# Printical Component Analysis – PCA

## Giới thiệu PCA

Printical Component Analysis là phép phân tích thành phần chính mục đích chính nhằm biến đổi cũng như giảm chiều (feature/attribute) tập dataset dựa vào phương sai và các hệ số tương qua giữa các chiều, tập dataset mới sẽ có số chiều ít hơn mang hầu hết các thông tin của tập dataset cũ.

## Giới thiệu Software Defect Prediction Dataset

Dataset với các số liệu của một file source code như số rẽ nhánh, số điều kiện, số dòng code, số comment, … để dự đoán file source code đó có deffective hay không.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên file:** | dataset\_for\_PCA\_LDA.csv | **Số datapoint (dòng):** | 124 |
| **Số attribute:** | 39 | **Số classtification:** | 2 (YES/NO) |

## Hiện thực PCA trên Software Defect Prediction Dataset

### ***Lấy dữ liệu và chuẩn hóa dataset***

****

Tách dataset thành X là tập 124 datapoint, y là target, đồng thời dùng LabelEncoder để chuyển label dạng chuỗi thành số tức YES/NO thành 1/0

Sử dụng thư viện StandardScaler để chuẩn hóa data



### ***Sử dụng thuật toán Logistic Regression***

Lý do chúng ta sử dụng Logistic Regression để phân lóp là do các attribute của Software Defect Prediction có giá trị liên tục (thuộc tập số thực R)



Chia tập dataset trên với 20% để testing 80% để training, sau đó dùng GaussiNB của Scikitlearn để training và dự đoán cho ra kết quả y\_predict

Bây giờ chúng ta sẽ xuất các thông số đánh giá thử model



Confusion matrix và các thông số khác như accuracy, precision, recall và F1 như sau



Trong 20% testing data (X\_test, y\_test) tức 25 datapoint model Linear Regression đã đoán đúng 6 YES và 12 NO.

### ***Đánh giá trước khi sử dụng PCA***

Tính toán và in ma trận hiệp phương sai cov\_mat với ma trận hệ số tương quan cor\_mat





Tính toán các trị riêng eig\_vals, các vector riêng eig\_vecs theo từng cặp, sau đó sắp xếp chúng theo thứ tự giảm giần của trị riêng.





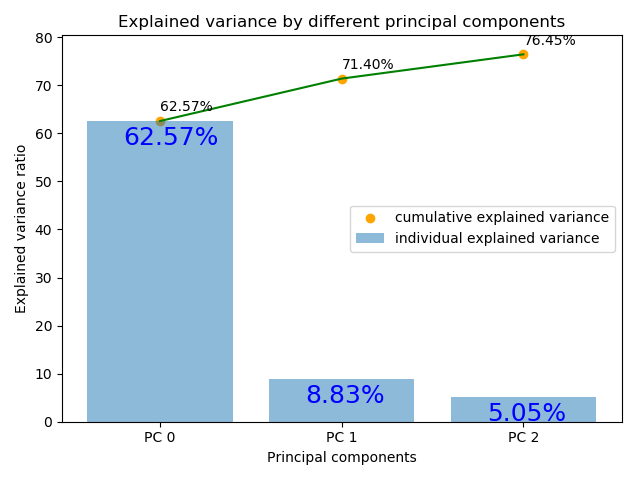
Bây giờ tính tỉ lệ của các PC xem chúng chứa bao nhiêu thông tin cũ





Có thể thấy với 3 PC đầu tiên đã chứa tới 76.45% thông tin từ dataset ban đầu. Vẽ thử đồ thị biễu diễn tỉ lệ thông tin mà các PC này mang.





### ***Áp dụng PCA***

Ghép 3 vector riêng chúng ta đã chọn bên trên thành matrix\_w



Sau đó chiếu hết datapoint ban đầu xuống không gian mới 3 chiều này



Nếu sử dụng thư viện chúng ta không cần làm 2 việc trên mà chỉ cần



Xong vậy là chúng ta có X\_new và y sử dụng lại Linear Regression và đánh giá lần 2.

### ***Sử dụng lại thuật toán Linear Regression và đánh giá***

Vì tính khách quan chúng ta không nên dùng train\_test\_split trên X\_new vì nó sẽ trộn (shuffle) và chọn ngẫu nhiên để chia lại cho training/testing data gây ra sự khác biệt ở lần đầu sử dụng Linear Regression. Hãy sử dụng matrix\_w để chiếu training/testing data cũ thành mới



Phân lớp lại với Linear Regression

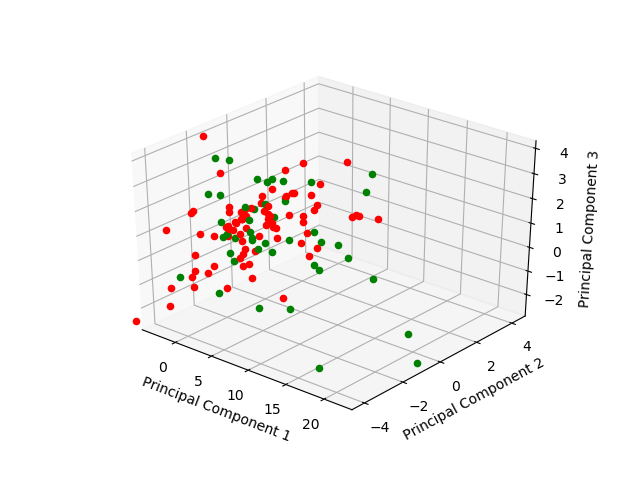




## Nhận xét

Theo confusion matrix thì các thông số giảm đôi chút đơn giản là tập dataset mới chiều đã giảm chỉ mang 76.45% thông tin cũ nhưng nhờ áp dụng PCA chúng ta có thể giảm/bỏ đi các attribute không cần thiết. Vẽ thử tập dataset mới





Có thể thấy đây là đặc tính của PCA sự tách biệt giữa các class không rõ ràng.

# Linear Discriminant Analysis – LDA

## Giới thiệu LDA

Linear Discriminant Analysis là phương pháp giảm chiều dữ liệu đồng thời cũng là phương pháp phân lớp. LDA là cho tập dataset mới sẽ có số chiều ít hơn và mỗi datapoint mang đặc trưng của class đó.

## Hiện thực LDA trên Software Defect Prediction Dataset

### ***Lấy dữ liệu, chuẩn hóa dataset và sử dụng thuật toán Logistic Regression***

Tương tự ở phần PCA trên, chúng ta cũng sử dụng Logistic Regression để phân lớp





### ***Áp dụng LDA***

Thư viện Sklearn cung cấp cho chúng ta công cụ để tính toán ra LDA tốt nhất



Ma trận để chiếu matrix\_w cũng là ma trận chứa các LDA cũng như vector riêng



Đối với dataset này chỉ có 2 label (YES/NO) để phân lớp nên số chiều mới mà LDA cần giảm về là 1 vì thế đối số truyền vào LDA với n\_component=1. Cũng như phần PCA ở trên, chúng ta không dùng tập dataset mới được tự chuyển đổi X\_new vì khi split nó sẽ khác dataset cũ, do đó chúng ta sẽ sử dụng ma trận matrix\_w để chiếu X\_train và X\_test.



Xong vậy là chúng ta có X\_new và y sử dụng lại Linear Regression và đánh giá lần 2.

### ***Sử dụng lại thuật toán Linear Regression và đánh giá***

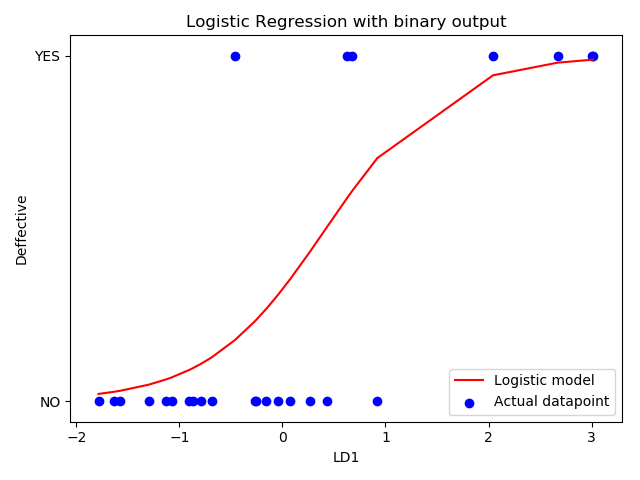
Phân lớp lại với Linear Regression





Vẽ thử biểu đồ





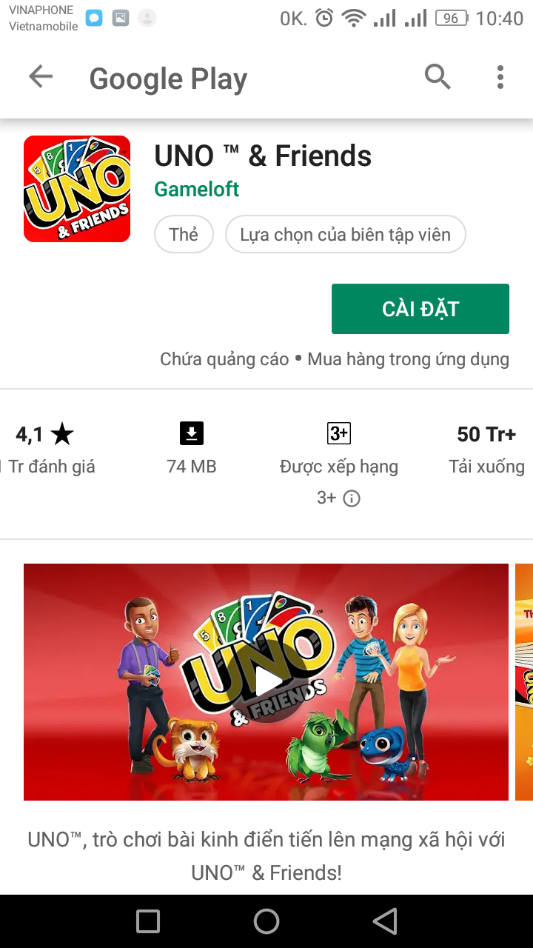
## Nhận xét

Chúng ta có thể thấy LDA giảm từ 39 chiều xuống 1 chiều và nó chú trọng tới việc phân lớp kết hợp với Logistic Regression đã làm các chỉ số precision, recall tăng đáng kể.

# Multiple Linear Regression

## Giới thiệu App Install Prediction Dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên file:** | dataset\_for\_multi\_linear\_regression.csv | | | | |
| **Số attribute:** | 3 | **Số datapoint (dòng):** | 7150 | **Số classtification:** |  |

Đối với người dùng smart phone trước khi họ tải ứng dụng/ trò chơi từ App Store của nhà cung cấp dịch vụ họ thường chú trọng tới các thông như số sao đánh giá cho ứng dụng đó, số lượt tải, kích cỡ ứng dụng hoặc họ cũng có thể review bằng cách xem nhận xét từ người đã trải nghiệm trước, xem video…

Tập dataset\_for\_for\_multi\_linear\_regression.csv chứa các atrribute như:

* Rating: điểm đánh giá trung bình của ứng dụng
* Reviews: tổng số lượt review
* Size: kích thước ứng dụng

Và 1 output để hồi quy đó là Install cho biết lượt tải của ứng dụng đó.

## Hiện thực Multiple Linear Regression

### ***Lấy dữ liệu và tiền xử lý dữ liệu***

Nhìn vào một phần nhỏ data trong file chúng ta có thể thấy attribute Size là dạng chuỗi với kí tự ‘M’ hoặc ‘k’ ở cuối tương ứng kích thước ứng dụng Megabytes hoặc Kilobytes, bên cạnh đó Output Install không phải kiểu số mà là chuỗi với kí tự ‘+’ ví dụ ’50,000+’ cho thấy có hơn 50000 lượt tải. Chúng ta cần xử lý chúng trước khi hồi quy.

Đọc dữ liệu từ file



Định nghĩa phương thức chuyển đổi chuỗi có ‘M’ và ‘K’ thành kiểu số với giá trị là kích thước tệp ở đơn vị Kilobytes



Định nghĩa phương thức khác để xử lý Output Install



Kế tiếp chúng ta sử dụng 2 phương thức trên xử lý dữ liệu



### ***Áp dụng Multiple Linear Regression và đánh giá***

Chia 20% dataset cho việc testing



In các trọng số của phương trình và đánh giá về model



### ***Áp dụng PCA và đánh giá***

Mục đích áp dụng PCA để giảm chiều dữ liệu để có thể vẽ giống như việc biến Multiple Linear Regression thành Simple Linear Regression. Vậy sao không dùng LDA? Đơn giản là bài toán chúng ta không phải phân lớp, không cần mối quan hệ giữa các lớp mà chỉ cần dataset mới có nhiều thông tin từ dataset cũ



In tỉ lệ thông tin mà PCA duy nhất kia mang và các thông số đánh giá sau khi sử dụng PCA



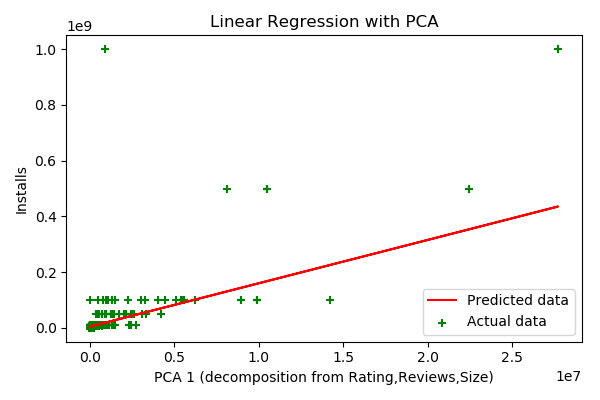


Có thể thấy PCA duy nhất kia mang gần như toàn bộ thông tin cũ nên việc chỉ số r2\_score không thay đổi là điều đương nhiên.

## Nhận xét

Vẽ thử data và model





Các acutal datapoint không được rải rác là do chúng có cùng output vì khi tiền xử lý dữ liệu ở trên cho ra ouput không thật sự thực tế ví dụ ‘5000+’ là hơn 5000 lượt cài đặt chúng ta xử lý ra 5000 trong khi thực tế hơn 5000 là một con số nào đó 5691 chẳng hạn.

# Polymial Linear Regression

## Giới thiệu Bodyfat Dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên file:** | dataset\_for\_poly\_linear\_regression.csv | | | | |
| **Số attribute:** | 14 | **Số datapoint (dòng):** | 252 | **Số classtification:** |  |

Dataset gồm chỉ số cân nặng dưới nước ( và các kích thước liên đến chu vi cơ thể như chiều cao, cân nặng, bắt tay,… của 252 người đàn ông nhằm ước lượng lượng mỡ trong máu của họ.

## Hiện thực Poly Linear Regression

### ***Lấy dữ liệu***



### ***Tìm attribute và degree tốt nhất***

Chúng ta sẽ định nghĩa một phương thức nhận vào dataset (datapoint lẫn output) và khoảng để tạo degree, phương thức này trả về X\_train, X\_test, Y\_train, Y\_test, best\_feature\_index, degree\_array, r2\_array trong đó:

* best\_feature\_index vị trí attribute tốt nhất (có cao nhất) từ dataset đưa vào (vị trí đầu tiên là 0)
* degree\_array mảng chứa số bậc đạt điểm cao nhất xét trong từng attribute
* r2\_array mảng chứa số điểm cao nhất của từng attribute
* X\_train, X\_test, y\_train, y\_test tập data đã split 20% dựa trên attribute và degree tốt nhất



Sử dụng phương thức trên để tìm

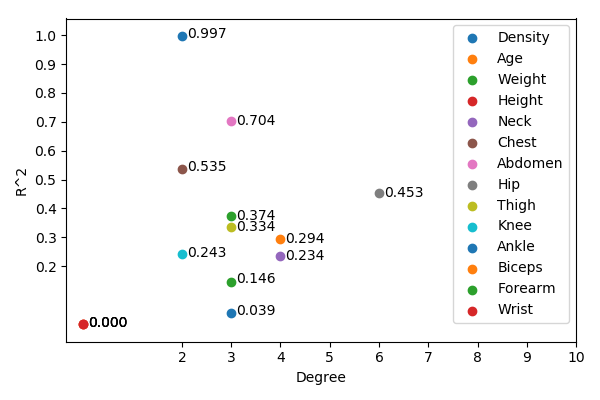


Xem kết quả tốt nhất cũng như những attribute khác



Để dễ quan sát chúng ta vẽ thử biểu đồ sau





### ***Tạo model, predict và đánh giá***

Chúng ta đã tìm ra attribute tốt nhất trong số 14 attribute đồng thời còn tìm được số bậc degree và tập data của nó. Sở dĩ chúng ta giữ lại tập dataset đã split tại thời điểm tìm degree đó là do tính khác quan bởi vì nếu split lại data thì degree tốt nhất sẽ thay đổi. Bây giờ chỉ cần tạo lại model.



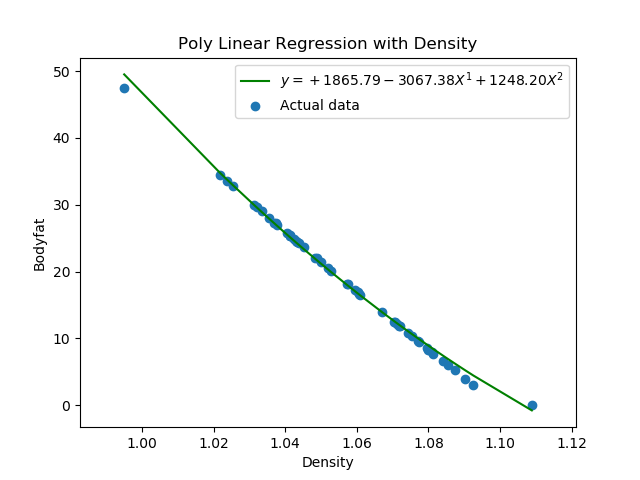
Và in một số thông số đánh giá model





Cuối cùng vẽ thử model này, muốn dường hồi quy vẽ ra không bị lộn xộn chúng ta cần sắp xếp X\_test và y\_pred tương ứng trước





## Nhận xét

Bằng việc tìm ra attribute chính là cột đầu tiên Density và degree tốt nhất chúng ta đã có một model tốt với điểm gần như hoàn hảo.

# Logistic Linear Regresssion

## Giới thiệu Dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tên file:** | dataset\_for\_logistic\_regression.csv | | | | |
| **Số attribute:** | 3 | **Số datapoint (dòng):** | 87 | **Số classtification:** | 2 |

## Hiện thực Logistic Regression

### ***Lấy dữ liệu và chuẩn hóa***



### ***Tìm attribute tốt nhất***

Tìm vị trí attribute tốt nhất với số accuracy cao nhất, đồng thời lấy dataset đã split lúc đó





Như vậy attribute đầu tiên là tốt nhất, chúng ta sẽ đánh giá model với cột này

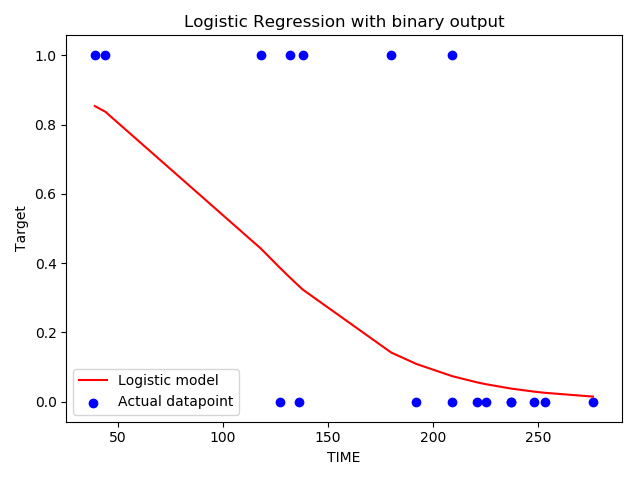


Kết quả



Vẽ model của Logistic Regression





### ***Dùng LDA***

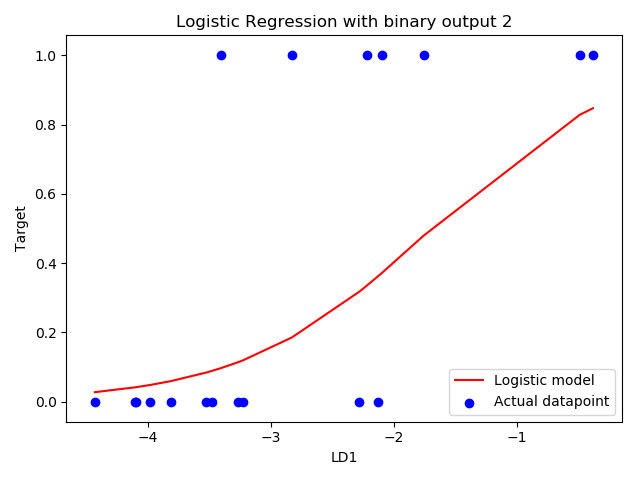
Bây giờ chúng ta sẽ giảm 3 chiều cũ xuống 1 chiều bằng LDA và sử dụng lại Logistic Regression





Vẽ model mới này





## Nhận xét

Có thể trên là 2 cách để giảm chiều dữ liệu 1 là selection feature và 1 extract feature với LDA và cho thông số precision, recall cũng không thay đổi là do dataset quá ít, attribute đầu mang nhiều thông tin phù hợp với Logistic Regression.