به نام خدا

مبانی سیستم های هوشمند

گزارش مینی پروژه اول

استاد: دکتر مهدی علیاری شوره دلی

دانشجو: رضا آقاجری

شماره دانشجویی: ۹۸۱۹۵۸۳

پاییز ۱۴۰۲

سوال اول

۱. با استفاده از sklearn.datasets، یک دیتاست با ۱۰۰۰ نمونه، ۲ کلاس و ۲ ویژگی تولید کنید.

برای ایجاد دیتاست لازم است از کتابخانه sklearn.datasets، make classification و برای نمایش دیتاست ساخته شده matplotlib.pyplot را import کنیم.

```
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt

X,y = make_classification(n_samples=1000,
                          n_features=2,
                          n_redundant=0,
                          n_classes=2,
                          n_clusters_per_class=1,
                          class_sep=1,
                          random_state=83)
plt.scatter(X[:,0],X[:,1],c=y)
```

برای تعیین ویژگی های دیتاست از آرگومان های تابع make_classification استفاده میکنیم. با استفاده از n_samples تعداد نمونه را برابر با ۱۰۰۰، برای تعیین تعداد ویژگی ها n_features را ۲ و برای تعداد کلاس ها از n_classes را برابر با ۲ قرار می دهیم.

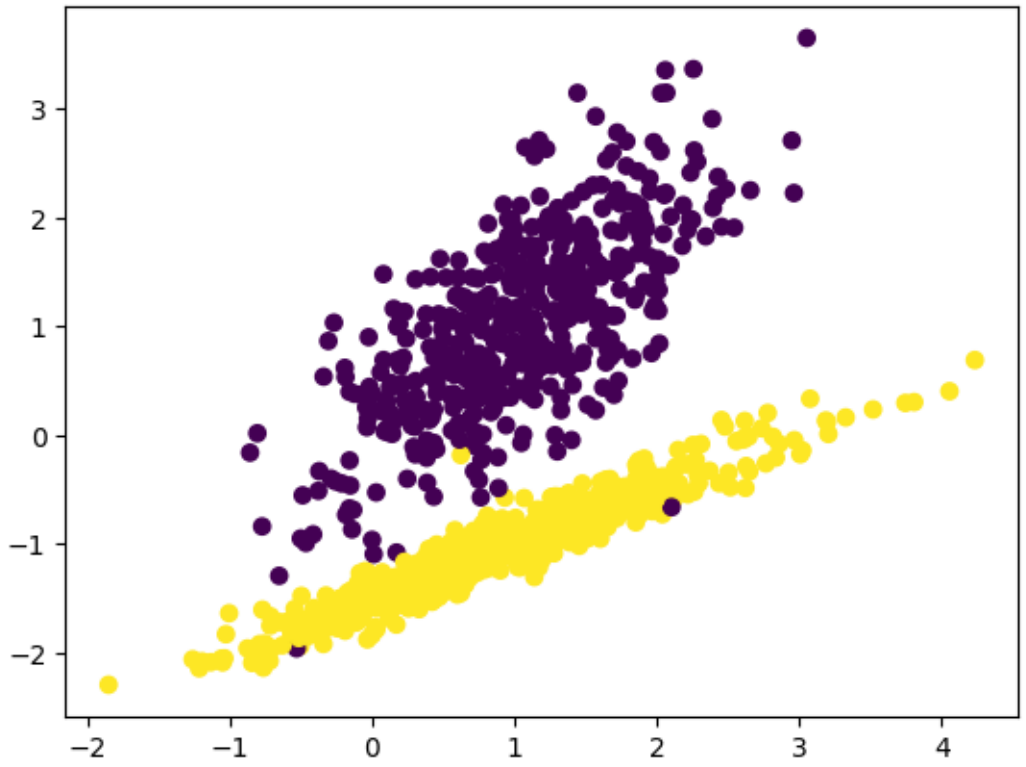
n_redundant ویژگی های تکراری را مشخص میکند که آن را روی مقدار پیشفرض آن که صفر میباشد قرار می دهیم.

n_cluster_per_class تعداد خوشه های هر کلاس را معین میکند که برای راحتی کار آن را در این بخش روی ۱ قرار می دهیم.

Class_sep میزان در هم تنیدگی و تفکیک داده ها را مشخص میکند که هر چقدر این عدد به صفر نزدیک تر باشد داده ها به یک دیگر نزدیک تر و هر چقدر از صفر بزرگتر باشد داده ها با فاصله بیشتری نسبت به یکدیگر قرار دارند.

Random_state حالت تصادفی داده ها را معین میکند و برای اینکه نتیجه اجرای کد با هر بار اجرا تغییر نکند باید مقداری برای آن مشخص کنیم. در این بخش دو رقم آخر شماره دانشجویی خود که برابر ۸۳ می باشد را قرار می دهیم.

در نهایت برای نمایش بصری دیتاست ایجاد شده با استفاده از تابع scatter از کتابخانه matplotlib دیتاست را plot میکنیم که به صورت زیر شکل میگیرد.



۲. با استفاده از حداقل دو طبقه‌بند آمادهٔ پایتون و در نظر گرفتن فراپارامترهای مناسب، دو کلاس موجود در دیتاست قسمت قبلی را از هم تفکیک کنید. ضمن توضیح روند انتخاب فراپارامترها (مانند تعداد دورهٔ آموزش و نرخ یادگیری)، نتیجهٔ دقت آموزش و ارزیابی را نمایش دهید. برای بهبود نتیجه از چه تکنیک‌هایی استفاده کردید؟

با استفاده از تابع train_test_split دیتا های خود را به دو بخش train و test تقسیم می‌کنیم. نسبت این تقسیم بندی را با آرگومان test_size به روی ۲۰ درصد قرار می دهیم و random_state را همانند بخش های قبل برابر ۲ رقم آخر شماره دانشجویی که ۸۳ میباشد قرار می‌دهیم.

سپس برای طبقه بندی داده ها از تابع LogisticRegression استفاده کرده و الگوریتم بهینه سازی را 'sag' یا همان Stochastic Average Gradient را انتخاب میکنیم که بر مبنای گرادیان فرایند بهینه سازی را انجام می‌دهد.

Max_iter نمایانگر تعداد تکرار الگوریتم مورد نظر می باشد که آن را باید برابر با مقداری نه چندان کم و نه چندان زیاد بگذاریم که مانع از همگرا نشدن و زمان محاسبه بالای الگوریتم شویم.

دقت test و train را نیز با استفاده از model.score بدست می آوریم.

سپس از طبقه بند دیگری به نام SGDClassifier استفاده میکنیم. این طبقه بند نیز بر اساس گرادیان کار میکند. پارامتر های این تابع را همانند تابع قبل انتخاب میکنیم با این تفاوت که الگوریتم بهینه سازی آن را log_loss انتخاب میکنیم.

دقت test و train را نیز با استفاده از model.score همانند بخش قبا بدست می آوریم و پرینت میکنیم.

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, SGDClassifier
#split
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=83)
#logistic regression
model1 = LogisticRegression(solver='sag',max_iter=2000,random_state=83)
#train
model1.fit(X_train,y_train)
#prediction
model1.predict(X_test),y_test

print('LogisticRegression Train score is:',model1.score(X_train,y_train))
print('LogisticRegression Test score is:',model1.score(X_test,y_test))
#SGDClassifier
model2=SGDClassifier(loss='log_loss', max_iter=2000, random_state=83)
#train
model2.fit(X_train,y_train)
#prediction
model2.predict(X_test),y_test

print('SGDClassifier Train score is:',model2.score(X_train,y_train))
print('SGDClassifier Test score is:',model2.score(X_test,y_test))
```

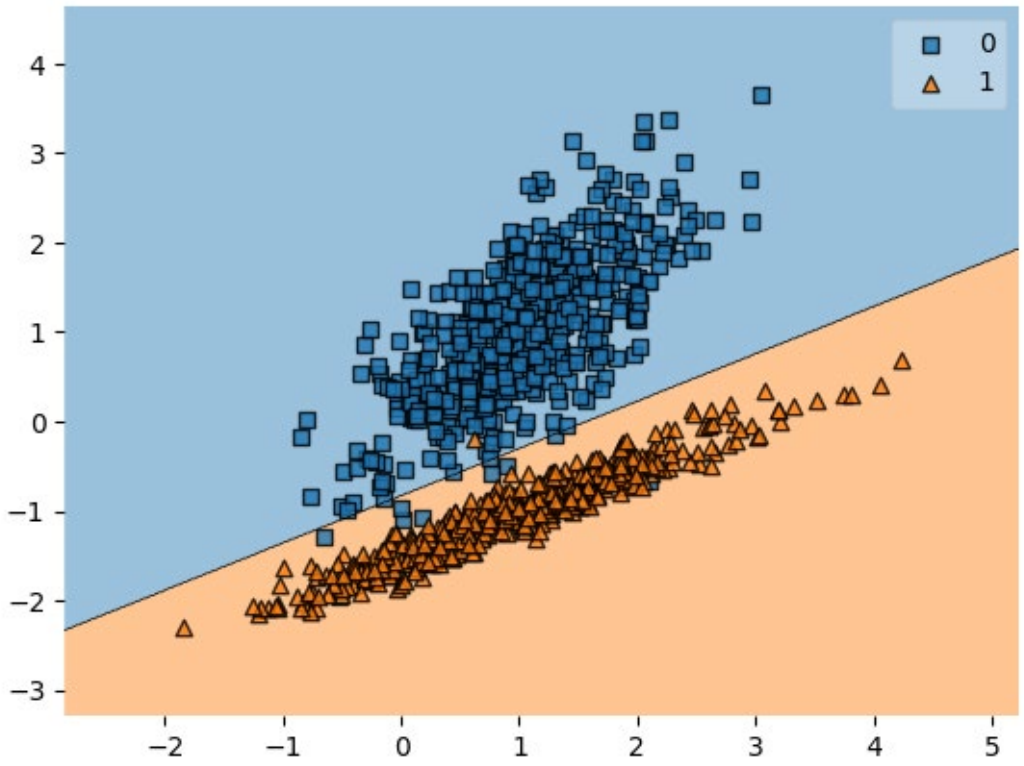
```
LogisticRegression Train score is: 0.99
LogisticRegression Test score is: 0.99
SGDClassifier Train score is: 0.99125
SGDClassifier Test score is: 0.99
```

۳.مرز و نواحی تصمیم گیری برآمده از مدل آموزش دیدهٔ خود را به همراه نمونه ها در یک نمودار نشان دهید. اگر می توانید نمونه هایی که اشتباه طبقه بندی شده اند را با شکل متفاوت نمایش دهید.

با استفاده از تابع plot_decision_regions مرز و نواحی تصمیم گیری مشخص می‌شوند.

```
from mlxtend.plotting import plot_decision_regions

# Logistic Regression
plot_decision_regions(X, y , clf=model1)
```

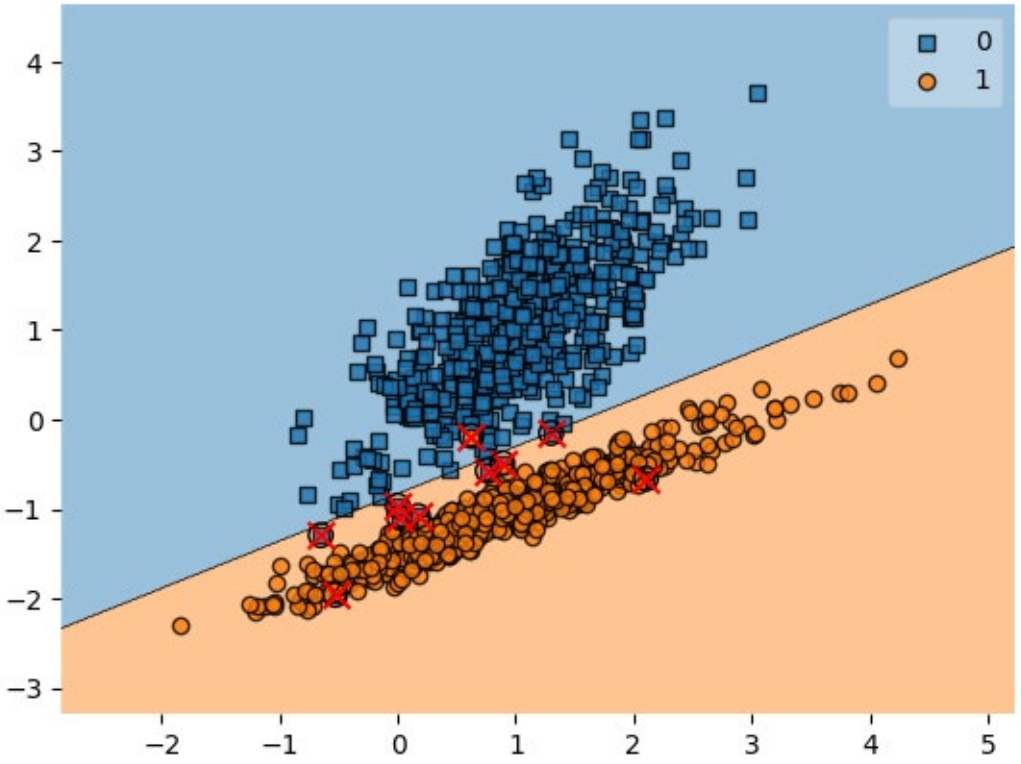
که به شکل زیر به نمایش در می آید.



نمونه های اشتباه تشخیص داده شده نیز با استفاده از دستورات زیر بر روی پلات به نمایش در می آیند.

```
misclassified1 = model1.predict(X) != y
plot_decision_regions(X, y, clf=model1, markers='so', X_highlight=X[misclassified1])
plt.scatter(X[misclassified1][:, 0],
            X[misclassified1][:, 1],
            c='red', marker='x',
            s=100, label='Misclassified')
plt.show()
```

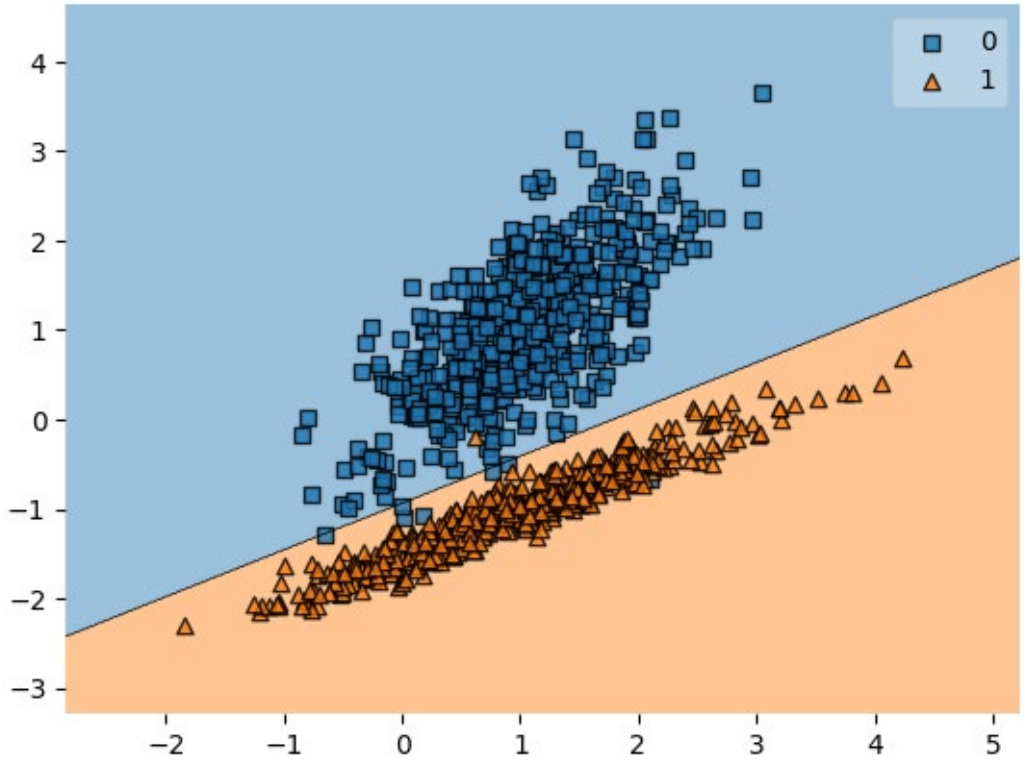
نتیجه به شکل زیر می باشد.



همین فرایند را برای طبقه بند دوم نیز تکرار می‌کنیم.

```
# SGD Classifier
plot_decision_regions(X, y , clf=model2)
```

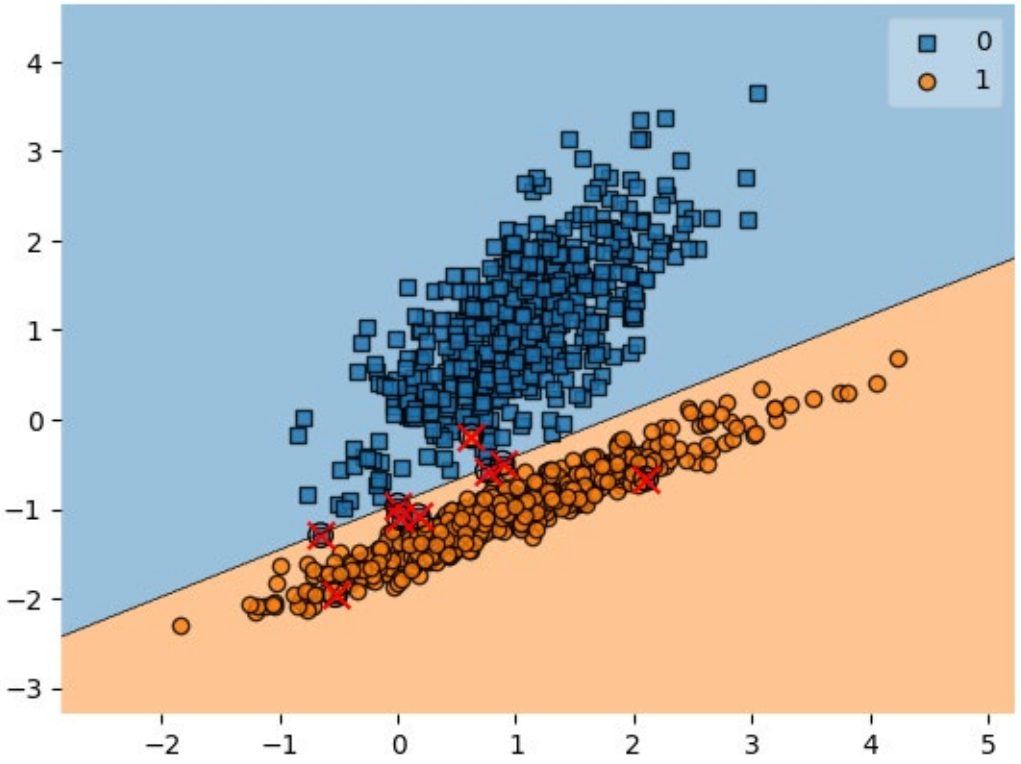
ناحیه بندی به شکل زیر صورت میگیرد.



داده های misclassify شده نیز با دستور زیر نمایش داده می شوند.

```
misclassified2 = model2.predict(X) != y
plot_decision_regions(X, y, clf=model2, markers='so', X_highlight=X[misclassified2])
plt.scatter(X[misclassified2][:, 0],
            X[misclassified2][:, 1],
            c='red', marker='x',
            s=100, label='Misclassified')
plt.show()
```

که در شکل زیر قابل مشاهده می‌باشند.



« »

« » « »

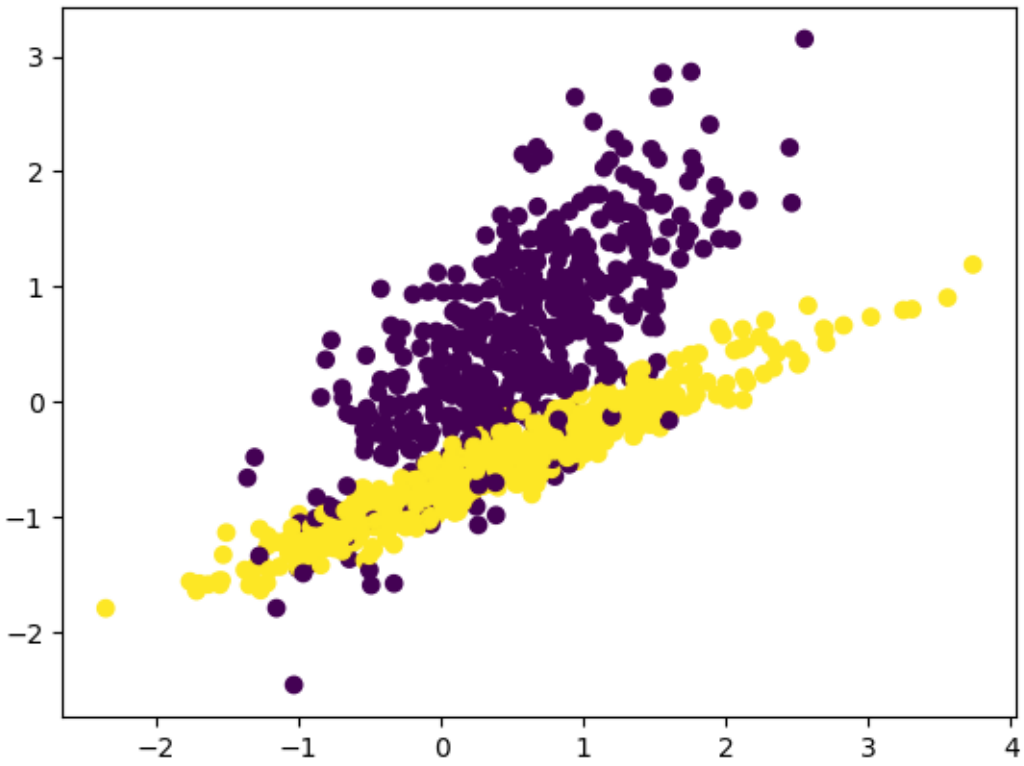
.

برای افزایش چالش برای classification داده ها می توان از دو آرگومان class_sep و clusters_per_class استفاده کرد.

با کاهش Class_sep میزان در هم تنیدگی داده ها افزایش یافته و کلاس بندی آن ها دشوار تر می‌شود.

Clusters_per_class نیز تعیین کننده تعداد خوشه های داده های هر کلاس می باشد.

```
X,y = make_classification(n_samples=1000,
                          n_features=2,
                          n_redundant=0,
                          n_classes=2,
                          n_clusters_per_class=1,
                          class_sep=0.5,
                          random_state=83)
plt.scatter(X[:,0],X[:,1],c=y)
```



همانگونه که انتظار می‌رفت با کاهش میزان class_sep به ۰.۵ در هم تنیدگی داده ها بیشتر شد.

در این حالت با انجام classification داده های بیشتری به صورت اشتباه تشخیص داده می‌شوند و به تبع آن دقت فرایند طبقه بندی کاهش می یابد.

بخش ۲ و ۳ را برای این حالت جدید تکرار می کنیم.

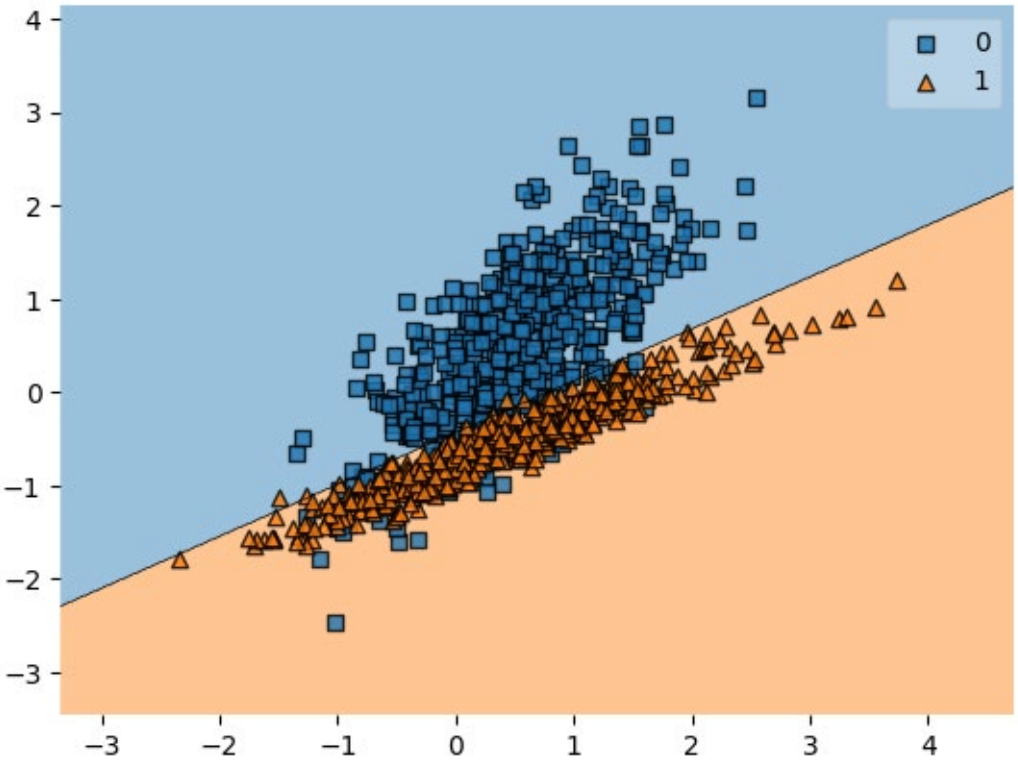
```
#split
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=83)
#logistic regression
model1 = LogisticRegression(solver='sag',max_iter=2000,random_state=83)
#train
model1.fit(X_train,y_train)
#prediction
model1.predict(X_test),y_test

print('LogisticRegression Train score is:',model1.score(X_train,y_train))
print('LogisticRegression Test score is:',model1.score(X_test,y_test))
#SGDClassifier
model2=SGDClassifier(loss='log_loss', max_iter=2000, random_state=83)
#train
model2.fit(X_train,y_train)
#prediction
model2.predict(X_test),y_test

print('SGDClassifier Train score is:',model2.score(X_train,y_train))
print('SGDClassifier Test score is:',model2.score(X_test,y_test))
```

```
LogisticRegression Train score is: 0.91875
LogisticRegression Test score is: 0.94
SGDClassifier Train score is: 0.9175
SGDClassifier Test score is: 0.92
```

همانگونه که انتظار می‌رفت دقت کاهش یافته است.

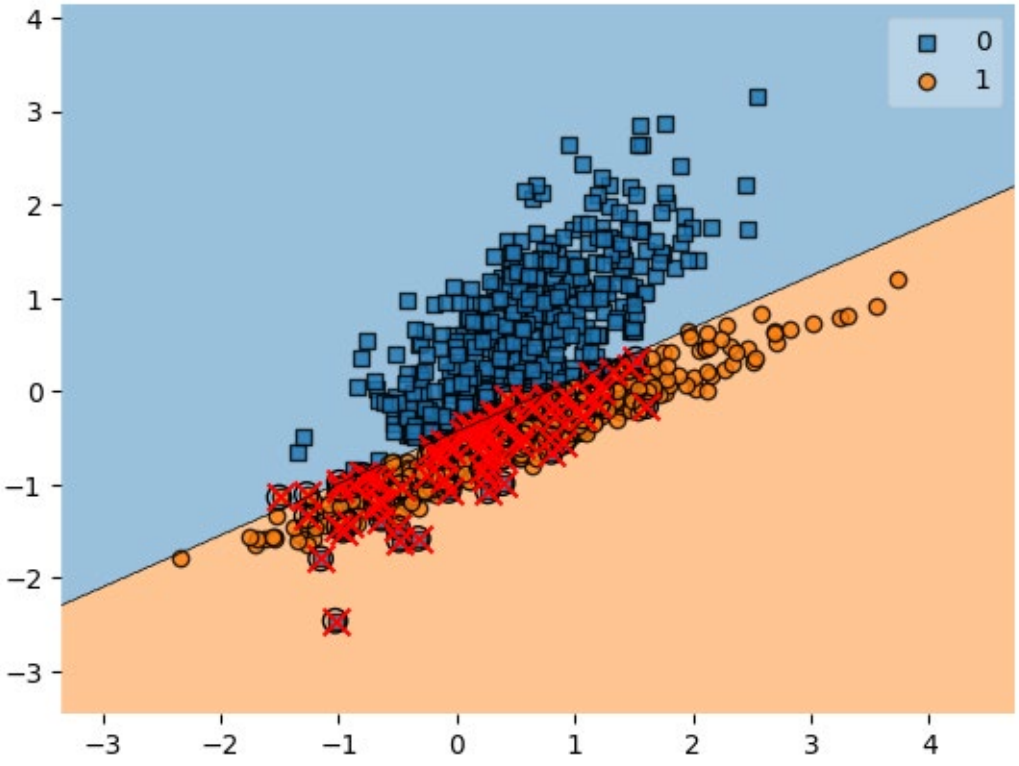
سپس ناحیه بندی را انجام می دهیم.

```
# Logistic Regression
plot_decision_regions(X, y , clf=model1)
```



داده های misclassify شده را رسم میکنیم.

```
#LogisticRegression misclassified data
misclassified1 = model1.predict(X) != y
plot_decision_regions(X, y, clf=model1, markers='so', X_highlight=X[misclassified1])
plt.scatter(X[misclassified1][:, 0],
            X[misclassified1][:, 1],
            c='red', marker='x',
            s=100, label='Misclassified')
plt.show()
```

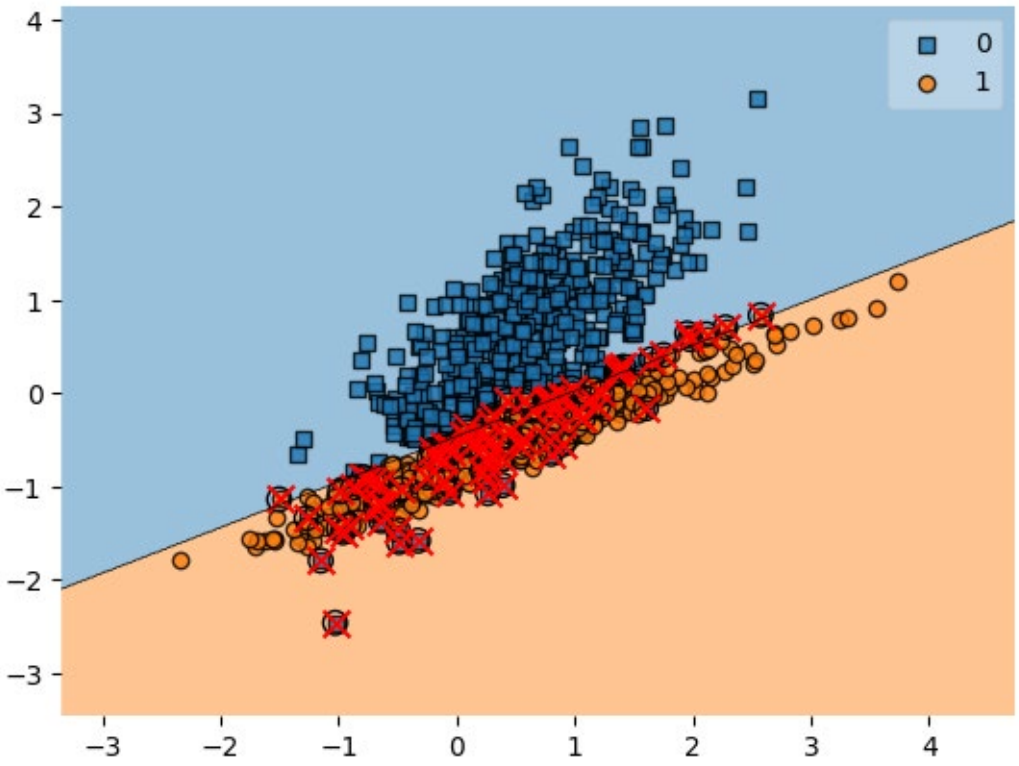


برای طبقه بندی دیگر نیز این کار را تکرار میکنیم.

```
#SGDClassifier misclassified data
misclassified2 = model2.predict(X) != y
plot_decision_regions(X, y, clf=model2, markers='so', X_highlight=X[misclassified2])
plt.scatter(X[misclassified2][:, 0],
            X[misclassified2][:, 1],
            c='red', marker='x',
            s=100, label='Misclassified')
plt.show()
```

اگر یک کلاس به داده های تولیدشده در قسمت «۱» اضافه شود، در کدام قسمت ها از بلوک دیاگرام آموزش و ارزیابی تغییراتی ایجاد می شود؟ در مورد این تغییرات توضیح دهید. آیا می توانید در این حالت پیاده

سازی را به راحتی و با استفاده از کتابخانه ها و کدهای آمادهٔ پایتونی انجام دهید؟ پیاده سازی کنید.



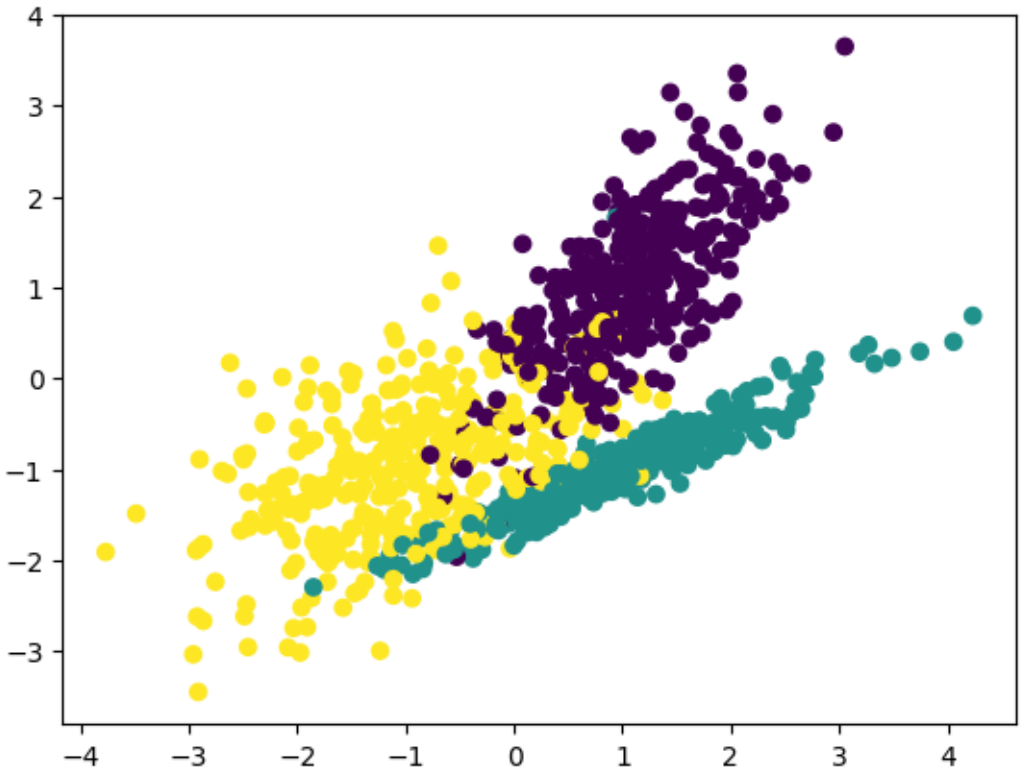
۵. اگر یک کلاس به داده های تولیدشده در قسمت «۱» اضافه شود، در کدام قسمت ها از بلوک دیاگرام آموزش و ارزیابی تغییراتی ایجاد می شود؟ در مورد این تغییرات توضیح دهید. آیا می توانید در این حالت پیاده سازی را به راحتی و با استفاده از کتابخانه ها و کدهای آمادۀ پایتونی انجام دهید؟ پیاده سازی کنید.

به دلیل افزایش تعداد کلاس در تعداد داده معین دقت کلاس بندی کاهش می یابد زیرا نمونه های کمتری از هر داده برای train در دسترس می باشد.

با استفاده از آرگومان n_classes تعداد کلاس ها را به ۳ افزایش میدهیم.

```
X,y = make_classification(n_samples=1000,
                          n_features=2,
                          n_redundant=0,
                          n_classes=3,
                          n_clusters_per_class=1,
                          class_sep=1,
                          random_state=83)
plt.scatter(X[:,0],X[:,1],c=y)
```

چیدمان داده ها به شکل زیر در می آید.



برای طبقه بندی داده ها مانند قبل عمل میکنیم با این تفاوت که multi_class را از ovr که یکی در مقابل همه بود به multinominal تغییر می‌دهیم.

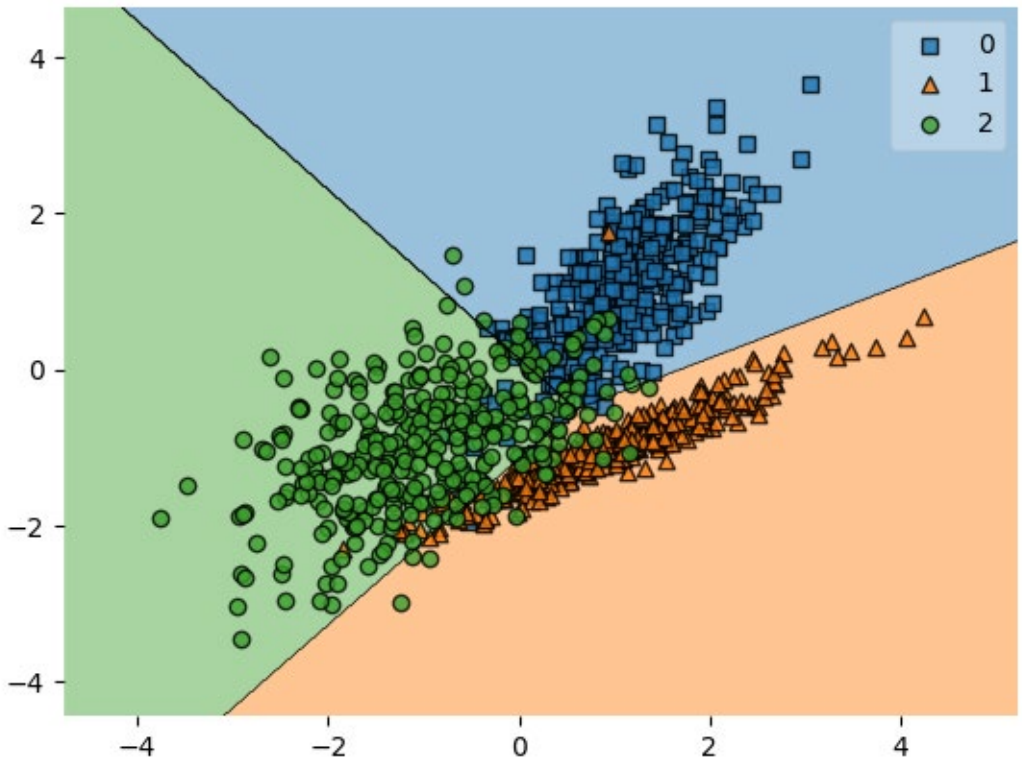
همچنین چون تعداد داده ها بیشتر از ۲ می باشد تارگت ها از حالت صفر و یک خارج شده بنابراین نمی توان از log_loss استفاده نمود و از sag استفاده میکنیم.

```
X_train,X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=83)

#LogisticRegression
model1 = LogisticRegression(solver='sag',max_iter=2000,multi_class='multinomial',random_state=83)
#train
model1.fit(X_train,y_train)
#prediction
model1.predict(X_test),y_test

print('LogisticRegression Train score is:',model1.score(X_train,y_train))
print('LogisticRegression Test score is:',model1.score(X_test,y_test))
```

```
LogisticRegression Train score is: 0.9075
LogisticRegression Test score is: 0.92
```

همانند بخش های قبل مرز و نواحی تصمیم گیری را معین میکنیم.

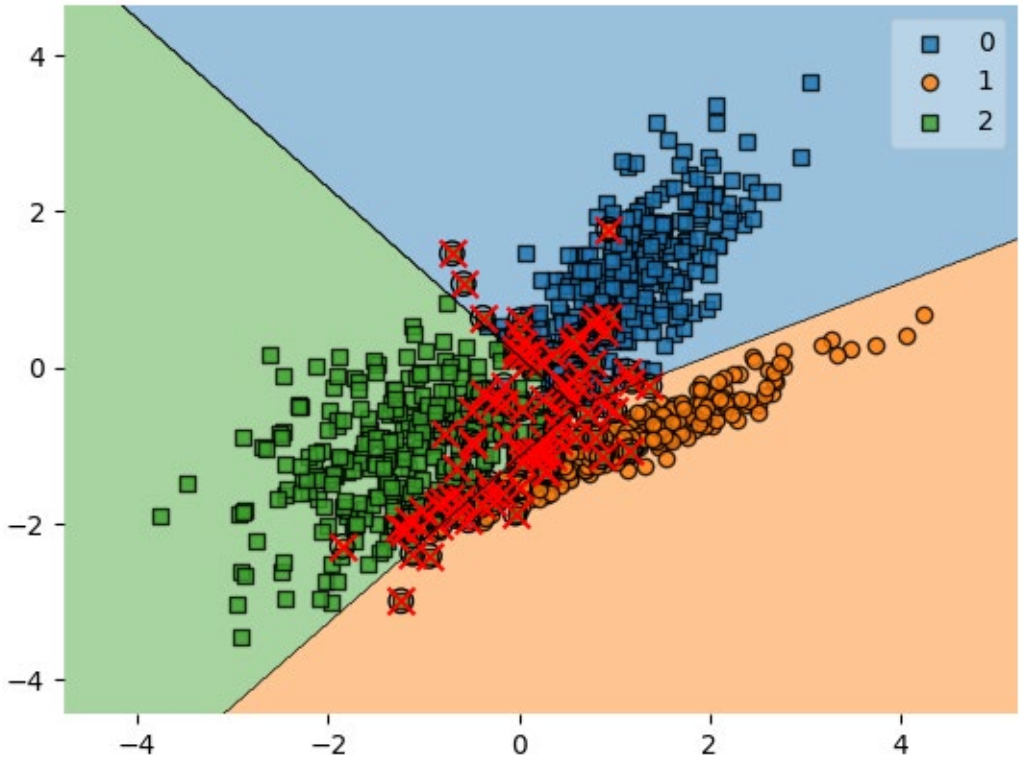
```
plot_decision_regions(X, y , clf=model1)
```



داده های misclassify شده را نیز به صورت زیر مشخص میکنیم.

```
misclassified1 = model1.predict(X) != y
plot_decision_regions(X, y, clf=model1, markers='so', X_highlight=X[misclassified1])
plt.scatter(X[misclassified1][:, 0],
            X[misclassified1][:, 1],
            c='red', marker='x',
            s=100, label='Misclassified')
plt.show()
```

نتیجه به شکل زیر در می آید.



سوال ۲

۱. با مراجعه به این پیوند با یک دیتاست مربوط به حوزهٔ «بانکی» آشنا شوید و ضمن توضیح کوتاه اهداف و ویژگی‌هایش، فایل آن را دانلود کرده و پس از بارگذاری در گوگل‌درایو خود، آن را با دستور gdown در محیط گوگل‌کولب قرار دهید. اگر تغییر فرمتی برای فایل این دیتاست نیاز می‌بینید، این کار را با دستورهای پایتونی انجام دهید.

داده ها از تصاویری که از نمونه های واقعی و جعلی شبیه اسکناس گرفته شده بودند استخراج شد .برای دیجیتالی کردن، از یک دوربین صنعتی که معمولاً برای بازرسی چاپ استفاده می شود استفاده می شود .تصاویر نهایی دارای ۴۰۰*۴۰۰ پیکسل هستند .با توجه به لنز شی و فاصله تا جسم مورد بررسی، تصاویری در مقیاس خاکستری با وضوح حدود ۶۶۰ نقطه در اینچ به دست آمد .ابزار تبدیل موجک برای استخراج ویژگی ها از تصاویر استفاده شد.

این دیتاست شامل ۱۳۷۲ داده از اسکناس می باشد که هر نمونه ۴ ویژگی زیر را در بر دارد:

واریانس تصویر، کجی تصویر، چولگی تصویر و انتروپی تصویر

هدف این دیتاست تمایز اسکناس جعلی از اصلی می‌باشد.

ابتدا دیتاست مورد نظر را در گوگل درایو اپلود کرده و با دستور زیر آن را بارگیری می‌کنیم.

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1oibux5Yko5FCxJ2mJd6aRnr_mX4NkRE9
```

در مرحله بعد ابتدا کتابخانه های مورد نیاز را import کرده و با استفاده از دستور pd.read_csv دیتاست را لود میکنیم. توجه می شود به دلیل اینکه این دیتاست header ندارد، header آن را برابر None قرار میدهیم.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

df = pd.read_csv('/content/data_banknote_authentication.txt' ,header=None)
df
```

دیتا ست به شکل زیر لود می‌شود.

	0	1	2	3	4
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

سپس برای دیتاست خود header تعریف کرده و ویژگی هر ستون را وارد میکنیم.

```
columns_name = ['variance','skewness','curtosis','entropy','target']
df.columns = columns_name
df
```

	variance	skewness	curtosis	entropy	target
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

۲. ضمن توضیح اهمیت فرآیند برزدن (مخلوط کردن)، داده‌ها را مخلوط کرده و با نسبت تقسیم دلخواه و معقول به دو بخش «آموزش» و «ارزیابی» تقسیم کنید.

اهمیت shuffling این است که باعث می شود مدل یادگیری ماشین به طور برابر از همه داده ها یاد بگیرد. اگر داده ها بدون بر زدن استفاده شوند ممکن است مدل به طور تصادفی از بعضی داده ها بیشتر از بقیه یاد بگیرد که این پدیده موجب کاهش دقت مدل می‌شود.

برای shuffle کردن داده ها از df.sample استفاده می‌کنیم و random_state را برابر ۸۳ قرار می دهیم.

سپس ایندکس ها را ریست می کنیم.

```
df_shuffled = df.sample(frac=1,random_state=83)
df_shuffled.reset_index(inplace = True, drop = True)
df_shuffled
```

داده ها را به X و y، assign کرده و با استفاده از train_test_split آن ها را با نسبت ۸۰ به ۲۰ تقسیم می‌کنیم.

```
X = df_shuffled.iloc[:, 0:4]
y = df_shuffled.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=83)
```

۳.بدون استفاده از کتابخانه‌های آمادهٔ پایتون، مدل، تابع اتلاف و الگوریتم یادگیری و ارزیابی را کدنویسی کنید تا دو کلاس موجود در دیتاست به خوبی از یکدیگر تفکیک شوند. نمودار تابع اتلاف را رسم کنید و نتیجهٔ دقت ارزیابی روی داده‌های تست را محاسبه کنید. نمودار تابع اتلاف را تحلیل کنید. آیا می‌توان از روی نمودار تابع اتلاف و قبل از مرحلهٔ ارزیابی با قطعیت در مورد عمل‌کرد مدل نظر داد؟ چرا و اگر نمی‌توان، راه‌حل چیست؟

ابتدا تابع سیگموید را برای logistic regression تعریف کرده، سپس خود تابع logistic regression را تعریف میکنیم که با استفاده از ضرب x ها در w ها y_hat را تولید کنیم.

سپس تابع اتلاف binary cross entropy را تعریف می کنیم.

در مرحله بعد گرادیان را تعریف کرده و با استفاده از Eta (ضریب یادگیری) و حاصل ضرب x transpose در اختلاف y_hat و y (Error) w را آپدیت میکنیم.

```python
def sigmoid(x):
  return 1 / (1+np.exp(-x))

def logistic_regression(x,w):
  y_hat = sigmoid(x @ w)
  return y_hat

def bce(y, y_hat):
  loss = -(np.mean(y*np.log(y_hat)+(1-y)*np.log(1-y_hat)))
  return loss

def gradient(x, y, y_hat):
  grads = (x.T @ (y_hat - y)) / len(y)
  return grads

def gradient_descent(w, eta, grads):
  w -= eta*grads
  return w

def accuracy(y , y_hat):
  acc = np.sum(y == np.round(y_hat)) / len(y)
  return acc
```

X_train را به آرایه تبدیل کرده و یک ستون تماما یک به آن اضافه می‌کنیم تا بایاس آن نیز لحاظ شود.

Y_train را با استفاده از دستور reshape به یک آرایه ۲ بعدی تبدیل میکنیم که ضرب ماتریسی ممکن شود.

سپس یک آرایه ۵ در ۱ (تعداد ویژگی ها + بایاس) تصادفی از w میسازیم و ضریب یادگیری را برابر ۰.۰۱ و epoch را برابر ۱۰۰۰ قرار می دهیم تا الگوریتم ۱۰۰۰ بار تکرار شود و w ها آپدیت بشوند.

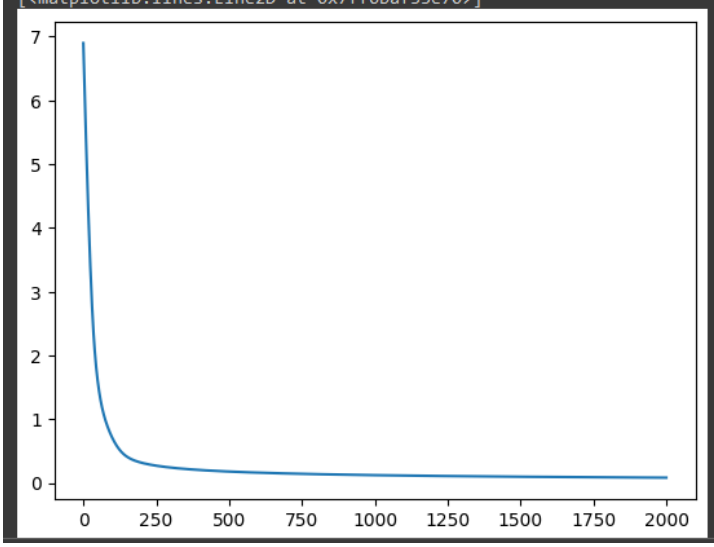
```python
w = np.random.randn(5,1)
eta = 0.01
n_epochs = 2000
```

آرایه خالی error_hist را میسازیم و Error ها را در آن ذخیره میکنیم سپس آن را پلات میکنیم.

```python
error_hist = []

for epoch in range(n_epochs):
  y_hat = logistic_regression(X_train, w)

  e = bce(y_train, y_hat)
  error_hist.append(e)

  grads = gradient(X_train, y_train, y_hat)

  w = gradient_descent(w, eta, grads)

  if (epoch+1) % 100 == 0:
    print(f'Epoch={epoch},\t E={e:.4f}, \t w={w.T[0]}')

plt.plot(error_hist)
```

نتیجه به شکل زیر حاصل میگردد.

```
Epoch=99,    E=0.7104,   w=[-0.58408727 -0.1375081  -0.19825317 -0.09205598  0.39533299]
Epoch=199,   E=0.3180,   w=[-0.41783694 -0.58199955 -0.17409421 -0.13296821  0.056955  ]
Epoch=299,   E=0.2418,   w=[-0.30264171 -0.75214914 -0.2572989  -0.22577216 -0.07047836]
Epoch=399,   E=0.2035,   w=[-0.20406729 -0.85765824 -0.32967839 -0.30633139 -0.14339556]
Epoch=499,   E=0.1792,   w=[-0.11567663 -0.93568271 -0.38908376 -0.37273352 -0.19160141]
Epoch=599,   E=0.1619,   w=[-0.03496206 -0.99815414 -0.4387901  -0.42879855 -0.22543009]
Epoch=699,   E=0.1488,   w=[ 0.03959806 -1.05051378 -0.48132243 -0.47733631 -0.24998274]
Epoch=799,   E=0.1384,   w=[ 0.10905484 -1.09573502 -0.51841194 -0.52019062 -0.26818033]
Epoch=899,   E=0.1298,   w=[ 0.17417758 -1.13563264 -0.5512566  -0.55860056 -0.28184014]
Epoch=999,   E=0.1226,   w=[ 0.23555141 -1.17140087 -0.58070855 -0.59343333 -0.29215808]
Epoch=1099,  E=0.1163,   w=[ 0.29363486 -1.20386991 -0.60739151 -0.62531945 -0.29995492]
Epoch=1199,  E=0.1109,   w=[ 0.34879618 -1.23364153 -0.63177403 -0.65473242 -0.30581296]
Epoch=1299,  E=0.1061,   w=[ 0.40133717 -1.26116654 -0.65421623 -0.68203753 -0.31015663]
Epoch=1399,  E=0.1018,   w=[ 0.4515095  -1.28679175 -0.67500045 -0.70752304 -0.3133023 ]
Epoch=1499,  E=0.0980,   w=[ 0.49952605 -1.31078983 -0.69435202 -0.73142078 -0.31549029]
Epoch=1599,  E=0.0945,   w=[ 0.54556914 -1.3333791  -0.71245348 -0.75392017 -0.31690606]
Epoch=1699,  E=0.0914,   w=[ 0.58979664 -1.35473708 -0.72945478 -0.77517824 -0.31769477]
Epoch=1799,  E=0.0885,   w=[ 0.63234647 -1.37501005 -0.74548058 -0.79532667 -0.31797133]
Epoch=1899,  E=0.0858,   w=[ 0.67334009 -1.3943199  -0.76063563 -0.8144771  -0.31782768]
Epoch=1999,  E=0.0834,   w=[ 0.71288528 -1.41276926 -0.77500882 -0.83272506 -0.31733804]
[<matplotlib.lines.Line2D at 0x7ff6baf35e70>]
```



همانطور که انتظار میرود، با پیشرفت آموزش، تابع اتلاف به تدریج کاهش پیدا میکند. این نشان دهنده بهینه سازی و بهبود وزن ها در جهتی که تابع هزینه کمینه شود، است. نمودار نشان میدهد که تابع اتلاف همگراست و به یک مقدار پایدار همگرا میشود. این نشاندهنده این است که فرآیند آموزش به یک وزن بهینه رسیده است.

تحلیل نمودار تابع اتلاف مهم است تا مشاهده کنیم که آیا مدل به اندازه کافی بهینه شده است یا نیاز به افزایش تعداد ایپاک ها داریم. از آنجایی که تابع اتالف به سرعت کاهش پیدا کرده و سپس به یک مقدار ثابت رسیده، نشاندهنده برازش خوب مدل است.

نمودار تابع اتالف میتواند به ما اطالعات مهمی درباره عملکرد مدل بدهد، اما این تا حدی است که به علتی به نامoverfitting، ممکن است مدل به دادههای آموزش بسیار خوب برازش یافته باشد ولی بر روی دادههای تست یا دادههای جدید عملکرد مناسبی نداشته باشد. به عبارت دیگر، این ممکن است نشاندهنده این باشد که مدل به دادههای آموزش "خاص" شده و توانمندی عمومی برای تفکیک موارد جدید را نداشته باشد به همین دلیل، نمودار تابع اتلاف به تنهایی کافی نیست و نیاز به ارزیابی روی دادههای جدید تست داریم.

همانند مراحل طی شده برای داده های train ، داده های تست را نیز classify میکنیم با این تفاوت که این بار وزن ها اپدیت نمیشوند و از اخرین w به دست آمده در فرایند train استفاده می‌شود.

```
[13] X_test = np.asarray(X_test)
     X_test = np.hstack((np.ones((len(X_test), 1)), X_test))

     print(X_test)
```

```
[[ 1.       1.8967  -2.5163   2.8093  -0.79742]
 [ 1.      -1.4454  -8.4385   8.8483   0.96894]
 [ 1.       4.1654  -3.4495   3.643    1.0879 ]
 ...
 [ 1.      -2.899   -0.60424  2.6045   1.3776 ]
 [ 1.       1.645    7.8612  -0.87598 -3.5569 ]
 [ 1.       1.5268  -5.5871   8.6564  -1.722  ]]
```

```
[14] y_test = np.array(y_test)
     y_test = y_test.reshape(-1,1)
     y_hat = np.array(y_hat)
     y_hat = y_hat.reshape(-1,1)
```

```
[15] w=[ 0.71288528, -1.41276926, -0.77500882, -0.83272506, -0.31733804]
     w = np.array(w)
     w = w.reshape(-1,1)
```

```
y_hat = logistic_regression(X_test,w)
accuracy(y_test, y_hat)
```

```
0.9745454545454545
```

۴.حداقل دو روش برای نرمال‌سازی داده‌ها را با ذکر اهمیت این فرآیند توضیح دهید و با استفاده از یکی از این روش‌ها، داده‌ها را نرمال کنید. آیا از اطلاعات بخش «ارزیابی» در فرآیند نرمال‌سازی استفاده کردید؟ چرا؟

نرمال‌سازی یا استانداردسازی داده‌ها یک مرحله مهم در پیش‌پردازش داده‌ها در مدل‌های یادگیری ماشین است. این عملیات به تبدیل داده‌ها به یک مقیاس یکسان و استاندارد می‌پردازد. اهمیت نرمال‌سازی داده‌ها در مدل‌های یادگیری ماشین به چند دلیل مهم برمی‌گردد:

سرعت آموزش:

مدل‌های یادگیری ماشین بسیار وابسته به عملکرد بهینه‌ای هستند. نرمال‌سازی باعث می‌شود که مدل به سرعت‌تر و بهتر از داده‌ها یاد بگیرد. اگر تفاوت مقیاس در ویژگی‌ها وجود داشته باشد (برای مثال، یک ویژگی با مقیاس ۰ تا ۱ و دیگری با مقیاس ۰ تا ۱۰۰۰)، فرآیند یادگیری ممکن است به سختی و یا حتی به طور کلی متوقف شود.

عملکرد بهتر مدل:

نرمال‌سازی موجب می‌شود که مدل بهتر و کارآمدتر عمل کند. اگر توزیع داده‌ها به شدت انحراف داشته باشد، ممکن است مدل به یک طرفه (biased) یاد بگیرد و برای داده‌های جدید به درستی پاسخ ندهد.

پایداری آموزش:

نرمال‌سازی موجب می‌شود که فرآیند یادگیری پایدارتر باشد. مدل با توجه به مقیاس یکسان داده‌ها، به سمت مینیمم محلی بهبود می‌یابد و به احتمال زیاد در مسیر مناسب‌تری بهینه می‌شود

جلوگیری از مشکل مشتق ناپذیری:

در الگوریتم‌های بهینه‌سازی مانند گرادیان کاهشی، مشکل مشتقات ناپذیر (vanishing gradients)ممکن است پیش بیاید. این مشکل زمانی رخ می‌دهد که یک یا چند ویژگی با مقیاس بسیار بزرگ باشند. نرمال‌سازی کمک می‌کند تا این مشکلات کاهش یابد.

به طور کلی، نرمال‌سازی داده‌ها باعث بهبود پایداری، عملکرد و سرعت یادگیری مدل‌های یادگیری ماشین می‌شود.

دو روش اصلی برای نرمال‌سازی داده‌ها در مدل‌های یادگیری ماشین عبارتند از:

Min-Max Scaling (Normalization)

در این روش، داده‌ها به گونه‌ای تبدیل می‌شوند که به یک بازه مشخص محدود می‌شوند، معمولاً به بازه [۰, ۱] یا [-۱, ۱]. فرمول Min-Max Scaling برای یک ویژگی x به صورت زیر است:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Z-score Standardization (Standard Scaling)

در این روش، داده‌ها به واحد استاندارد تبدیل می‌شوند، به این معنا که میانگین داده‌ها برابر با صفر می‌شود و انحراف معیار آنها برابر با یک. این روش برای داده‌هایی که دارای توزیع نزدیک به نرمال هستند مناسب است.

$$X_{normalized} = \frac{X - \mu}{\sigma}$$

اکنون به روش Min-Max scaling به نرمال سازی داده ها میپردازیم.

```
normalized_df = (df - df.min())/(df.max()-df.min())
normalized_df
```

	variance	skewness	curtosis	entropy	target
0	0.769004	0.839643	0.106783	0.736628	0.0
1	0.835659	0.820982	0.121804	0.644326	0.0
2	0.786629	0.416648	0.310608	0.786951	0.0
3	0.757105	0.871699	0.054921	0.450440	0.0
4	0.531578	0.348662	0.424662	0.687362	0.0
...	...	...	...	...	...
1367	0.537124	0.565855	0.165249	0.726398	1.0
1368	0.407690	0.332868	0.506753	0.808350	1.0
1369	0.237385	0.011768	0.985603	0.524755	1.0
1370	0.250842	0.201701	0.761587	0.660675	1.0
1371	0.324528	0.490747	0.343348	0.885949	1.0

1372 rows × 5 columns

در نرمال سازی داده ها باید هم داده های test و هم داده های train را نرمال کنیم.

۵. تمام قسمت‌های «۱» تا «۳» را با استفاده از داده‌های نرمال‌شده تکرار کنید و نتایج پیش‌بینی مدل را برای پنج نمونه داده نشان دهید.

مراحل مانند سوال های قبل انجام می شود.

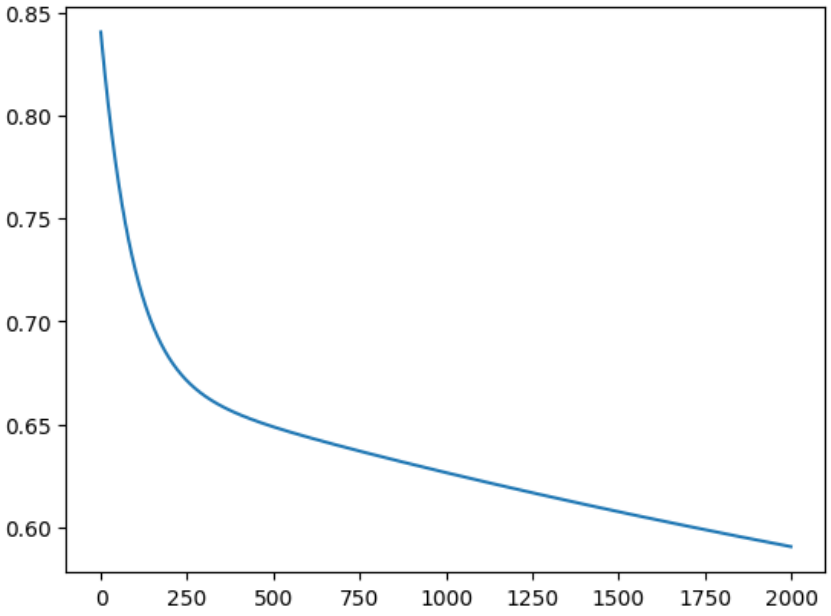
```
X1 = normalized_df.iloc[:, 0:4]
y1 = normalized_df.iloc[:, -1]

X_train1,X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.2,random_state=83)
X_train1.shape, y_train1.shape, X_test1.shape, y_test1.shape
```

```
((1097, 4), (1097,), (275, 4), (275,))
```

```
X_train1 = np.asarray(X_train1)

X_train1 = np.hstack((np.ones((len(X_train1), 1)), X_train1))

y_train1 = np.array(y_train1)
y_train1 = y_train1.reshape(-1,1)

w = np.random.randn(5,1)
```

```
error_hist = []

for epoch in range(n_epochs):
  y_hat = logistic_regression(X_train1, w)

  e = bce(y_train1, y_hat)
  error_hist.append(e)

  grads = gradient(X_train1, y_train1, y_hat)

  w = gradient_descent(w, eta, grads)

  if (epoch+1) % 100 == 0:
    print(f'Epoch={epoch},\t E={e:.4f}, \t w={w.T[0]}')

plt.plot(error_hist)
```

```
Epoch=99,  E=0.7258,     w=[ 0.02864999  0.15755866 -0.03639169  0.58320378  0.03484548]
Epoch=199,     E=0.6820,     w=[-0.07271897  0.03151151 -0.14017491  0.56382116 -0.03717537]
Epoch=299,     E=0.6642,     w=[-0.11949003 -0.06352507 -0.21072396  0.55937059 -0.07246443]
Epoch=399,     E=0.6550,     w=[-0.13484437 -0.14048087 -0.26192927  0.56332621 -0.08666882]
Epoch=499,     E=0.6489,     w=[-0.13217656 -0.20682419 -0.3017919   0.57189189 -0.08884174]
Epoch=599,     E=0.6439,     w=[-0.11912996 -0.26681228 -0.33486974  0.5828937  -0.08415037]
Epoch=699,     E=0.6393,     w=[-0.10009861 -0.32289175 -0.36378335  0.59508182 -0.07556407]
Epoch=799,     E=0.6350,     w=[-0.07763222 -0.37648376 -0.39005697  0.60773411 -0.06480392]
Epoch=899,     E=0.6308,     w=[-0.05322077 -0.42842139 -0.41458635  0.62043291 -0.0528736 ]
Epoch=999,     E=0.6267,     w=[-0.02773898 -0.47919683 -0.43790286  0.63293782 -0.04036016]
Epoch=1099,    E=0.6227,     w=[-0.00170161 -0.52910307 -0.46032488  0.64511225 -0.02760689]
Epoch=1199,    E=0.6188,     w=[ 0.02458819 -0.57831608 -0.48204556  0.65688074 -0.01481383]
Epoch=1299,    E=0.6150,     w=[ 0.05095215 -0.62694276 -0.50318412  0.66820375 -0.00209649]
Epoch=1399,    E=0.6113,     w=[ 0.07728579 -0.67504922 -0.52381597  0.67906297  0.01047946]
Epoch=1499,    E=0.6077,     w=[ 0.10352836 -0.72267736 -0.54399047  0.68945253  0.0228779 ]
Epoch=1599,    E=0.6041,     w=[ 0.12964499 -0.76985467 -0.5637414   0.69937382  0.03508007]
Epoch=1699,    E=0.6007,     w=[ 0.15561614 -0.81660007 -0.58309324  0.70883245  0.04707749]
Epoch=1799,    E=0.5973,     w=[ 0.1814314  -0.86292734 -0.60206475  0.71783641  0.05886771]
Epoch=1899,    E=0.5940,     w=[ 0.20708571 -0.90884715 -0.62067125  0.72639502  0.07045178]
Epoch=1999,    E=0.5907,     w=[ 0.2325772  -0.95436831 -0.63892587  0.73451822  0.08183275]
```



مشاهده می شود که بعد از نرمال سازی داده ها مقدار تابع اتلاف افزایش یافته و به ۰.۶ رسیده است.

```
w=[ 0.2325772,  -0.95436831, -0.63892587,  0.73451822,  0.08183275]
w = np.array(w)
w = w.reshape(-1,1)

X_test1 = np.asarray(X_test1)
X_test1 = np.hstack((np.ones((len(X_test1), 1)), X_test1))

y_test1 = np.array(y_test1)
y_test1 = y_test1.reshape(-1,1)
y_hat = np.array(y_hat)
y_hat = y_hat.reshape(-1,1)

y_hat = logistic_regression(X_test1,w)
accuracy(y_test1, y_hat)
```

```
0.6181818181818182
```

ارزیابی دقت داده های تست نیز به ۰.۶۱۸۲ کاهش یافته است.

۶.با استفاده از کدنویسی پایتون وضعیت تعادل داده‌ها در دو کلاس موجود در دیتاست را نشان دهید. آیا تعداد نمونه‌های کلاس‌ها با هم برابر است؟ عدم تعادل در دیتاست

می‌تواند منجر به چه مشکلاتی شود؟ برای حل این موضوع چه اقداماتی می‌توان انجام داد؟ پیاده‌سازی کرده و نتیجه را مقایسه و گزارش کنید

تعداد داده ها را به شکل زیر به دست می آوریم.

```
a=df[df['target']==1]
b=df[df['target']==0]
print(f'len a:{len(a)}')
print(f'len b:{len(b)}')
```

```
len a:610
len b:762
```

وجود تعداد ناصحیح نمونه‌ها در هر کلاس می‌تواند منجر به تأثیرات منفی بر عملکرد مدل‌های یادگیری ماشین شود. مدل‌هایی که با داده‌های نامتوازن آموزش داده شده‌اند، ممکن است تمایل به پیش‌بینی کلاس اکثریت داشته باشند و در تشخیص کلاس‌های کمتری دچار مشکل شوند. معیارهای ارزیابی مانند دقت (Accuracy) در مواجهه با دیتاست‌های نامتوازن ممکن است تا حدودی مطلوبیت خود را از دست بدهند. به عنوان مثال، اگر یک کلاس دارای تعداد نمونه کمی باشد و سایر کلاس‌ها دارای تعداد بیشتری نمونه داشته باشند، مدلی که تمام نمونه‌ها را به عنوان عضو اکثریت تشخیص دهد، با دقت بالایی عمل می‌کند که این مورد معمولا نمایانگر یک عملکرد نامطلوب است. وجود تعداد نامتوازن نمونه‌ها می‌تواند باعث شود که الگوهای کمتر مشاهده شوند و در نتیجه توانایی مدل در تشخیص و یادگیری این الگوها کاهش یابد. این موضوع ممکن است در مسائلی که تشخیص کلاس‌های کمتر مهم است (مانند تشخیص بیماری‌های نادر)، اثر مخربی داشته باشد. همچنین ممکن است تعمیم‌پذیری مدل را کاهش دهند. در صورتی که مدل تنها با داده‌های کلاس اکثریت آموزش ببیند، احتمال بروز overfitting به داده‌های این کلاس بیشتر است و توانایی عمومی‌سازی مدل کاهش می‌یابد. در برخی موارد، اگر داده‌ها نامتوازن باشند، اعتبارپذیری نتایج و استنتاج‌ها ممکن است کاهش یابد. این مسئله می‌تواند وجود داشتن تعداد کمی از یک کلاس را نادیده گرفته و تحلیل‌های نادرستی را به دنبال

داشته باشد. برای حل این موضوع اگر تعداد داده‌های ما زیاد بود، می‌توانیم تعداد داده‌های کلاس بیشتر را کم کنیم تا تعداد یکسانی داشته باشند که البته این روش روش خوبی نیست. روش دیگری برای درست کردن این موضوع، ایجاد دادهی فیک است. این کار را در این پروژه با میانگین گیری از دو سطر و ایجاد سطر جدید انجام دادیم. دیتا فریم جدید و خالی ای به اسم new-row ایجاد می‌کنیم و به تعداد اختلاف a و b، با استفاده از میانگین سطرهای بالایی و پایینی در a دادهی جدید ایجاد کرده و به دلیل آن که کلاس a، ۱ است target تمام داده‌های تولید شده را برابر ۱ قرار می‌دهیم و در نهایت تمامی داده‌های تولید شدهی جدید را در new-row قرار داده و آن را با a ابتدایی، مخلوط می‌کنیم.

```
new_rows = pd.DataFrame()

for i in range(len(b)-len(a)):
  v= a.iloc[i:i+2, :-1].mean()

  new_row = v.append(pd.Series({'target': 1}))
  new_rows = new_rows.append(new_row, ignore_index=True)


a.reset_index(drop=True, inplace=True)
new_rows.reset_index(drop=True, inplace=True)

updated_df = pd.concat([a.reset_index(drop=True), new_rows], ignore_index=True)

updated_df
```

```
combined_df = updated_df.append(b)
combined_df
```

```
<ipython-input-23-9b7c210afdb9>:1: FutureWarning: T
  combined_df = updated_df.append(b)
```

	variance	skewness	curtosis	entropy	target
0	-1.39710	3.31910	-1.392700	-1.99480	1.0
1	0.39012	-0.14279	-0.031994	0.35084	1.0
2	-1.66770	-7.15350	7.892900	0.96765	1.0
3	-3.84830	-12.80470	15.682400	-1.28100	1.0
4	-3.56810	-8.21300	10.083000	0.96765	1.0
...	...	...	...	...	...
757	2.66060	3.16810	1.961900	0.18662	0.0
758	3.93100	1.85410	-0.023425	1.23140	0.0
759	0.01727	8.69300	1.398900	-3.96680	0.0
760	3.24140	0.40971	1.401500	1.19520	0.0
761	2.25040	3.57570	0.352730	0.28360	0.0

1524 rows × 5 columns

```
X2 = combined_df.iloc[:,0:4]
y2 = combined_df.iloc[:,-1]
```

```
X_train2,X_test2, y_train2, y_test2 = train_test_split(X2, y2, test_size=0.2,random_state=83)
X_train2.shape, y_train2.shape, X_test2.shape, y_test2.shape
```

```
((1219, 4), (1219,), (305, 4), (305,))
```

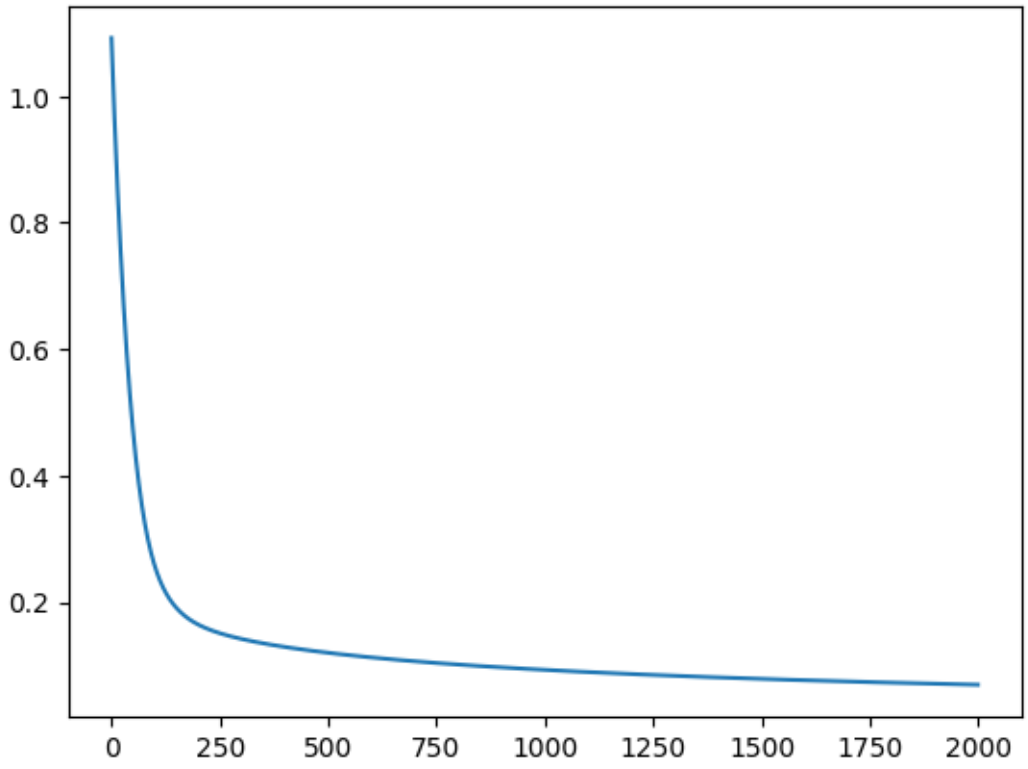
```
X_train2 = np.asarray(X_train2)
X_train2 = np.hstack((np.ones((len(X_train2), 1)), X_train2))

print(X_train2)
```

```
[[ 1.         -2.2482      3.0915     -2.3969     -2.6711    ]
 [ 1.          1.8216     -6.4748      8.0514     -0.41855   ]
 [ 1.         -2.3277      1.4381     -0.82114    -1.2862    ]
 ...
 [ 1.          4.2406     -2.4852      1.608       0.7155    ]
 [ 1.          3.9994      0.90427     1.1693      1.6892    ]
 [ 1.          0.0096613   3.5612     -4.407      -4.4103    ]]
```

```
y_train2 = np.array(y_train2)
y_train2 = y_train2.reshape(-1,1)
```

```
error_hist = []

for epoch in range(n_epochs):
  y_hat = logistic_regression(X_train2, w)

  e = bce(y_train2, y_hat)
  error_hist.append(e)

  grads = gradient(X_train2, y_train2, y_hat)

  w = gradient_descent(w, eta, grads)

  if (epoch+1) % 100 == 0:
    print(f'Epoch={epoch}, \t E={e:.4f}, \t w={w.T[0]}')

plt.plot(error_hist)
```

```
Epoch=99,     E=0.2591,     w=[ 0.2768822  -1.07096465 -0.27981436 -0.01300257 -0.10859849]
Epoch=199,    E=0.1649,     w=[ 0.32170829 -1.07653306 -0.33030794 -0.29516562 -0.12397574]
Epoch=299,    E=0.1419,     w=[ 0.37786202 -1.10128194 -0.40140991 -0.41011015 -0.13961549]
Epoch=399,    E=0.1291,     w=[ 0.43605892 -1.13314341 -0.45772881 -0.4778355  -0.16064002]
Epoch=499,    E=0.1199,     w=[ 0.49259002 -1.16553126 -0.50297609 -0.52797227 -0.18129844]
Epoch=599,    E=0.1125,     w=[ 0.54648571 -1.19654849 -0.54103218 -0.56924787 -0.1995278 ]
Epoch=699,    E=0.1064,     w=[ 0.59763323 -1.22576143 -0.5741215  -0.60508486 -0.21498035]
Epoch=799,    E=0.1012,     w=[ 0.64617818 -1.25318825 -0.60353853 -0.63714287 -0.22786704]
Epoch=899,    E=0.0968,     w=[ 0.69233273 -1.27897951 -0.63009716 -0.66635376 -0.2385384 ]
Epoch=999,    E=0.0928,     w=[ 0.73631205 -1.30331151 -0.65434742 -0.69330015 -0.24734461]
Epoch=1099,   E=0.0893,     w=[ 0.77831412 -1.32635115 -0.67668292 -0.71837712 -0.25459411]
Epoch=1199,   E=0.0862,     w=[ 0.81851494 -1.34824586 -0.697398   -0.74186965 -0.26054622]
Epoch=1299,   E=0.0834,     w=[ 0.85706906 -1.36912261 -0.71672032 -0.76399316 -0.26541516]
Epoch=1399,   E=0.0809,     w=[ 0.89411183 -1.38908971 -0.73483061 -0.78491644 -0.26937681]
Epoch=1499,   E=0.0785,     w=[ 0.92976205 -1.40823954 -0.75187539 -0.80477548 -0.2725756 ]
Epoch=1599,   E=0.0764,     w=[ 0.96412434 -1.42665111 -0.76797545 -0.82368222 -0.27513044]
Epoch=1699,   E=0.0744,     w=[ 0.99729128 -1.44439229 -0.78323176 -0.84173041 -0.27713962]
Epoch=1799,   E=0.0726,     w=[ 1.02934519 -1.46152173 -0.79772975 -0.85899971 -0.2786848 ]
Epoch=1899,   E=0.0709,     w=[ 1.06035965 -1.47809037 -0.81154238 -0.87555854 -0.27983414]
Epoch=1999,   E=0.0693,     w=[ 1.09040069 -1.49414275 -0.82473258 -0.89146631 -0.28064479]
```



```
X_test2 = np.asarray(X_test2)
X_test2 = np.hstack((np.ones((len(X_test2), 1)), X_test2))

print(X_test2)
```

```
[[ 1.      -3.5681  -8.213   10.083    0.96765]
 [ 1.       4.5707   7.2094  -3.2794  -1.4944 ]
 [ 1.       5.0429  -0.52974  0.50439  1.106  ]
 ...
 [ 1.       3.8969   7.4163  -1.8245   0.14007]
 [ 1.       4.0715   7.6398  -2.0824  -1.1698 ]
 [ 1.       2.0911   0.94358  4.5512   1.234  ]]
```

```
y_test2 = np.array(y_test2)
y_test2 = y_test2.reshape(-1,1)
y_hat = np.array(y_hat)
y_hat = y_hat.reshape(-1,1)
```

```
w=[ 1.09040069, -1.49414275, -0.82473258, -0.89146631, -0.28064479]
w = np.array(w)
w = w.reshape(-1,1)
```

```
y_hat = logistic_regression(X_test2,w)
accuracy(y_test2, y_hat)
```

```
0.980327868852459
```

مشاهده می شود دقت مدل یادگیری ماشین به طور چشمگیری بهبود پیدا میکند.

۷.فرآیند آموزش و ارزیابی مدل را با استفاده از یک طبقه‌بند آمادهٔ پایتونی انجام داده و این‌بار در این حالت چالش عدم تعادل داده‌های کلاس‌ها را حل کنید.

دقت قبل از متعادل سازی داده ها

```
from sklearn.linear_model import LogisticRegression, SGDClassifier

X_train3,X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.2)
model = LogisticRegression(solver='sag', max_iter=2000, random_state=83)
model.fit(X_train3, y_train3)
```

```
LogisticRegression(max_iter=2000, random_state=83, solver='sag')
```

```
model.score(X_test3, y_test3)
```

```
0.9890909090909091
```

با استفاده از تابع resample از کتابخانه sklearn.units داده ها را به صورت زیر متعادل کرده سپس با train داده های حاصل و به دست آوردن دقت بر روی داده های تست مشاهده می شود دقت الگوریتم افزایش یافته است.

```
from sklearn.utils import resample

class_0 = df[df['target'] == 0]
class_1 = df[df['target'] == 1]

min_class_size = min(len(class_0), len(class_1))

balanced_class_0 = resample(class_0, replace=True, n_samples=min_class_size, random_state=83)
balanced_class_1 = resample(class_1, replace=True, n_samples=min_class_size, random_state=83)

balanced_df = pd.concat([balanced_class_0, balanced_class_1])

features = balanced_df.drop('target', axis=1)
target = balanced_df['target']
```

```
X_train4, X_test4, y_train4, y_test4 = train_test_split(features, target, test_size=0.2, random_state=83)

model = LogisticRegression(solver='sag', max_iter=2000, random_state=83)
model.fit(X_train4, y_train4)
```

```
LogisticRegression
LogisticRegression(max_iter=2000, random_state=83, solver='sag')
```

```
accuracy = model.score(X_test4, y_test4)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9918032786885246
```

سوال سوم

۱.به این پیوند مراجعه کرده و یک دیتاست مربوط به «بیماری قلبی» را دریافت کرده و توضیحات مختصری در مورد هدف و ویژگی‌های آن بنویسید. فایل دانلودشدهٔ دیتاست را روی گوگل‌درایو خود قرار داده و با استفاده از دستور gdown آن را در محیط گوگل‌کولب بارگذاری کنید.

این مجموعه داده شامل شاخص های مختلف مرتبط با سلامت برای نمونه ای از افراد است. در اینجا توضیح مختصری از هر ستون آورده شده است:

Heart Diseaseor Attack: نشان می دهد که آیا فرد دچار بیماری قلبی یا حمله قلبی شده است (دودویی: ۰ = خیر، ۱ = بله).

HighBP: وضعیت فشار خون بالا (باینری: ۰ = خیر، ۱ = بله).

HighChol: وضعیت کلسترول بالا (دودویی: ۰ = خیر، ۱ = بله).

CholCheck: دفعات بررسی کلسترول (طبقه ای).

BMI: شاخص توده بدن (مستمر).

سیگاری: وضعیت سیگار کشیدن (دودویی: ۰ = خیر، ۱ = بله).

سکته مغزی: سابقه سکته مغزی (باینری: ۰ = خیر، ۱ = بله).

دیابت: وضعیت دیابت (دودویی: ۰ = خیر، ۱ = بله).

PhysActivity: سطح فعالیت بدنی (طبقه ای).

میوه ها: فراوانی مصرف میوه (قسمتی).

سبزیجات: فراوانی مصرف سبزیجات (قسمتی).

HvyAlcoholConsump: وضعیت مصرف الکل سنگین (باینری: ۰ = خیر، ۱ = بله).

AnyHealthcare: دسترسی به هر مراقبت بهداشتی (باینری: ۰ = خیر، ۱ = بله).

NoDocbcCost: بدون پزشک به دلیل هزینه (باینری: ۰ = خیر، ۱ = بله).

GenHlth: ارزیابی سلامت عمومی (طبقه ای).

MentHlth: ارزیابی سلامت روان (مقوله ای).

PhysHlth: ارزیابی سلامت جسمانی (طبقه ای).

DiffWalk: وضعیت دشواری راه رفتن (باینری: ۰ = خیر، ۱ = بله).

جنسیت: جنسیت فرد (دودویی: ۰ = زن، ۱ = مرد).

سن: سن فرد (مستمر).

تحصیلات: مقطع تحصیلی (قسمتی).

-درآمد: سطح درآمد (مقوله ای).

این مجموعه داده حاوی انواع اطلاعات مرتبط با سلامت، عوامل سبک زندگی و اطلاعات جمعیتی برای گروهی از افراد است که آن را برای بررسی همبستگی ها و عوامل خطر بالقوه بیماری قلبی و سایر شرایط سلامتی مناسب می کند.

همانند سوال قبل دیتاست مربوطه را در گوگل درایو قرار داده و با gdown آن را بارگذاری کرده و سپس فراخوانی میکنیم.

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1ze2KX99xKI8BheBenqK6bNRV1DUScesr
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.7.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from gdown) (1.16.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.11.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2023.11.17)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7.1)
Downloading...
From: https://drive.google.com/uc?id=1ze2KX99xKI8BheBenqK6bNRV1DUScesr
To: /content/heart_disease_health_indicators.csv
100% 11.8M/11.8M [00:00<00:00, 61.4MB/s]
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
df = pd.read_csv('heart_disease_health_indicators.csv')
df
```

	HeartDiseaseorAttack	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	Diabetes	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0	0	1	1	1	40	1	0	0	0	0	...	1	0	5	18	15	1	0	9	4	3
1	0	0	0	0	25	1	0	0	1	0	...	0	1	3	0	0	0	0	7	6	1
2	0	1	1	1	28	0	0	0	0	1	...	1	1	5	30	30	1	0	9	4	8
3	0	1	0	1	27	0	0	0	1	1	...	1	0	2	0	0	0	0	11	3	6
4	0	1	1	1	24	0	0	0	1	1	...	1	0	2	3	0	0	0	11	5	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
253656	0	0	0	1	25	0	0	0	1	1	...	1	0	1	0	0	0	0	4	6	8
253657	0	0	1	1	24	0	0	0	0	0	...	1	0	3	0	0	0	0	7	5	3
253658	0	0	0	0	27	0	0	0	1	0	...	1	1	2	0	0	0	0	3	6	5
253659	0	0	1	1	37	0	0	2	0	0	...	1	0	4	0	0	0	0	6	4	1
253660	0	0	1	1	34	1	0	0	0	1	...	1	0	3	0	2	1	0	7	4	3

53661 rows × 22 columns

۲.ضمن توجه به محل قرارگیری هدف و ویژگی‌ها، دیتاست را به‌صورت یک دیتافریم درآورده و با استفاده از دستورات پایتونی، ۱۰۰ نمونه‌داده مربوط به کلاس «۱» و ۱۰۰ نمونه‌داده مربوط به کلاس «۰» را در یک دیتافریم جدید قرار دهید و در قسمت‌های بعدی با این دیتافریم جدید کار کنید.

```
df = pd.DataFrame(df)
firstcolumn = df.pop('HeartDiseaseorAttack')
df.insert(len(df.columns), 'HeartDiseaseorAttack', firstcolumn)
df_out1 = df[df['HeartDiseaseorAttack'] == 1].head(100)
df_out0 = df[df['HeartDiseaseorAttack'] == 0].head(100)
combined_df = df_out1.append(df_out0)
combined_df
```

ابتدا با pop و insert ستون مورد نظر را به اخر منتقل کرده و سپس از کلاس صفر و یک ۱۰۰ نمونه بر میداریم. آن ها را در دیتا فریم جدید قرار میگذاریم.

۳.با استفاده از حداقل دو طبقه‌بند آمادهٔ پایتون و در نظر گرفتن فراپارامترهای مناسب، دو کلاس موجود در دیتاست را از هم تفکیک کنید. نتیجهٔ دقت آموزش و ارزیابی را نمایش دهید.

مانند سوال های قبل دیتا هارا طبقه بندی میکنیم.

```
X = combined_df.iloc[:,0:-1]
y = combined_df.iloc[:,-1]

x_train,x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=83)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((160, 21), (160,), (40, 21), (40,))
```

```
from sklearn.linear_model import LogisticRegression, SGDClassifier

model = LogisticRegression(solver='sag', max_iter=1000, random_state=83)
model.fit(x_train, y_train)
model.predict(x_test), y_test
```

سپس داده های test و train را ارزیابی میکنیم.

```
print(model.score(x_train, y_train))
```

```
0.7375
```

```
print(model.score(x_test, y_test))
```

```
0.575
```

همین کار را با sgdclassifier نیز انجام میدهیم نتیجه به شکل زیر قابل مشاهده است.

```
model1 = SGDClassifier(loss= 'log_loss', random_state=83)
model1.fit(x_train, y_train)
```

```
SGDClassifier
SGDClassifier(loss='log_loss', random_state=83)
```

```
print(model1.score(x_train, y_train))
```

```
0.69375
```

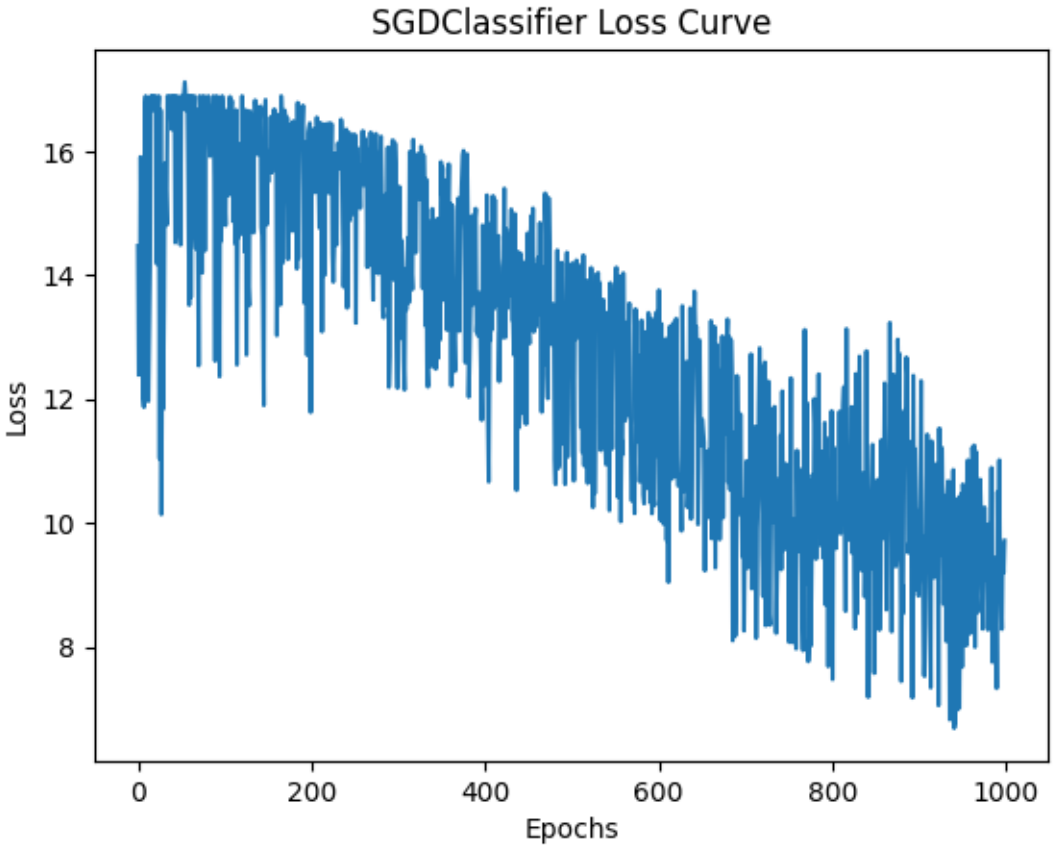
```
print(model1.score(x_test, y_test))
```

```
0.575
```

۴.در حالت استفاده از دستورات آمادهٔ سایکیت‌لرن، آیا راهی برای نمایش نمودار تابع اتلاف وجود دارد؟ پیاده‌سازی کنید.

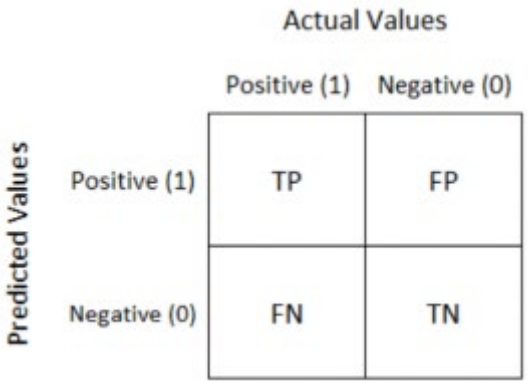
با استفاده از model.predict_proba اختلاف تارگت را با نتیجه کلاس بندی مدل یادگیری ماشین این کار را انجام می دهیم سپس تابع اتلاف را برای تعداد تکرار الگوریتم به شکل زیر ترسیم می نماییم.

```
from sklearn.metrics import log_loss
model1 = SGDClassifier(loss='log_loss', random_state=83)
model1.fit(x_train, y_train)

model1_proba = model1.predict_proba(x_train)

losses = []
for epoch in range(1, 1000):
    model1.partial_fit(x_train, y_train, classes=np.unique(y_train))
    epoch_loss = log_loss(y_train, model1.predict_proba(x_train))
    losses.append(epoch_loss)

plt.plot(losses)
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('SGDClassifier Loss Curve')
plt.show()
```



۵.یک شاخصهٔ ارزیابی (غیر از Accuracy) تعریف کنید و بررسی کنید که از چه طریقی می‌توان این شاخص جدید را در ارزیابی داده‌های تست نمایش داد. پیاده‌سازی کنید.

یکی از روش های ارزیابی مدل طبقه بندی، ماتریس درهم ریختگی یا confusion matrix می باشد. این ماتریس یک ماتریس ۲ در ۲ می باشد که به درایه های آن به نمایش تعداد حالت های مختلف (True Positive, True Negative, False Positive, False Negative)پیش بینی داده ها میپردازد



```python
from sklearn.metrics import confusion_matrix

y_pred = model.predict(x_test)
confusion_matrix = confusion_matrix(y_test, y_pred)

plt.imshow(confusion_matrix, cmap='Blues', interpolation='nearest')
plt.title('Confusion Matrix')
plt.colorbar()

classes = [0, 1]
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)

thresh = confusion_matrix.max() / 2.
for i in range(confusion_matrix.shape[0]):
    for j in range(confusion_matrix.shape[1]):
        plt.text(j, i, format(confusion_matrix[i, j], 'd'),
                 ha="center", va="center",
                 color="white" if confusion_matrix[i, j] > thresh else "black")

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.show()
```