ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN

MÔN HỌC: THỊ GIÁC MÁY TÍNH NÂNG CAO

ĐỀ TÀI

TÌM HIỂU VỀ LOGISTIC REGRESSION

Giảng viên hướng dẫn: TS. Nguyễn Vinh Tiệp

Sinh viên thực hiện: Trương Quốc Bình -19521270
Nguyễn Hữu Hưng – 19521571
Nguyễn Quan Huy - 19521622

Lớp: CS331.M21.KHCL

Thành phố Hồ Chí Minh, tháng 4 năm 2022

MỤC LỤC

LỜI MỞ ĐẦU

Trong những năm gần đây, các lĩnh vực nghiên cứu của ngành Khoa học máy tính phát triển hết sức mạnh mẽ, nhiều thuật toán ra đời với nhiều hướng nghiên cứu khác nhau. Trong đó Deep learning là một hướng nghiên cứu sâu của Machine Learning đã xuất hiện gần đây và đạt rất nhiều thành tựu. Nhiều thuật toán Deep Learning được ứng dụng trong thực tế mang lại hiệu quả trong các lĩnh vực như: Ứng dụng xe tự đông, Trợ lí ảo Siri, Mô phỏng và nhận diện ảnh,. . . Các thuật toán được dùng phổ biến như: Linear Regression, Logistic Regression, Neural Network,… Mỗi thuật toán đều có những ưu và nhược điểm khác nhau, được cải tiến trong nhiều năm và ảnh hưởng rất lớn đến những thành tựu khoa học kĩ thuật tương lai.

Trong tiểu luận này, mục tiêu của nhóm chúng tôi là tìm hiểu vềLogistic Regression, bài toán áp dụng, phương pháp sử dụng Logistic Regression, ưu và nhược điểm của phương pháp, cài đặt phương pháp và bài tập về Logistic Regression.

Chúng tôi cũng xin chân thành cảm ơn thầy Nguyễn Vinh Tiệp đã tận tình giảng dạy chúng tôi môn học này để em có thể hoàn thành tiểu luận này một cách tốt nhất.

Chương 1. GIỚI THIỆU

1.1 Giới thiệu Logistic Regression.

Logistic Regression là thuật toán trong deep learning, đây là thuật toán đơn giản nhưng lại rất hiệu quả trong bài toán phân loại (Classification).

Logistic Regression được áp dụng trong bài toán phân loại nhị phân (Binary classification) tức ta sẽ có hai output, hoặc có thể gọi là hai nhãn (ví dụ như 0 và 1 hoặc cat và non-cat).

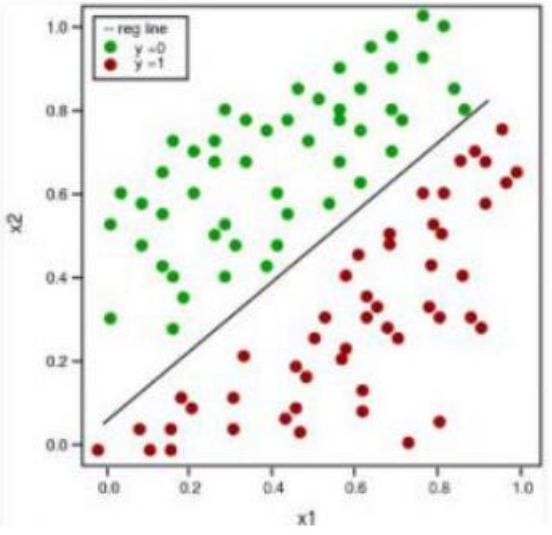
Logistic Regression thường được sử dụng để so sánh với các thuật toán phân loại khác.

Logistic Regression là một phương pháp phân tích thống kê được sử dụng để dự đoán giá trị dữ liệu dựa trên các quan sát trước đó của tập dữ liệu.

Mục đích của Logistic Regression là ước tính xác suất của các sự kiện, bao gồm xác định mối quan hệ giữa các tính năng từ đó đự đoán xác suất của các kết quả, nên đối với hồi quy logistic ta sẽ có:

Input: dữ liệu input (ta sẽ coi có hai nhãn là 0 và 1).

Output: Xác suất dữ liệu input rơi vào nhãn 0 hoặc nhãn 1.



Hình 1. Mô hình Logistic Regression

Ở hình trên ta gọi các điểm màu xanh là nhãn 0 và các điểm màu đỏ là nhãn 1 đối với hồi quy logistic ta sẽ biết được với mỗi điểm thì xác xuất rơi vào nhãn 0 là bao nhiêu và xác suất rơi vào nhãn 1 là bao nhiêu, ta có thể thấy giữa hai màu xanh và màu đỏ có một đường thẳng để phân chia rất rõ ràng nhưng nếu các điểm dữ liệu mà không nằm sang hai bên mà nằm trộn lẫn nhiều vào nhau thì ta sẽ phân chia như nào ? khi đó ta sẽ gọi tập dữ liệu có nhiều nhiễu và ta phải xử lí trước các nhiễu đó.

1.2 So sánh Logistic Regression với Linear Regression

Linear Regression và Logistic Regression là hình thức hồi quy cơ bản nhất thường được sử dụng. Sự khác biệt cơ bản giữa hai điều này là hồi quy Logistic được sử dụng khi biến phụ thuộc có bản chất là nhị phân. Ngược lại, hồi quy tuyến tính được sử dụng khi biến phụ thuộc là liên tục và bản chất của đường hồi quy là tuyến tính.

Regression là một kỹ thuật được sử dụng để dự đoán giá trị của một biến phản hồi (phụ thuộc), từ một hoặc nhiều biến dự báo (độc lập), trong đó biến là số. Có nhiều dạng hồi quy khác nhau như tuyến tính, nhiều, logistic, đa thức, không tham số, …

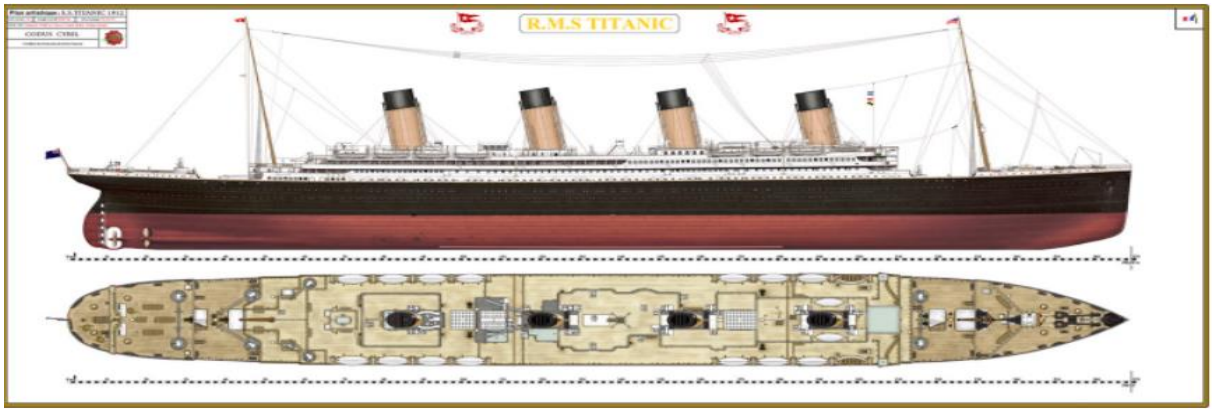
Bảng so sánh:

Cơ sở để so sánh	Linear Regression	Logistic Regression
Căn bản	Dữ liệu được mô hình hóa bằng cách sử dụng một đường thẳng.	Xác suất của một số sự kiện thu được được biểu diễn dưới dạng một hàm tuyến tính của tổ hợp các biến dự báo.
Mối quan hệ tuyến tính giữa các biến phụ thuộc và độc lập	Bắt buộc	Không yêu cầu
Biến độc lập	Có thể tương quan với nhau. (Đặc biệt trong hồi quy tuyến tính nhiều lần)	Không nên tương quan với nhau (không tồn tại đa cộng tuyến)

1.3 Đưa vào ngữ cảnh bài toán

Trong bài đăng trên một blog, chúng tôi sẽ giới thiệu cho các bạn toàn bộ quá trình tạo ra một mô hình Machine Learning trên bộ dữ liệu Titanic nổi tiếng, được nhiều người trên thế giới sử dụng. Nó cung cấp thông tin về số phận của các hành khách trên tàu Titanic, được tóm tắt theo tình trạng kinh tế (hạng hang khách), giới tính, tuổi tác và khả năng sống sót.

Ban đầu, tập dữ liệu được lấy từ kaggle.com, như một phần của Cuộc thi "Titanic: Machine Learning from Disaster". Trong tiểu luận này, chúng tôi được sẽ thực hiện bài toán dự đoán liệu một hành khách trên tàu Titanic có sống sót hay không.



Hình 2. Hình ảnh về con tàu Titanic

Nhóm chúng tôi sẽ sử dụng thuật toán Logistic Regression để build model cho bài toán dự đoán liệu một hành khách trên tàu Titanic có sống sót hay không. Vấn đề này sẽ được nói rõ hơn ở chương 2.

Chương 2. PHƯƠNG PHÁP

2.1 Bài toán Logistic Regression

2.1.1 Ý tưởng bài toán

Chúng em sẽ sử dụng tập dữ liệu Titanic kết hợp với mô hình Logistic Regression trong Python.

Đầu vào bao gồm mã khách hàng, tên tuổi, giới tính, mã vé , khoang , ..

Sau khi xử lí dữ liệu và xây dựng mô hình logictic regression thì e dự đoán xem liệu một hành khách có sống sót sau vụ tai nạn tàu Titanic hay không.

2.1.2 Dữ liệu của phương pháp

Ta thấy tập dữ liệu train.csv được lấy từ trang kaggle.com ở đây, là 1 bảng dữ liệu với 891 dòng dữ liệu và 12 cột dữ liệu.

Tử bảng dữ liệu train.csv ta có thể biết được rằng chúng ta cần chuyển đổi nhiều tính năng thành số để các thuật toán Machine Learning có thể dễ dàng xử lý chúng. Cần phải chuyển đổi cho phù hợp các tỷ lệ như giới tính với tỉ lệ sống sót, hay tỉ lệ hạng (hành khách) với tỉ lệ sống sót, … Ngoài ra, ta thấy vẫn còn nhiều giá trị còn thiếu mà ta cần xử lí.

2.2.3 Chuẩn hóa dữ liệu

Chúng ta cần phải chuẩn hóa dữ liệu thành những cột giá trị đơn giản hơn, để dễ dàng xử lí phân tích chúng.

Scale các biến input để mô hình có thể dễ dàng dự đoán sau này. Các bài toán có thể phức tạp hoặc không rõ ràng nên ta không xác định được việc sử dụng kỹ thuật nào để scale dữ liệu là tốt nhất. Vì thế nên thường thì mình hay thử nghiệm scale dữ liệu và không scale có khác biệt nhau thế nào bằng việc cho mô hình chạy rồi tiến hành đánh giá.

Scale biến output là biến đầu ra được dự đoán bởi mô hình. Nếu đầu ra của activation function thuộc vào miền [0, 1] thì giá trị biến đầu ra cũng phải nằm trong miền giá trị này.

Có 2 cách để scale dữ liệu đó là normalization và standardization (chuẩn hóa dữ liệu). Normalization là phương pháp scale dữ liệu từ miền giá trị bất kì sang miền giá trị nằm trong khoảng 0 đến 1. Chuẩn hóa dữ liệu là việc scale dữ liệu về một phân bố trong đó giá trị trung bình của các quan sát bằng 0 và độ lệch chuẩn = 1. Nhờ việc chuẩn hóa, các thuật toán như linear regression, logistic regression được cải thiện.

Khi sử dụng các kỹ thuật Machine Learning để mô hình hóa các bài toán Classification, nên hiểu về tỷ lệ giữa các danh mục. Đối với vấn đề cụ thể này, sẽ hữu ích khi xem có bao nhiêu người sống sót so với những người không sống sót tồn tại trong dữ liệu train.csv.

2.2 Ví dụ minh họa

Ví dụ sẽ được nói rõ trong phần cài đặt chương trình ở chương 3.

2.3 Đặc điểm của Logistic Regression

a. Ưu điểm:

- Không cần chọn tốc độ học.

- Thường chạy nhanh hơn một số phương pháp khác.

- Có thể ước tính độ dốc về mặt số học cho bạn (không phải lúc nào cũng hoạt động tốt).

b. Nhược điểm:

- Phức tạp hơn một số phương pháp khác.

- Thêm một hộp đen trừ khi bạn tìm hiểu các chi tiết cụ thể.

- Hồi quy logistic đa thức.

2.4 Ứng dụng của thuật toán Logistic Regression

Trong tiểu luận này, nhóm chúng tôi đưa ra bốn ứng dụng của việc sử dụng thuật toán Logistic Regression: Ngân hàng, Y học, Thị trường chứng khoán và Thương mại điện tử. Những ứng dụng này, các bạn nên tìm hiểu để hiểu rõ hơn về Logistic Regression.

Chương 3. CÀI ĐẶT CHƯƠNG TRÌNH

3.1. Giới thiệu bài toán

Chúng tôi sẽ sử dụng tập dữ liệu Titanic kết hợp với mô hình Logistic Regression trong Python để dự đoán xem liệu một hành khách có sống sót sau vụ tai nạn tàu Titanic hay không.

Báo cáo và code của bài toán sẽ được đăng lên Kho lưu trữ Github của nhóm - https://github.com/noeffortnomoney/CS331.M21.KHCL/tree/main/Project%20Topic

3.2. Tập dữ liệu

Trong đồ án này, tôi sẽ áp dụng phương pháp Deep Learning với thuật toán model Logistic Regression để dự đoán xem liệu một hành khách có sống sót sau vụ tai nạn tàu Titanic hay không, bằng cách sử dụng một trong những tập dữ liệu được sử dụng nhiều nhất - tập dữ liệu train.csv lấy trên website: https://www.kaggle.com/

Tập dữ liệu trong file train.csv gồm 891 dòng dữ liệu và 12 cột dữ liệu.

```
[ ]  titanic_data.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

Hình 23. 12 cột dữ liệu từ file train.csv

Đây là tên của các cột trong DataFrame. Dưới đây là giải thích ngắn gọn về từng điểm dữ liệu:

PassengerId: một số nhận dạng cho mọi hành khách trên tàu Titanic.

Survived: một mã định danh nhị phân cho biết hành khách có sống sót sau vụ tai nạn tàu Titanic hay không. Biến này sẽ giữ giá trị 1nếu chúng sống sót và 0nếu chúng không tồn tại.

Pclass: hạng hành khách của hành khách được đề cập. Điều này có thể có giá trị 1, 2hoặc 3, tùy thuộc vào vị trí của hành khách trên tàu.

Name: tên của hành khách. '

Sex: Nam hay nữ.

Age: tuổi (tính bằng năm) của hành khách.

SibSp: số anh chị em, vợ chồng trên tàu.

Parch: số lượng cha mẹ và trẻ em trên tàu.

Ticket: số vé của hành khách.

Fare: hành khách đã trả bao nhiêu cho vé của họ trên tàu Titanic.

Cabin: số cabin của hành khách.

Embarked: cảng mà hành khách xuống (C = Cherbourg, Q = Queenstown, S = Southampton)

*** Split Dataset:**

+ Dataset được chia làm 2 set theo tỉ lệ 7:3:

- 624 hàng dữ liệu cho Training set (70%).

- 267 hàng dữ liệu cho Validation set (30%).

```
x_training_data, x_test_data, y_training_data, y_test_data = train_test_split(x_data, y_data, test_size = 0.3)
```

.

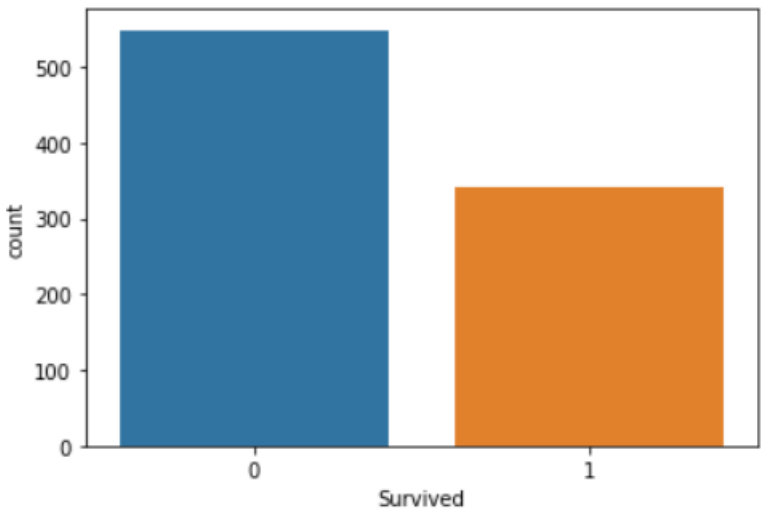
3.3 Xử lí và trực quan hóa dữ liệu

+ Ta cũng xử lí rõ dữ liệu bằng thư viện Seaborn và thư viện Matplotlib để dễ dàng hơn khi train model.

*** Data Exploration**

- Xử lí dữ liệu cột Survived: Xét xem tỷ lệ người sống sót

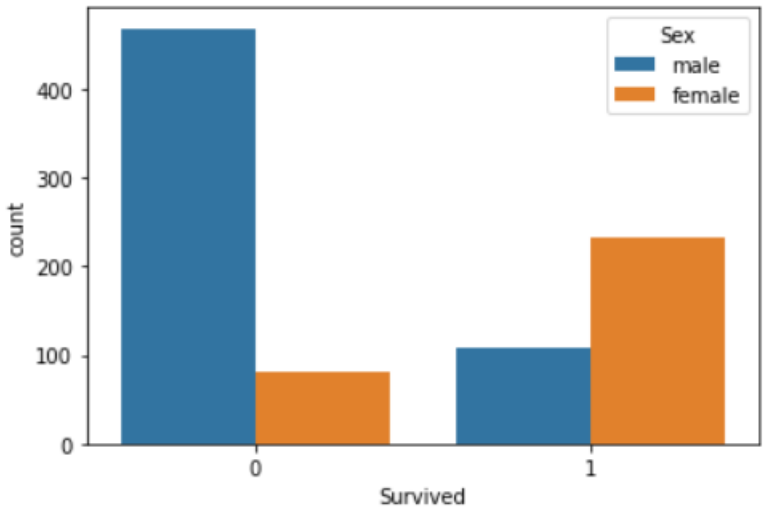
→ Survived = 1: Còn sống
Survived = 0: Đã chết



Hình 21. Biểu đồ Survived

- Xử lí dữ liệu cột Survived: Xét xem tỷ lệ người sống sót với giới tính

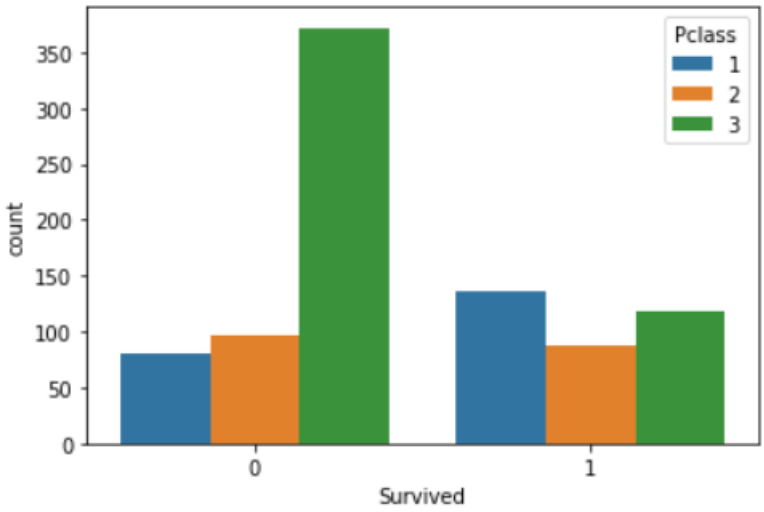
→ Survived = 1: Còn sống
Survived = 0: Đã chết



Hình 22. Biểu đồ Survived và Sex

- Xử lí dữ liệu cột Survived và Pclass: Xét xem tỷ lệ sống sót với hạng hành khách

→ Survived = 1: So
Survived = 0: Nữ



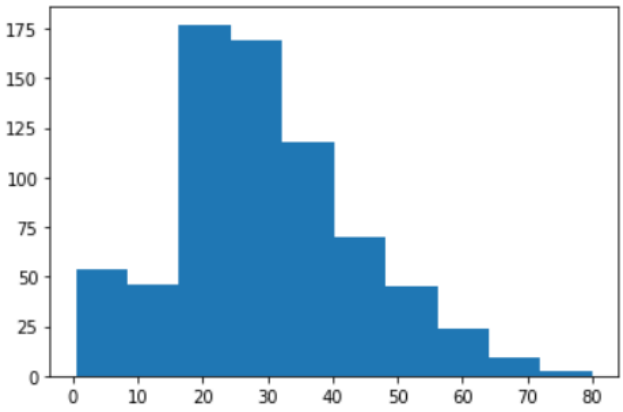
Hình 22. Biểu đồ Survived và Pclass

- Tổng hợp dữ liệu cột theo Age, Ticket, Fare, Pclass

Phân bố tuổi của các hành khác trên tàu.

```
plt.hist(titanic_data['Age'].dropna())
```

```
(array([ 54.,  46., 177., 169., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42 ,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <a list of 10 Patch objects>)
```

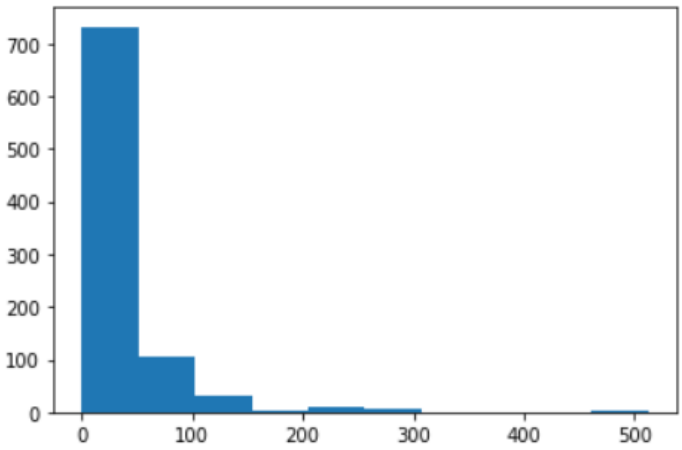


Hình 22. Biểu đồ Heart Disease Frequency for Ages

Phân bố giá vé của các hành khách

```
plt.hist(titanic_data['Fare'])
```

```
(array([732., 106.,  31.,   2.,  11.,   6.,   0.,   0.,   0.,   3.]),
 array([  0.     ,  51.23292, 102.46584, 153.69876, 204.93168, 256.1646 ,
        307.39752, 358.63044, 409.86336, 461.09628, 512.3292 ]),
 <a list of 10 Patch objects>)
```



```
titanic_data.isnull()#ô chứa True nếu nó là giá trị null và False ngược lại
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

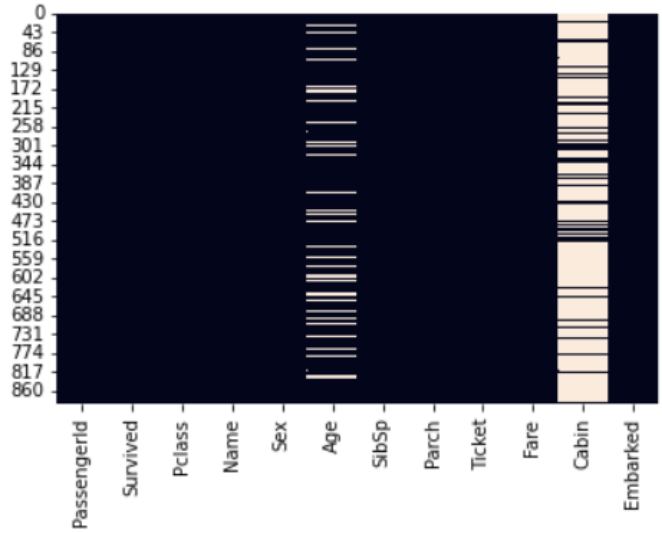
891 rows × 12 columns

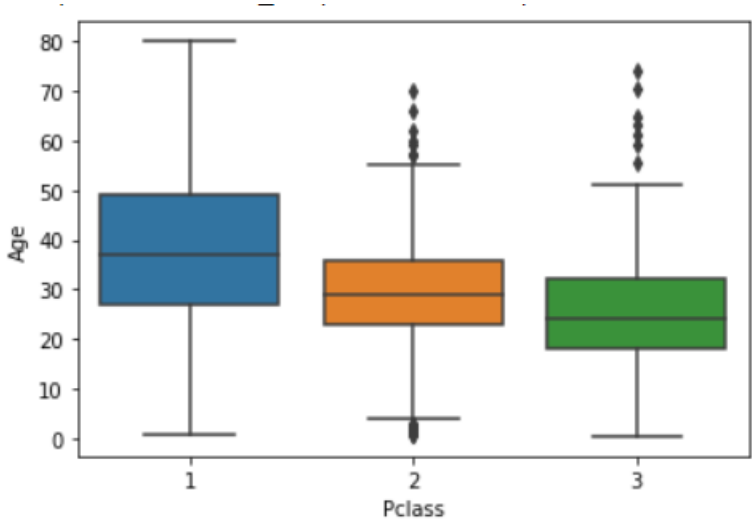
- Xóa dữ liệu và thêm dữ liệu cho cột Age còn thiếu:

Trực quan hóa dữ liệu còn thiếu bằng seaborn

```
sns.heatmap(titanic_data.isnull(), cbar=False)
#Các đường màu trắng cho biết các giá trị bị thiếu trong tập dữ liệu
#Cột Age thiếu ít nên ta có thể dụng imputation để thêm vào.
#Cột Cabin thiếu hầu hết dữ liệu nên ta có thể xóa nó đi
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f09787ad1d0>
```

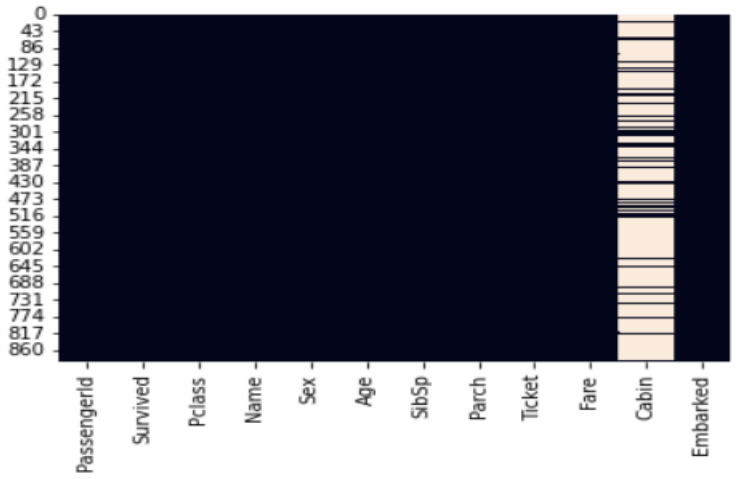





```
#Xác định độ tuổi trung bình cho từng hạng
#Pclass value 1
titanic_data[titanic_data['Pclass'] == 1]['Age'].mean()
#Pclass value 2
titanic_data[titanic_data['Pclass'] == 2]['Age'].mean()
#Pclass 3
titanic_data[titanic_data['Pclass'] == 2]['Age'].mean()
```

```
29.87763005780347
```

```
#Thêm dữ liệu Age còn thiếu
def impute_missing_age(columns):
    age = columns[0]
    passenger_class = columns[1]
    if pd.isnull(age):
        if(passenger_class == 1):
            return titanic_data[titanic_data['Pclass'] == 1]['Age'].mean()
        elif(passenger_class == 2):
            return titanic_data[titanic_data['Pclass'] == 2]['Age'].mean()
        elif(passenger_class == 3):
            return titanic_data[titanic_data['Pclass'] == 3]['Age'].mean()
    else:
        return age
```



- Mục đích của Data Exploration: là tiền xử lí dữ liệu để ta hiểu để ta có thể hiểu rõ được dữ liệu này, làm rõ vấn đề ở đâu rồi đồng bộ chúng để để dữ liệu train dữ liệu 1 cách dễ dàng hơn.

3.4. Train model

Sau khi chia tập dữ liệu thành 7:3, thì ta tiến hành train model.

Train model:

- Sử dụng thư viện skicit-learn để gọi LogisticRegression().

```
model = LogisticRegression()
```

```
model.fit(x_training_data, y_training_data)
```

```
predictions = model.predict(x_test_data)
```

- Độ chính xác của model là : 81%

```
print(classification_report(y_test_data, predictions))
```

```
              precision    recall  f1-score   support

           0       0.81      0.88      0.85       161
           1       0.79      0.69      0.74       106

    accuracy                           0.81       267
   macro avg       0.80      0.79      0.79       267
weighted avg       0.80      0.81      0.80       267
```

Chương 4. BÀI TẬP

Mô tả bài toán: Cài đặt Logistic Regression với Python và Numpy với dữ liệu sinh ra ngẫu nhiên.

- Import thư viện:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
import seaborn as sns
```

- Sau đó, chúng ta sẽ định nghĩa hàm sigmoid theo công thức sigmoid(x)\ =\ 1\ /\ (1\ +\ e^{-x})

```
# Sigmoid function
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

- Tiếp theo là hàm cost, lưu ý rằng hàm cost trong logistic khác hàm cost trong linear. Trong logistic regression, chúng ta sử dụng hàm sigmoid để tính tổng trọng số để cho ra kết quả phi tuyến tính.

```
# Cost function
def compute_cost(X,y,theta):
    m = len(y)
    h=sigmoid(X@theta)
    epsilon = 1e-5
    cost = (1/m)*(((-y).T @ np.log(h+epsilon)) - ((1-y).T @ np.log(1-h + epsilon)))
    return cost
```

- Tiếp theo là tính gradient descent để tối ưu hóa các tham số. Việc thực hiện gradient descent không khác nhiều so với linear regression trừ hàm được sử dụng ở đây là hàm sigmoid.

```
# Gradient Descent
def gradient_descent(X,y,params,learning_rate, iterations):
    m = len(y)
    cost_history =  np.zeros((iterations,1))

    for i in range(iterations):
        params = params - (learning_rate/m) * (X.T @ (sigmoid(X @ params) - y))
        cost_history[i] = compute_cost(X,y,params)

    return (cost_history, params)
```

- Trong hàm dự đoán cho mô hình, nếu giá trị trên 0.5 chúng ta sẽ làm tròn thành 1,

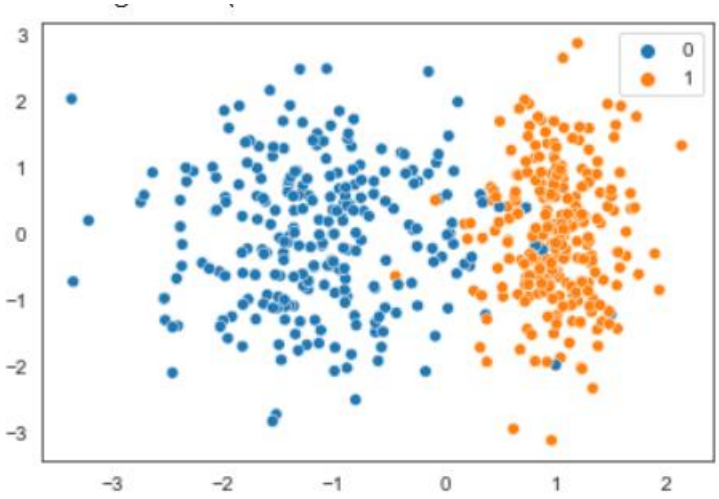
nghĩa là mẫu thuộc về class 1 và ngược lại nếu giá trị dưới 0.5, mẫu thuộc về class 0.

```
# predict function
def predict(X, params):
    return np.round(sigmoid(X @ params))
```

- Implement in random dataset:

- Tiếp theo chúng ta sẽ tạo ra random dataset với hàm make_classification của sklearn.datasets. Chúng ta sẽ tạo ra 500 điểm dữ liệu với 2 class và được visualize theo hình bên dưới

```
# Make random dataset
X,y = make_classification(n_samples=500, n_features=2, n_redundant=0,
                          n_informative=1, n_clusters_per_class=1, random_state=14)

y = y[:, np.newaxis]

sns.set_style('white')
sns.scatterplot(X[:,0], X[:,1], hue=y.reshape(-1));
```

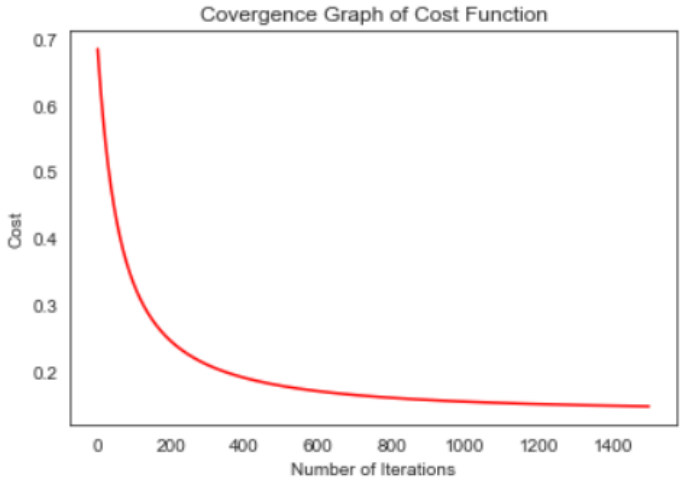


- Bây giờ chúng ta sẽ chạy thuật toán tính toán các tham số.

```
m = len(y)

X = np.hstack((np.zeros((m,1)), X))
n = np.size(X,1)
params = np.zeros((n,1))

iterations = 1500
learning_rate = 0.03
initial_cost = compute_cost(X,y,params)

print(f'Initial Cost is {initial_cost} \n')
(cost_history, params_optimal) = gradient_descent(X,y, params, learning_rate, iterations)

print('Optimal Parameters are: \n', params_optimal, '\n')
```

```
Initial Cost is [[0.69312718]]

Optimal Parameters are:
 [[0.        ]
 [3.17742536]
 [0.03857884]]
```

```
plt.figure()
sns.set_style('white')
plt.plot(range(len(cost_history)), cost_history, 'r')
plt.title('Covergence Graph of Cost Function')
plt.xlabel('Number of Iterations')
plt.ylabel('Cost')
plt.show()
```



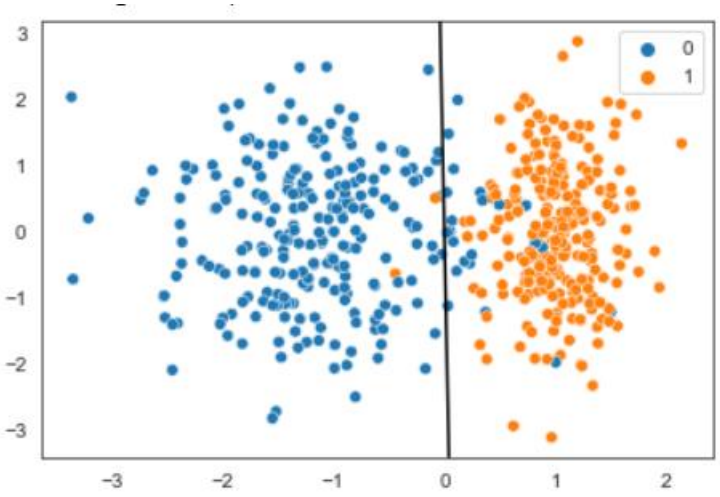
- Độ chính xác: 94,8%

```
y_pred = predict(X, params_optimal)
score = float(sum(y_pred==y))/ float(len(y))

print(score)
```

```
0.948
```

- Sau khi đã chạy thuật toán, chúng ta sẽ vẽ ra đường phân cách giữa 2 class. Như hình bên dưới thì kết quả thu được có độ chính xác khá cao.

```
slope = -(params_optimal[1] / params_optimal[2])
intercept = -(params_optimal[0] / params_optimal[2])

sns.set_style('white')
sns.scatterplot(X[:,1], X[:,2], hue=y.reshape(-1));

ax= plt.gca()
ax.autoscale(False)
x_vals = np.array(ax.get_xlim())
y_vals = intercept + (slope*x_vals)
plt.plot(x_vals, y_vals, c='k');
```


0
1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2

Chương 5. KẾT LUẬN

5.1 Kết luận

Phần báo cáo tiểu luận đã trình bày những giới thiệu, nêu ra Logistic Regression áp dụng cho những bài toán nào, so sánh Logistic Regression với Linear Regression, ngữ cảnh bài toán về Logistic Regression, các phương pháp áp dụng và ưu nhược điểm phương pháp … xây dựng bài toán và cài đặt chương trình, cài đặt Logistic Regression với Python và Numpy với dữ liệu sinh ra ngẫu nhiên.

Trong báo cáo đồ án chúng tôi đã tìm hiểu cơ sở lý thuyết thuật toán Logistic Regression. Ta có thể thấy Logistic Regression là một thuật toán hiệu quả, chính xác cao, cần được đầu tư phát triển cho các ứng dụng thực tiễn và đời sống. Tuy nhiên thuật toán vẫn còn tồn tại một số nhược điểm cần phải được cải thiện sau này. Có thể sau này ta có thể cải thiện thuật toán bằng nhiều phương pháp khác và chúng tôi hi vọng sẽ thấy được điều đó sau này.

Thông qua những nghiên cứu về thuật toán Logistic Regression và ứng dụng, cho thấy nó là một thuật toán hiệu quả, chính xác cao, cần được đầu tư phát triển cho các ứng dụng thực tiễn và đời sống. Bản thân chúng tôi đã hoàn thành tiểu luận tuy nhiên vẫn còn nhiều thiếu xót, cần đào sâu nghiên cứu và trau dồi thêm nhiều kiến thức.

TÀI LIỆU THAM KHẢO

[1] https://www.geeksforgeeks.org/understanding-logistic-regression

[2] http://maitrongnghia.com/2020/04/logistic-regression/

[3] https://machinelearningcoban.com/2017/08/31/evaluation/

[4] https://www.dathoangblog.com/2019/01/logistic-regression.html

[5]https://viblo.asia/p/logistic-regression-bai-toan-co-ban-trong-machine-learning-924lJ4rzKPM

[6] https://cole.vn/blog/quy-trinh-phan-tich-du-lieu/

[7] J. Ross Quinlan ,C4.5 Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning), 1993.

[8] http://tutorials.aiclub.cs.uit.edu.vn/index.php/2021/05/12/logistic-regression/

[9] https://vi.living-in-belgium.com/difference-between-linear-and-logistic-regression-292