



# Machine Learning with Python

## Bài 3: Supervised Learning – Classification - Logistic Regression

Phòng LT & Mạng

[https://csc.edu.vn/lap-trinh-va-cSDL/Machine-Learning-with-Python\\_197](https://csc.edu.vn/lap-trinh-va-cSDL/Machine-Learning-with-Python_197)

2020



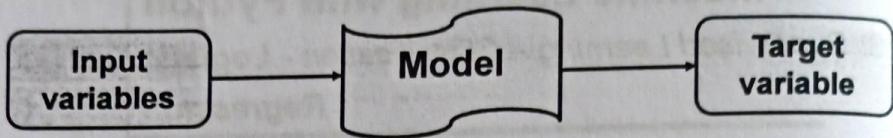
## Nội dung

### 1. Classification

### 2. Logistic Regression



# Classification



Mục tiêu: dự đoán target variable (category) từ input variables



# Classification

## ❑ Dữ liệu dành cho Classification

	Input Variables				Target Variable
	sepal length	sepal width	petal length	petal width	
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...					
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica



# Classification



## Classification là Supervised

					Target Variable
	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...					
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

Target  
Label  
Output  
Class variable  
Class  
Category

# Classification



## Phân loại

### Binary Classification

value1

value2

Target có 2 value

### Multi-class Classification

value  
1

value  
2

...

value  
N

Target có nhiều value

VD:

Ngày mai trời sẽ mưa hay  
không mưa?

Giao dịch này hợp lệ hay  
không hợp lệ?

VD:

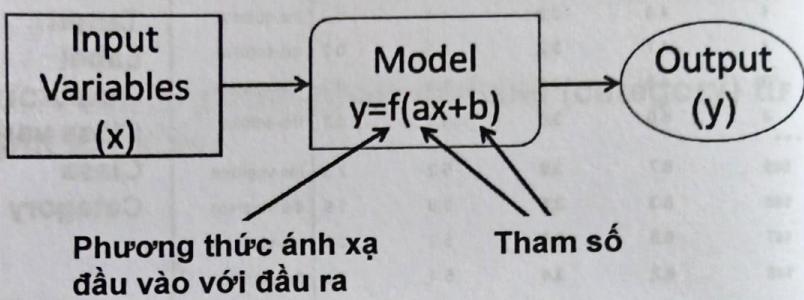
Khách hàng này sẽ mua loại  
sản phẩm nào?

Tweet này tích cực, tiêu cực  
hay trung tính

## Classification

### □ Machine Learning Model

- Là mô hình toán học với các tham số ánh xạ đầu vào với đầu ra

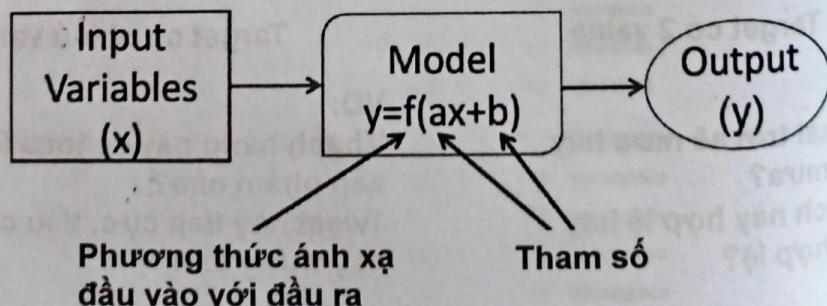


Machine Learning with Python

## Classification

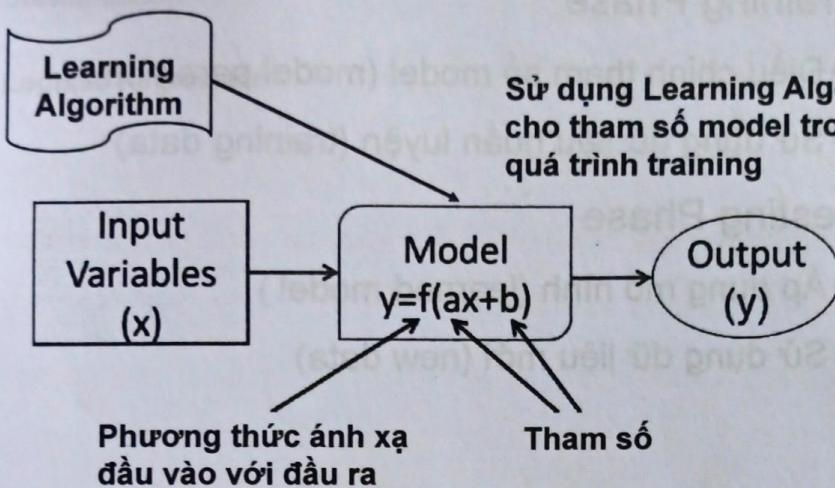
### □ Xây dựng Machine Learning Model

- Các tham số mẫu được điều chỉnh trong suốt quá trình training model để thay đổi ánh xạ input-output.

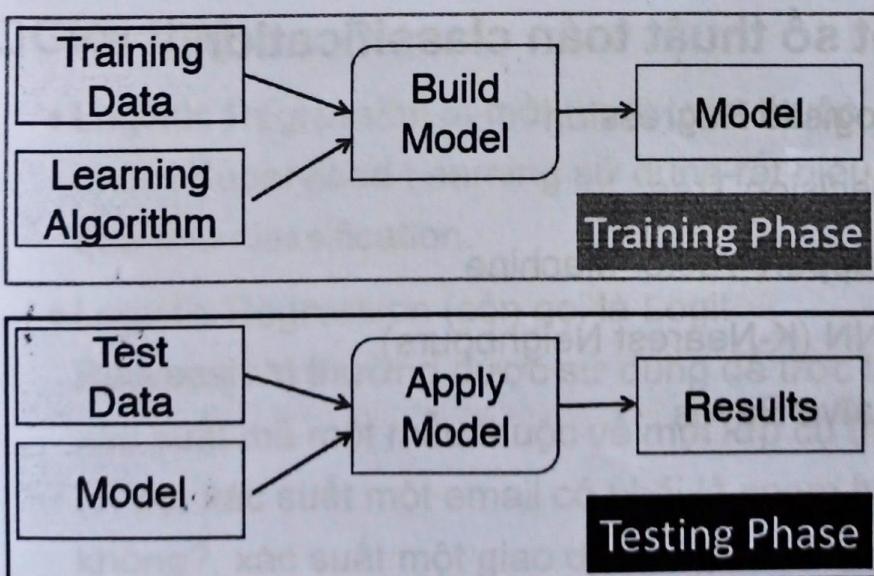


Machine Learning with Python

# Classification



# Classification



## Classification

---

- Training Phase

- Điều chỉnh tham số model (model parameter)
- Sử dụng dữ liệu huấn luyện (training data)

- Testing Phase

- Áp dụng mô hình (learned model)
- Sử dụng dữ liệu mới (new data)



Machine Learning with Python

---

## Classification

---

### ❑ Một số thuật toán classification

- Logistic Regression
- Decision Tree
- Support Vector Machine
- kNN (K-Nearest Neighbours)
- Naïve Bayes
- ...



Machine Learning with Python



## 1. Classification

### 2. Logistic Regression

# Logistic Regression



## □ Giới thiệu

- Logistic Regression là một thuật toán thuộc nhóm Supervised Learning sử dụng rất hiệu quả cho classification.
- Logistic Regression (còn gọi là Logit Regression) thường được sử dụng để ước tính xác suất mà một mẫu thuộc về một lớp cụ thể (ví dụ: xác suất một email có phải là spam hay không?, xác suất một giao dịch có phải là gian lận hay không?...)

## Logistic Regression

- Nếu xác suất ước tính lớn hơn 50%, thì mô hình dự đoán mẫu thuộc về lớp đó (được gọi là lớp positive, có nhãn "1"), hoặc ngược lại dự đoán rằng nó không thuộc về lớp đó (được gọi là lớp negative, có nhãn "0") => tạo ra một phân loại nhị phân, Binary Classifier
- Binary Classifier (phân loại nhị phân): outcome chỉ có 1 và 0 hay đúng và sai dù trong tên có Regression, có thể gọi là Conditional Class probabilities.



## Logistic Regression

### ❑ Các ứng dụng

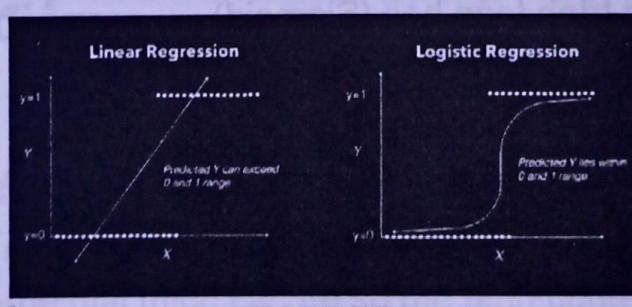
- Marketing (Tiếp thị): Dự đoán liệu một khách hàng cụ thể có mua sản phẩm bảo hiểm hay không?
- Kinh doanh: Dự đoán liệu khách hàng có ngưng sử dụng sản phẩm/ dịch vụ không?
- Banking (Ngân hàng): Dự đoán liệu khách hàng sẽ mặc định cho một khoản vay không?





## ☐ Lý do không áp dụng Linear Regression với bài toán chỉ có hai outcome

- Khi response variable chỉ có hai giá trị cụ thể, ta mong muốn có một mô hình dự đoán giá trị là 0 hoặc 1 hoặc là một điểm xác suất nằm trong khoảng từ 0 đến 1. Linear Regression không có khả năng này. Do đó, nếu ta sử dụng nó để mô hình binary response variable, mô hình kết quả có thể không hạn chế các giá trị Y được dự đoán trong 0 và 1



Y được dự đoán có thể vượt quá khoảng 0..1

Y được dự đoán nằm trong khoảng 0..1

Trong logistic regression, ta nhận được một probability score, điểm xác suất, phản ánh xác suất xảy ra sự kiện.



# Logistic Regression

## □ Thuật toán

- Là một mô hình hồi quy, trong đó response variable (biến phụ thuộc) có các giá trị phân loại như TRUE/FALSE hoặc 1/0. Nó đo lường xác suất của một phản ứng nhị phân như giá trị của response variable dựa trên phương trình toán học liên quan tới response variable và các biến predictor variable.



Machine Learning with Python

# Logistic Regression

- Phương trình toán học (Sigmoid)

$$S(z) = \frac{1}{1 + e^{-z}}$$

Với  $z = \log(p/(1-p)) = (c + w_1*x_1 + w_2*x_2 + w_3*x_3 + \dots)$

$$\Rightarrow S(z) = p = \frac{1}{1 + e^{-(c + w_1*x_1 + w_2*x_2 + w_3*x_3 + \dots)}}$$

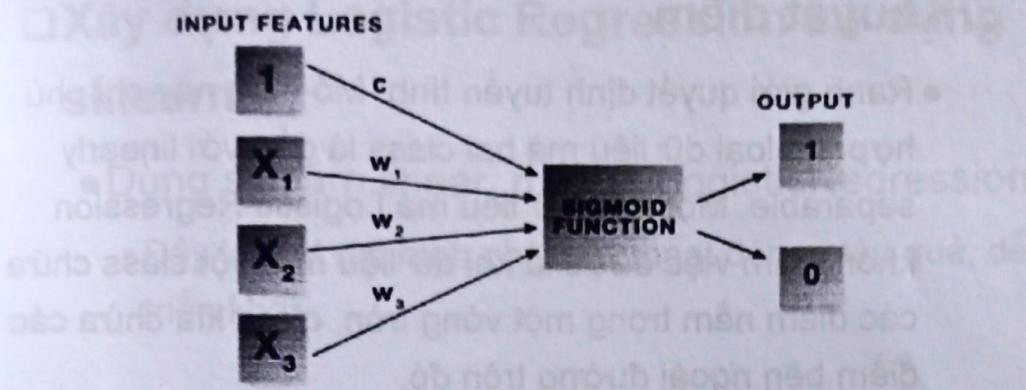
- Trong đó:

- $S(z) = p = y$ : response variable  $\Rightarrow$  xác suất dự đoán, với  $p \geq 0.5$  thì class = 1, còn  $p < 0.5$  thì class = 0
- $x_1, x_2, \dots, x_n$ : các predictor variable
- $c, w_1, w_2, w_3, \dots, w_n$  là những hằng số được gọi là hệ số (coefficient).



Machine Learning with Python

# Logistic Regression



$$y = \text{logistic}(c + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \dots + x_n * w_n)$$

$$y = 1 / (1 + e^{-(c + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \dots + x_n * w_n)})$$



# Logistic Regression



## Ưu điểm

- Dễ dàng mở rộng cho bài toán target có nhiều hơn hai loại
- Huấn luyện nhanh
- Độ chính xác cao cho nhiều tập dữ liệu đơn giản
- Có thể giải thích các hệ số mô hình cũng như các chỉ số về tầm quan trọng của tính năng

• Dánh giá độ chính xác

• Trực quan hóa kết quả

• Dự đoán mới

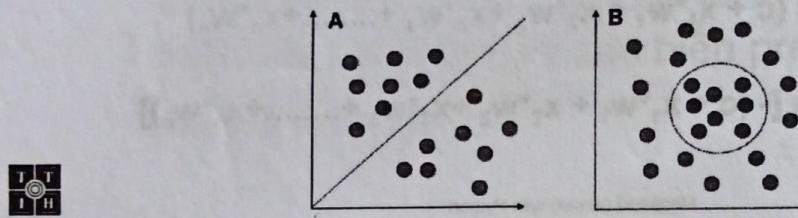


# Logistic Regression

## ❑ Khuyết điểm

- Ranh giới quyết định tuyến tính: Mô hình này chỉ phù hợp với loại dữ liệu mà hai class là gần với linearly separable. Một kiểu dữ liệu mà Logistic Regression không làm việc được là khi dữ liệu mà một class chứa các điểm nằm trong một vòng tròn, class kia chứa các điểm bên ngoài đường tròn đó.

Linear vs. nonlinear problems



# Logistic Regression

## ❑ Khuyết điểm

- Một hạn chế nữa của Logistic Regression là nó yêu cầu các điểm dữ liệu được tạo ra một cách *độc lập* với nhau. Trên thực tế, các điểm dữ liệu có thể bị ảnh hưởng bởi nhau.





## ❑ Xây dựng Logistic Regression sử dụng sklearn

- Dùng `sklearn.linear_model. LogisticRegression`

- Đây là một mô hình phân lớp hoạt động hiệu quả, dễ triển khai.



## • Các bước thực hiện

- Chọn model sẽ sử dụng là: `LogisticRegression`
- Đọc dữ liệu, tiền xử lý dữ liệu
- Tạo một tập dữ liệu features (inputs) và một tập target (output) chứa các nhãn cho các mẫu => chia dữ liệu thành hai bộ train và test
- Xây dựng model với training data
- Đánh giá model với test data
- Đánh giá độ chính xác
- Trực quan hóa kết quả
- Dự đoán mới



# Logistic Regression

- Ví dụ: Xây dựng model dự đoán một khách hàng có mua xe hay không dựa trên thông tin về 'Age' và 'EstimatedSalary\_K' (mức lương ước tính – đơn vị tính 1000\$)



Machine Learning with Python

## Logistic Regression

### Đọc dữ liệu

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

data = pd.read_csv("Social_Network_Ads.csv",
                   usecols=['Age', 'EstimatedSalary_K', 'Purchased'])

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
Age           400 non-null int64
EstimatedSalary_K 400 non-null int64
Purchased     400 non-null int64
dtypes: int64(3)
memory usage: 9.5 KB

data.head()
```

	Age	EstimatedSalary_K	Purchased
0	19	19	0
1	35	20	0
2	26	43	0
3	27	57	0
4	19	76	0



## Logistic Regression



Tạo inputs, output  
=> chia bộ dữ liệu  
training dataset :  
test dataset

```
X = data[['Age', 'EstimatedSalary_K']]  
X.head()
```

<u>Age</u>	<u>EstimatedSalary_K</u>
0	19
1	35
2	26
3	27
4	19

```
Y = data['Purchased']
Y.head()

0      0
1      0
2      0
3      0
4      0
Name: Purchased, dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3)
```

## Logistic Regression



## Xây dựng model

```
from sklearn.linear_model import LogisticRegression
```

```
clf = LogisticRegression()
```

```
clf.fit(X_train, Y_train)
```

```
print('Train score: ', clf.score(X_train,Y_train))
```

Train score: 0.8428571428571429

## Đánh giá model

```
print('Test score: ', clf.score(X_test,Y_test))
```

Test score: 0.8333333333333334

```
Xtest = clf.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
```

```
print("Test Accuracy is ", accuracy_score(Y_test,Yhat_test)*100,"%")
```

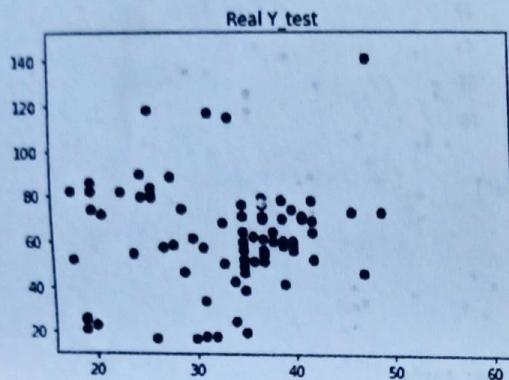
Test Accuracy is 83.33333333333334 %



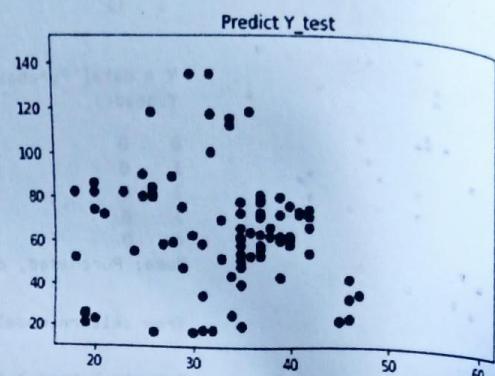
# Logistic Regression

## Trục quan hóa kết quả

```
plt.scatter(X_test.Age , X_test.EstimatedSalary_K,  
           c=Y_test)  
plt.title('Real Y_test')  
plt.show()
```



```
plt.scatter(X_test.Age , X_test.EstimatedSalary_K,  
           c=Yhat_test)  
plt.title('Predict Y_test')  
plt.show()
```



Machine Learning with Python

# Logistic Regression

## Dự đoán mới

```
X_now = [[40,120]]  
Y_now = clf.predict(X_now)  
Y_now  
  
array([1], dtype=int64)
```



## Chapter 3: Demo Multi-class Classification

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

In [2]: from google.colab import drive
drive.mount("/content/gdrive", force_remount=True)

path = '/content/gdrive/My Drive/LDS6_MachineLearning/'

Mounted at /content/gdrive

In [3]: iris = pd.read_excel(path + "practice/Chapter3_Logistic_Regression/Iris.xls")
iris.info()

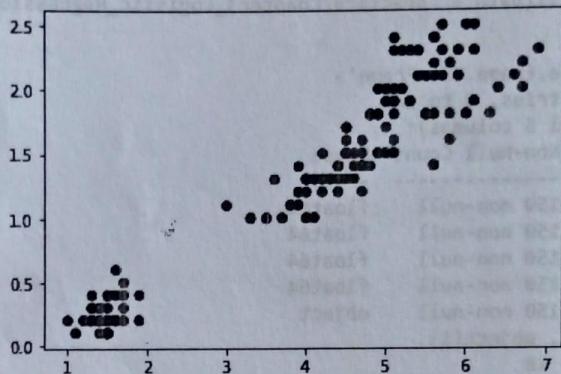
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   sepalength   150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   iris         150 non-null    object  
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
In [4]: iris_class = {'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2}
iris['species_num'] = [iris_class[i] for i in iris.iris]
iris.head()
```

	sepalength	sepalwidth	petallength	petalwidth	iris	species_num
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0

```
In [5]: def make_color(value):
    color = 'yellow'
    if value == 0:
        color = 'red'
    elif value == 1:
        color = 'green'
    else:
        color = 'blue'
    return color
```

```
In [6]: petallength = iris.petallength.values
petalwidth = iris.petalwidth.values
types = iris.species_num.values
color= [make_color(x) for x in types]
plt.scatter(petallength, petalwidth, color=color)
plt.show()
```

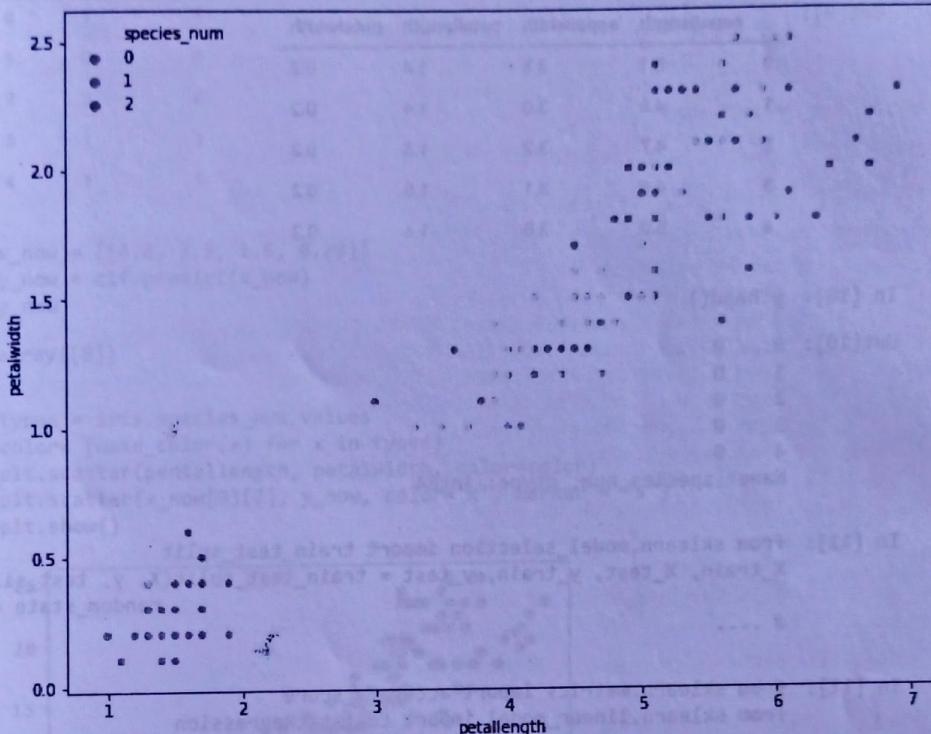


mean_sepal_wid	std_sepal_wid	mean_petal_length	std_petal_length	mean_petal_width	std_petal_width
5.0	0.3739	3.77	1.78	1.78	0.43
5.8	0.3739	4.35	1.74	1.98	0.43
6.5	0.3739	5.15	1.97	2.05	0.37
7.0	0.3739	5.8	1.8	2.3	0.37



```
In [7]: import seaborn as sns
plt.figure(figsize=(10,8))
sns.scatterplot(x="petallength", y="petalwidth",
                 hue="species_num", palette="Set2", data=iris)
plt.show()

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: Future
Warning: pandas.util.testing is deprecated. Use the functions in the public API
at pandas.testing instead.
import pandas.util.testing as tm
```



```
In [8]: X = iris.drop(['iris', 'species_num'], axis=1)
y = iris.species_num
```

```
In [9]: X.head()
```

```
Out[9]:
```

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
In [10]: y.head()
```

```
Out[10]: 0    0
1    0
2    0
3    0
4    0
Name: species_num, dtype: int64
```

```
In [11]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state = 42)
# ...
```

```
In [12]: from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
```

```
In [13]: # https://scikit-Learn.org/stable/modules/generated/sklearn.LinearModel.html
clf = LogisticRegression(solver='lbfgs', multi_class='multinomial')
```

```
In [14]: clf.fit(X_train, y_train)
# Tham so C???? => value???
```

```
Out[14]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                           intercept_scaling=1, l1_ratio=None, max_iter=100,
                           multi_class='multinomial', n_jobs=None, penalty='l2',
                           random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                           warm_start=False)
```

```
In [15]: y_pred = clf.predict(X_test)
```



```
In [16]: # Kiểm tra độ chính xác
print("The prediction accuracy is: ", clf.score(X_test,y_test)*100,"%")
The prediction accuracy is: 100.0 %
```

```
In [17]: df = pd.DataFrame({'Actual': pd.DataFrame(y_test.values)[0].values,
                           'Prediction': pd.DataFrame(y_pred)[0].values})
df.head()
```

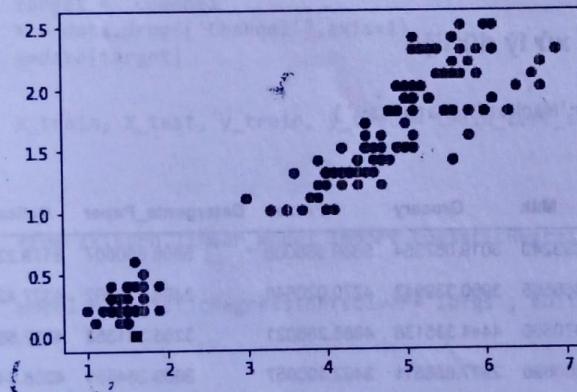
Out[17]:

	Actual	Prediction
0	1	1
1	0	0
2	2	2
3	1	1
4	1	1

```
In [18]: x_now = [[4.8, 3.3, 1.6, 0.25]]
y_now = clf.predict(x_now)
y_now
```

Out[18]: array([0])

```
In [19]: types = iris.species_num.values
color= [make_color(x) for x in types]
plt.scatter(petallength, petalwidth, color=color)
plt.scatter(x_now[0][2], y_now, color='k', marker = 's')
plt.show()
```



In [19]:

## Chapter 3 - Ex1: Marketing Data

- Cho dữ liệu MarketingData.csv chứa số tiền chi tiêu hàng năm của 20.000 khách hàng của một công ty bán lẻ lớn. Nhóm tiếp thị của công ty đã sử dụng các kênh khác nhau để bán hàng hóa và đã phân loại khách hàng dựa trên các giao dịch mua được thực hiện bằng các kênh khác nhau, như sau: 0-Retail (Bán lẻ), 1-Road Show, 2-Social Media và 3-Television.
- Là phụ trách bộ phận Sale, bạn được giao nhiệm vụ xây dựng một mô hình Machine Learning có thể dự đoán kênh hiệu quả nhất có thể được sử dụng để nhắm mục tiêu khách hàng dựa trên chi tiêu hàng năm cho các sản phẩm (features) do công ty bán: Fresh (sản phẩm tươi sống), Milk(sữa), Grocery (tạp hóa), Frozen (sản phẩm đông lạnh), Detergents\_Paper (chất tẩy rửa và giấy) và Delicassen (đồ ăn nhanh).

### Gợi ý

```
In [1]: import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import precision_recall_fscore_support
import matplotlib.pyplot as plt
import seaborn as sns
```

### Đọc dữ liệu, tiền xử lý dữ liệu

```
In [2]: data= pd.read_csv(r'MarketingData.csv')
data.head(5)
```

Out[2]:

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Chann
0	6623.613537	5513.093240	6019.057354	5669.568008	5898.660607	5179.234947	
1	5642.542497	5829.866565	3960.339943	4270.020548	3498.818262	4327.423268	
2	5292.078175	6634.370556	4444.335138	4888.286021	3265.391352	4887.560190	
3	5595.227928	4754.860698	2977.856511	3462.490957	3609.264559	4268.641413	
4	5126.693267	6009.649079	3811.569943	4744.115976	3829.516831	5097.491872	

```
In [3]: data.shape
```

Out[3]: (20000, 7)



```
In [4]: data.isnull().values.any()
```

```
Out[4]: False
```

```
In [5]: data.describe()
```

```
Out[5]:
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Count
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000
mean	5853.350191	5267.873868	4873.362341	4899.477763	4786.331781	5613.672184	5000.000000
std	1128.370297	1177.563192	1265.579790	1220.923393	1154.682284	1343.743103	1100.000000
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	5155.249455	4438.167387	3983.317183	4071.997222	3877.943500	4705.582182	3500.000000
50%	5988.720207	5337.741327	4828.100401	5048.099489	4857.070488	5425.888761	4500.000000
75%	6573.895741	6081.755179	5784.992859	5684.876863	5602.146034	6574.281056	5000.000000
max	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000

```
In [6]: data['Channel'].value_counts()
```

```
Out[6]: 0    5007
        3    5002
        1    5001
        2    4990
Name: Channel, dtype: int64
```

```
In [7]: target = 'Channel'
X = data.drop(['Channel'], axis=1)
y = data[target]
```

```
In [8]: X_train, X_test, y_train, y_test = train_test_split(X.values, y,
                                                       test_size=0.20,
                                                       random_state=42)
```

```
In [9]: from sklearn.linear_model import LogisticRegression
```

```
In [10]: model = LogisticRegression(solver='lbfgs', multi_class='multinomial')
```

```
In [11]: model.fit(X_train,y_train)
c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py
7: ConvergenceWarning: lbfgs failed to converge. Increase the number of iterations.
"of iterations.", ConvergenceWarning)

Out[11]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=100,
                             multi_class='multinomial', n_jobs=None, penalty='l2',
                             random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                             warm_start=False)

In [12]: y_pred=model.predict(X_test)

In [13]: model.score(X_train, y_train)
Out[13]: 0.6745625

In [14]: model.score(X_test, y_test)
Out[14]: 0.67925

In [15]: precision_recall_fscore_support(y_test, y_pred, average='macro')
Out[15]: (0.6750866854839646, 0.6785115125221878, 0.6761930481883371, None)

In [16]: precision_recall_fscore_support(y_test, y_pred, average='micro')
Out[16]: (0.67925, 0.67925, 0.67925, None)

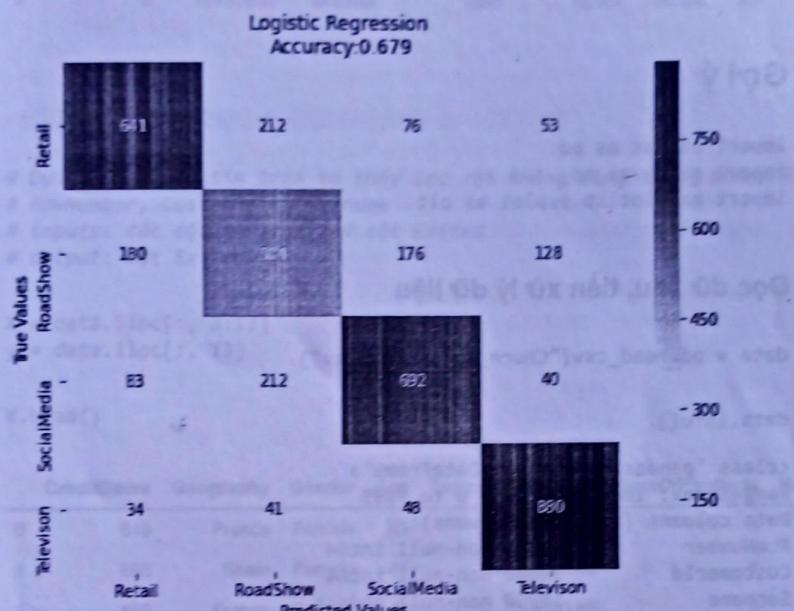
In [17]: target_names = ["Retail","RoadShow","SocialMedia","Televison"]
         print(classification_report(y_test, y_pred,target_names=target_names))
         precision    recall   f1-score   support
          Retail      0.68      0.65      0.67      982
          RoadShow     0.52      0.51      0.52      988
          SocialMedia   0.70      0.67      0.69     1027
          Televison     0.80      0.88      0.84     1003
          accuracy           0.68      0.68      0.68      4000
          macro avg       0.68      0.68      0.68      4000
          weighted avg    0.68      0.68      0.68      4000

In [18]: cm = confusion_matrix(y_test, y_pred)
cm

Out[18]: array([[641, 212, 76, 53],
               [180, 504, 176, 128],
               [ 83, 212, 692, 40],
               [ 34,  41,  48, 880]], dtype=int64)
```



```
In [19]: cm_df = pd.DataFrame(cm,
                           index = target_names,
                           columns = target_names)
plt.figure(figsize=(8,6))
sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
plt.title('Logistic Regression \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, y_pred)))
plt.ylabel('True Values')
plt.xlabel('Predicted Values')
plt.show()
```



#### Nhận xét:

- Model cho độ chính xác ~70% chưa được cao
- Có giải pháp nào khác không?
- Nếu chưa tìm được giải pháp nào phù hợp hơn thì có thể nghĩ đến việc phải thay đổi thuật toán (sẽ học sau)

## Chapter 3 - Ex2: Predicting Customer Churn

- Cho dữ liệu Churn\_Modelling.csv chứa thông tin của 10000 khách hàng của công ty.
- Là phụ trách bộ phận chăm sóc khách hàng bạn nhận thấy việc phải xây dựng một mô hình Machine Learning để dự đoán việc khách hàng sẽ ra đi hay ở lại. Công việc này vô cùng quan trọng vì giữ chân được khách hàng càng lâu doanh nghiệp của bạn sẽ càng tiết kiệm được chi phí và tăng doanh thu.

### Gợi ý

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

### Đọc dữ liệu, tiền xử lý dữ liệu

```
In [2]: data = pd.read_csv("Churn_Modelling.csv")
```

```
In [3]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber      10000 non-null int64
CustomerId     10000 non-null int64
Surname        10000 non-null object
CreditScore    10000 non-null int64
Geography      10000 non-null object
Gender         10000 non-null object
Age            10000 non-null int64
Tenure         10000 non-null int64
Balance        10000 non-null float64
NumOfProducts   10000 non-null int64
HasCrCard      10000 non-null int64
IsActiveMember 10000 non-null int64
EstimatedSalary 10000 non-null float64
Exited         10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```



In [4]: `data.head()`

Out[4]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	Hargrave	619	France	Female	42	2	0.00
1	2	Hill	608	Spain	Female	41	1	83807.86
2	3	Onio	502	France	Female	42	8	159660.80
3	4	Boni	699	France	Female	39	1	0.00
4	5	Mitchell	850	Spain	Female	43	2	125510.82

In [5]: `# Dựa trên thông tin trên ta thấy các cột không dùng trong model là:`

`# RowNumber, CustomerId, Surname`  
`# inputs: các cột còn Lại trừ cột Exited`  
`# output: cột Exited`

In [6]: `X = data.iloc[:, 3:13]`  
`y = data.iloc[:, 13]`

In [7]: `X.head()`

Out[7]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	619	France	Female	42	2	0.00	1	1	1
1	608	Spain	Female	41	1	83807.86	0	1	0
2	502	France	Female	42	8	159660.80	3	1	1
3	699	France	Female	39	1	0.00	2	0	0
4	850	Spain	Female	43	2	125510.82	1	1	1

In [8]: `y.where(y==0).count() # khách hàng ở Lại`

Out[8]: 7963

In [9]: `y.where(y==1).count() # khách hàng ra đi`

Out[9]: 2037

In [10]: `# Các thuộc tính phân loại`

`objects = [f for f in X.columns if X.dtypes[f] == 'object']`  
`objects`

Out[10]: `['Geography', 'Gender']`

In [11]: # Xem xét thuộc tính phân loại: Geography  
 X.groupby(by='Geography')['CreditScore'].count()

Out[11]: Geography  
 France 5014  
 Germany 2509  
 Spain 2477  
 Name: CreditScore, dtype: int64

In [12]: # Dựa trên kết quả ta thấy có 3 quốc gia  
 # => cần chuyển sang dữ liệu kiểu số

In [13]: # Xem xét thuộc tính phân loại: Gender  
 X.groupby(by='Gender')['CreditScore'].count()

Out[13]: Gender  
 Female 4543  
 Male 5457  
 Name: CreditScore, dtype: int64

In [14]: # Dựa trên kết quả ta thấy có 2 giới tính  
 # => cần chuyển sang dữ liệu kiểu số

In [15]: X\_new = pd.get\_dummies(X)

In [16]: X\_new.head()

Out[16]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	En
0	619	42	2	0.00		1	1	1
1	608	41	1	83807.86		1	0	1
2	502	42	8	159660.80		3	1	0
3	699	39	1	0.00		2	0	0
4	850	43	2	125510.82		1	1	1

In [17]: # from sklearn.preprocessing import StandardScaler  
 # sc = StandardScaler()

In [18]: # X\_new\_1 = sc.fit\_transform(X\_new)

### Áp dụng model, nhận xét kết quả

In [19]: from sklearn.model\_selection import train\_test\_split



```
In [20]: X_train, X_test, y_train, y_test = train_test_split(X_new, y,
                                                       test_size = 0.2,
                                                       random_state = 0)

In [21]: from sklearn.linear_model import LogisticRegression

In [22]: lr = LogisticRegression()

In [23]: lr.fit(X_train, y_train)
        c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py:43
        2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
        solver to silence this warning.
        FutureWarning)

Out[23]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                           intercept_scaling=1, l1_ratio=None, max_iter=100,
                           multi_class='warn', n_jobs=None, penalty='l2',
                           random_state=None, solver='warn', tol=0.0001, verbose=0,
                           warm_start=False)

In [24]: print('Train score: ', lr.score(X_train,y_train))
        Train score:  0.788625

In [25]: print('Test score: ', lr.score(X_test,y_test))
        Test score:  0.786

In [26]: yhat_test = lr.predict(X_test)
        yhat_test

Out[26]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [27]: from sklearn.metrics import accuracy_score, precision_score, recall_score
        \_\_init\_\_()

In [28]: print("Test Accuracy is ", accuracy_score(y_test,yhat_test)*100,"%")
        Test Accuracy is  78.60000000000001 %

In [29]: from sklearn.metrics import confusion_matrix, precision_score, recall_score
        \_\_init\_\_()

In [30]: cm = confusion_matrix(y_test, yhat_test)

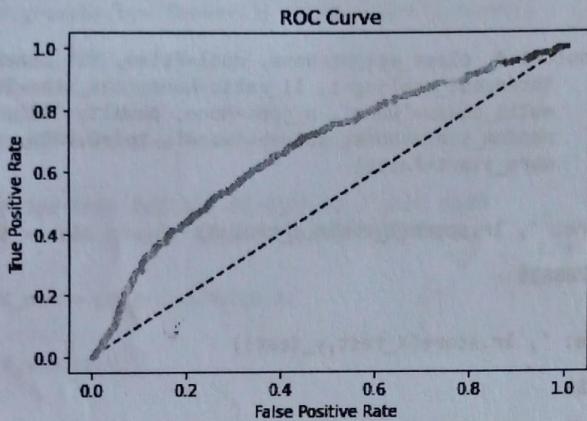
In [31]: cm
        Out[31]: array([[1544,    51],
                      [ 377,   28]], dtype=int64)

In [32]: from sklearn.metrics import roc_curve,auc
```

```
In [33]: # Print ROC_AUC score using probabilities
probs = lr.predict_proba(X_test)
```

```
In [34]: scores = probs[:,1]
fpr, tpr, thresholds = roc_curve(y_test, scores)
```

```
In [35]: plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```



```
In [36]: auc(fpr, tpr)
```

```
Out[36]: 0.6781191222570533
```

### Predicting new samples

```
In [37]: # Cần phải chuẩn hóa dữ liệu để mẫu mới có cùng cấu trúc với dữ liệu đang
```

```
In [38]: columns = X_new.columns
columns
```

```
Out[38]: Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
   'IsActiveMember', 'EstimatedSalary', 'Geography_France',
   'Geography_Germany', 'Geography_Spain', 'Gender_Female', 'Gender_Male'],
  dtype='object')
```

```
In [39]: new_samples = X.iloc[[0, 1, 2]]
```

```
In [40]: new_samples = pd.get_dummies(new_samples)
```



```
In [41]: new_samples.columns
Out[41]: Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
   'IsActiveMember', 'EstimatedSalary', 'Geography_France',
   'Geography_Spain', 'Gender_Female'],
  dtype='object')

In [42]: missing_cols = set(X_new.columns) - set(new_samples.columns)
missing_cols
Out[42]: {'Gender_Male', 'Geography_Germany'}

In [43]: for c in missing_cols:
    new_samples[c] = 0
    # Ensure the order of column in the test set
    # is in the same order than in train set
new_samples = new_samples[X_new.columns]

In [44]: new_samples.columns
Out[44]: Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
   'IsActiveMember', 'EstimatedSalary', 'Geography_France',
   'Geography_Germany', 'Geography_Spain', 'Gender_Female', 'Gender_Male'],
  dtype='object')

In [45]: #new_samples = sc.transform(new_samples)

In [46]: new_predictions = lr.predict(new_samples)
new_predictions
Out[46]: array([0, 0, 0], dtype=int64)

In [47]: # Nhận xét kết quả
# Có giải pháp nào giúp cho kết quả cải thiện hơn không?
```

## Giải pháp

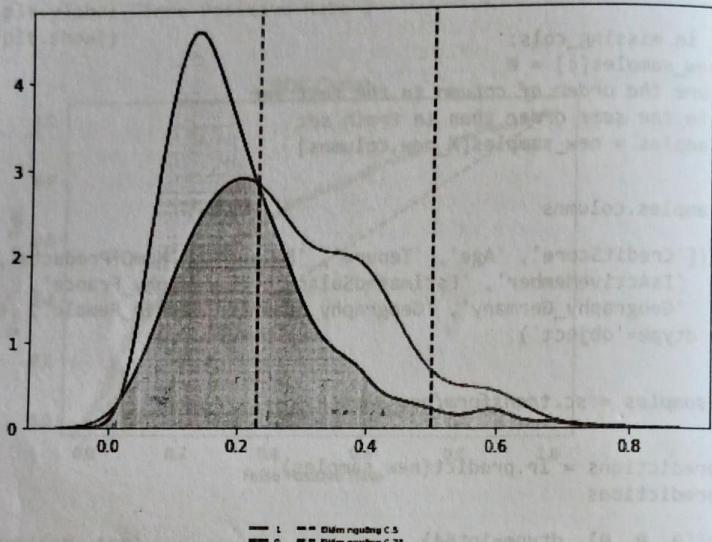
### Đề xuất 1: Điều chỉnh ngưỡng

```
In [48]: # Dự đoán xác suất khách hàng bỏ đi
pos_label= 1
pos_index= np.where(lr.classes_ == pos_label)[0][0]
neg_index= np.where(lr.classes_ != pos_label)[0][0]
neg_label= lr.classes_[neg_index]
# Dự đoán xác suất
y_predict_proba= lr.predict_proba(X_test)
# Tách xác suất khách hàng bỏ đi
pos_proba= y_predict_proba[:, pos_index]
```

```
In [49]: import seaborn as sns
```

In [50]: # Trực quan phân phối xác suất khách hàng ra đi để điều chỉnh điểm ngưỡng

```
plt.figure(figsize= (8, 5))
sns.distplot(pos_proba[y_test== pos_label], label= pos_label,
             color= 'r', kde_kws={"shade": True}, hist=False, norm_hist=True)
sns.distplot(pos_proba[y_test== neg_label], label= neg_label,
             color= 'b', kde_kws={"shade": True}, hist=False, norm_hist=True)
plt.axvline(x= 0.5, linestyle= '--', color= 'black', label= 'Điểm ngưỡng 0.5')
plt.axvline(x= 0.23, linestyle= '--', color= 'g', label= 'Điểm ngưỡng 0.23')
plt.legend(fontsize= 'xx-small', bbox_to_anchor=(0.3, -0.3, 0.3, 0.2),
           ncol=2, loc='lower center')
plt.show()
```



In [51]: # Với ngưỡng mới: 0.23

In [52]: y\_hat\_test\_pro = lr.predict\_proba(X\_test)

In [53]: y\_hat\_test\_pro[:5]

Out[53]: array([[0.79754585, 0.20245415],  
[0.64651623, 0.35348377],  
[0.82521651, 0.17478349],  
[0.90879767, 0.09120233],  
[0.8208282 , 0.1791718 ]])

In [54]: y\_hat\_now = y\_hat\_test\_pro[:,1] > 0.23  
print(y\_hat\_now)

[False True False ... True False False]

In [55]: print("Test Accuracy is ", accuracy\_score(y\_test,y\_hat\_now)\*100,"%")  
Test Accuracy is 66.95 %



```
In [56]: cm1 = confusion_matrix(y_test, y_hat_now)
cm1
```

```
Out[56]: array([[1109,  486],
   [ 175,  230]], dtype=int64)
```

```
In [57]: fpr1, tpr1, thresholds1 = roc_curve(y_test, y_hat_now)
```

```
In [58]: auc(fpr1, tpr1)
```

```
Out[58]: 0.6315995201052671
```

### Predicting new samples with new threshold

```
In [59]: lr.predict_proba(new_samples)[:,1]>0.23
```

```
Out[59]: array([False,  True,  True])
```

```
In [60]: # Nhận xét kết quả
# Có giải pháp nào giúp cho kết quả cải thiện hơn không?
```

## Đề xuất 2: Resampling

```
In [61]: from imblearn.over_sampling import SMOTE
method = SMOTE(kind='borderline1')
```

```
Using TensorFlow backend.
```

```
In [62]: # Apply resampling to the training data only
X_resampled, y_resampled = method.fit_sample(X_train, y_train)
```

```
In [63]: # Count the occurrences of fraud and no fraud and print them
occ_no = y_resampled[y_resampled==0].size
print(occ_no)
```

```
6368
```

```
In [64]: occ_fraud = y_resampled[y_resampled==1].size
print(occ_fraud)
```

```
6368
```

10/02/2020

```
In [65]: # Continue fitting the model and obtain predictions  
model = LogisticRegression()  
model.fit(X_resampled, y_resampled)
```

```
c:\program files\python36\lib\site-packages\sklearn\linear_model\logistic.py:102: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specifying 'newton-cg', 'lbfgs', 'sag', 'saga' or 'sagam' as solver will silence this warning.  
FutureWarning
```

```
Out[65]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
                             intercept_scaling=1, l1_ratio=None, max_iter=100,  
                             multi_class='warn', n_jobs=None, penalty='l2',  
                             random_state=None, solver='warn', tol=0.0001, verbose=0,  
                             warm_start=False)
```

```
In [66]: # training score  
model.score(X_resampled, y_resampled)
```

```
Out[66]: 0.6812971105527639
```

```
In [67]: # testing score  
model.score(X_test, y_test)
```

```
Out[67]: 0.666
```

```
In [68]: # Get your performance metrics  
y_pred = model.predict(X_test)
```

```
In [69]: conf_mat = confusion_matrix(y_true=y_test, y_pred=y_pred)  
print('Confusion matrix:\n', conf_mat)
```

```
Confusion matrix:  
[[1045 550]  
 [118 287]]
```

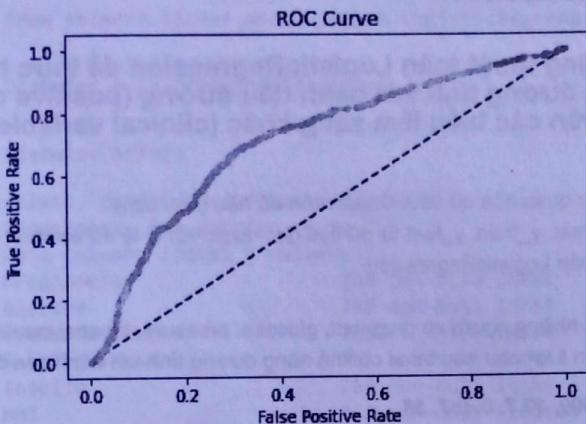
```
In [70]: # Print ROC_AUC score using probabilities  
probs = model.predict_proba(X_test)
```

```
In [71]: from sklearn.metrics import roc_curve, auc
```

```
In [72]: scores = model.predict_proba(X_test)[:,1]  
fpr, tpr, thresholds = roc_curve(y_test, scores)
```



```
In [73]: plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr, tpr, marker='.')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```



```
In [74]: auc(fpr, tpr)
```

```
Out[74]: 0.714701033321723
```

### Predicting new samples

```
In [75]: new_predictions = model.predict(new_samples)
new_predictions
```

```
Out[75]: array([0, 1, 1], dtype=int64)
```

### Kết luận:

- Kết quả nào phù hợp hơn với bài toán này? Tại sao?
- Nếu chưa tìm được giải pháp nào phù hợp hơn thì có thể nghĩ đến việc phải thay đổi thuật toán (sẽ học sau)

## Chapter 3 - Ex3: Diabetes

Cho dữ liệu diabetes.csv

**Yêu cầu:** Áp dụng thuật toán LogisticRegression để thực hiện việc đoán khả năng dương tính với bệnh tiểu đường (positive diabetes outputs) dựa trên các biến lâm sàng khác (clinical variables - inputs)

1. Đọc dữ liệu, trực quan hóa dữ liệu. Chuẩn hóa dữ liệu (nếu cần)
2. Tạo X\_train, X\_test, y\_train, y\_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.3
3. Áp dụng thuật toán LogisticRegression
4. Tìm kết quả
5. Hãy cho biết với những người có pregnant, glucose, pressure, triceps, insulin, mass, pedigree, age lần lượt như sau thì ai có khả năng dương tính với bệnh tiểu đường. ai không

8, 176, 90, 34, 300, 33.7, 0.467, 58

1, 100, 66, 15, 56, 23.6, 0.666, 26

12, 88, 74, 40, 54, 35.3, 0.378, 48

## Diabetes

Thông tin các cột dữ liệu

1. Pregnancies: số lần mang thai
2. Glucose: Nồng độ glucose huyết tương 2 giờ trong thử nghiệm dung nạp glucose đường uống
3. BloodPressure: Huyết áp tâm trương (mm Hg)
4. SkinThickness: độ dày da gấp Triceps skin fold thickness (mm)
5. Insulin: 2-Hour serum insulin (mu U/ml). insulin huyết thanh 2-giờ
6. BMI: (weight in kg/(height in m)^2)
7. DiabetesPedigreeFunction: Diabetes pedigree function
8. Age: Age (years)
9. Outcome: Class variable (0 or 1)

**Chú ý:** Tất cả các biến trên liên tục, mục đích là dự đoán ai đó có bị tiểu đường hay không (Outcome=1) dựa trên các biến khác. Các mẫu lấy từ phụ nữ trên 21 years old.

```
In [1]: # from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter3_Logis'
```



```
In [2]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model_selection import train_test_split  
import math
```

```
In [3]: from sklearn.linear_model import LogisticRegression
```

```
In [4]: Diabetes = pd.read_csv("diabetes.csv")
```

```
In [5]: Diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 768 entries, 0 to 767  
Data columns (total 9 columns):  
Pregnancies           768 non-null int64  
Glucose               768 non-null int64  
BloodPressure         768 non-null int64  
SkinThickness         768 non-null int64  
Insulin               768 non-null int64  
BMI                  768 non-null float64  
DiabetesPedigreeFunction 768 non-null float64  
Age                  768 non-null int64  
Outcome              768 non-null int64  
dtypes: float64(2), int64(7)  
memory usage: 54.1 KB
```

```
In [6]: Diabetes.head()
```

```
Out[6]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	
0	6	148	72	35	0	33.6		0.627
1	1	85	66	29	0	26.6		0.351
2	8	183	64	0	0	23.3		0.672
3	1	89	66	23	94	28.1		0.167
4	0	137	40	35	168	43.1		2.288

## Chapter 3 - Ex4: LowBirthWeight

Cung cấp dữ liệu birthweight\_reduced.csv

**Yêu cầu: Áp dụng Logistic Regression để thực hiện việc xác định liệu có thiếu cân hay không dựa vào thông tin còn lại.**

1. Hãy đọc dữ liệu từ tập tin này. Chuẩn hóa dữ liệu nếu cần.
2. Tạo X\_train, X\_test, y\_train, y\_test từ dữ liệu đọc được với tỷ lệ dữ liệu test là 0.3
3. Áp dụng thuật toán Logistic Regression
4. Kiểm tra độ chính xác
5. Tìm kết quả Cho dữ liệu Test: X\_now = [[12, 18, 4.5, 35, 1, 41, 7, 65, 125, 37, 14, 25, 68, 1]]]

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import math
from pandas import DataFrame
from sklearn import preprocessing
```

```
In [2]: # https://www.sheffield.ac.uk/mash/data
data = pd.read_csv("birthweight_reduced.csv")
```

```
In [3]: data.head()
```

```
Out[3]:
```

0	1313		12	17	5.8	33	0	24	0				
1	431		12	19	4.2	33	1	20	7				
2	808		13	19	6.4	34	0	26	0				
3	300		12	18	4.5	35	1	41	7				
4	516		13	18	5.8	35	1	20	35				

In [4]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
Data columns (total 17 columns):
id           42 non-null int64
headcircumference 42 non-null int64
length        42 non-null int64
Birthweight   42 non-null float64
Gestation     42 non-null int64
smoker        42 non-null int64
motherage     42 non-null int64
mnocig        42 non-null int64
mheight       42 non-null int64
mppwt         42 non-null int64
fage          42 non-null int64
fedyrs        42 non-null int64
fnocig        42 non-null int64
fheight       42 non-null int64
lowbwt        42 non-null int64
mage35        42 non-null int64
LowBirthWeight 42 non-null object
dtypes: float64(1), int64(15), object(1)
memory usage: 5.7+ KB
```

In [5]: `data.describe()`

Out[5]:

	<b>id</b>	<b>headcircumference</b>	<b>length</b>	<b>Birthweight</b>	<b>Gestation</b>	<b>smoker</b>	<b>motherage</b>	
count	42.000000	42.000000	42.000000	42.000000	42.000000	42.000000	42.000000	42
mean	894.071429	13.261905	19.928571	7.264286	39.190476	0.523810	25.547619	9
std	467.616186	0.766987	1.112958	1.329739	2.643336	0.505487	5.666342	12
min	27.000000	12.000000	17.000000	4.200000	33.000000	0.000000	18.000000	0
25%	537.250000	13.000000	19.000000	6.450000	38.000000	0.000000	20.250000	0
50%	821.000000	13.000000	20.000000	7.250000	39.500000	1.000000	24.000000	4
75%	1269.500000	14.000000	21.000000	8.000000	41.000000	1.000000	29.000000	15
max	1764.000000	15.000000	22.000000	10.000000	45.000000	1.000000	41.000000	50