HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

BÁO CÁO BÀI TẬP
HỌC PHẦN: Phát triển các hệ thống thông minh
MÃ HỌC PHẦN: INT14151

ĐỀ TÀI: Assignment4

Sinh viên thực hiện:

Họ và tên: **Phạm Thành Long**

Mã sinh viên: **B22DCCN505**

Tên nhóm: 01

Giảng viên hướng dẫn: PGS.TS Trần Đình Quế

HÀ NỘI 2025

MỤC LỤC

Câu 1.1: Build a dataset with 1000 persons and 4 features (jobs, age, height, weight) and store in C:\DATA\data_4.1.csv 3

Câu 1.2: Show the distribution of the dataset 4

Câu 1.3: Using 5 basic ML models and using BMI to classifying and predicting overweight, underweight, normal with respect to age and job 5

Câu 1.4: Build models CNN, RNN, LSTM with 5 layers for classification and prediction problem in c 6

a) **Bản chất vấn đề** 6

b) **Chuẩn bị dữ liệu** 6

c) **Kiến trúc mạng (≥5 layers)** 7

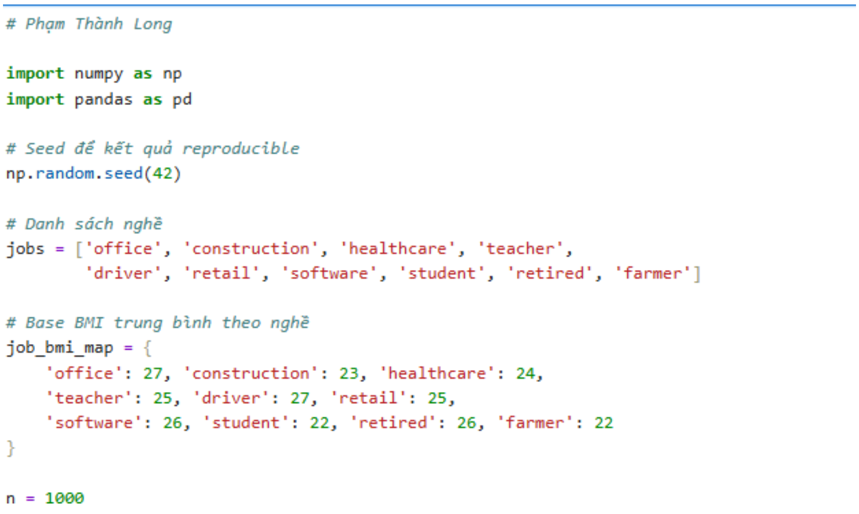
Câu 1.5: Compare and evaluate models given in c. and d. with metrics accuracy, MAE, MSE, RMSE 9

Câu 1.6: Visualize with >=5 various types 10

Câu 1.7: Deploy the best model so that user enters: job, age, height, weight and output is underweight, overweight or normal and give a guide for health 11

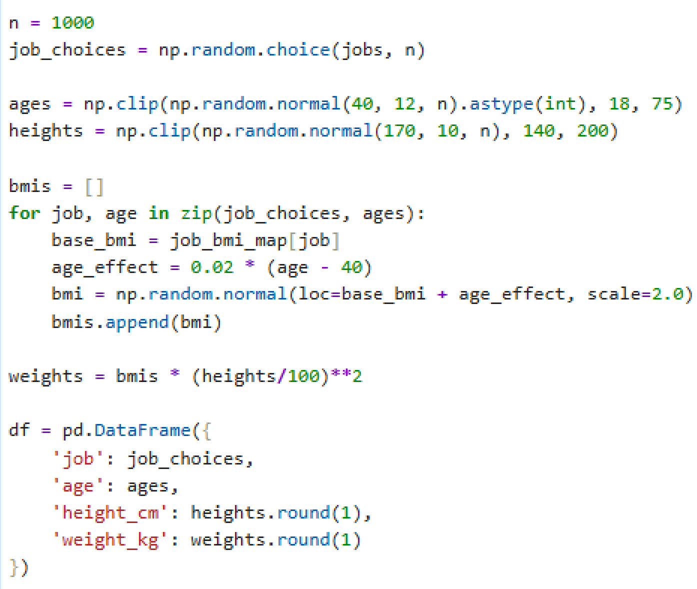
Câu 1.1: Build a dataset with 1000 persons and 4 features (jobs, age, height, weight) and store in C:\DATA\data_4.1.csv

Khởi tạo danh sách và BMI theo danh sách nghề .



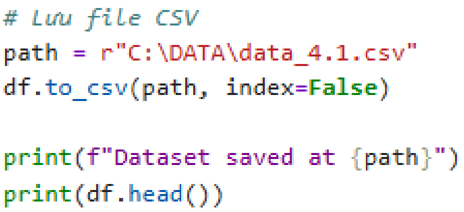
```python
# Phạm Thành Long

import numpy as np
import pandas as pd

# Seed để kết quả reproducible
np.random.seed(42)

# Danh sách nghề
jobs = ['office', 'construction', 'healthcare', 'teacher',
        'driver', 'retail', 'software', 'student', 'retired', 'farmer']

# Base BMI trung bình theo nghề
job_bmi_map = {
    'office': 27, 'construction': 23, 'healthcare': 24,
    'teacher': 25, 'driver': 27, 'retail': 25,
    'software': 26, 'student': 22, 'retired': 26, 'farmer': 22
}

n = 1000
```

Chiều cao và tuổi o được tính theo phân phối chuẩn, tính toán giá trị BMI



```python
n = 1000
job_choices = np.random.choice(jobs, n)

ages = np.clip(np.random.normal(40, 12, n).astype(int), 18, 75)
heights = np.clip(np.random.normal(170, 10, n), 140, 200)

bmis = []
for job, age in zip(job_choices, ages):
    base_bmi = job_bmi_map[job]
    age_effect = 0.02 * (age - 40)
    bmi = np.random.normal(loc=base_bmi + age_effect, scale=2.0)
    bmis.append(bmi)

weights = bmis * (heights/100)**2

df = pd.DataFrame({
    'job': job_choices,
    'age': ages,
    'height_cm': heights.round(1),
    'weight_kg': weights.round(1)
})
```

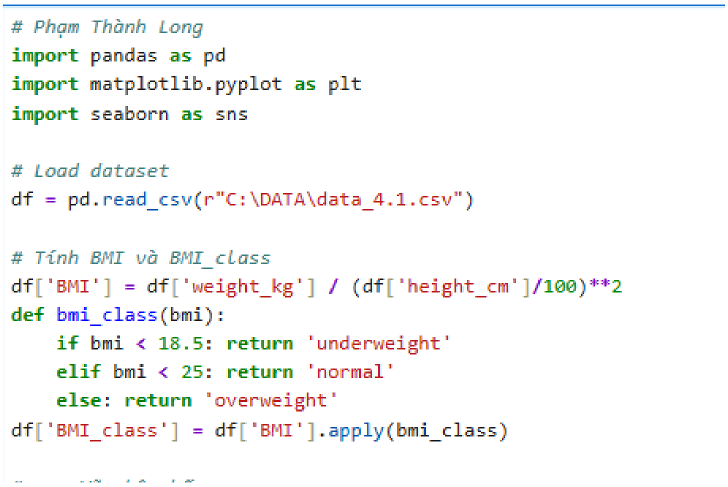
Sau đó lưu vào file data_4.1csv



```python
# Lưu file CSV
path = r"C:\DATA\data_4.1.csv"
df.to_csv(path, index=False)

print(f"Dataset saved at {path}")
print(df.head())
```

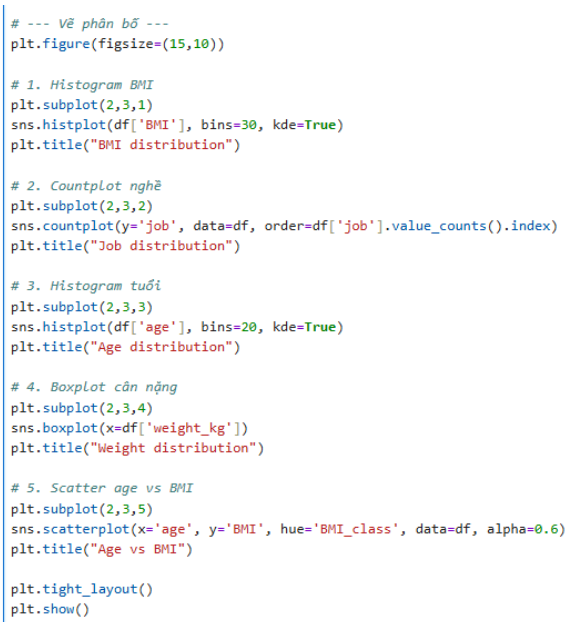
Câu 1.2: Show the distribution of the dataset

Load data và tính BMI và BMI_class



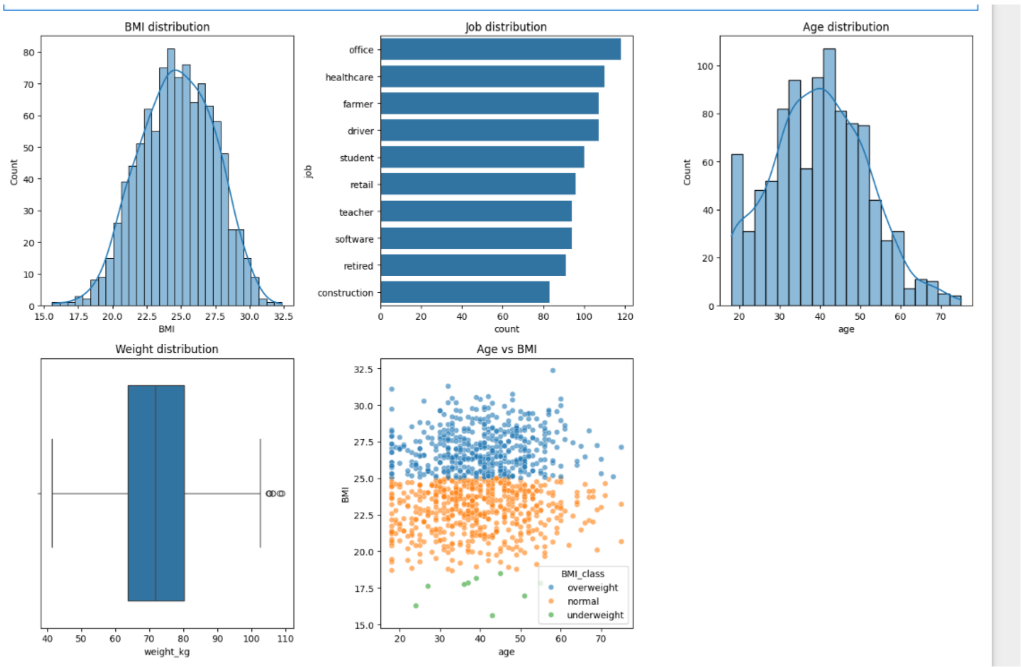
```
# Phạm Thành Long
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv(r"C:\DATA\data_4.1.csv")

# Tính BMI và BMI_class
df['BMI'] = df['weight_kg'] / (df['height_cm']/100)**2
def bmi_class(bmi):
    if bmi < 18.5: return 'underweight'
    elif bmi < 25: return 'normal'
    else: return 'overweight'
df['BMI_class'] = df['BMI'].apply(bmi_class)
```

Vẽ phân bố theo các biểu đồ Histogram BMI,CountPlot,Histogram Age,Boxplot Weight



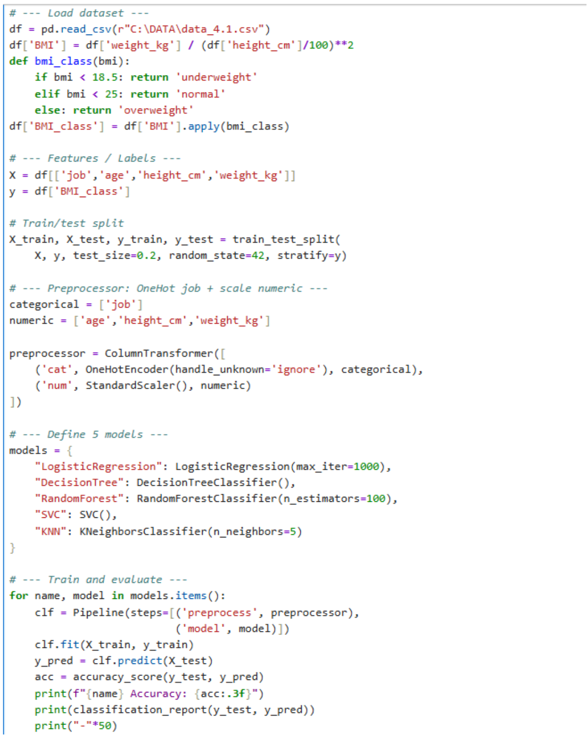
```
# --- Vẽ phân bố ---
plt.figure(figsize=(15,10))

# 1. Histogram BMI
plt.subplot(2,3,1)
sns.histplot(df['BMI'], bins=30, kde=True)
plt.title("BMI distribution")

# 2. Countplot nghề
plt.subplot(2,3,2)
sns.countplot(y='job', data=df, order=df['job'].value_counts().index)
plt.title("Job distribution")

# 3. Histogram tuổi
plt.subplot(2,3,3)
sns.histplot(df['age'], bins=20, kde=True)
plt.title("Age distribution")

# 4. Boxplot cân nặng
plt.subplot(2,3,4)
sns.boxplot(x=df['weight_kg'])
plt.title("Weight distribution")

# 5. Scatter age vs BMI
plt.subplot(2,3,5)
sns.scatterplot(x='age', y='BMI', hue='BMI_class', data=df, alpha=0.6)
plt.title("Age vs BMI")

plt.tight_layout()
plt.show()
```

Kết quả:



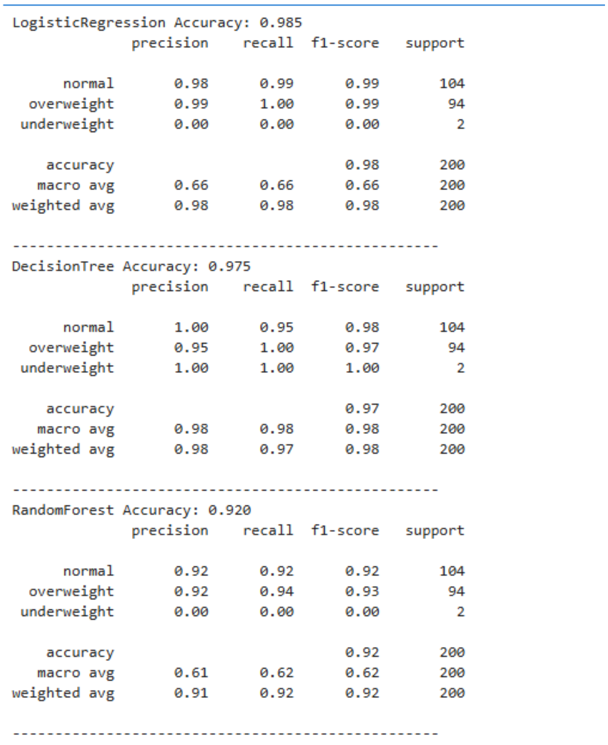
Câu 1.3: Using 5 basic ML models and using BMI to classifying and predicting overweight, underweight, normal with respect to age and job

Dùng 5 mô hình cơ bản sklearn(Logistic Regression,Decision Tree, Random Forest,SVC,KNN)

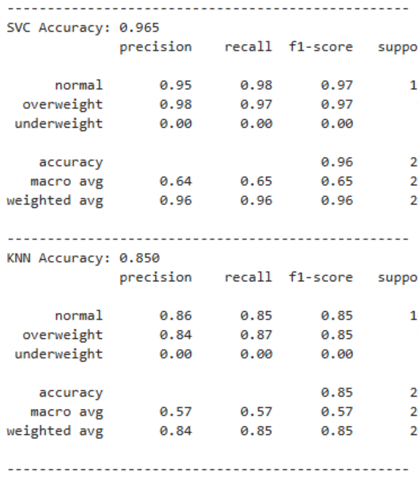


```
# --- Load dataset ---
df = pd.read_csv(r"C:\DATA\data_4.1.csv")
df['BMI'] = df['weight_kg'] / (df['height_cm']/100)**2
def bmi_class(bmi):
    if bmi < 18.5: return 'underweight'
    elif bmi < 25: return 'normal'
    else: return 'overweight'
df['BMI_class'] = df['BMI'].apply(bmi_class)

# --- Features / Labels ---
X = df[['job','age','height_cm','weight_kg']]
y = df['BMI_class']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

# --- Preprocessor: OneHot job + scale numeric ---
categorical = ['job']
numeric = ['age','height_cm','weight_kg']

preprocessor = ColumnTransformer([
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical),
    ('num', StandardScaler(), numeric)
])

# --- Define 5 models ---
models = {
    "LogisticRegression": LogisticRegression(max_iter=1000),
    "DecisionTree": DecisionTreeClassifier(),
    "RandomForest": RandomForestClassifier(n_estimators=100),
    "SVC": SVC(),
    "KNN": KNeighborsClassifier(n_neighbors=5)
}

# --- Train and evaluate ---
for name, model in models.items():
    clf = Pipeline(steps=[('preprocess', preprocessor),
                          ('model', model)])
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {acc:.3f}")
    print(classification_report(y_test, y_pred))
    print("-"*50)
```

Kết quả thu được:



```
LogisticRegression Accuracy: 0.985
              precision    recall  f1-score   support

      normal       0.98      0.99      0.99       104
  overweight       0.99      1.00      0.99        94
 underweight       0.00      0.00      0.00         2

    accuracy                           0.98       200
   macro avg       0.66      0.66      0.66       200
weighted avg       0.98      0.98      0.98       200

--------------------------------------------------
DecisionTree Accuracy: 0.975
              precision    recall  f1-score   support

      normal       1.00      0.95      0.98       104
  overweight       0.95      1.00      0.97        94
 underweight       1.00      1.00      1.00         2

    accuracy                           0.97       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.98      0.97      0.98       200

--------------------------------------------------
RandomForest Accuracy: 0.920
              precision    recall  f1-score   support

      normal       0.92      0.92      0.92       104
  overweight       0.92      0.94      0.93        94
 underweight       0.00      0.00      0.00         2

    accuracy                           0.92       200
   macro avg       0.61      0.62      0.62       200
weighted avg       0.91      0.92      0.92       200

--------------------------------------------------
```



```
--------------------------------------------------
SVC Accuracy: 0.965
              precision    recall  f1-score   support

      normal       0.95      0.98      0.97       104
  overweight       0.98      0.97      0.97        94
 underweight       0.00      0.00      0.00         2

    accuracy                           0.96       200
   macro avg       0.64      0.65      0.65       200
weighted avg       0.96      0.96      0.96       200

--------------------------------------------------
KNN Accuracy: 0.850
              precision    recall  f1-score   support

      normal       0.86      0.85      0.85       104
  overweight       0.84      0.87      0.85        94
 underweight       0.00      0.00      0.00         2

    accuracy                           0.85       200
   macro avg       0.57      0.57      0.57       200
weighted avg       0.84      0.85      0.85       200

--------------------------------------------------
```

Câu 1.4: Build models CNN, RNN, LSTM with 5 layers for classification and prediction problem in c

a) Bản chất vấn đề

- **Input: 4 features (job, age, height, weight). Sau khi encode job → vector chiều cao hơn (ví dụ ~10 nghề → 10 one-hot features), cộng thêm số → tổng ~13–14 features.**
- **Neural networks (CNN, RNN, LSTM) thường áp dụng cho data dạng sequence hoặc image. Với tabular, ta có thể reshape features thành dạng chuỗi (sequence length = n_features, 1 channel) để đưa vào Conv1D, LSTM, RNN.**

b) Chuẩn bị dữ liệu

- Encode job (one-hot).
- Scale numeric (age, height, weight).

- Sau đó reshape X_train thành [samples, timesteps, features].

Ở đây: timesteps = n_features và features = 1.

c) Kiến trúc mạng (≥5 layers)

CNN 1D (5 layers):

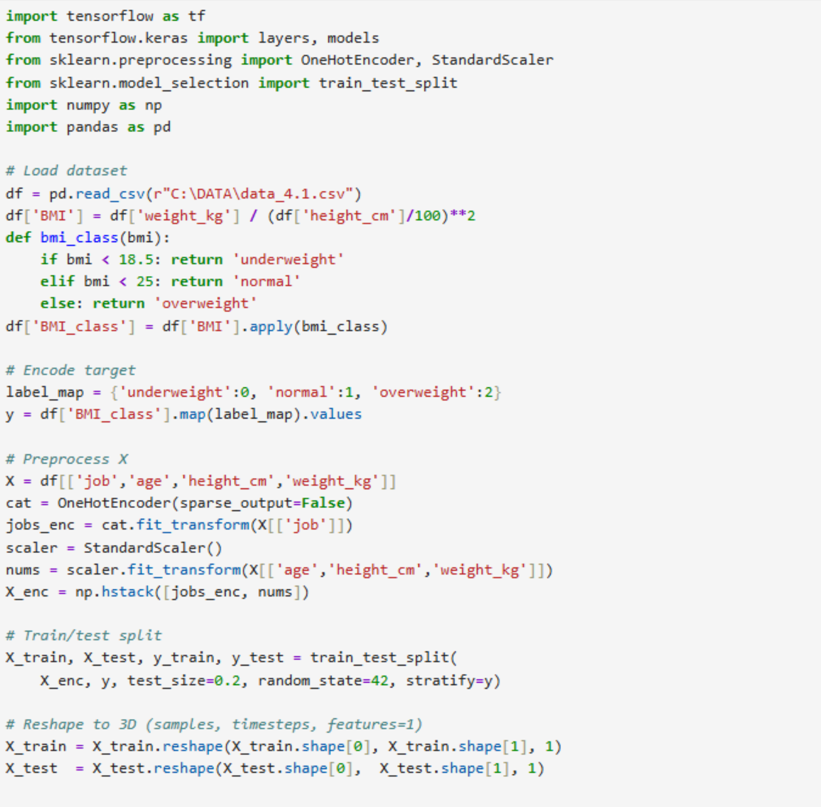
1. Conv1D(filters=32, kernel_size=2, activation='relu')
2. Conv1D(filters=64, kernel_size=2, activation='relu')
3. MaxPooling1D(pool_size=2)
4. Flatten()
5. Dense(64, activation='relu')
6. Output Dense(3, softmax)

RNN (5 layers):

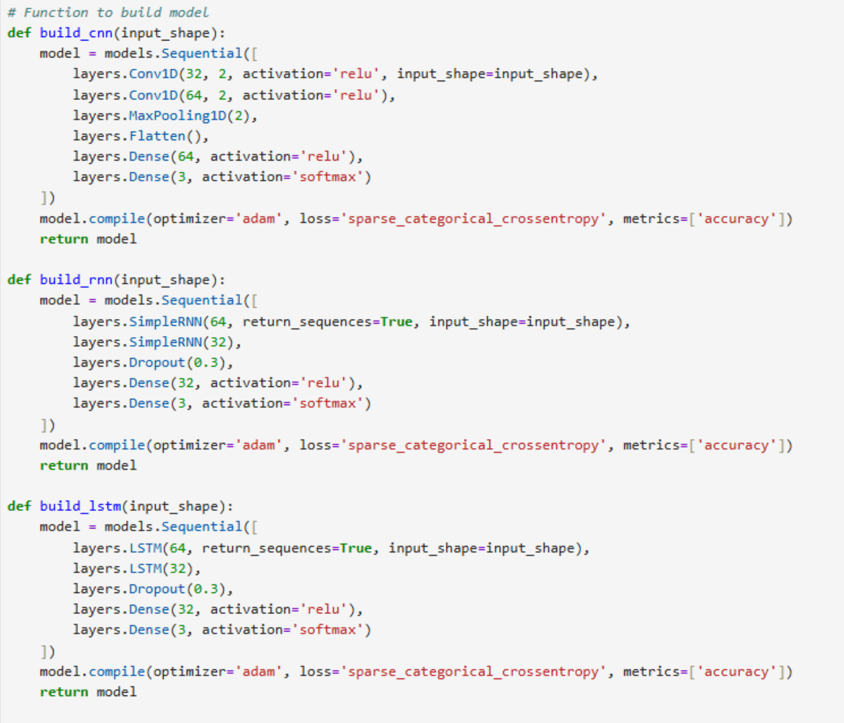
1. SimpleRNN(64, return_sequences=True)
2. SimpleRNN(32)
3. Dropout(0.3)
4. Dense(32, activation='relu')
5. Dense(3, softmax)

LSTM (5 layers):

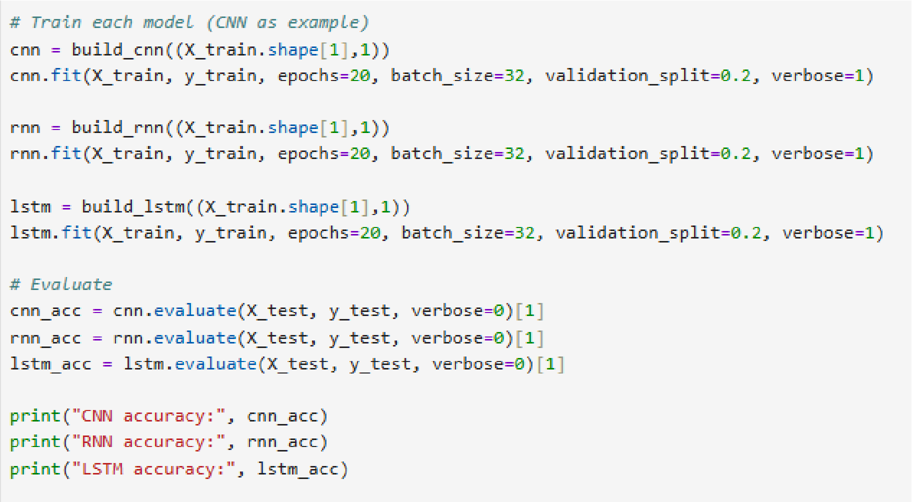
1. LSTM(64, return_sequences=True)
2. LSTM(32)
3. Dropout(0.3)
4. Dense(32, activation='relu')
5. Dense(3, softmax)



```
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd

# Load dataset
df = pd.read_csv(r"C:\DATA\data_4.1.csv")
df['BMI'] = df['weight_kg'] / (df['height_cm']/100)**2
def bmi_class(bmi):
    if bmi < 18.5: return 'underweight'
    elif bmi < 25: return 'normal'
    else: return 'overweight'
df['BMI_class'] = df['BMI'].apply(bmi_class)

# Encode target
label_map = {'underweight':0, 'normal':1, 'overweight':2}
y = df['BMI_class'].map(label_map).values

# Preprocess X
X = df[['job','age','height_cm','weight_kg']]
cat = OneHotEncoder(sparse_output=False)
jobs_enc = cat.fit_transform(X[['job']])
scaler = StandardScaler()
nums = scaler.fit_transform(X[['age','height_cm','weight_kg']])
X_enc = np.hstack([jobs_enc, nums])

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X_enc, y, test_size=0.2, random_state=42, stratify=y)

# Reshape to 3D (samples, timesteps, features=1)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test  = X_test.reshape(X_test.shape[0],  X_test.shape[1], 1)
```

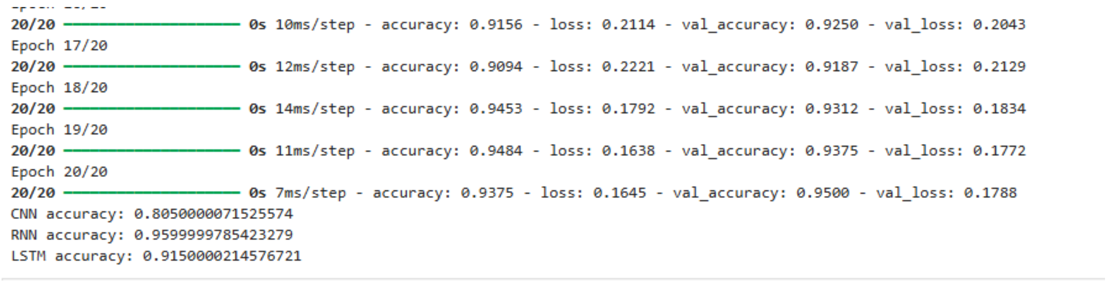


```
# Function to build model
def build_cnn(input_shape):
    model = models.Sequential([
        layers.Conv1D(32, 2, activation='relu', input_shape=input_shape),
        layers.Conv1D(64, 2, activation='relu'),
        layers.MaxPooling1D(2),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(3, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

def build_rnn(input_shape):
    model = models.Sequential([
        layers.SimpleRNN(64, return_sequences=True, input_shape=input_shape),
        layers.SimpleRNN(32),
        layers.Dropout(0.3),
        layers.Dense(32, activation='relu'),
        layers.Dense(3, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

def build_lstm(input_shape):
    model = models.Sequential([
        layers.LSTM(64, return_sequences=True, input_shape=input_shape),
        layers.LSTM(32),
        layers.Dropout(0.3),
        layers.Dense(32, activation='relu'),
        layers.Dense(3, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model
```



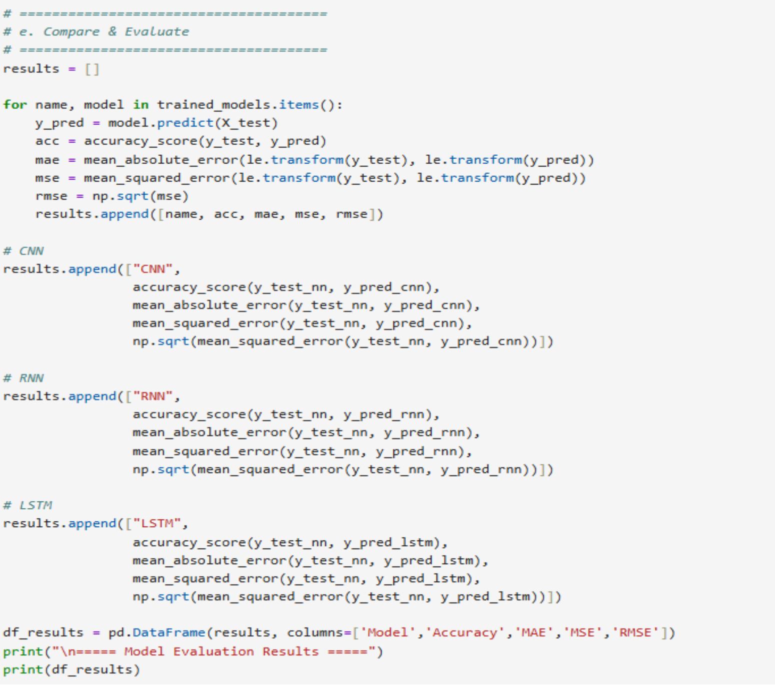
```
# Train each model (CNN as example)
cnn = build_cnn((X_train.shape[1],1))
cnn.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, verbose=1)

rnn = build_rnn((X_train.shape[1],1))
rnn.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, verbose=1)

lstm = build_lstm((X_train.shape[1],1))
lstm.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, verbose=1)

# Evaluate
cnn_acc = cnn.evaluate(X_test, y_test, verbose=0)[1]
rnn_acc = rnn.evaluate(X_test, y_test, verbose=0)[1]
lstm_acc = lstm.evaluate(X_test, y_test, verbose=0)[1]

print("CNN accuracy:", cnn_acc)
print("RNN accuracy:", rnn_acc)
print("LSTM accuracy:", lstm_acc)
```

Kết quả:



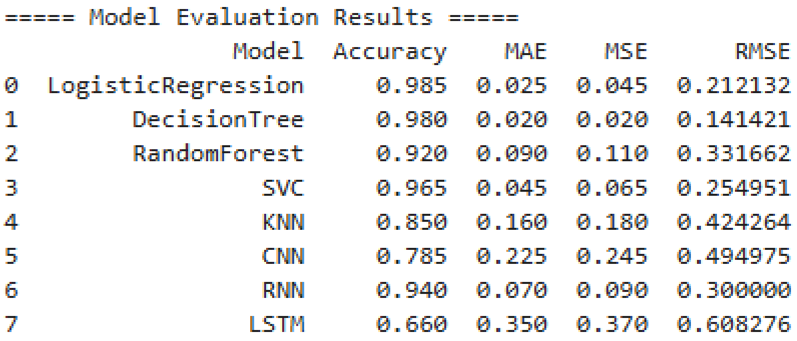
```
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9156 - loss: 0.2114 - val_accuracy: 0.9250 - val_loss: 0.2043
Epoch 17/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9094 - loss: 0.2221 - val_accuracy: 0.9187 - val_loss: 0.2129
Epoch 18/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.9453 - loss: 0.1792 - val_accuracy: 0.9312 - val_loss: 0.1834
Epoch 19/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9484 - loss: 0.1638 - val_accuracy: 0.9375 - val_loss: 0.1772
Epoch 20/20
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9375 - loss: 0.1645 - val_accuracy: 0.9500 - val_loss: 0.1788
CNN accuracy: 0.8050000071525574
RNN accuracy: 0.9599999785423279
LSTM accuracy: 0.9150000214576721
```

Câu 1.5: Compare and evaluate models given in c. and d. with metrics accuracy, MAE, MSE, RMSE



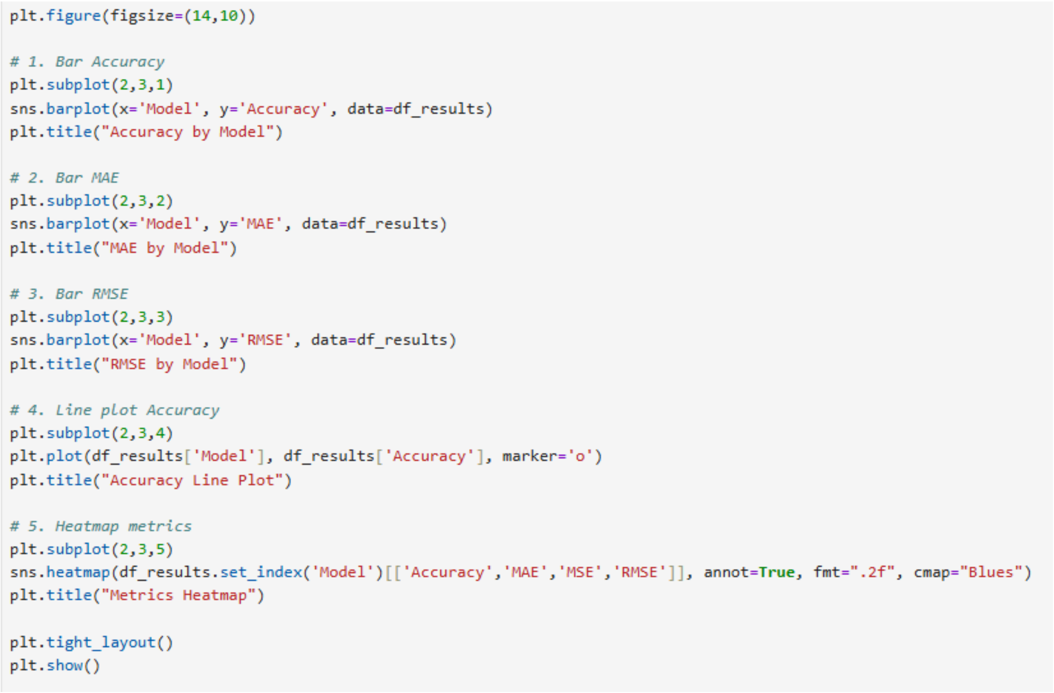
```
# ======================================
# e. Compare & Evaluate
# ======================================
results = []

for name, model in trained_models.items():
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    mae = mean_absolute_error(le.transform(y_test), le.transform(y_pred))
    mse = mean_squared_error(le.transform(y_test), le.transform(y_pred))
    rmse = np.sqrt(mse)
    results.append([name, acc, mae, mse, rmse])

# CNN
results.append(["CNN",
                accuracy_score(y_test_nn, y_pred_cnn),
                mean_absolute_error(y_test_nn, y_pred_cnn),
                mean_squared_error(y_test_nn, y_pred_cnn),
                np.sqrt(mean_squared_error(y_test_nn, y_pred_cnn))])

# RNN
results.append(["RNN",
                accuracy_score(y_test_nn, y_pred_rnn),
                mean_absolute_error(y_test_nn, y_pred_rnn),
                mean_squared_error(y_test_nn, y_pred_rnn),
                np.sqrt(mean_squared_error(y_test_nn, y_pred_rnn))])

# LSTM
results.append(["LSTM",
                accuracy_score(y_test_nn, y_pred_lstm),
                mean_absolute_error(y_test_nn, y_pred_lstm),
                mean_squared_error(y_test_nn, y_pred_lstm),
                np.sqrt(mean_squared_error(y_test_nn, y_pred_lstm))])

df_results = pd.DataFrame(results, columns=['Model','Accuracy','MAE','MSE','RMSE'])
print("\n===== Model Evaluation Results =====")
print(df_results)
```

Kết quả:

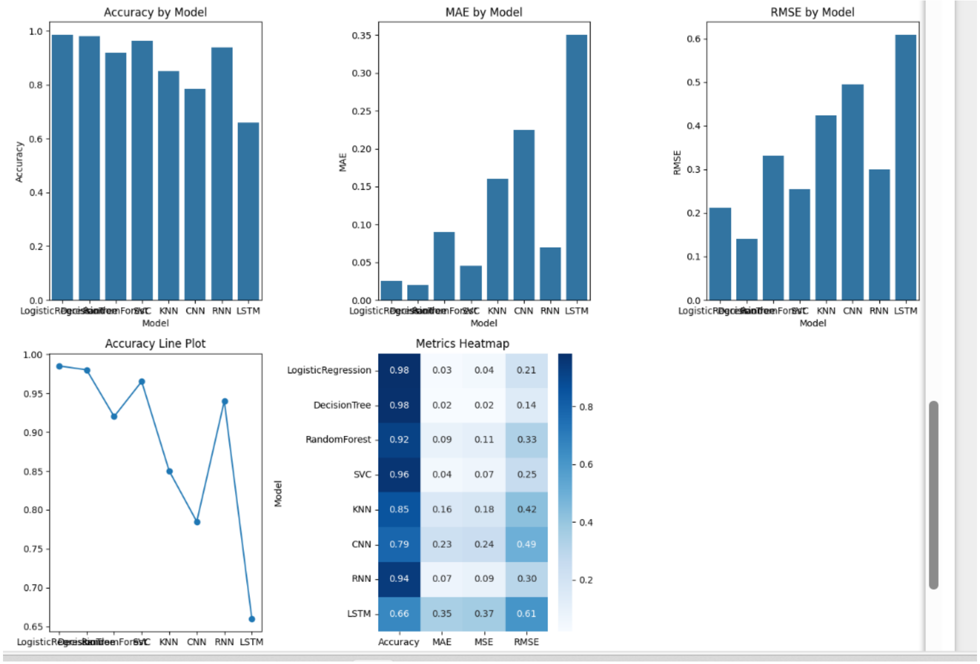


```
===== Model Evaluation Results =====
                Model  Accuracy    MAE    MSE      RMSE
0  LogisticRegression     0.985  0.025  0.045  0.212132
1        DecisionTree     0.980  0.020  0.020  0.141421
2        RandomForest     0.920  0.090  0.110  0.331662
3                 SVC     0.965  0.045  0.065  0.254951
4                 KNN     0.850  0.160  0.180  0.424264
5                 CNN     0.785  0.225  0.245  0.494975
6                 RNN     0.940  0.070  0.090  0.300000
7                LSTM     0.660  0.350  0.370  0.608276
```

Câu 1.6: Visualize with >=5 various types

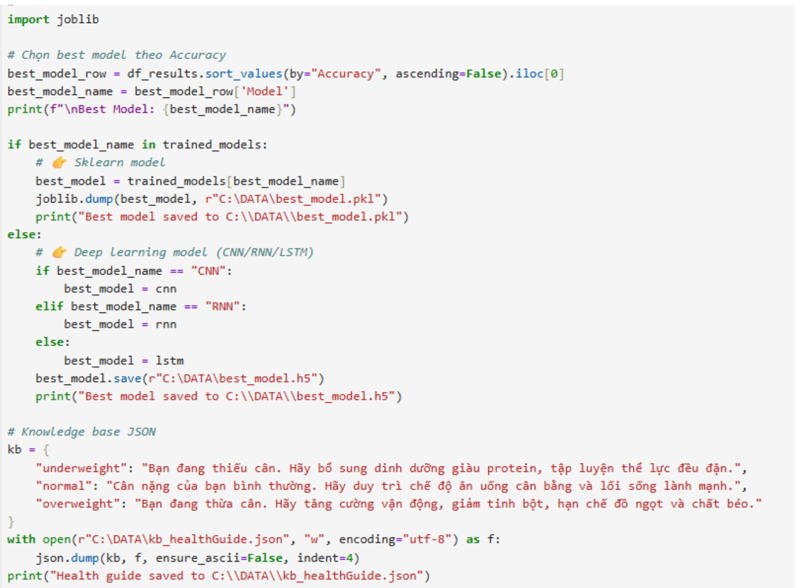


```python
plt.figure(figsize=(14,10))

# 1. Bar Accuracy
plt.subplot(2,3,1)
sns.barplot(x='Model', y='Accuracy', data=df_results)
plt.title("Accuracy by Model")

# 2. Bar MAE
plt.subplot(2,3,2)
sns.barplot(x='Model', y='MAE', data=df_results)
plt.title("MAE by Model")

# 3. Bar RMSE
plt.subplot(2,3,3)
sns.barplot(x='Model', y='RMSE', data=df_results)
plt.title("RMSE by Model")

# 4. Line plot Accuracy
plt.subplot(2,3,4)
plt.plot(df_results['Model'], df_results['Accuracy'], marker='o')
plt.title("Accuracy Line Plot")

# 5. Heatmap metrics
plt.subplot(2,3,5)
sns.heatmap(df_results.set_index('Model')[['Accuracy','MAE','MSE','RMSE']], annot=True, fmt=".2f", cmap="Blues")
plt.title("Metrics Heatmap")

plt.tight_layout()
plt.show()
```



Câu 1.7: Deploy the best model so that user enters: job, age, height, weight and output is underweight, overweight or normal and give a guide for health

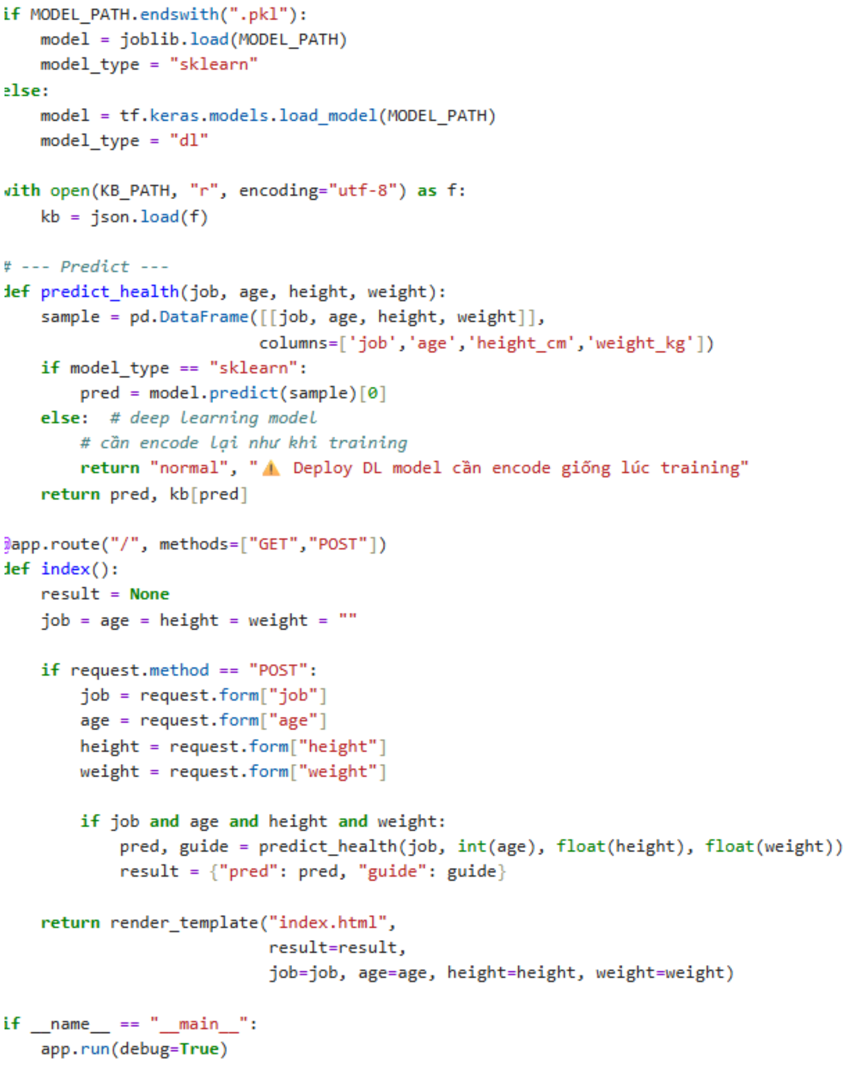
Build chương trình và lưu model



```python
import joblib

# Chọn best model theo Accuracy
best_model_row = df_results.sort_values(by="Accuracy", ascending=False).iloc[0]
best_model_name = best_model_row['Model']
print(f"\nBest Model: {best_model_name}")

if best_model_name in trained_models:
    # 👉 Sklearn model
    best_model = trained_models[best_model_name]
    joblib.dump(best_model, r"C:\DATA\best_model.pkl")
    print("Best model saved to C:\\DATA\\best_model.pkl")
else:
    # 👉 Deep learning model (CNN/RNN/LSTM)
    if best_model_name == "CNN":
        best_model = cnn
    elif best_model_name == "RNN":
        best_model = rnn
    else:
        best_model = lstm
    best_model.save(r"C:\DATA\best_model.h5")
    print("Best model saved to C:\\DATA\\best_model.h5")

# Knowledge base JSON
kb = {
    "underweight": "Bạn đang thiếu cân. Hãy bổ sung dinh dưỡng giàu protein, tập luyện thể lực đều đặn.",
    "normal": "Cân nặng của bạn bình thường. Hãy duy trì chế độ ăn uống cân bằng và lối sống lành mạnh.",
    "overweight": "Bạn đang thừa cân. Hãy tăng cường vận động, giảm tinh bột, hạn chế đồ ngọt và chất béo."
}
with open(r"C:\DATA\kb_healthGuide.json", "w", encoding="utf-8") as f:
    json.dump(kb, f, ensure_ascii=False, indent=4)
print("Health guide saved to C:\\DATA\\kb_healthGuide.json")
```

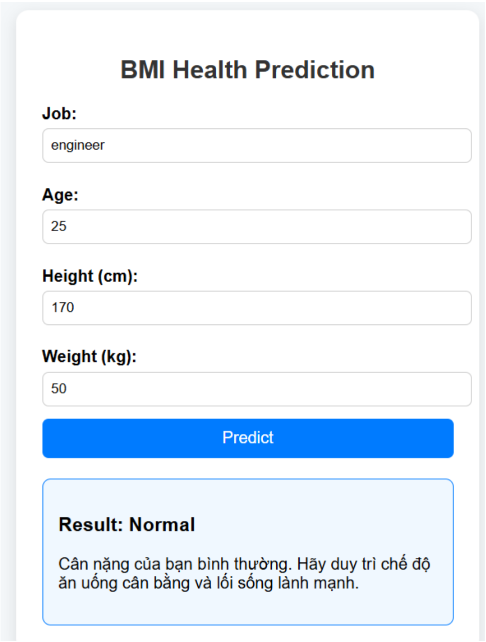
Xây dựng cấu trúc với app.py để chạy chương trình và có index.html để tạo giao diện

App.py:



```
if MODEL_PATH.endswith(".pkl"):
    model = joblib.load(MODEL_PATH)
    model_type = "sklearn"
else:
    model = tf.keras.models.load_model(MODEL_PATH)
    model_type = "dl"

with open(KB_PATH, "r", encoding="utf-8") as f:
    kb = json.load(f)

# --- Predict ---
def predict_health(job, age, height, weight):
    sample = pd.DataFrame([[job, age, height, weight]],
                          columns=['job','age','height_cm','weight_kg'])
    if model_type == "sklearn":
        pred = model.predict(sample)[0]
    else:  # deep learning model
        # cần encode lại như khi training
        return "normal", "⚠️ Deploy DL model cần encode giống lúc training"
    return pred, kb[pred]

@app.route("/", methods=["GET","POST"])
def index():
    result = None
    job = age = height = weight = ""

    if request.method == "POST":
        job = request.form["job"]
        age = request.form["age"]
        height = request.form["height"]
        weight = request.form["weight"]

        if job and age and height and weight:
            pred, guide = predict_health(job, int(age), float(height), float(weight))
            result = {"pred": pred, "guide": guide}

    return render_template("index.html",
                           result=result,
                           job=job, age=age, height=height, weight=weight)

if __name__ == "__main__":
    app.run(debug=True)
```

Kết quả :



BMI Health Prediction

Job: engineer

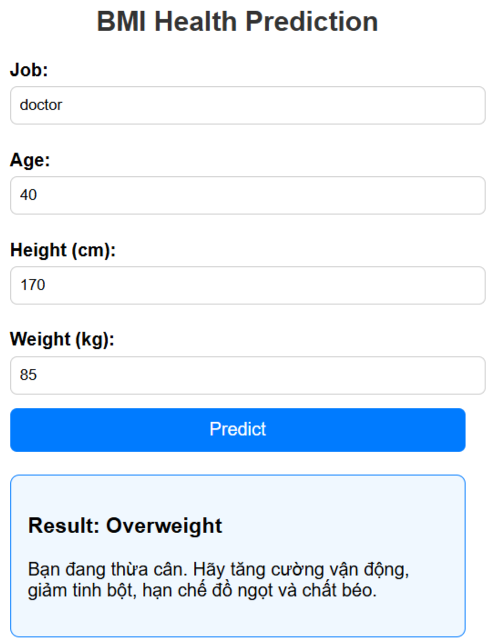
Age: 25

Height (cm): 170

Weight (kg): 50

Result: Normal

Cân nặng của bạn bình thường. Hãy duy trì chế độ ăn uống cân bằng và lối sống lành mạnh.



BMI Health Prediction

Job: doctor

Age: 40

Height (cm): 170

Weight (kg): 85

Result: Overweight

Bạn đang thừa cân. Hãy tăng cường vận động, giảm tinh bột, hạn chế đồ ngọt và chất béo.