STUDY TRONG CHAP 12 (TÀI LIỆU 1)

2.1 Cài đặt và chạy case study

2.1.1 Loading the Data

```python
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes.csv')
df.info()
```

Hình 2.1: Loading data

```
PS C:\Projects\Python\REST API> python case.py
<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
```

Hình 2.2: Loading data output

2.1.2 Cleaning the Data

```python
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes.csv')
#---check for null values---
print("Nulls")
print("=====")
print(df.isnull().sum())
```

Hình 2.3: Check for null values

```
PS C:\Projects\Python\REST API> python case.py
Nulls
=====
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Hình 2.4: Check for null values output

```python
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes.csv')
#---check for 0s---
print("0s")
print("==")
print(df.eq(0).sum())
```

Hình 2.5: Check for 0s

```
PS C:\Projects\Python\REST API> python case.py
0s
==
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Hình 2.6: Check for 0s output

```python
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes.csv')

df[['Glucose','BloodPressure','SkinThickness',
    'Insulin','BMI','DiabetesPedigreeFunction','Age']] = \
df[['Glucose','BloodPressure','SkinThickness',
    'Insulin','BMI','DiabetesPedigreeFunction','Age']].replace(0, np.nan)

df.fillna(df.mean(), inplace=True)  # replace NaN with the mean
print(df.eq(0).sum())
```

Hình 2.7: Replace 0s with the mean

```
PS C:\Projects\Python\REST API> python case.py
Pregnancies                 111
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                       0
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

Hình 2.8: Replace 0s with the mean output

2.1.3 Examining the Correlation Between the Feature

```python
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes.csv')

corr = df.corr()
print(corr)
```

Hình 2.9: Examining the correlation

```
PS C:\Projects\Python\REST API> python case.py
                          Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  DiabetesPedigreeFunction       Age   Outcome
Pregnancies                  1.000000  0.129459       0.141282      -0.081672 -0.073535  0.017683                 -0.033523  0.544341  0.221898
Glucose                      0.129459  1.000000       0.152590       0.057328  0.331357  0.221071                  0.137337  0.263514  0.466581
BloodPressure                0.141282  0.152590       1.000000       0.207371  0.088933  0.281805                  0.041265  0.239528  0.065068
SkinThickness               -0.081672  0.057328       0.207371       1.000000  0.436783  0.392573                  0.183928 -0.113970  0.074752
Insulin                     -0.073535  0.331357       0.088933       0.436783  1.000000  0.197859                  0.185071 -0.042163  0.130548
BMI                          0.017683  0.221071       0.281805       0.392573  0.197859  1.000000                  0.140647  0.036242  0.292695
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928  0.185071  0.140647                  1.000000  0.033561  0.173844
Age                          0.544341  0.263514       0.239528      -0.113970 -0.042163  0.036242                  0.033561  1.000000  0.238356
Outcome                      0.221898  0.466581       0.065068       0.074752  0.130548  0.292695                  0.173844  0.238356  1.000000
```

Hình 2.10: Examining the correlation output

2.1.4 Plotting the Correlation Between Features

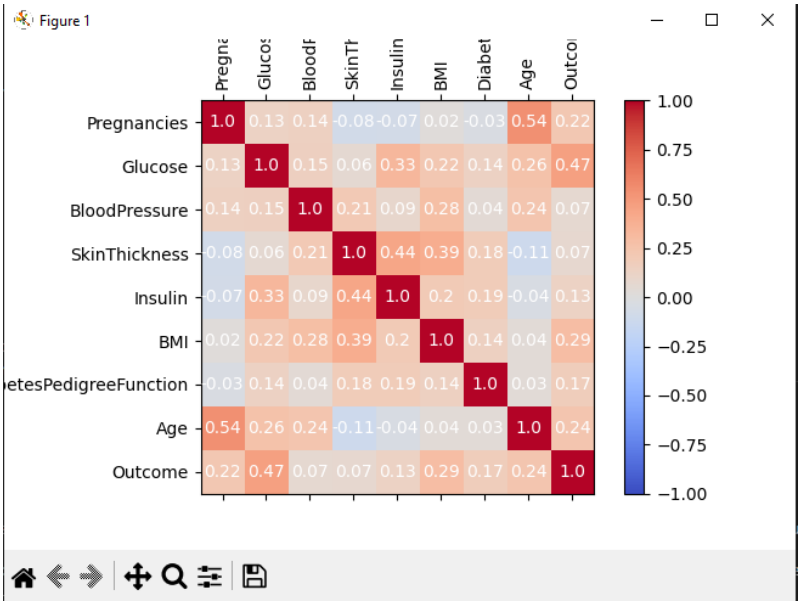
case.py > ...

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('diabetes.csv')
corr = df.corr()

fig, ax = plt.subplots(figsize=(10, 10))
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(df.columns),1)
ax.set_xticks(ticks)
ax.set_xticklabels(df.columns)
plt.xticks(rotation = 90)
ax.set_yticklabels(df.columns)
ax.set_yticks(ticks)
#---print the correlation factor---
for i in range(df.shape[1]):
    for j in range(9):
        text = ax.text(j, i, round(corr.iloc[i][j],2),
            ha="center", va="center", color="w")
plt.show()
```

Hình 2.11: Plotting the correlation



Hình 2.12: Matrix showing the various correlation factors

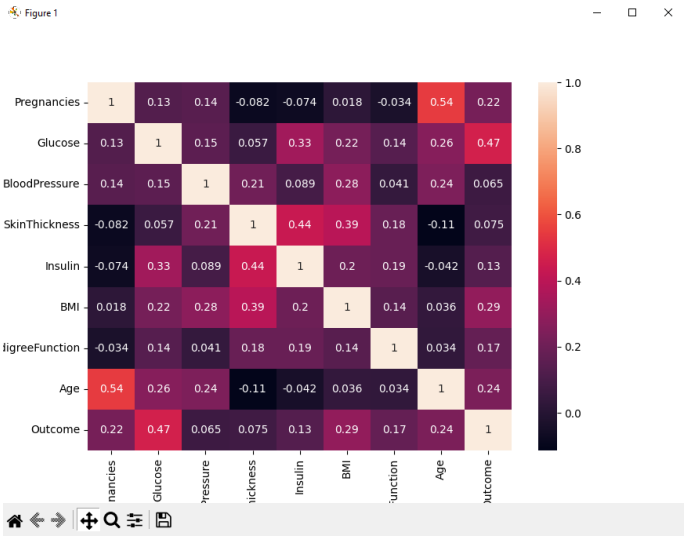
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('diabetes.csv')
corr = df.corr()

sns.heatmap(df.corr(),annot=True)
#---get a reference to the current figure and set its size---
fig = plt.gcf()
fig.set_size_inches(8,8)

fig, ax = plt.subplots(figsize=(10, 10))
cax = ax.matshow(corr,cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,len(df.columns),1)
ax.set_xticks(ticks)
ax.set_xticklabels(df.columns)
plt.xticks(rotation = 90)
ax.set_yticklabels(df.columns)
ax.set_yticks(ticks)
#---print the correlation factor---
for i in range(df.shape[1]):
    for j in range(9):
        text = ax.text(j, i, round(corr.iloc[i][j],2),
            ha="center", va="center", color="w")
plt.show()
```

Hình 2.13: Using Seaborn's heatmap() function



Hình 2.14: Heatmap produced by Seaborn showing the correlation factors

```python
import numpy as np
import pandas as pd

df = pd.read_csv('diabetes.csv')
corr = df.corr()

#---get the top four features that has the highest correlation---
print(df.corr().nlargest(4, 'Outcome').index)
#---print the top 4 correlation values---
print(df.corr().nlargest(4, 'Outcome').values[:,8])
```

Hình 2.15: Get the top four features that has the highest correlation.

```
PS C:\Projects\Python\REST API> python case.py
Index(['Outcome', 'Glucose', 'BMI', 'Age'], dtype='object')
[1.         0.4665814  0.29269466 0.23835598]
```

Hình 2.16: The top four features that have the highest correlation with the Outcome.

2.1.5 Logistic Regression

```python
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn.model_selection import cross_val_score

df = pd.read_csv('diabetes.csv')

#---features---
X = df[['Glucose','BMI','Age']]
#---label---
y = df.iloc[:,8]
log_regress = linear_model.LogisticRegression()
log_regress_score = cross_val_score(log_regress, X, y, cv=10, scoring='accuracy').mean()
print(log_regress_score)
```

Hình 2.17: Sử dụng Logistic Regression, kết quả trả về: 0.765704032809296

2.1.6 K-Nearest Neighbors

case.py > ...

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv('diabetes.csv')

#---features---
X = df[['Glucose','BMI','Age']]
#---label---
y = df.iloc[:,8]

#---empty list that will hold cv (cross-validates) scores---
cv_scores = []
#---number of folds---
folds = 10
#---creating odd list of K for KNN---
ks = list(range(1,int(len(X) * ((folds - 1)/folds)), 2))
#---perform k-fold cross validation---
for k in ks:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X, y, cv=folds, scoring='accuracy').mean()
    cv_scores.append(score)
#---get the maximum score---
knn_score = max(cv_scores)
#---find the optimal k that gives the highest score---
optimal_k = ks[cv_scores.index(knn_score)]
print(f"The optimal number of neighbors is {optimal_k}")
print(knn_score)
```

Hình 2.18: Sử dụng KNN, kết quả trả về:
The optional number of neighbors is 19
0.7669514695830485

2.1.7 Support Vector Machines

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn import svm

df = pd.read_csv('diabetes.csv')

#---features---
X = df[['Glucose','BMI','Age']]
#---label---
y = df.iloc[:,8]

linear_svm = svm.SVC(kernel='linear')
linear_svm_score = cross_val_score(linear_svm, X, y,
cv=10, scoring='accuracy').mean()
print(linear_svm_score)
```

Hình 2.19: Sử dụng SVM Linear, kết quả trả về: 0.7630724538619276

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn import svm

df = pd.read_csv('diabetes.csv')

#---features---
X = df[['Glucose','BMI','Age']]
#---label---
y = df.iloc[:,8]

rbf = svm.SVC(kernel='rbf')
rbf_score = cross_val_score(rbf, X, y, cv=10, scoring='accuracy').mean()
print(rbf_score)
```

Hình 2.20: Sử dụng SVM RBF kernel, kết quả trả về: 0.7617908407382091

2.1.8 Training and Saving the Model

Algorithm	Accuracy
K Nearest Neighbors	0.7669514695830485
Logistic Regression	0.7657040328092960
SVM Linear Kernel	0.7630724538619276
SVM RBF Kernel	0.7617908407382091

Bảng 2.1: Xếp hạng độ chính xác của các thuật toán khác nhau.

case.py > ...

```python
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
import pickle

df = pd.read_csv('diabetes.csv')

#---features---
X = df[['Glucose','BMI','Age']]
#---label---
y = df.iloc[:,8]

knn = KNeighborsClassifier(n_neighbors=19)
knn.fit(X, y)

#---save the model to disk---
filename = 'diabetes.sav'
#---write to the file using write and binary mode---
pickle.dump(knn, open(filename, 'wb'))
```

Hình 2.21: Save the Model

case.py > Age

```python
import numpy as np
import pandas as pd
import pickle

#---save the model to disk---
filename = 'diabetes.sav'
#---load the model from disk---
loaded_model = pickle.load(open(filename, 'rb'))

Glucose = 65
BMI = 70
Age = 50
prediction = loaded_model.predict([[Glucose, BMI, Age]])
print(prediction)

if (prediction[0]==0):
    print("Non-diabetic")
else:
    print("Diabetic")
```

Hình 2.22: Example

2.2 Triển khai mô hình (Deploying the Model)

rest_api.py > ...

```python
import pickle
from flask import Flask, request, json, jsonify
import numpy as np

app = Flask(__name__)
#---the filename of the saved model---
filename = 'diabetes.sav'
#---load the saved model---
loaded_model = pickle.load(open(filename, 'rb'))

@app.route('/diabetes/v1/predict', methods=['POST'])
def predict():
    #---get the features to predict---
    features = request.json
    #---create the features list for prediction---
    features_list = [features["Glucose"],
    features["BMI"],
    features["Age"]]
    #---get the prediction class---
    prediction = loaded_model.predict([features_list])
    #---get the prediction probabilities---
    confidence = loaded_model.predict_proba([features_list])
    #---formulate the response to return to client---
    response = {}
    response['prediction'] = int(prediction[0])
    response['confidence'] = str(round(np.amax(confidence[0]) * 100 ,2))
    return jsonify(response)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Hình 2.23: Rest API source code

2.2.1 Kiểm thử model (Testing the Model)

```
PS C:\Projects\Python\REST API> $headers = @{
>>     "Content-Type" = "application/json"
>> }
PS C:\Projects\Python\REST API>
PS C:\Projects\Python\REST API> $body = @{
>>     "Glucose" = 148
>>     "BMI" = 33.6
>>     "Age" = 50
>> } | ConvertTo-Json
PS C:\Projects\Python\REST API>
PS C:\Projects\Python\REST API> Invoke-WebRequest -Uri "http://127.0.0.1:5000/diabetes/v1/predict" `
>>     -Method POST `
>>     -Headers $headers `
>>     -Body $body `
>>     -ContentType "application/json"
```

Hình 2.24: Gửi 1 HTTP request đến http://127.0.0.1:5000/diabetes/v1/predict

```
StatusCode        : 200
StatusDescription : OK
Content           : {"confidence":"78.95","prediction":0}

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 38
                    Content-Type: application/json
                    Date: Tue, 11 Mar 2025 08:16:48 GMT
                    Server: Werkzeug/3.1.2 Python/3.12.5

                    {"confidence":"78.95","prediction":...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 38], [Content-Type, application/json], [Date, Tue, 11 Mar
                    2025 08:16:48 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 38
```

Hình 2.25: Kết quả trả về dưới dạng JSON: "confidence":"78.95","prediction":0

2.2.2 Tạo ứng dụng client (Creating the Client Application to Use the Model)

```python
import json
import requests

def predict_diabetes(BMI, Age, Glucose):
    url = 'http://127.0.0.1:5000/diabetes/v1/predict'
    data = {"BMI":BMI, "Age":Age, "Glucose":Glucose}
    data_json = json.dumps(data)
    headers = {'Content-type':'application/json'}
    response = requests.post(url, data=data_json, headers=headers)
    result = json.loads(response.text)
    return result
if __name__ == "__main__":
    predictions = predict_diabetes(30,40,100)
    print("Diabetic" if predictions["prediction"] == 1 else "Not Diabetic")
    print("Confidence: " + predictions["confidence"] + "%")
```

Hình 2.26: Kết quả trả về:
Not Diabetic
Confidence: 68.42%

```python
import json
import requests

def predict_diabetes(BMI, Age, Glucose):
    url = 'http://127.0.0.1:5000/diabetes/v1/predict'
    data = {"BMI":BMI, "Age":Age, "Glucose":Glucose}
    data_json = json.dumps(data)
    headers = {'Content-type':'application/json'}
    response = requests.post(url, data=data_json, headers=headers)
    result = json.loads(response.text)
    return result

if __name__ == "__main__":
    BMI = input('BMI?')
    Age = input('Age?')
    Glucose = input('Glucose?')
    predictions = predict_diabetes(BMI,Age,Glucose)
    print("Diabetic" if predictions["prediction"] == 1 else "Not Diabetic")
    print("Confidence: " + predictions["confidence"] + "%")
```

Hình 2.27: Tùy chỉnh Input, kết quả trả về:
Not Diabetic
Confidence: 57.89%