Báo cáo

▼ Giới thiệu bài tập lớn

Mục tiêu của Bài Tập

Bài tập lớn này tập trung vào việc xây dựng và đánh giá các mô hình học máy để chẩn đoán ung thư vú. Mục tiêu chính của dự án là sử dụng các thuật toán học máy để phân loại các mẫu tế bào thành lành tính hoặc ác tính dựa trên các đặc điểm sinh học. Đây là một vấn đề quan trọng trong y học vì việc chẩn đoán sớm và chính xác ung thư vú có thể cứu sống nhiều bệnh nhân và cải thiện chất lượng cuộc sống của họ.

Dữ Liệu từ Kaggle

Dữ liệu được sử dụng trong bài tập này được thu thập từ Kaggle, một nền tảng trực tuyến nổi tiếng dành cho các cuộc thi khoa học dữ liệu và học máy. Bộ dữ liệu "Breast Cancer Wisconsin (Diagnostic) Data Set" trên Kaggle cung cấp một tập hợp các đặc điểm được tính toán từ các ảnh chụp X-quang của tế bào vú.

Chi Tiết Bộ Dữ Liệu

Bộ dữ liệu bao gồm các đặc điểm như:

- **Radius (bán kính)**
- **Texture (kết cấu)**
- **Perimeter (chu vi)**
- **Area (diện tích)**
- **Smoothness (độ mịn)**
- ... và nhiều đặc điểm khác.
- Bên cạnh đó có các đặc điểm khác như của các đặc trưng dữ liệu
    - Mean : giá trị trung bình
    - SE : Phương sai

- Worst : Tệ nhất (Thường là giá trị lớn nhất)

Mỗi mẫu tế bào được phân loại là lành tính (B - Benign) hoặc ác tính (M - Malignant)

▼ Tiền sử lí dữ liệu

Đầu tiên chúng ta cần đọc dữ liệu mà chúng ta đã thu thập được từ file Data mà chúng ta đã thu thập được và lưu vào biến `data`

```
data = pd.read_csv("data.csv")
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

Lượng data chúng ta thu thập được gồm 569 hàng và 33 cột

Tiếp theo ta tiến hành xử lí dữ liệu

```
data=data.drop(['Unnamed: 32', 'id'], axis = 1)
data.diagnosis = [1 if each == "M" else 0 for each in data.di
```

Xóa cột có tên là `Unnamed: 32` `id` vì nó không cần thiết trong quá trình huấn luyện

Chuẩn hóa kết quả dữ liệu ở cột `diagnosis` về dạng số để thuật tiện cho quá trình huấn luyện mô hình . Nếu là M thì chuyển về 1 tức là `True` nếu là B thì chuyển về 0 tức là `False`.

Sau đó ta tiến hành lưu dữ liệu đã làm sạch vào file `data_clean.csv`

Tách dữ liệu thành 2 phần là tập test và data

```
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size = 0.15, random_state = 42)
```

```
x_train = x_train.T
x_test = x_test.T
y_train = y_train.T
y_test = y_test.T

print("x train: ", x_train.shape)
print("x test: ", x_test.shape)
print("y train: ", y_train.shape)
print("y test: ", y_test.shape)
```

Ta thu được các kết quả của các tập sau:

x train: (31, 483)
x test: (31, 86)
y train: (483,)
y test: (86,)

Tức tập kiểm thử có 86 phần tử và tập huấn luyện có 483 phần tử.

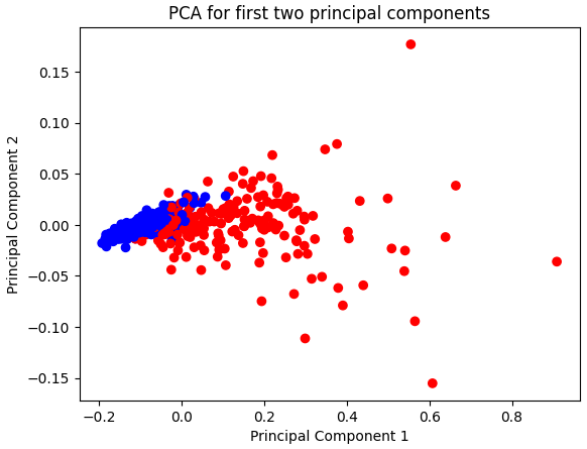
Đồng thời ta cũng chuẩn hóa dữ liệu sử dụng phương pháp Min-Max Scaling:

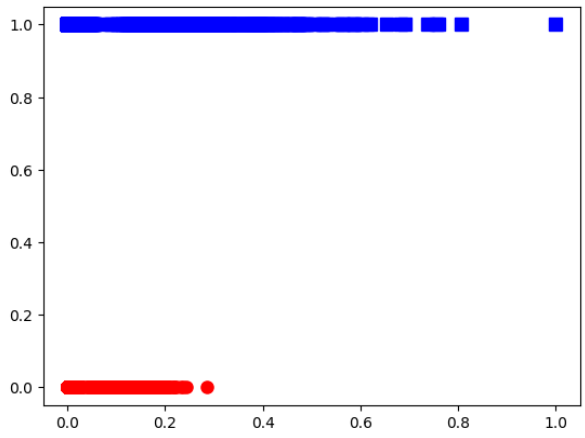
```
x = (x_data - np.min(x_data))/(np.max(x_data) - np.min(x_data
```

▼ Huấn luyện mô hình

Sau khi chuẩn hóa và tiền sử lí dữ liệu xong chúng ta tiến hành huấn luyện mô hình

Hãy nhìn vào các biểu đồ dưới đây:

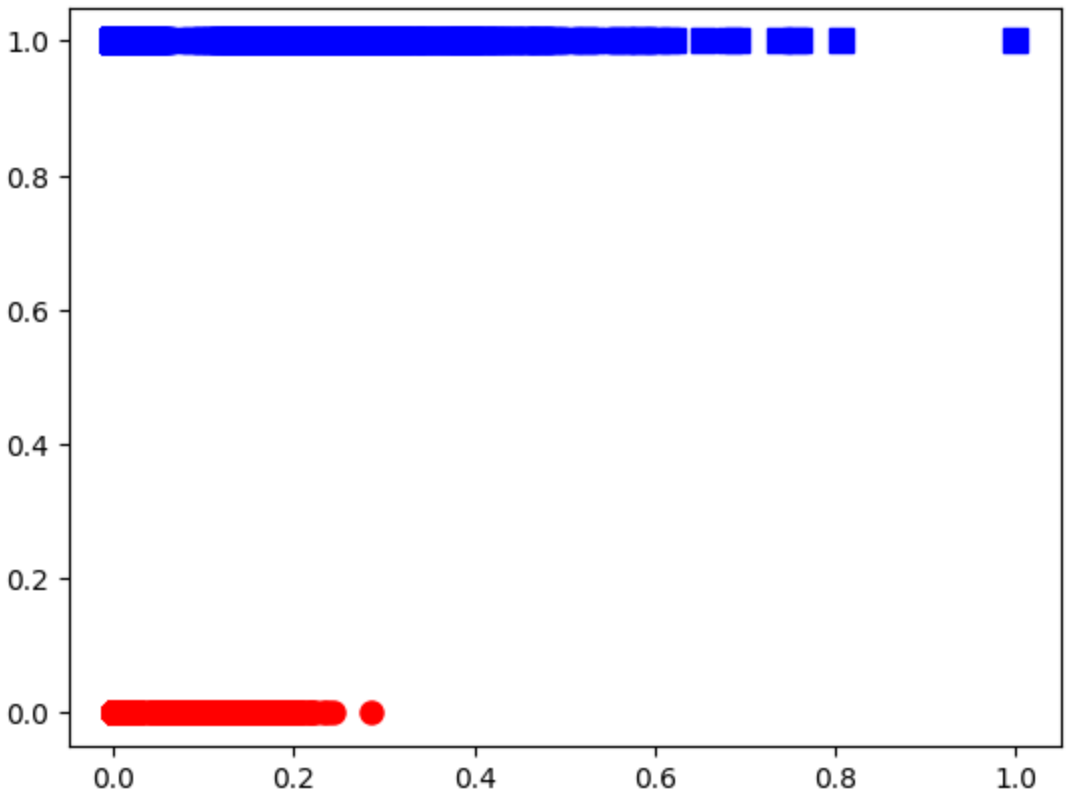




Qua biêu đồ trên chúng tôi quyết định sử dụng 3 thuật toán Logistic Regression, K-nearest- neighbors và Perceptron Learning Algorithm để huấn luyện mô hình học máy dự đoán ung thư vú.

Màu đỏ biểu thị cho các điểm dữ liệu dương tính. Màu xanh biểu diễn các điểm dữ liệu âm tính

▼ Logistic Regression




```
# z = np.dot(w.T, x_train)+b
def sigmoid(z):
    y_head = 1/(1 + np.exp(-z))
    return y_head
```

Việc tính độ lệch chuẩn y_head sẽ dựa vào trọng số w của dữ liệu với xông thức

`z = np.dot(w.T, x_train)+b`

`y_head= 1/(1 + np.exp(-z))`

(với w.T là trọng số số của dữ liệu huấn luyện, x_train là dữ liệu huấn luyện)

Sau đó ta thực hiện lan truyền tiến và lùi để tìm ra độ dốc của dữ liệu huấn luyện

```
# thực hiện lan truyền tiến và lùi và tìm chi phí và độ dốc
        cost, gradients = forward_backward_propagation(w, b,
        cost_list.append(cost)
```

Và bước cuối cùng trước khi đi vào huấn luyện mô hình là đưa ra dự đoán cho các giá trị y thực tế

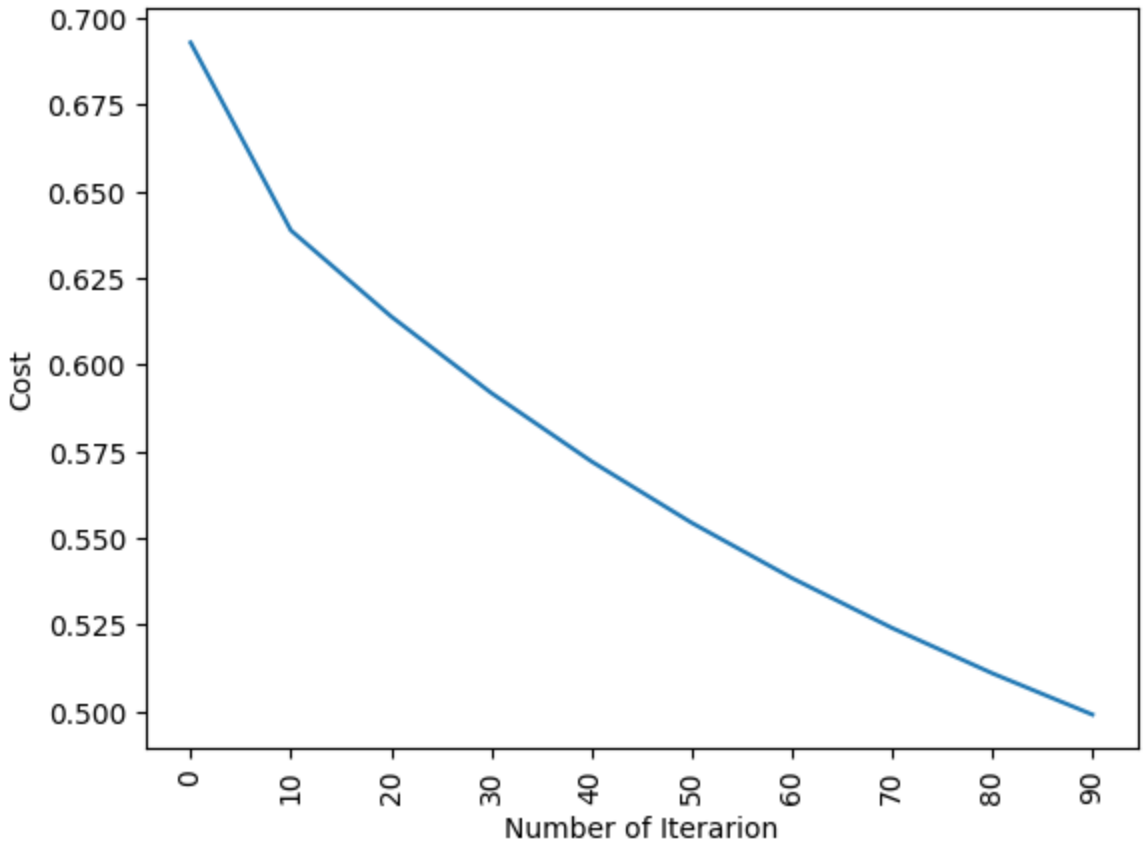
```
def predict(w, b, x_test):
    z = sigmoid(np.dot(w.T, x_test)+b)
    Y_prediction = np.zeros((1, x_test.shape[1]))

    # if z is bigger than 0.5, our prediction is sign one (y_
    # if z is smaller than 0.5, our prediction is sign zero (
    for i in range(z.shape[1]):
        if z[0, i]<= 0.5:
            Y_prediction[0, i] = 0
        else:
            Y_prediction[0, i] = 1

    return Y_prediction
```

Nếu z được tính theo công thức như trên mà cho ra kết quả nhỏ hơn hoặc bằng 0.5 thì kết quả dự đoán của mô hình sẽ là 1 và ngược lại thì kết quả dự đoán sẽ là 0

Sau khi huấn luyện vói 90 lần lặp ta sẽ được kết quả như biểu đồ sau



Sau khi huấn luyện mô hình ta thu được kết quả sau đây

- test accuracy: 0.8953488372093024
- train accuracy: 0.8633540372670807

Có vẻ như điểm số không được cao cho lắm

▼ K-nearest-neighbors

Để thuật toán này

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train).score(X_train, y_train)
```

Điểm số thu được khi sử dụng KNN là

- Với dữ liệu huấn luyện là : 0.946058091286307
- Với dữ liệu kiểm thử là : 0.9302325581395349

⇒ Điểm số khá thấp so với mong đợi

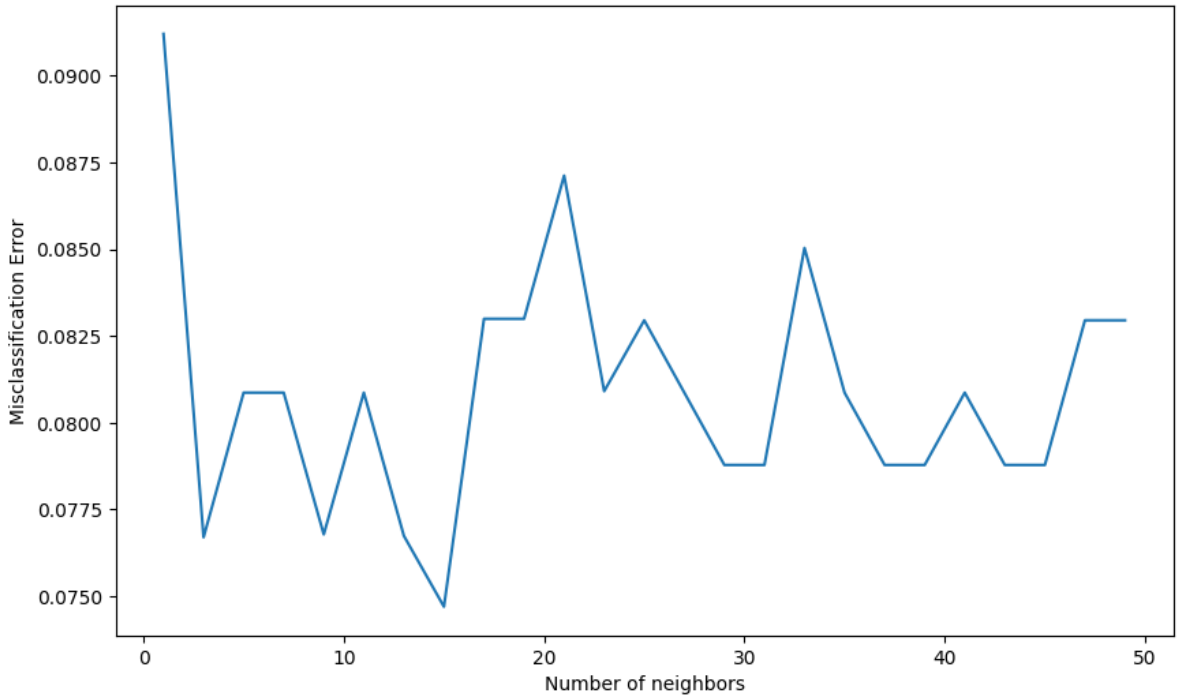
Tối ưu KNN

Khi thực hiện huấn luyện mô hình trên. Trở ngại lớn nhất là tìm giá trị của k sao cho mô hình huấn luyện là tối ưu nhất. Bằng cách sử dụng kỹ thuật cross-validation và vẽ biểu đồ để đánh giá hiệu suất của mô hình dựa trên số lượng láng giềng qua đó tìm được K tối ưu nhất là 13.

```
neighbors = []
cv_scores = []

# thực hiện xác nhận chéo 10 lần
for k in range(1, 51, 2):
    neighbors.append(k)
    knn = KNeighborsClassifier(n_neighbors = k)
    scores = cross_val_score(
        knn, X, y, cv = 10, scoring = 'accuracy')
    cv_scores.append(scores.mean())
```

Ta xem xét số lượng láng giềng từ 1 đến 51 với bước nhảy bằng 2



Theo như biểu đồ thì điểm càng thấp càng tốt. Quá đó t cũng dễ thấy điểm thấp nhất có k là 13.

Đồng thời xác suất đoán đúng cũng rất cao

- Với tập kiểm thử : 0.9767441860465116
- Với tập huấn luyện : 0.9358178053830227

So sánh

	Model	cross_val	Accuracy	Precision	Recall	F1 Score	AUC
0	KNN	0.929762	0.930233	0.964286	0.84375	0.900000	0.912616
1	Knn_Best_Params	0.931516	0.976744	1.000000	0.93750	0.967742	0.968750

1. **Cross_val**:
    - Knn_Best_Params có giá trị cross_val cao hơn một chút so với KNN cơ bản, cho thấy nó có thể có độ chính xác cao hơn một chút khi áp dụng trên tập dữ liệu khác.
2. **Accuracy**:

- Knn_Best_Params có độ chính xác cao hơn đáng kể (0.976744) so với KNN cơ bản (0.930233).

3. **Precision**:
   - Precision của Knn_Best_Params là 1.000000, nghĩa là tất cả các dự đoán "positive" đều đúng. Trong khi đó, Precision của KNN cơ bản là 0.964286, cũng khá cao nhưng không bằng Knn_Best_Params.
4. **Recall**:
   - Recall của Knn_Best_Params (0.93750) cao hơn so với KNN cơ bản (0.84375). Điều này cho thấy Knn_Best_Params có khả năng nhận diện chính xác nhiều mẫu "positive" hơn.
5. **F1 Score**:
   - F1 Score của Knn_Best_Params là 0.967742, vượt trội hơn so với KNN cơ bản (0.900000). Điều này cho thấy sự cân bằng tốt hơn giữa Precision và Recall trong mô hình với tham số tốt nhất.
6. **AUC**:
   - AUC của Knn_Best_Params (0.968750) cao hơn so với KNN cơ bản (0.912616), cho thấy mô hình Knn_Best_Params có khả năng phân biệt giữa các lớp tốt hơn.

▼ Perceptron Learning Algorithm

Theo đồ thị trên thì thuật toán Perceptron Learning Algorithm khá phù hợp với dữ liệu của chúng ta nó sẽ tìm ra 1 đường thẳng để chia 2 tập dữ liệu trên thành 2 nửa mặt phẳng

```
from sklearn.linear_model import Perceptron
clf = Perceptron(tol=1e-3, random_state=42)
clf.fit(x_train, y_train)
Perceptron()
train = clf.score(x_train, y_train)
test = clf.score(x_test, y_test)
```

Ta sẽ chọn điểm hội tụ là `1e-3` nhằm cho mô hình được tối ưu nhất.

Thuận toán cũng cho ra kết quả rất với:

- Với tập kiểm thử : 0.8953488372093024
- Với tập huấn luyện : 0.855072463768116

Khá tương đồng so với Logistic Regression nhưng để đánh giá thuật toán nào tối ưu nhất ta cần phải đánh giá thuật toán từ nhiêu phía hơn

▼ So sánh các mô hình

	Model	cross_val	Accuracy	Precision	Recall	F1 Score	AUC
0	Logistic_regression	0.868233	0.895349	1.0	0.71875	0.836364	0.859375
1	KNN	0.931516	0.976744	1.0	0.93750	0.967742	0.968750
2	PLA	0.889254	0.895349	1.0	0.71875	0.836364	0.859375

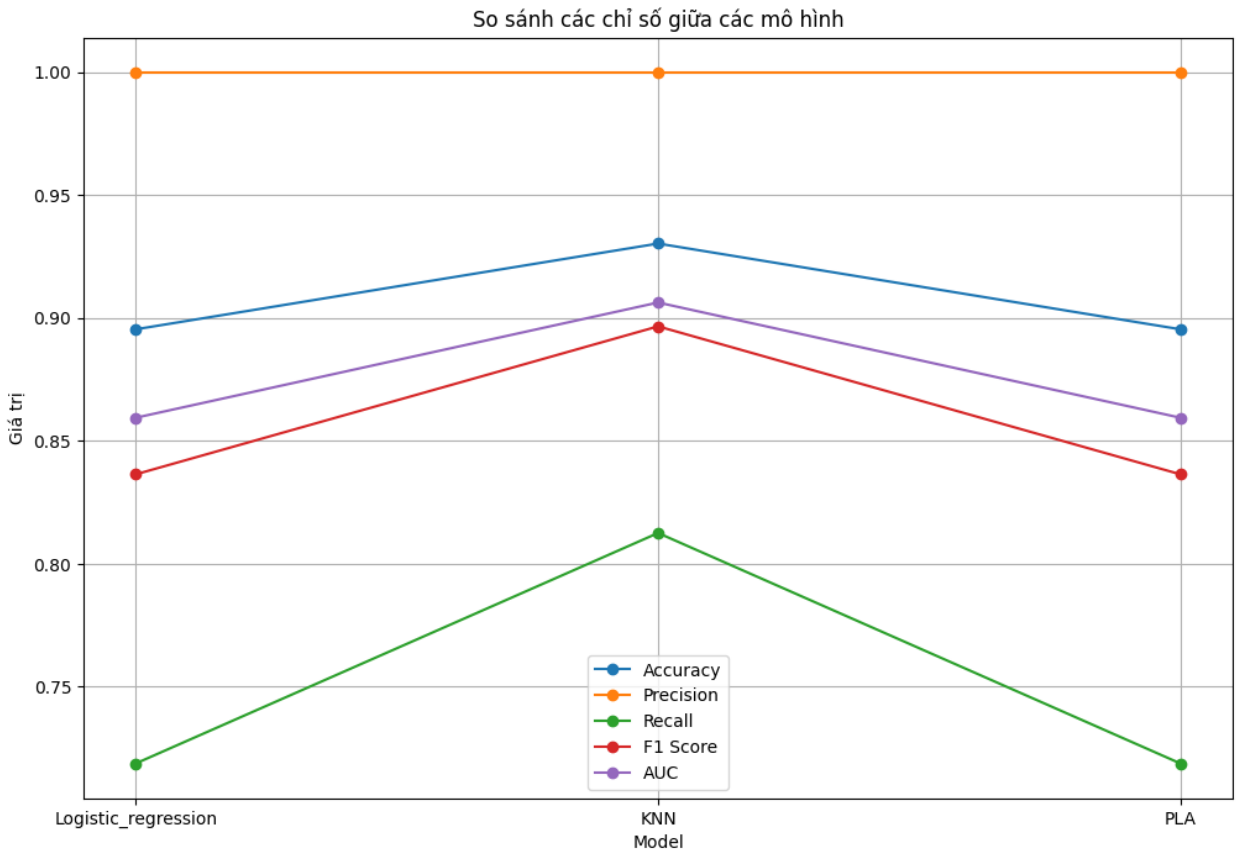
Dựa vào bảng kết quả của các mô hình học máy, chúng ta có các chỉ số hiệu suất như Accuracy, Precision, Recall, F1 Score, và AUC cho ba mô hình: Logistic Regression, K-Nearest Neighbors (KNN), và PLA (Perceptron Learning Algorithm). Hãy cùng phân tích và so sánh các chỉ số này để đưa ra nhận xét về hiệu suất của từng mô hình.

1. **Cross-validation(**Xác thực chéo**):**
    - **KNN** có độ chính xác cao nhất (0.934994).
    - **PLA là** 0.889254 và **Logistic Regression** là 0.868233.
2. **Accuracy** (Độ chính xác):
    - **KNN** có độ chính xác cao nhất (0.976744).
    - **PLA** và **Logistic Regression** có độ chính xác tương tự (0.895349).
3. **Precision** (Độ chính xác dự đoán):
    - Cả ba mô hình đều đạt giá trị Precision cao nhất là 1.0, có nghĩa là tất cả các dự đoán dương tính đều chính xác.
4. **Recall** (Khả năng phát hiện dương tính thực sự):
    - **KNN** có Recall cao nhất (0.93750), thể hiện khả năng phát hiện dương tính thực sự tốt hơn.

- **PLA** và **Logistic Regression** có Recall tương tự nhau (0.71875), thấp hơn so với KNN.

5. **F1 Score** (Điểm F1):
   - **KNN** có F1 Score cao nhất (0.967742), cân bằng tốt giữa Precision và Recall.
   - **PLA** và **Logistic Regression** có F1 Score tương tự nhau (0.836364).
6. **AUC** (Diện tích dưới đường cong ROC):
   - **KNN** có AUC cao nhất (0.968750), cho thấy mô hình có khả năng phân loại tốt hơn.
   - **PLA** và **Logistic Regression** có AUC tương tự nhau (0.859375).

Để dễ hình dung hơn hãy nhìn vào biểu đồ sau:



Nhận xét

- **KNN** là mô hình vượt trội nhất trong cả năm chỉ số (Accuracy, Precision, Recall, F1 Score, AUC). Điều này cho thấy KNN có hiệu suất tốt nhất cho tập dữ liệu này.
- **Logistic Regression** và **PLA** có hiệu suất tương tự nhau, nhưng kém hơn so với KNN, đặc biệt ở chỉ số Recall và F1 Score.

Kết luận

KNN sẽ là lựa chọn tốt nhất vì nó đạt kết quả tốt nhất ở tất cả các chỉ số quan trọng. Tiếp theo đó là **PLA** và **Logistic Regression**