Đồ án 3: Linear Regression

1. Thông tin sinh viên

- Họ và tên: Nguyễn Huy Hoàn
- MSSV: 20127166
- Lớp: 20CLC02

2. Import thư viện

```
import pandas as pd
import numpy as np

# Import thêm dữ thư viện nếu cần
import sklearn as sk
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
```

Ở đây em dùng thư viện sklearn để làm việc với Linear Regression và KFold Cross Validation.

Lý do em sử dụng thư viện sklearn là để làm việc với Linear Regression và KFold Cross Validation.

Lý do em sử dụng thư viện numpy là để chuyển dữ liệu từ dataframe sang numpy.array.

Lý do em sử dụng pandas là để đọc dữ liệu từ file excel hoặc csv

3. Đọc dữ liệu

```
# Đọc dữ liệu bằng pandas
train = pd.read_csv('train.csv', sep=',')
test = pd.read_csv('test.csv', sep=',')

# Lấy các đặc trưng X và giá trị mục tiêu y cho các tập huấn luyện (train) và kiểm tra (test)
X_train = train.iloc[:, :-1]    # Dataframe (chứa 10 đặc trưng huấn luyện)
y_train = train.iloc[:, -1]     # Series    (chứa 1 giá trị mục tiêu kiểm tra)

X_test = test.iloc[:, :-1]      # Dataframe (chứa 10 đặc trưng kiểm tra)
y_test = test.iloc[:, -1]       # Series    (chứa 1 giá trị mục tiêu kiểm tra)
```

```
# Sinh viên có thể sử dụng các khác nếu cần
```

Phần đọc dữ liệu em sử dụng code hướng dẫn để đọc dữ liêu từ file csv.

Dùng pandas để đọc dữ liệu từ file csv.

Lệnh `train.iloc` chỉ lấy các cột có index từ 0 đến n-1.

4. Cài đặt hàm

```
name_Columns = list(train.columns.values)
print(name_Columns)

data = np.array(train)
data_test = np.array(test)
```

```
['Adult Mortality', 'BMI', 'Polio', 'Diphtheria', 'HIV/AIDS', 'GDP', 'Thinness age 10-19
```

Em dùng một biến danh sách có tên là name_Columns để lưu tên cột của dữ liệu.

```
name_Columns = ['x', 'y']
```

Sau đó em dùng numpy để chuyển dữ liệu thành mảng. Tương tự dùng numpy để chuyển dữ liệu tập kiểm tra thành mảng.

Cách này giúp em có thể train và test dữ liệu dễ hơn

Yêu cầu 1a: Sử dụng toàn bộ 10 đặc trưng đề bài cung cấp (2 điểm)

```
# Phần code cho yêu cầu 1a
model = LinearRegression()
model.fit(X_train, y_train)
prediction = model.predict(X_test)
# print(model.coef_)
# print(prediction)
```

Ta xây dựng model với tất cả các đặc trưng đề bài cung cấp. Bằng linear regression, ta có thể tìm được hệ số của tất cả các đặc trưng đề bài cung cấp. Sau đó ta có thể tính được giá trị y dựa vào hệ số của tất cả các đặc trưng đề bài cung cấp.

Lệnh `model.fit(x_train, y_train)` sẽ train model với dữ liệu x_train và y_train.

`prediction = model.predict(x_test)` sẽ tính được giá trị y dựa vào hệ số của tất cả các đặc trưng đề bài cung cấp.

Sau đó dùng numpy để chuyển đổi giá trị y dựa vào hệ số của tất cả các đặc trưng đề bài cung cấp.

```
# Gọi hàm RMSE (tự cài đặt hoặc từ thư viện) trên tập kiểm tra
RMSE = np.sqrt(np.mean((prediction - y_test) ** 2))
print("RMSE: ", RMSE)
```

```
RMSE:  3.461055081296453
```

Hàm RMSE (Root Mean Squared Error) sẽ tính được sai số trung bình của dữ liệu.

RMSE được tính bằng công thức:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_{prediction})^2}$$

Công thức hồi quy

$$\text{Life expectancy} = Life expectancy = 52.43 -0.02 AM + 0.0457 BMI + 0.0015 P + 0.03 Diph -0.50 HIV/AIDS + 0.000063 GDP -0.0403 Age10_19 -0.0565 Age5_19 + 12.71 ICOR + 0.03 School $$

Yêu cầu 1b: Xây dựng mô hình sử dụng duy nhất 1 đặc trưng, tìm mô hình cho kết quả tốt nhất (2 điểm)

Lưu ý: khi sử dụng cross-validation, sinh viên cần xáo trộn dữ liệu 1 lần duy nhất và thực hiện trên toàn bộ đặc trưng

```
# Phần code cho yêu cầu 1b
# Tìm ra đặc trưng tốt nhất
# Huấn luyện lại mô hình best_feature_model với đặc trưng tốt nhất trên toàn bộ tập huấn luyệ
def cross_validation(data, k=5, f_index = 0):
    kf = KFold(n_splits=k, shuffle=True)
    rmse_list = []
```

```
    for train_index, test_index in kf.split(data):
        X_train, X_test = data[train_index, f_index].reshape(-1, 1), data[test_index, f_index
        y_train, y_test = data[train_index, -1], data[test_index, -1]
        model = LinearRegression()
        model.fit(X_train, y_train)
        prediction = model.predict(X_test)
        rmse = np.sqrt(np.mean((prediction - y_test) ** 2))
        rmse_list.append(rmse)

    return np.min(rmse_list)
```

Trước khi sử dụng cross-validation, em đã xáo trộn dữ liệu thành công bằng cách sử dụng hàm KFold

Em để biến shuffle là True, nghĩa là em sẽ xáo trộn dữ liệu trước khi sử dụng cross-validation.

Ở hàm cross_validation, em sử dụng hàm KFold để chia dữ liệu thành các phần tử, ở đây em chia ra làm 5 phần tử.

Em lấy các tính chất của phần tử đầu tiên vào model mang nó đi train với các fold khác. Và sau đó em lấy các tính chất của phần tử thứ 2 vào model mang nó đi train với các fold khác. Và như vậy cho đến hết.

Em sử dụng vòng lặp for để train model với các fold khác. Sau đó em đưa hết vào model để tính được RMSE, sau khi tính được RMSE thì em sẽ lấy giá trị nhỏ nhất để đưa vào list RMSE.

Và giá trị nhỏ nhất đó là đặc trưng tốt nhất.

```
# In ra các kết quả cross-validation như yêu cầu 1b
rmse_best_list = []
for i in range(len(data[0]) - 1):
    rmse = cross_validation(data, k=5, f_index=i)
    rmse_best_list.append(rmse)

def create_table(rmse_best_list, name_Columns):
    table = []
    for i in range(len(rmse_best_list)):
        row = []
        row.append(name_Columns[i])
        row.append(rmse_best_list[i])
        table.append(row)
    return pd.DataFrame(table, columns=['Mô Hình với 1 đặc trưng', 'RMSE'])

def find_best_feature(rmse_best_list, name_Columns):
    best_feature = rmse_best_list.index(min(rmse_best_list))
    return name_Columns[best_feature]
```

```
#print table
print(create_table(rmse_best_list, name_Columns))
```

```
                Mô Hình với 1 đặc trưng      RMSE
     0                    Adult Mortality  5.802005
     1                                BMI  6.720119
     2                              Polio  8.004572
     3                         Diphtheria  7.619996
     4                            HIV/AIDS  6.906212
     5                                GDP  7.265052
     6                Thinness age 10-19  7.711883
     7                  Thinness age 5-9  7.006402
     8  Income composition of resources  5.250445
```

Em sử dụng vòng lặp for để lặp cross-validation với số lần lặp là số cột, sau đó em lấy giá trị nhỏ nhất để đưa vào list RMSE.

Hàm create_table_rmse sẽ tạo bảng RMSE với các giá trị RMSE.

Hàm find_best_rmse sẽ tìm giá trị nhỏ nhất trong list RMSE.

Công thức hồi quy

$$\text{Life expectancy} = \text{RMSE}^{\frac{1}{2}}$$$$\text{Life expectancy} = \text{RMSE}^{\frac{1}{2}}$$

Yêu cầu 1c: Sinh viên tự xây dựng mô hình, tìm mô hình cho kết quả tốt nhất (3 điểm)

Lưu ý: khi sử dụng cross-validation, sinh viên cần xáo trộn dữ liệu 1 lần duy nhất và thực hiện trên toàn bộ $m$$m$ mô hình mà sinh viên thiết kế

```
# Phần code cho yêu cầu 1c
# Tìm ra mô hình tốt nhất (tự thiết kế bởi sinh viên)
# In ra các kết quả cross-validation như yêu cầu
# Tính sai số từ 1 cặp bộ dữ liệu train và test.
def scale_data(data):
    # scale model to data from 0 to 1
    # scale trên từng cột của data
    print(data.shape)
    for i in range(1,len(data[1])):
        data[:, i] = (data[:, i] - min(data[:, i])) / (max(data[:, i]) - min(data[:, i]))
    return data

# data = scale_data(data)
newdata = scale_data(data)
```

```
def cross_val(new_data, k = 5, f_index = 0):
    kf = KFold(n_splits=k, shuffle=True)
    rmse_list = []
    for train_index, test_index in kf.split(new_data):
        X_train, X_test = new_data[train_index, f_index].reshape(-1, 1), new_data[test_index,
        y_train, y_test = new_data[train_index, -1], new_data[test_index, -1]
        model = LinearRegression()
        model.fit(X_train, y_train)
        prediction = model.predict(X_test)
        rmse = np.sqrt(np.mean((prediction - y_test) ** 2))
        rmse_list.append(rmse)

    return np.min(rmse_list)

rmse_data_scaled = []
for i in range(len(data[0]) - 1):
    rmse = cross_val(data, k=5, f_index=i)
    rmse_data_scaled.append(rmse)

print(create_table(rmse_data_scaled, name_Columns))
```

```
 (1085, 11)
            Mô Hình với 1 đặc trưng      RMSE
 0                   Adult Mortality  0.130204
 1                               BMI  0.152618
 2                             Polio  0.168724
 3                        Diphtheria  0.174985
 4                          HIV/AIDS  0.155193
 5                               GDP  0.165790
 6                Thinness age 10-19  0.167939
 7                  Thinness age 5-9  0.161462
 8  Income composition of resources  0.120751
 9                          Schooling  0.125031
```

Mô hình thứ nhất

Mô hình này em chọn cách scale dữ liệu về các dữ liệu có giá trị từ 0 đến 1.

Lý do em chọn cách scale dữ liệu này là vì nó sẽ giúp cho mô hình này có thể tính được đặc trưng tốt nhất.

Sau đó em dùng phương pháp cross-validation để train mô hình này và tìm ra đặc trưng tốt nhất.

Hàm scale_data sẽ scale dữ liệu về các dữ liệu có giá trị từ 0 đến 1.

```
# biến đổi mô hình bằng cách bình phương
def square_data(data):
    for i in range(1,len(data[1])):
        data[:, i] = data[:, i] ** 2
```

```
    return data

square_data_ = square_data(data)
# làm tương tự như trên
def cross_val_square(square_data, k = 5, f_index = 0):
    kf = KFold(n_splits=k, shuffle=True)
    rmse_list = []
    for train_index, test_index in kf.split(square_data):
        X_train, X_test = square_data[train_index, f_index].reshape(-1, 1), square_data[test_
        y_train, y_test = square_data[train_index, -1], square_data[test_index, -1]
        model = LinearRegression()
        model.fit(X_train, y_train)
        prediction = model.predict(X_test)
        rmse = np.sqrt(np.mean((prediction - y_test) ** 2))
        rmse_list.append(rmse)

    return np.min(rmse_list)

rmse_data_square = []
for i in range(len(data[0]) - 1):
    rmse = cross_val_square(data, k=5, f_index=i)
    rmse_data_square.append(rmse)

print(create_table(rmse_data_square, name_Columns))
```

	Mô Hình với 1 đặc trưng	RMSE
0	Adult Mortality	0.142103
1	BMI	0.144534
2	Polio	0.176530
3	Diphtheria	0.174079
4	HIV/AIDS	0.172661
5	GDP	0.175401
6	Thinness age 10-19	0.191932
7	Thinness age 5-9	0.193403
8	Income composition of resources	0.094639
9	Schooling	0.120834

Mô hình thứ 2

Mô hình này được tạo ra sau khi biến đổi bình phương của dữ liệu.

Lý do em chọn mô hình này là vì nó sẽ giúp cho mô hình này có thể tính được đặc trưng tốt nhất.

Sau đó em dùng phương pháp cross-validation để train mô hình này và tìm ra đặc trưng tốt nhất.

Hàm square_data sẽ biến đổi dữ liệu thành dữ liệu có giá trị bình phương.

```
# mô hình có sự kết hợp của 2 hoặc nhiều đặc trưng
def combine_features(data):
    # chọn ra 2 đặc trưng có độ chính xác cao nhất từ data và tạo thành 1 data mới
    data_new = data[:, [0, 1]]\
```

```
    # kết hợp 2 đặc trưng
    data_new = np.concatenate((data_new, data[:, [2, 3]]), axis=1)
    return data_new

combine_data = combine_features(data)

def cross_val_combine(combine_data, k = 5, f_index = 0):
    kf = KFold(n_splits=k, shuffle=True)
    rmse_list = []
    for train_index, test_index in kf.split(combine_data):
        X_train, X_test = combine_data[train_index, f_index].reshape(-1, 1), combine_data[tes
        y_train, y_test = combine_data[train_index, -1], combine_data[test_index, -1]
        model = LinearRegression()
        model.fit(X_train, y_train)
        prediction = model.predict(X_test)
        rmse = np.sqrt(np.mean((prediction - y_test) ** 2))
        rmse_list.append(rmse)

    return np.min(rmse_list)

rmse_data_combine = []
for i in range(len(data[0]) - 1):
    rmse = cross_val_combine(data, k=5, f_index=i)
    rmse_data_combine.append(rmse)

# find min of rmse_data_combine
def find_min(rmse_data_combine):
    # sort rmse_data_combine from small to big
    rmse_data_combine.sort()
    # find index of min of rmse_data_combine
    index_min = rmse_data_combine.index(min(rmse_data_combine))
    return index_min


print(create_table(rmse_data_combine, name_Columns))
```

	Mô Hình với 1 đặc trưng	RMSE
0	Adult Mortality	0.142855
1	BMI	0.135670
2	Polio	0.166321
3	Diphtheria	0.172984
4	HIV/AIDS	0.192897
5	GDP	0.174361
6	Thinness age 10-19	0.184232
7	Thinness age 5-9	0.181860
8	Income composition of resources	0.096158
9	Schooling	0.124251

Công thức hồi quy

$$\text{Life expectancy} = \text{RMSE}^{\frac{1}{2}}$$$$\text{Life expectancy} =

Mô hình thứ 3

Mô hình này được tạo ra sau khi chọn ra 2 đặc trưng tốt nhất kết hợp với nhau.

Lý do em chọn mô hình này là vì nó sẽ giúp cho mô hình này có thể tính được đặc trưng tốt nhất.

Sau đó em dùng phương pháp cross-validation để train mô hình này và tìm ra đặc trưng tốt nhất.

Hàm combine_features sẽ kết hợp 2 đặc trưng tốt nhất.

Sau khi train 3 mô hình, em sẽ chọn mô hình có RMSE nhỏ nhất.

Kết quả thu được em thấy dữ liệu được chuẩn hóa và được train được tốt hơn.

Tài liệu tham khảo

1. Tài liệu về linear regression trong thư viện sklearn:

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

2. Tài liệu về cross-validation trong thư viện sklearn:

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

3. Tài liệu về đặc trưng tốt nhất trong thư viện sklearn:

- https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

4. Tài liệu về KFold trong thư viện sklearn:

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

5. Tài liệu về linear model trong thư viện sklearn:

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Ngoài ra em còn kiếm 1 số code khác nhau trong việc train mô hình. các code để fix lỗi từ các trang như stackoverflow, youtube, etc.

Double-click (or enter) to edit