Báo cáo bài tập giữa kỳ
Học máy: Hồi quy Logistic trên tập Mobie Price

18020157 - Đỗ Tuấn Anh
18020317 - Trần Hữu Quốc Đông
18020596 - Phạm Quang Hùng
18020936 - Bùi Duy Nam
18020944 - Đường Thị Thủy Ngân

Ngày 14 tháng 11 năm 2020

Tóm tắt nội dung

Bản báo cáo về phương pháp Hồi quy Logistic và phương pháp thực hiện, cách xây dựng mô hình học máy cho tập dữ liệu Mobile Price.

1 Định nghĩa

1.1 Bài toán phân loại

Hồi quy Logistic là một mô hình thống kê ở dạng cơ bản của nó sử dụng một hàm logistic để mô hình một biến phụ thuộc nhị phân, mặc dù tồn tại nhiều phần mở rộng phức tạp hơn. Trong phân tích hồi quy, hồi quy logistic là ước lượng các tham số của mô hình logistic. Về mặt toán học, mô hình logistic nhị phân có một biến phụ thuộc với hai giá trị có thể có, chẳng hạn như đạt/không đạt được đại diện bởi một biến chỉ báo, trong đó hai giá trị được gắn nhãn "0" và "1". Xác suất tương ứng của giá trị được gắn nhãn "1" có thể thay đổi giữa 0 và 1.
Phương pháp này là sử dụng hồi quy tuyến tính và ánh xạ tất cả các dự đoán (phỏng đoán) lớn hơn 0,5 là 1 và tất cả nhỏ hơn 0,5 là 0. Tuy nhiên, phương pháp này không hoạt động tốt vì phân loại không thực sự là một hàm tuyến tính.
Việc phân loại giống như một bài toán hồi quy. Bây giờ, chúng ta tập trung vào "bài toán phân loại nhị phân" trong đó y chỉ có thể nhận hai giá trị, 0 và 1. Do đó $y \in \{0, 1\}$. 0 còn được gọi là lớp phủ định, và 1 là lớp dương, và chúng đôi khi cũng được biểu thị bằng các ký hiệu "$-$" và "$+$". Cho trước x_i, y_i tương ứng còn được gọi là nhãn cho ví dụ huấn luyện.

1.2 Hàm giả thuyết

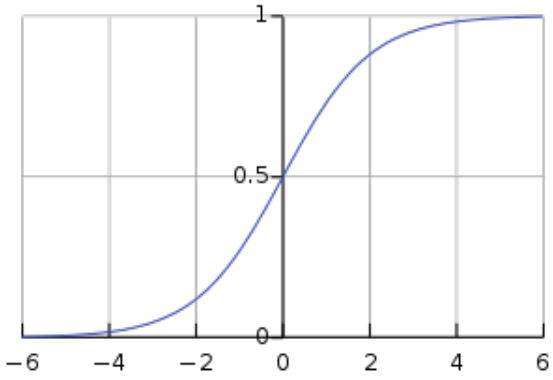
Hàm Sigmoid hay còn gọi là hàm Logistic:

$$h_\theta(x) = g(\theta^T x)$$
$$z = \theta^T x$$
$$g(z) = \frac{1}{1+e^{-z}}$$

Biểu diễn đồ thị của hàm Sigmoid như Hình 1:

Hàm $g(z)$ ánh xạ bất kỳ số thực nào đến khoảng $(0, 1)$, làm cho nó hữu ích cho việc chuyển đổi một hàm có giá trị tùy ý thành một hàm phù hợp hơn để phân loại.
$h_\theta(x)$ sẽ cho xác suất đầu ra là 1.

$$h_\theta(x) = P(y = 1|x; \theta) = 1 - P(y = 0|x; \theta)$$
$$P(y = 1|x; \theta) + P(y = 0|x; \theta) = 1$$



Hình 1: Đồ thị hàm Sigmoid

1.3 Ranh giới quyết định

Để có được phân loại 0 hoặc 1 rời rạc, phân loại đầu ra của hàm giả thuyết như sau:

$$h_\theta(x) \geq 0,5 \rightarrow y = 1$$
$$h_\theta(x) < 0,5 \rightarrow y = 0$$

Cách hoạt động của hàm Logistic g là khi đầu vào của nó lớn hơn hoặc bằng 0, thì đầu ra của nó lớn hơn hoặc bằng 0,5:

$$z \geq 0 \rightarrow g(z) \geq 0,5$$

và:

$$z = 0, e^0 = 1 \Rightarrow g(z) = 0,5$$
$$z \rightarrow \infty, e^{-\infty} \rightarrow 0 \Rightarrow g(z) = 1$$
$$z \rightarrow -\infty, e^{\infty} \rightarrow \infty \Rightarrow g(z) = 0$$

Với đầu vào của hàm g là $\theta^T x$, điều đó có nghĩa là:

$$\theta^T \geq 0 \rightarrow h_\theta(x) = g(\theta^T x) \geq 0,5$$

Từ đó, ta có:

$$\theta^T x \geq 0 \Rightarrow y = 1$$
$$\theta^T x < 0 \Rightarrow y = 0$$

Ranh giới quyết định là đường phân cách vùng $y = 0$ và $y = 1$. Nó được tạo bởi hàm giả thuyết.
Đầu vào cho hàm Sigmoid $g(z)$ không cần phải tuyến tính và có thể là một hàm mô tả một hình tròn hoặc bất kỳ hình dạng nào để phù hợp với dữ liệu.

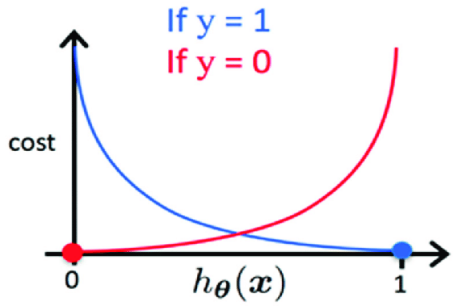
2 Mô hình hồi quy Logistic

2.1 Hàm chi phí: Cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{n} Cost(h_\theta(x^{(i)}), y^{(i)})$$
$$Cost(h_\theta(x^{(i)}), y^{(i)}) = -\log h_\theta(x) \text{ nếu } y = 1$$
$$Cost(h_\theta(x^{(i)}), y^{(i)}) = -\log 1 - h_\theta(x) \text{ nếu } y = 0$$

Đồ thị biểu diễn Cost function biểu diễn tại Hình 2:

$$Cost(h_\theta(x^{(i)}), y^{(i)}) = 0 \text{ nếu } h_\theta(x) = y$$
$$Cost(h_\theta(x^{(i)}), y^{(i)}) \rightarrow \infty \text{ nếu } y = 0 \text{ và } h_\theta(x) \rightarrow 1$$
$$Cost(h_\theta(x^{(i)}), y^{(i)}) \rightarrow \infty \text{ nếu } y = 1 \text{ và } h_\theta(x) \rightarrow 0$$



Hình 2: Đồ thị biểu diễn Cost funtion

2.2 Cost function đơn giản và Gradient descent

- Cost function đơn giản
  Cost function kết hợp:

$$Cost(h_\theta(x^{(i)}), y^{(i)}) = -y \log h_\theta(x) - (1-y) \log(1 - h_\theta(x))$$

  Cost function tổng quát:

$$J(\theta) = -\frac{1}{m}[y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

  Vector hóa:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m}(-y^T \log(h) - (1-y)^T \log(1-h))$$

- Gradient descent Dạng tổng quát:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

  Đạo hàm:

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

  Khai triển vector:

$$\theta = \theta - \frac{\alpha}{m} X^T (g(X\theta) - y)$$

3 Tập dữ liệu Mobile Price

3.1 Lựa chọn vào phân tích tập dữ liệu

Tập giá trị Mobile Price là tập giá trị sử dụng để phân loại mức giá của sản phẩm (price_range) dựa trên các thuộc tính của nó.
Tập dữ liệu Mobile Price có 21 giá đặc trưng và mỗi đặc trưng có 2000 giá trị.
Trong bài toán Học máy sử dụng mô hình Hồi quy Logistic, nhóm lựa chọn 12 đặc trưng ban đầu dùng làm tập giá trị đầu vào và đặc trưng "price_range" làm đầu ra.
Các giá trí của tập dữ liệu không đồng nhất (Hình 3), nhóm thực hiện chuẩn hóa tập dữ liệu về trên đoạn [0;1].

Hàm sử dụng để chuẩn hóa các giá trị trong tập dữ liệu:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15

Hình 3: Giá trị chưa chuẩn hóa

```
% Normalize function
def normalize(X):
    _min = np.min(X, axis=0)
    _max = np.max(X, axis=0)
    _range = _max - _min
    norm_X = 1 - (_max - X)/_range
    return norm_X
```

Sau khi chuẩn hóa dữ liệu về [0;1] nhóm thu được kết quả như Hình 4:

```
array([[0.22778891, 0.        , 0.68      , ..., 0.9       , 0.14285714,
        0.1       ],
       [0.34736139, 1.        , 0.        , ..., 0.46666667, 0.28571429,
        0.3       ],
       [0.04141617, 1.        , 0.        , ..., 0.54166667, 0.57142857,
        0.3       ],
       ...,
       [0.94188377, 0.        , 0.16      , ..., 0.23333333, 1.        ,
        0.15      ],
       [0.6753507 , 0.        , 0.16      , ..., 0.54166667, 0.57142857,
        0.25      ],
       [0.00601202, 1.        , 0.6       , ..., 0.73333333, 0.71428571,
        0.8       ]])
```

Hình 4: Giá trị chuẩn hóa

3.2 Chia tập dữ liệu huấn luyện/kiểm tra

Nhóm sử dụng tập dữ liệu trong tệp "train.csv". Chia tập dữ liệu: 65% dùng để huấn luyện và 35% dùng để kiểm tra. Các dữ liệu dùng để huấn luyện/kiểm tra được lấy ngẫu nhiên.

4 Hồi quy Logistic của thư viện sklearn

4.1 Các tham số và thuộc tính

4.1.1 Tham số

+ penalty: {'l1', 'l2', 'elasticnet', 'none', mặc định ='l2'}
Được sử dụng để chỉ định mức được sử dụng trong penalization. Các bộ giải 'newton-cg', 'sag' và 'lbfgs' chỉ hỗ trợ các penalty = l2. 'Asticnet 'chỉ được hỗ trợ bởi bộ giải 'saga'. Nếu 'none' (không được hỗ trợ bởi bộ giải liblinear), thì không có quy định nào được áp dụng.
+ dual: bool, mặc định = False
Công thức dual hoặc primal. dual chỉ được thực hiện cho penalty = l2 với bộ giải liblinear. dual = False khi n_samples> n_features.
+ tol: float, mặc định = 1e-4
Dung sai cho các tiêu chí dừng.
+ C: float, mặc định = 1.0
Nghịch đảo của sức mạnh chính quy; phải là một float dương. Giống như trong các máy vectơ hỗ trợ, các giá trị nhỏ hơn xác định chính quy mạnh hơn.
+ fit_intercept: bool, mặc định = True
Chỉ định xem có nên thêm một hằng số (bias hoặc intercept) vào hàm quyết định hay không.
+ intercept_scaling: float, mặc định = 1

Chỉ hữu ích khi bộ giải 'liblinear' được sử dụng và self.fit_intercept được đặt thành True. Trong trường hợp này, x trở thành [x, self.intercept_scaling].
+ class_weight: 'dict' hoặc 'balance', mặc định = none
Trọng số được liên kết với các lớp ở dạng {class_label: weight}. Nếu không được đưa ra, tất cả các lớp phải có trọng số là một.
Chế độ "cân bằng" sử dụng các giá trị của y để tự động điều chỉnh trọng số tỷ lệ nghịch với tần số lớp trong dữ liệu đầu vào dưới dạng n_samples / (n_classes * np.bincount (y)).
+ random_state: int, mặc định = none
Được sử dụng khi bộ giải == 'sag', 'saga' hoặc 'liblinear' để xáo trộn dữ liệu.
+ solver: {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, mặc định = 'lbfgs'
Thuật toán sử dụng trong bài toán tối ưu hóa.

- Đối với các tập dữ liệu nhỏ, 'liblinear' là một lựa chọn tốt, trong khi 'sag' và 'saga' nhanh hơn cho các tập lớn.
- Đối với các vấn đề đa thức, chỉ có 'newton-cg', 'sag', 'saga' và 'lbfgs' xử lý mất đa thức; 'liblinear' được giới hạn trong các lược đồ một so với phần còn lại.
- 'Newton-cg', 'lbfgs', 'sag' và 'saga' xử lý L2 hoặc không penalty.
- 'liblinear' và 'saga' cũng xử lý penalty = L1
- 'saga' cũng hỗ trợ hình phạt 'thunnet'
- 'liblinear' không hỗ trợ đặt penalty = 'none'

+ max_iter: int, mặc định = 100
Số lần lặp lại tối đa được thực hiện để các bộ giải hội tụ.
+ multi_class {'auto', 'ovr', 'multinomial',mặc định = 'auto'
Nếu tùy chọn được chọn là 'ovr', thì một vấn đề nhị phân phù hợp với từng nhãn. Đối với 'multinomial', tổn thất được giảm thiểu là tổn thất đa thức phù hợp trên toàn bộ phân phối xác suất, ngay cả khi dữ liệu là nhị phân. 'multinomial' không khả dụng khi solver = 'liblinear'. 'Auto' chọn 'ovr' nếu dữ liệu là nhị phân hoặc nếu solver = 'liblinear', và nếu không sẽ chọn 'multinomial'.
+ verbose: int, mặc định = 0 Đối với bộ giải liblinear và lbfgs, hãy đặt verbose thành bất kỳ số dương nào.
+ warm_start: bool, mặc định = False
Khi được đặt thành True, hãy sử dụng lại giải pháp của lần gọi trước đó để vừa với quá trình khởi tạo, nếu không, chỉ cần xóa giải pháp trước đó. Vô ích cho bộ giải liblinear.
+ n_jobs: int, mặc định = none
Số lõi CPU được sử dụng khi ghép song song qua các lớp nếu multi_class = 'ovr' ". Tham số này bị bỏ qua khi bộ giải được đặt thành 'liblinear' bất kể 'multi_class' có được chỉ định hay không. Không có nghĩa là 1 trừ khi trong ngữ cảnh joblib.parallel_backend. -1 có nghĩa là sử dụng tất cả các bộ xử lý. Xem Bảng chú giải thuật ngữ để biết thêm chi tiết.
+ l1_ratio: float, mặc định = none
Tham số trộn Elastic-Net, với 0 <= l1_ratio <= 1. Chỉ được sử dụng nếu penalty ='asticnet'. Đặt l1_ratio = 0 tương đương với sử dụng penalty = 'l2', trong khi đặt l1_ratio = 1 tương đương với sử dụng penaltyt = 'l1'. Đối với 0 <l1_ratio < 1, penalty là sự kết hợp của L1 và L2.

4.1.2 Thuộc trính

+ classes_: ndarray of shape (n_classes,)
Danh sách các nhãn lớp mà bộ phân loại đã biết.
+ coef_: ndarray của hình dạng (1, n_features) hoặc (n_classes, n_features)
Hệ số của các tính năng trong hàm quyết định.
coef_ có dạng (1, n_features) khi bài toán đã cho là hệ nhị phân. Đặc biệt, khi multi_class = 'multinomial', coef_ tương ứng với kết quả 1 (Đúng) và -coef_ tương ứng với kết quả 0 (Sai).
+ intercept_: ndarray of shape (1,) or (n_classes,)
Intercept (bias) được thêm vào hàm quyết định. Nếu fit_intercept được đặt thành False, thì giá trị chặn được đặt thành 0. intercept_ có dạng (1,) khi bài toán đã cho là hệ nhị phân. Đặc biệt, khi multi_class =

'multinomial', intercept_ tương ứng với kết quả 1 (True) và -intercept_ tương ứng với kết quả 0 (False).
+ n_iter_: ndarray of shape (n_classes,) or (1,)
Số lần lặp thực tế cho tất cả các lớp. Nếu nhị phân hoặc đa thức, nó chỉ trả về 1 phần tử. Đối với bộ giải liblinear, chỉ số lần lặp tối đa trên tất cả các lớp được đưa ra.

4.2 Giải thích code

Khai báo các thư viện cần thiết:

```
from sklearn.linear_model import LogisticRegression
import numpy as np
import pandas as pd
```

Hàm sử dụng để chuẩn hóa dữ liệu:

```
def normalize(X):
    _min = np.min(X, axis=0)
    _max = np.max(X, axis=0)
    _range = _max - _min
    norm_X = 1 - (_max - X)/_range
    return norm_X
```

Hiển thị các thuộc tính cần thiết của tập dữ liệu trên màn hình:

```
df = pd.read_csv('mobile_price/train.csv')
df.shape
df.columns
```

Khai báo mô hình Logistic Regression của thư viện sklearn với các thuộc tính:

- Sai số: tol = 0.05
- Số vòng lặp cực đại = 10000
- multi class = multinomial

```
logisticReg = LogisticRegression(tol=0.05,
        max_iter=10000, multi_class='multinomial')
```

Chọn các đặc trưng sử dụng làm đầu vào và đâu ra của mô hình huấn luyện. Sử dụng 11 đặc trưng đầu tiên để là đầu vào và đặc trung cuối cùng (price_range) là đặc trưng đầu ra.

```
X = df[df.columns[:11]].values
X = normalize(X)
Y = df[df.columns[-1]].values
```

Lựa chọn ngẫu nhiên 65% trong các cặp giá trị đầu vào và đầu ra để huấn luyện, 35% còn lại sử dụng để kiểm tra độ chính xác.

```
np.random.seed(1)
p = np.random.permutation(len(X))
x_train = X[p[:int(len(X)*0.65)]].copy()
y_train = Y[p[:int(len(X)*0.65)]].copy()
x_test = X[p[int(len(X)*0.65):]].copy()
y_test = Y[p[int(len(X)*0.65):]].copy()
```

Huấn luyện cho tập dữ liệu sử dụng hàm fit() và dữ đoán dữ liệu bằng hàm predict()

```
logisticReg.fit(x_train, y_train)
y_predict = logisticReg.predict(x_test)
```

Kiểm tra độ chính xác của mô hình sử dụng:

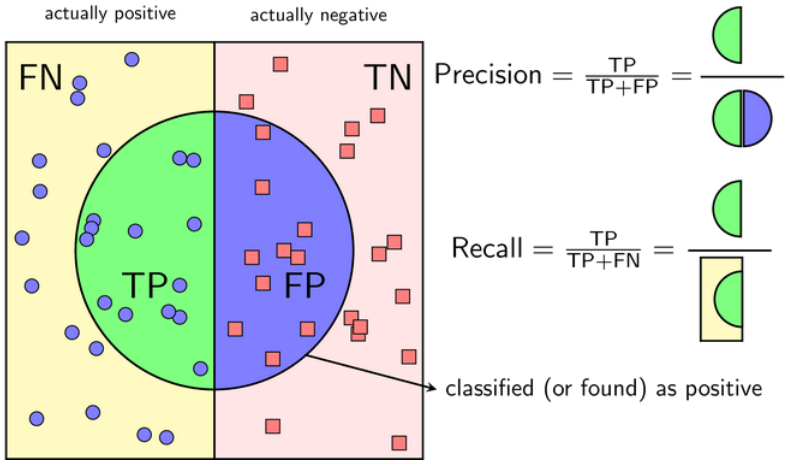
```
num_correct = len(y_predict[y_predict!=y_test])
num_total = len(y_predict)
print("True predict = {}/{}".format(num_correct, num_total))
print("Logistic Regression model accuracy: {}%".format(
        round(100*num_correct/num_total, 2)))
```

Kết quả thu được độ chính xác của mô hình được huấn luyện là 73,14% tương đương với 512/700 mẫu.

5 F1 score

5.1 Precision và Recall

Hình 5 mô tả về Precision và Recall trong bài toán nhị phân.



Hình 5: Precision và Recall

Precision là tỉ lệ số điểm Positive mô hình đoán đúng trên tổng số mô hình dự đoán là Position.
Recall là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm Positive thực sự (tổng số điểm được gán nhãn là Positive ban đầu).
Precision càng cao, tức là số điểm mô hình dự đoán là positive đều là positive càng nhiều. Precision = 1, tức là tất cả số điểm mô hình dự doán là Positive đều đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.
Recall càng cao, tức là số điểm là positive bị bỏ sót càng ít. Recall = 1, tức là tất cả số điểm có nhãn là Positive đều được mô hình nhận ra.

5.2 F1 score

Tuy nhiên, chỉ có Precision hay chỉ có Recall thì không đánh giá được chất lượng mô hình.
Chỉ dùng Precision, mô hình chỉ đưa ra dự đoán cho một điểm mà nó chắc chắn nhất. Khi đó Precision = 1, tuy nhiên ta không thể nói là mô hình này tốt.
Chỉ dùng Recall, nếu mô hình dự đoán tất cả các điểm đều là positive. Khi đó Recall = 1, tuy nhiên ta cũng không thể nói đây là mô hình tốt.
Khi đó F1-score được sử dụng. F1-score là trung bình điều hòa (harmonic mean) của precision và recall (giả sử hai đại lượng này khác 0).F1-score được tinh theo công thức:

$$f1score = \frac{2 * precision * recall}{precision + recall} \tag{1}$$

Đối với bài toán F1 score trong bài toán nhiều lớp, đơn giản ta phân loại thành từng bài toán nhị phân nhỏ hơn và tính Precision, Recall và F1 score cho từng bài toán nhị phân đó.
Hàm thống kê các giá trị trong confusion matrix:

```
def f1_statistic(Y_truth, Y_predict):
    Y_set = list(set(Y_truth))
    length = len(Y_set)
    #print(length)
    f1 = np.zeros([length, length])
    print(f1.size)
    for i in range(len(Y_truth)):
        f1[Y_predict[i], Y_truth[i]] += 1
    return f1
```

Hàm tính các giá trị Precision, Recall và F1 score cho từng phân lớp.

```
def f1_value(f1):
    score = []
    for i in range(f1.shape[0]):
        pt_at = f1[i,i]
        pt_af = np.sum(f1,axis=1)[i] - pt_at
        pf_at = np.sum(f1,axis=0)[i] - pt_at
        pf_af = np.sum(f1) - (pt_at+pt_af+pf_at)

        precision = 100*pt_at/(pt_at+pt_af) # %
        recall = 100*pt_at/(pt_at+pf_at)    # %
        f1_score = 2*precision*recall/(precision+recall)    # %
        score.append([precision, recall, f1_score])
    return np.array(score)
```

Các giá trị Precision, Recall và F1 score (%) khi áp dụng mô hình Logistic Regression:

Classes	Precision	Recall	F1 score
0	26.04	23.28	24.58
1	26.53	14.21	18.51
2	20.86	25.16	22.81
3	32.11	45.66	37.71

6 So sánh với mô hình học máy khác

Trong lựa chọn so sánh với mô hình học máy khác, nhóm đã tìm hiểu trên thư viện sklearn một loại mô hình học máy khác cũng được sử dụng để phân loại là Ridge Classifier.
Cách tiến hành so sánh hai mô hình bằng cách thực hiện lấy cùng một lượng đặc trưng để sử dụng. Số lượng mẫu dùng để chia thành tập huấn luyện và tập kiểm tra là như nhau.
Kết quả thu được độ chính xác của mô hình là 72,14% tương đương 505/700 mẫu. Sau khi thu được kết quả từ hai mô hình là Logistic Regression và Ridge Classifier, nhóm nhận thấy đối với tập dữ liệu Mobile Price và cách chia tập dữ liệu là 65% huấn luyện - 35% kiểm tra, thì mô hình Logistic Regression có độ chính xác tốt hơn.