UNIVERSITY OF SCIENCE

VIETNAM NATIONAL UNIVERSITY

LINEAR REGRESSION

Lecturer

Vu Quoc Hoang
Nguyen Van Quang Huy
Le Thanh Tung
Phan Thi Phuong Uyen

Ho Chi Minh City, July 24th, 2022

TABLE OF CONTENTS

1. INFORMATION 2
2. CONTENTS 2
2.1. Library Support 2
2.2. Function 3
2.2.1. fit(self, X, y) 3
2.2.2. get_params(self) 3
2.2.3. predict(self, X) 3
2.2.4. mse(y, y_hat) 3
2.2.5. rmse(y, y_hat) 3
2.2.6. FrameToNumpy(X_train, y_train, X_test, y_test) 4
2.2.7. Init_Data() 4
2.2.8. Shuffle_Data(X_train_clone, y_train_clone) 4
2.2.9. cross_validation(X_train_feature, y_train_feature, numberSet) 5
2.2.10. first_Model_Featurel(x) 6
2.2.11. second_Model_Featurel(x) 7
2.2.12. third_Model_Featurel(x) 7
2.3. Requirements 7
2.3.1. Model using all 10 features the topic provides 7
2.3.2. Models using all 1 feature the topic provides, find the best model 8
2.3.3. Freedom models, find the best model 9
3. RESULTS & EVALUTIONS 10
3.1. Model using all 10 features the topic provides 10
3.2. Models using all 1 feature the topic provides, find the best model 11
3.3. Freedom models, find the best model 12
4. REFERENCES 14

LINEAR REGRESSION

Applied Mathematics And Statistics

1. INFORMATION

Lecterer:

- Vu Quoc Hoang
- Nguyen Van Quang Huy
- Le Thanh Tung
- Phan Thi Phuong Uyen

Class: 20CLC08

Student:

- **ID:** 20127039
- **Name:** Tran Dam Gia Huy

Completion

Requirement	Detail	Completion
1	Use all 10 features the topic provides	100%
2	Build a model using only 1 feature, find the model that gives the best results	100%
3	Build the freedom model, find the model that gives the best results	100%
4	Explain all functions and the implement to the report	100%

2. CONTENTS

2.1.Library Support

```
import pandas as pd
import numpy as np
# Import thêm dữ thư viện nếu cần
import math
```

- pandas library is used for reading two files namely 'train.csv' and 'test.csv'
- numpy library is used for matrix processing

- math library: I use it to calculate RMSE by square root of 2 of MSE by math.sqrt()

2.2.Function

Four methods in class **OLSLinearRegression** (**fit**, **get_params**, **predict**, **mse**) I refer from **lab04.ipynb** on moodle of lecturer. Other functions, do it by myself

2.2.1. fit(self, X, y)

Input: matrix X (mxn), matrix y (m)

Output: Object with X,y,w properties

Approach: Calculate the w value by fomular w=(X^T.X)$^{-1}$.X^T.y

2.2.2. get_params(self)

Input: None

Output: w value in object

Approach: Return self.w

2.2.3. predict(self, X)

Input: matrix X (mxn)

Output: matrix y (m)

Approach: Use the w value calculated in previous to calculate the predict y value

2.2.4. mse(y, y_hat)

Input: matrix y result, matrix y_hat result that predicted by calculating with w value

Output: the MSE value

Approach: Calculate MSE value by fomular

$$\boxed{MSE = \frac{1}{n}\sum_{n}^{i=1}(y_i - \hat{y}_i)^2}$$

2.2.5. rmse(y, y_hat)

Input: matrix y result, matrix y_hat result that predicted by calculating with w value

Output: the MSE value

Approach: Calculate RMSE value by by square root of 2 of MSE

2.2.6. FrameToNumpy(X_train, y_train, X_test, y_test)

I refer the function to_numpy() from Python Examples (**)

Input: 4 matrix with frame datatype

Output: 4 matrix with np.array datatype

Approach: Use function **to_numpy()** (**) to convert it

2.2.7. Init_Data()

Input: None

Output: 2 matrix clone and the number of set (after divided the total data set by 5)

Approach:

- Copy from the orgirinal matrix and make a shuffle (I will decribe later)
- Divide total data set by 5

Implement:

- Use .copy() to copy data from X,y_train to X,y_train_clone
- Make a shuffle by **Shuffle_Data(X_train_clone, y_train_clone)** function
- Get len of X_train_clone and divided by 5 to save in numberSet

2.2.8. Shuffle_Data(X_train_clone, y_train_clone)

The idea of code I refer from DelftStack (*)

Input: 2 matrix

Output: 2 matrix after shuffle data

Approach: Use the random array index to change the position of each element in matrix (The index of two matrixes are the same)

Implement:

- Use **np.arange(len(X_train))** to get an array index in order with size = X_train's length
- Use **np.random.shuffle(rand_result)** to shuffle array index
- **X_train_clone[rand_result]** to reassign value at each index with new index that defined in array random index **rand_result**

-

2.2.9. cross_validation(X_train_feature, y_train_feature, numberSet)

This function implement the 5-fold cross validation method to find the best model

Input: 2 matrix feature, the numer of set

Output: RMSE of the model

Approach:

- Divide the total sets by 5. In this project, there is 5 groups of set
- Use 1 group (include X_train_feature and y_train_feature) for validating. The others are use for training
- Changing validating group hay training group in loop until each group becomes validating group one time. For example, the first time use the 1^{st} group for validating and 2^{nd}, 3^{rd}. 4^{th}, 5^{th} group for training so that the second time use the 2^{nd} group for validating and 1^{sd}, 3^{rd}. 4^{th}, 5^{th} group for training,….
- Sum the RMSE at each time in loop. After 5 times the training and validating, divided the RMSE by 5 to get the average RMSE in this model

Implement:

- Use loop from 0 to 5 (i=0; i<5) to do with 5 times validating and training data
- There are 4 matrix to store data at each loop **X_j_val** and **y_j_val** (for validate), **X_j_train** and **y_j_train** (for training)
- Assign data to **X_j_val**

```
X_j_val = X_train_feature[numberSet*j:numberSet*(j+1),:]
```

  X_j_val will get data from **X_train_feature** with all data in column and specific rows in each loop (mutiple 217 with j). For example:

  The first loop: **X_j_val =** X_train_feature[0:numberSet,:] = X_train_feature[0:217,:]

  The second loop: **X_j_val =** X_train_feature[217:434,:]

  …….

- Assign data to **y_j_val** is the same with **X_j_val** without choosing column (‘:’) because **y_j_val** just has 1 column and n row. So we need to get all data in specific rows

```
y_j_val = y_train_feature[numberSet*j:numberSet*(j+1)]
```

- Assign data to **X_j_train**

X_j_train will get data from **X_train_feature** except **X_j_val**.

The above code I merge 2 matrix before (from 0 to j) and after (from j+1 to end) **X_j_val**

```
X_j_train = np.concatenate([X_train_feature[0:numberSet*j,:],
X_train_feature[numberSet*(j+1):len(X_train_feature),:]])
```

matrix by np.concatenate. For example

Supposed that, **X_j_val** = X_train_feature[0:217,:]. So that **X_j_train** will merge two matrix with get all columns data and specific rows: X_train_feature[0:0,:] and X_train_feature[217:1085,:]

Supposed that, **X_j_val** = X_train_feature[217:434,:]. So that **X_j_train** will merge two matrix with get all columns data and specific rows: X_train_feature[0:217,:] and X_train_feature[434:1085,:]

……

- Assign data to **y_j_train** is the same with **X_j_train** without choosing column (':') because **y_j_train** just has 1 column and n row. So we need to get all data in specific rows

```
y_j_train = np.concatenate([y_train_feature[0:numberSet*j],
y_train_feature[numberSet*(j+1):len(y_train_feature)]])
```

- Call the method **fit(X_j_train, y_j_train)** to calculate w values and predict the result by method **predict(X_j_val)** that stored in **y_j_val_pre.**
- Calculate the RMSE of each loop by **rmse(y_j_val, y_j_val_pre**) function and add it to **rmse_val**
- Divided RMSE value by 5 after finished and return it as average RMSE of this model

2.2.10. first_Model_Featurel(x)

Input: Original matrix after shuffle

Output: The feature matrix corresponding the model defined

Approach and Implement: The first model I implemented is

Life Expectancy= Polio*w1 + Diphtheria*w2 + Income composition of resources*w3 + Schooling*w4

I choose this model because this is 4 best features (min RMSE) in 10 features. So I make an addition between them to create an RMSE model.

I use concatenate method I merge corressponding column (column 2,3,8,9 with begin index=0) with each other to create the matrix feature X and return it

2.2.11. second_Model_FeatureI(x)

Input: Original matrix after shuffle

Output: The feature matrix corresponding the model defined

Approach and Implement: The second model I implemented is

Life Expectancy= (Schooling^3)*w1 + (Income composition of resources^2)*w2 + Diphtheria*w3

I choose this model because this is 3 best features (min RMSE) in 10 features. After a number of times tries, I find this model has the good RMSE with cubic function

I use concatenate method I merge corressponding column ((column 3)^1, (column 8)^2 ,(column 9)^3) with begin index=0) with each other to create the matrix feature X and return

2.2.12. third_Model_FeatureI(x)

Input: Original matrix after shuffle

Output: The feature matrix corresponding the model defined

Approach and Implement: The third model I implemented is

Life Expectancy = (Schooling * Income composition of resources)*w1 + (BMI + Polio + Diphtheria)*w2

I choose this model because this is 5 best features (min RMSE) in 10 features. After a number of times tries, I find this model has the good RMSE with combining multiplication and summation

I use concatenate method I merge corressponding column ((column 8*column 9) and (column 1 + column 2 + column 3)) with begin index=0) with each other to create the matrix feature X and return

2.3. Requirements

2.3.1. Model using all 10 features the topic provides

Approach:

- Calculating the w value by training data with 10 features (in file train.csv)
- Using w value to calculate the predict result
- Determining the RMSE between the predict result and the result in file test.csv

Implement:

- I call the method fit(X_train, y_train) in class OLSLinearRegression to calculate the **w** values and save the entire object to **lr** with features in file **train.csv**
- The object **lr** calls the method predict(X_test) to calculate the predict the result with 10 features in file test.csv and **w** value in object **lr** that calculated in the previous step
- Having the predict result calculated by **w**, I calculate the RMSE between the predict result and the result in file **test.csv**

2.3.2. Models using all 1 feature the topic provides, find the best model

Approach:

- Calculating the w value by training data with only 1 feature (in file train.csv)
- Using w value to calculate the predict result
- Determining the RMSE between the predict result and the result
- All of above are implemented by **5-fold cross validation (** Divide dataset into 5 groups , 1 group for testing, 4 groups remaining for training data, and calculate the RMSE, do it 5 times until each group becomes validating group and divided RMSE by 5 to get the average RMSE after executing)
- Do it with 10 features in **train.csv** (10 times, each time works with 1 feature)

Implement:

- Initializing **X_train_clone, y_train_clone** and **numberSet** by Init_Data() function (I have explained clearly in the previous topic **2.2.7**)
- Calculate the RMSE value in a model by calling the cross_validation(X_train_feature, y_train_feature, numberSet) (I have explained clearly in the previous topic **2.2.9**)
- To calculate 10 RMSE values in 10 features in train.csv, I initialize an array to store 10 RMSE values and call the cross_validation(X_train_feature, y_train_feature, numberSet) in loop with index i in range (0,10)
- In loop, **X_train_feature** = X_train_clone[:,i:i+1] and **y_train_feature** = y_train_clone.

For example

  - With index i=0: **X_train_feature** = X_train_clone[:,0:1] means getting all data at column 0 as the first feature in **train.csv**
  - With index i=1: **X_train_feature** = X_train_clone[:,1:2] means getting all data at column 1 as the second feature in **train.csv**
  - ……….
- Finally, after doing loop to calculate the RMSE of 10 features, I get an array storing 10 RMSE values (**rmse_arr).** I use argmin() function to get the index in the array that has minimum values (It means the i^{th} feature has smallest RMSE) and save to **bestIndex** because the model has smallest RMSE is the best model

- The best feature will be display by X_train[:,bestIndex] and save to X_best_feature (Get the column with bestIndex)
- I call the method fit(X_train, y_train) in class OLSLinearRegression with X_train = X_best_feature (The best feature) to calculate the **w** values and save the entire object to **lr** with features in file **train.csv**
- The object **lr** calls the method predict(X_test) to calculate the predict the result with the feature having the corresponding bestIndex in file **test.csv** and **w** value in object **lr** that calculated in the previous step
- Having the predict result calculated by **w**, I calculate the RMSE between the predict result and the result in file **test.csv**

2.3.3. Freedom models, find the best model

Approach:

- Create 3 models (**X_train_feature**) by 3 function (**first_Mode_Featurel**, **second_Mode_Featurel**, **third_Mode_Featurel**) that I have explained in 2.2.10, 2.2.11, 2.2.12
- Calculate 3 RMSE of 3 models
- Find the best model by minimum RMSE
- Train the best model in train.csv to get w values in class OLSLinearRegression
- Calculate the predict result by using w value with coressponding model in test.csv
- Calculate the RMSE between result in test.csv and the predict result

Implement:

- Create 3 models (**X_train_feature**) and save in list **X_train_feature_array** by 3 function(**first_Mode_Featurel(x)**,**second_Mode_Featurel(x)**,**third_Mode_Featurel(x))**) that I have explained in 2.2.10, 2.2.11, 2.2.12 with **x**=**X_train_clone**
- Create 3 models (**X_test_feature**) and save in list **X_test_feature_array** by 3 function(**first_Mode_Featurel(x)**,**second_Mode_Featurel(x)**,**third_Mode_Featurel(x))**) that I have explained in 2.2.10, 2.2.11, 2.2.12 with **x**=**X_test.** This step is used for when I get the best model in train.csv, I can get the best model in test.csv with same index and structure
- Calculate the RMSE value in each model by calling the cross_validation(X_train_feature, y_train_feature, numberSet) (I have explained clearly in the previous topic **2.2.9**) in list **RMSE_array**
- Finally, after calculating the RMSE of 3 models, I get an array storing 3 RMSE values (**RMSE_arr).** I use argmin() function to get the index in the array that has minimum

values (It means the i^{th} feature has smallest RMSE) and save to **bestIndex** because the model has smallest RMSE is the best model

- The best feature will be display by X_train_feature_array[bestIndex] and save to X_best_feature (Get the model with bestIndex)
- I call the method fit(X_train, y_train) in class OLSLinearRegression with X_train = X_best_feature (The best feature) to calculate the **w** values and save the entire object to **lr** with features in file **train.csv**
- The object **lr** calls the method predict(X_test) to calculate the predict the result with the feature having the corresponding bestIndex in file **test.csv** and **w** value in object **lr** that calculated in the previous step
- Having the predict result calculated by **w**, I calculate the RMSE between the predict result and the result in file **test.csv**

3. RESULTS & EVALUTIONS

3.1.Model using all 10 features the topic provides

W values

```
w values----------------
w1: 0.015101362735318279
w2: 0.09021998065775627
w3: 0.04292181752549435
w4: 0.13928911689488216
w5: -0.5673328270884068
w6: -0.0001007651148748953
w7: 0.7407134377587112
w8: 0.19093579767396474
w9: 24.505973591149445
w10: 2.393516607832779
```

RMSE on **test.csv**

```
RMSE: 7.064046430584705
```

Regression Formula

Life Expectancy = 0.015101*Adult Mortality + 0.090220*BMI + 0.042922*Polio + 0.139289*Diphtheria - 0.567333*HIV/AIDS - 0.000101*GDP + 0.740713*Thinness age 10-19 + 0.190936*Thinness age 5-9 + 24.505974*Income composition of resources + 2.393517*Schooling

Evaluation

- W values have inverse ratio with feature value
- The RMSE value on **test.cv** with 10 features is small
- If the features are modified, it means W values are changed that leads to the shift in RMSE value

3.2.Models using all 1 feature the topic provides, find the best model

RMSE 10 features on train.csv

```
RMSE feature 1: 46.29318840485384
RMSE feature 2: 27.89566460172918
RMSE feature 3: 18.03199333638845
RMSE feature 4: 15.999435116642521
RMSE feature 5: 67.1076781800127
RMSE feature 6: 60.208226896875715
RMSE feature 7: 51.75287025474305
RMSE feature 8: 51.665179224465724
RMSE feature 9: 13.291015950250417
RMSE feature 10: 11.786003414181945
```

Best feature

```
Best feature is feature: 10
Feature 10 [[ 9.9]
 [ 9.8]
 [ 9.5]
 ...
 [10. ]
 [ 9.8]
 [ 9.8]]
```

W values

```
W: 5.5573993976919205
```

RMSE on **test.csv**

```
RMSE: 10.26095039165537
```

Regression Formula

Life Expectancy = 5.5573994*Schooling

Evaluation

No.	Feature	RMSE
1	Adult Mortality	46.245725793822416
2	BMI	27.955334582292416
3	Polio	18.02915163159739
4	Diphtheria	15.845876548938545
5	HIV/AIDS	67.09917048487907
6	GDP	60.20975942082864
7	Thinness age 10-19	51.855291331394504
8	Thinness age 5-9	51.75814155839241
9	Income composition of resources	13.27861308365102
10	Schooling	11.771538278894493
RMSE of Model Schooling on **test.csv**		**10.26095039165537**

- W values have inverse ratio with feature value
- The RMSE of **Schooling** feature is smallest (11.77) that means **Schooling** is the best feature to create a model with 1 feature
- The RMSE of **HIV/AIDS** feature is biggest (67.1) that means **HIV/AIDS** is the worst feature that should not be used to create a model
- There is a large difference between RMSE value of each feature. Some features have small RMSE, but the others are bigger than a lot (**Polio** and **GDP**)
- There are some features that have RMSE value is a little bit the same with the other (**Schooling** and **Income composition of resources**, **Polio** and **Diphtheria**)
- The RMSE value on **test.cv** with 1 feature is greater than it with 10 features (10.26 > 7.06). So that the model with 1 feature is worse than the other with 1 feature.
- If the features are modified, it means W values are changed that leads to the shift in RMSE value

3.3.Freedom models, find the best model

Model 1

Life Expectancy= (Schooling^3)*w1 + (Income composition of resources^2)*w2 + Diphtheria*w3

Model 2

Life Expectancy= Polio*w1 + Diphtheria*w2 + Income composition of resources*w3 + Schooling*w4

Model 3

Life Expectancy = (Schooling * Income composition of resources)*w1 + (BMI + Polio + Diphtheria)*w2

RMSE 3 models on **train.csv**

```
RMSE model 1: 9.55407523675269
RMSE model 2: 13.200600645811434
RMSE model 3: 12.951990251649391
```

Best model

```
Best model is model: 1
Best model: [[99.    91.     0.822 17.2  ]
 [ 7.    69.     0.613 12.9  ]
 [42.    42.     0.507  9.7  ]
 ...
 [98.    98.     0.569  9.8  ]
 [78.    77.     0.668 11.   ]
 [91.    91.     0.662 12.8  ]]
```

W values

```
W: [[0.14820296]
 [0.27274154]
 [5.55143115]
 [2.31214422]]
```

RMSE on **test.csv**

```
RMSE: 9.692944694229913
```

Regression Formula

Life Expectancy = 0.14820296*Polio + 0.27274154*Diphtheria + 5.55143115*Income composition of resources + 2.31214422*Schooling

Evaluation

No.	Feature	RMSE
1	Model 1	9.55407523675269
RMSE of Model 1 on **test.csv**		**9.692944694229913**
2	Model 2	13.200600645811434
3	Model 3	12.951990251649391

- W values have inverse ratio with feature value
- The RMSE of **Model 1** is smallest (9.55) that means **Model 1** is the best model
- The RMSE of **Model 2** is biggest (13.2) that means **Model 2** is the worst model
- Difference between RMSE value of each model is not too high (9.55, 13.2, 12.95).
- The RMSE value on **test.cv** with **Model 1** is greater than it with 10 features at requirement 1 (9.69 > 7.06) and smaller than 1 feature at requirement 2 (9.69 < 10.26). So that the **Model 1** is worse than the model with 10 features and better than the model with 1 feature.
- The feature with small RMSE values creates small RMSE model
- If the features are modified, it means W values are changed that leads to the shift in RMSE value

4. REFERENCES

- **(*)** Shuffle Data: https://www.delftstack.com/howto/numpy/python-numpy-shuffle-two-arrays/
- **(**)** Convert Pandas DataFrame to Numpy Array (using the function DataFrame.to_numpy()): https://pythonexamples.org/convert-pandas-dataframe-to-numpy-array/#:~:text=To%20convert%20Pandas%20DataFrame%20to,returned%20ndarray%20is%202%2Ddimensional.
- Merge 2 numpy array with np.concatenate: https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html