```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression, SGDClassifier
from mlxtend.plotting import plot_decision_regions
from sklearn.utils import shuffle
```

```
https://drive.google.com/file/d/1Won6xkyYCcJLJ7eMpVt5VA_4P0tE1nb7/view?usp=sharing
```

```
  File "<ipython-input-2-103dc6819567>", line 1
    https://drive.google.com/file/d/1Won6xkyYCcJLJ7eMpVt5VA_4P0tE1nb7/view?usp=sharing
                                  ^
SyntaxError: invalid decimal literal
```

SUGGEST FIX

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1Won6xkyYCcJLJ7eMpVt5VA_4P0tE1nb7
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.7.3)
Collecting gdown
  Downloading gdown-5.0.0-py3-none-any.whl (16 kB)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.11.2)
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: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.1)
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)
Installing collected packages: gdown
  Attempting uninstall: gdown
    Found existing installation: gdown 4.7.3
    Uninstalling gdown-4.7.3:
      Successfully uninstalled gdown-4.7.3
Successfully installed gdown-5.0.0
Downloading...
From: https://drive.google.com/uc?id=1Won6xkyYCcJLJ7eMpVt5VA_4P0tE1nb7
To: /content/data_banknote_authentication.txt
100% 46.4k/46.4k [00:00<00:00, 90.7MB/s]
```

```
df = pd.read_csv('/content/data_banknote_authentication.txt')
df
```

	x1	x2	x3	x4	y
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

```
shuffled_data = shuffle(df)
shuffled_data.to_csv('created_data.csv', index=False)
print(shuffled_data)
```

```
             x1        x2        x3        x4  y
1093  0.744280  -3.77230   1.61310  1.575400  1
1030 -1.843900  -8.64750   7.67960 -0.666820  1
1206 -2.434900  -9.24970   8.99220 -0.500010  1
821  -4.017300  -8.31230  12.45470 -1.437500  1
246   1.647200   0.48213   4.74490  1.225000  0
...        ...       ...       ...       ... ..
235   2.046600   2.03000   2.17610 -0.083634  0
770   0.343400   0.12415  -0.28733  0.146540  1
680   3.446500   2.95080   1.02710  0.546100  0
726   0.040498   8.52340   1.44610 -3.930600  0
138   5.241800  10.53880  -4.11740 -4.279700  0

[1372 rows x 5 columns]
```

```
df1 = pd.read_csv('/content/created_data.csv')
df1
```

	x1	x2	x3	x4	y
0	0.744280	-3.77230	1.61310	1.575400	1
1	-1.843900	-8.64750	7.67960	-0.666820	1
2	-2.434900	-9.24970	8.99220	-0.500010	1
3	-4.017300	-8.31230	12.45470	-1.437500	1
4	1.647200	0.48213	4.74490	1.225000	0
...	...	...	...	...	...
1367	2.046600	2.03000	2.17610	-0.083634	0
1368	0.343400	0.12415	-0.28733	0.146540	1
1369	3.446500	2.95080	1.02710	0.546100	0
1370	0.040498	8.52340	1.44610	-3.930600	0
1371	5.241800	10.53880	-4.11740	-4.279700	0

1372 rows × 5 columns

Logistic Regression (from Scratch)

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

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

Binary Cross Entropy (BCE)

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

Gradient

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

Gradient Descent

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

Accuracy

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

آموزش داده های train

```
X = df1[['x1','x2','x3','x4']].values
y = df1[['y']].values
X, y
```

```
(array([[ 0.74428 , -3.7723  ,  1.6131  ,  1.5754  ],
        [-1.8439  , -8.6475  ,  7.6796  , -0.66682 ],
        [-2.4349  , -9.2497  ,  8.9922  , -0.50001 ],
        ...,
        [ 3.4465  ,  2.9508  ,  1.0271  ,  0.5461  ],
        [ 0.040498,  8.5234  ,  1.4461  , -3.9306  ],
        [ 5.2418  , 10.5388  , -4.1174  , -4.2797  ]]),
 array([[1],
        [1],
        [1],
        ...,
        [0],
        [0],
        [0]]))
```

تقسیم داده ها به دو دسته آموزش و ارزیابی

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)

x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((1097, 4), (275, 4), (1097, 1), (275, 1))
```

```
y_hat = logistic_regression(x_train, np.random.randn(4, 1))
print(y_hat.shape)
```

```
(1097, 1)
```

```
x_train = np.hstack((np.ones((len(x_train), 1)), x_train))
x_train.shape
```

```
(1097, 5)
```

```
m = 4
w = np.random.randn(m+1, 1)
print(w.shape)
```

```
(5, 1)
```

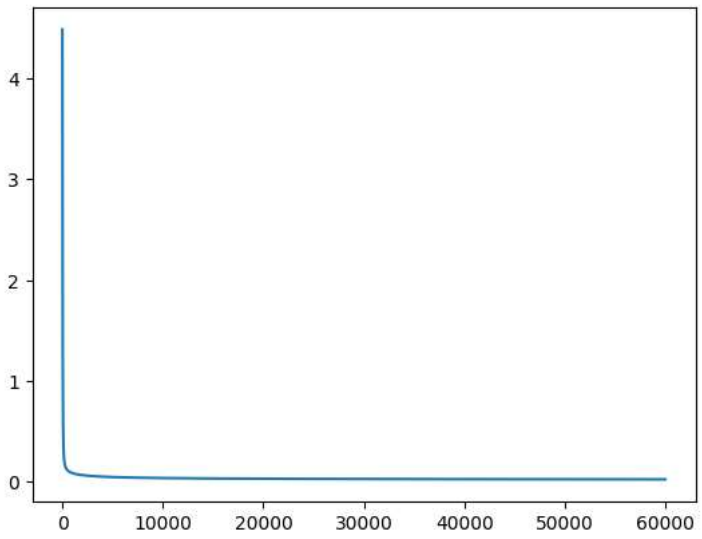
```
eta = 0.01
n_epochs = 60000 #N
```

```
error_hist = []

for epoch in range(n_epochs):
    # predictions
    y_hat = logistic_regression(x_train, w)

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

    # gradients
    grads = gradient(x_train, y_train, y_hat)

    # gradient descent
    w = gradient_descent(w, eta, grads)

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

```
Epoch=59942,     E=0.02314,      w=[ 3.560825   -3.63701299 -2.05972763 -2.50254987 -0.19671217]
Epoch=59943,     E=0.02314,      w=[ 3.56083889 -3.63703018 -2.05973635 -2.50256119 -0.19671339]
Epoch=59944,     E=0.02314,      w=[ 3.56085278 -3.63704737 -2.05974508 -2.50257252 -0.1967146 ]
Epoch=59945,     E=0.02314,      w=[ 3.56086666 -3.63706456 -2.05975381 -2.50258384 -0.19671582]
Epoch=59946,     E=0.02314,      w=[ 3.56088055 -3.63708175 -2.05976254 -2.50259517 -0.19671704]
Epoch=59947,     E=0.02314,      w=[ 3.56089443 -3.63709894 -2.05977126 -2.50260649 -0.19671826]
Epoch=59948,     E=0.02314,      w=[ 3.56090832 -3.63711613 -2.05977999 -2.50261781 -0.19671948]
Epoch=59949,     E=0.02314,      w=[ 3.56092221 -3.63713332 -2.05978872 -2.50262914 -0.1967207 ]
Epoch=59950,     E=0.02314,      w=[ 3.56093609 -3.63715051 -2.05979744 -2.50264046 -0.19672192]
Epoch=59951,     E=0.02314,      w=[ 3.56094998 -3.6371677  -2.05980617 -2.50265178 -0.19672314]
Epoch=59952,     E=0.02314,      w=[ 3.56096386 -3.63718489 -2.0598149  -2.50266311 -0.19672436]
Epoch=59953,     E=0.02314,      w=[ 3.56097775 -3.63720208 -2.05982363 -2.50267443 -0.19672558]
Epoch=59954,     E=0.02314,      w=[ 3.56099163 -3.63721927 -2.05983235 -2.50268575 -0.1967268 ]
Epoch=59955,     E=0.02314,      w=[ 3.56100551 -3.63723646 -2.05984108 -2.50269707 -0.19672802]
Epoch=59956,     E=0.02314,      w=[ 3.5610194  -3.63725365 -2.0598498  -2.5027084  -0.19672924]
Epoch=59957,     E=0.02314,      w=[ 3.56103328 -3.63727083 -2.05985853 -2.50271972 -0.19673046]
Epoch=59958,     E=0.02314,      w=[ 3.56104717 -3.63728802 -2.05986726 -2.50273104 -0.19673168]
Epoch=59959,     E=0.02314,      w=[ 3.56106105 -3.63730521 -2.05987598 -2.50274236 -0.1967329 ]
Epoch=59960,     E=0.02314,      w=[ 3.56107493 -3.6373224  -2.05988471 -2.50275368 -0.19673412]
Epoch=59961,     E=0.02314,      w=[ 3.56108882 -3.63733959 -2.05989343 -2.502765   -0.19673534]
Epoch=59962,     E=0.02314,      w=[ 3.5611027  -3.63735677 -2.05990216 -2.50277633 -0.19673656]
Epoch=59963,     E=0.02314,      w=[ 3.56111658 -3.63737396 -2.05991089 -2.50278765 -0.19673777]
Epoch=59964,     E=0.02314,      w=[ 3.56113047 -3.63739115 -2.05991961 -2.50279897 -0.19673899]
Epoch=59965,     E=0.02314,      w=[ 3.56114435 -3.63740833 -2.05992834 -2.50281029 -0.19674021]
Epoch=59966,     E=0.02314,      w=[ 3.56115823 -3.63742552 -2.05993706 -2.50282161 -0.19674143]
Epoch=59967,     E=0.02314,      w=[ 3.56117211 -3.6374427  -2.05994579 -2.50283293 -0.19674265]
Epoch=59968,     E=0.02314,      w=[ 3.561186   -3.63745989 -2.05995451 -2.50284425 -0.19674387]
Epoch=59969,     E=0.02314,      w=[ 3.56119988 -3.63747708 -2.05996324 -2.50285557 -0.19674509]
Epoch=59970,     E=0.02314,      w=[ 3.56121376 -3.63749426 -2.05997196 -2.50286689 -0.19674631]
Epoch=59971,     E=0.02314,      w=[ 3.56122764 -3.63751145 -2.05998069 -2.50287821 -0.19674753]
Epoch=59972,     E=0.02314,      w=[ 3.56124152 -3.63752863 -2.05998941 -2.50288953 -0.19674875]
Epoch=59973,     E=0.02314,      w=[ 3.5612554  -3.63754582 -2.05999814 -2.50290085 -0.19674997]
Epoch=59974,     E=0.02314,      w=[ 3.56126928 -3.637563   -2.06000686 -2.50291217 -0.19675119]
Epoch=59975,     E=0.02314,      w=[ 3.56128316 -3.63758018 -2.06001558 -2.50292349 -0.19675241]
Epoch=59976,     E=0.02314,      w=[ 3.56129704 -3.63759737 -2.06002431 -2.50293481 -0.19675363]
Epoch=59977,     E=0.02314,      w=[ 3.56131092 -3.63761455 -2.06003303 -2.50294613 -0.19675485]
Epoch=59978,     E=0.02314,      w=[ 3.5613248  -3.63763174 -2.06004176 -2.50295745 -0.19675607]
Epoch=59979,     E=0.02314,      w=[ 3.56133868 -3.63764892 -2.06005048 -2.50296877 -0.19675729]
Epoch=59980,     E=0.02314,      w=[ 3.56135256 -3.6376661  -2.0600592  -2.50298009 -0.19675851]
Epoch=59981,     E=0.02314,      w=[ 3.56136644 -3.63768329 -2.06006793 -2.50299141 -0.19675972]
Epoch=59982,     E=0.02314,      w=[ 3.56138032 -3.63770047 -2.06007665 -2.50300273 -0.19676094]
Epoch=59983,     E=0.02314,      w=[ 3.5613942  -3.63771765 -2.06008537 -2.50301404 -0.19676216]
Epoch=59984,     E=0.02314,      w=[ 3.56140808 -3.63773483 -2.0600941  -2.50302536 -0.19676338]
Epoch=59985,     E=0.02314,      w=[ 3.56142196 -3.63775202 -2.06010282 -2.50303668 -0.1967646 ]
Epoch=59986,     E=0.02314,      w=[ 3.56143584 -3.6377692  -2.06011154 -2.503048   -0.19676582]
Epoch=59987,     E=0.02314,      w=[ 3.56144972 -3.63778638 -2.06012027 -2.50305932 -0.19676704]
Epoch=59988,     E=0.02314,      w=[ 3.5614636  -3.63780356 -2.06012899 -2.50307063 -0.19676826]
Epoch=59989,     E=0.02314,      w=[ 3.56147747 -3.63782074 -2.06013771 -2.50308195 -0.19676948]
Epoch=59990,     E=0.02314,      w=[ 3.56149135 -3.63783792 -2.06014643 -2.50309327 -0.1967707 ]
Epoch=59991,     E=0.02314,      w=[ 3.56150523 -3.63785511 -2.06015516 -2.50310459 -0.19677192]
Epoch=59992,     E=0.02314,      w=[ 3.56151911 -3.63787229 -2.06016388 -2.5031159  -0.19677314]
Epoch=59993,     E=0.02314,      w=[ 3.56153298 -3.63788947 -2.0601726  -2.50312722 -0.19677436]
Epoch=59994,     E=0.02314,      w=[ 3.56154686 -3.63790665 -2.06018132 -2.50313854 -0.19677558]
Epoch=59995,     E=0.02314,      w=[ 3.56156074 -3.63792383 -2.06019005 -2.50314985 -0.1967768 ]
Epoch=59996,     E=0.02314,      w=[ 3.56157461 -3.63794101 -2.06019877 -2.50316117 -0.19677802]
Epoch=59997,     E=0.02314,      w=[ 3.56158849 -3.63795819 -2.06020749 -2.50317249 -0.19677924]
Epoch=59998,     E=0.02314,      w=[ 3.56160237 -3.63797537 -2.06021621 -2.5031838  -0.19678046]
Epoch=59999,     E=0.02314,      w=[ 3.56161624 -3.63799255 -2.06022493 -2.50319512 -0.19678167]
```

```
plt.plot(error_hist)
```

```
[<matplotlib.lines.Line2D at 0x7b4c1bae0dc0>]
```




```
y_hat = logistic_regression(x_train, w)
accuracy(y_train, y_hat)
```

```
0.9881494986326345
```

```
x_test = np.hstack((np.ones((len(x_test), 1)), x_test))
y_hat = logistic_regression(x_test, w)
accuracy(y_test, y_hat)
```

```
0.9890909090909091
```