```
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
from sklearn.preprocessing import StandardScaler
```

```
!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, 58.7MB/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
169   0.11739  6.27610 -1.54950 -2.474600  0
567   5.02140  8.07640 -3.05150 -1.715500  0
703   1.31140  4.54620  2.29350  0.225410  0
721  -0.45062 -1.36780  7.08580 -0.403030  0
1348 -2.97860  2.34450  0.52667 -0.401730  1
...       ...      ...      ...       ... ..
875  -1.86290 -0.84841  2.53770  0.097399  1
```

```
1100  1.43780  0.66837 -2.02670  1.027100  1
39    3.48050  9.70080 -3.75410 -3.437900  0
639   3.88460 -3.03360  2.53340  0.202140  0
43    0.96441  5.83950  2.32350  0.066365  0

[1372 rows x 5 columns]
```

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

	x1	x2	x3	x4	y
0	0.11739	6.27610	-1.54950	-2.474600	0
1	5.02140	8.07640	-3.05150	-1.715500	0
2	1.31140	4.54620	2.29350	0.225410	0
3	-0.45062	-1.36780	7.08580	-0.403030	0
4	-2.97860	2.34450	0.52667	-0.401730	1
...	...	...	...	...	...
1367	-1.86290	-0.84841	2.53770	0.097399	1
1368	1.43780	0.66837	-2.02670	1.027100	1
1369	3.48050	9.70080	-3.75410	-3.437900	0
1370	3.88460	-3.03360	2.53340	0.202140	0
1371	0.96441	5.83950	2.32350	0.066365	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
```

validatin آموزش داده های نرمالایز شده

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

```
(array([[ 0.11739 ,  6.2761  , -1.5495  , -2.4746  ],
        [ 5.0214  ,  8.0764  , -3.0515  , -1.7155  ],
        [ 1.3114  ,  4.5462  ,  2.2935  ,  0.22541 ],
        ...,
        [ 3.4805  ,  9.7008  , -3.7541  , -3.4379  ],
        [ 3.8846  , -3.0336  ,  2.5334  ,  0.20214 ],
        [ 0.96441 ,  5.8395  ,  2.3235  ,  0.066365]]),
 array([[0],
        [0],
        [0],
        ...,
        [0],
        [0],
        [0]]))
```

```
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))
```

```
# Create separate StandardScaler instances for each feature
scaler_x1 = StandardScaler()
scaler_x2 = StandardScaler()
scaler_x3 = StandardScaler()
scaler_x4 = StandardScaler()

# Fit and transform each feature separately
x_train_normalized_x1 = scaler_x1.fit_transform(x_train[:, [0]])
x_train_normalized_x2 = scaler_x2.fit_transform(x_train[:, [1]])
x_train_normalized_x3 = scaler_x3.fit_transform(x_train[:, [2]])
x_train_normalized_x4 = scaler_x4.fit_transform(x_train[:, [3]])

# Transform the corresponding test data using the same scalers
x_test_normalized_x1 = scaler_x1.fit_transform(x_test[:, [0]])
x_test_normalized_x2 = scaler_x1.fit_transform(x_test[:, [1]])
x_test_normalized_x3 = scaler_x1.fit_transform(x_test[:, [2]])
x_test_normalized_x4 = scaler_x1.fit_transform(x_test[:, [3]])

# Concatenate the normalized features back into a 2D array
x_train_normalized = np.hstack((x_train_normalized_x1,x_train_normalized_x2,x_train_normalized_x3,x_train_normalized_x4))
x_test_normalized = np.hstack((x_test_normalized_x1,x_test_normalized_x2,x_test_normalized_x3,x_test_normalized_x4))

# Check the shapes of the normalized data
x_train_normalized.shape, x_test_normalized.shape, y_train.shape, y_test.shape
```

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

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

```
(275, 1)
```

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

```
(275, 5)
```

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

```
(5, 1)
```

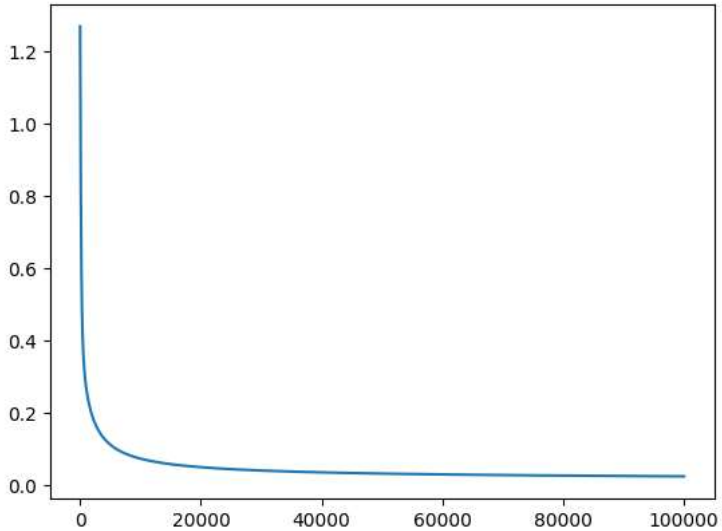
```
eta = 0.01
n_epochs = 100000 #N
```

```
error_hist = []

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

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

    # gradients
    grads = gradient(x_test_normalized, y_test, 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=99942,     E=0.02487,      w=[-1.43956744 -5.95488235 -6.88281263 -6.1883103   0.18117732]
Epoch=99943,     E=0.02487,      w=[-1.4395747  -5.95489662 -6.8828312  -6.18832689  0.18117679]
Epoch=99944,     E=0.02487,      w=[-1.43958197 -5.95491089 -6.88284978 -6.18834348  0.18117626]
Epoch=99945,     E=0.02487,      w=[-1.43958924 -5.95492517 -6.88286835 -6.18836007  0.18117573]
Epoch=99946,     E=0.02487,      w=[-1.43959651 -5.95493944 -6.88288692 -6.18837666  0.1811752 ]
Epoch=99947,     E=0.02487,      w=[-1.43960377 -5.95495371 -6.88290549 -6.18839324  0.18117467]
Epoch=99948,     E=0.02487,      w=[-1.43961104 -5.95496798 -6.88292407 -6.18840983  0.18117414]
Epoch=99949,     E=0.02487,      w=[-1.43961831 -5.95498225 -6.88294264 -6.18842642  0.1811736 ]
Epoch=99950,     E=0.02487,      w=[-1.43962558 -5.95499652 -6.88296121 -6.18844301  0.18117307]
Epoch=99951,     E=0.02487,      w=[-1.43963284 -5.95501079 -6.88297978 -6.1884596   0.18117254]
Epoch=99952,     E=0.02487,      w=[-1.43964011 -5.95502506 -6.88299835 -6.18847619  0.18117201]
Epoch=99953,     E=0.02487,      w=[-1.43964738 -5.95503934 -6.88301693 -6.18849277  0.18117148]
Epoch=99954,     E=0.02487,      w=[-1.43965464 -5.95505361 -6.8830355  -6.18850936  0.18117095]
Epoch=99955,     E=0.02487,      w=[-1.43966191 -5.95506788 -6.88305407 -6.18852595  0.18117042]
Epoch=99956,     E=0.02487,      w=[-1.43966918 -5.95508215 -6.88307264 -6.18854254  0.18116989]
Epoch=99957,     E=0.02487,      w=[-1.43967645 -5.95509642 -6.88309121 -6.18855912  0.18116936]
Epoch=99958,     E=0.02487,      w=[-1.43968371 -5.95511069 -6.88310978 -6.18857571  0.18116883]
Epoch=99959,     E=0.02487,      w=[-1.43969098 -5.95512496 -6.88312835 -6.1885923   0.1811683 ]
Epoch=99960,     E=0.02487,      w=[-1.43969825 -5.95513923 -6.88314692 -6.18860888  0.18116777]
Epoch=99961,     E=0.02487,      w=[-1.43970551 -5.9551535  -6.88316549 -6.18862547  0.18116723]
Epoch=99962,     E=0.02487,      w=[-1.43971278 -5.95516777 -6.88318406 -6.18864206  0.1811667 ]
Epoch=99963,     E=0.02487,      w=[-1.43972005 -5.95518204 -6.88320263 -6.18865864  0.18116617]
Epoch=99964,     E=0.02487,      w=[-1.43972731 -5.95519631 -6.8832212  -6.18867523  0.18116564]
Epoch=99965,     E=0.02487,      w=[-1.43973458 -5.95521058 -6.88323977 -6.18869182  0.18116511]
Epoch=99966,     E=0.02487,      w=[-1.43974185 -5.95522485 -6.88325834 -6.1887084   0.18116458]
Epoch=99967,     E=0.02487,      w=[-1.43974911 -5.95523912 -6.88327691 -6.18872499  0.18116405]
Epoch=99968,     E=0.02487,      w=[-1.43975638 -5.95525339 -6.88329548 -6.18874157  0.18116352]
Epoch=99969,     E=0.02487,      w=[-1.43976365 -5.95526766 -6.88331405 -6.18875816  0.18116299]
Epoch=99970,     E=0.02487,      w=[-1.43977091 -5.95528192 -6.88333262 -6.18877474  0.18116246]
Epoch=99971,     E=0.02487,      w=[-1.43977818 -5.95529619 -6.88335119 -6.18879133  0.18116193]
Epoch=99972,     E=0.02487,      w=[-1.43978544 -5.95531046 -6.88336976 -6.18880791  0.1811614 ]
Epoch=99973,     E=0.02487,      w=[-1.43979271 -5.95532473 -6.88338833 -6.1888245   0.18116087]
Epoch=99974,     E=0.02487,      w=[-1.43979998 -5.955339   -6.88340689 -6.18884108  0.18116033]
Epoch=99975,     E=0.02487,      w=[-1.43980724 -5.95535327 -6.88342546 -6.18885767  0.1811598 ]
Epoch=99976,     E=0.02487,      w=[-1.43981451 -5.95536754 -6.88344403 -6.18887425  0.18115927]
Epoch=99977,     E=0.02487,      w=[-1.43982177 -5.95538181 -6.8834626  -6.18889084  0.18115874]
Epoch=99978,     E=0.02487,      w=[-1.43982904 -5.95539607 -6.88348117 -6.18890742  0.18115821]
Epoch=99979,     E=0.02487,      w=[-1.43983631 -5.95541034 -6.88349973 -6.18892401  0.18115768]
Epoch=99980,     E=0.02487,      w=[-1.43984357 -5.95542461 -6.8835183  -6.18894059  0.18115715]
Epoch=99981,     E=0.02487,      w=[-1.43985084 -5.95543888 -6.88353687 -6.18895717  0.18115662]
Epoch=99982,     E=0.02487,      w=[-1.4398581  -5.95545315 -6.88355543 -6.18897376  0.18115609]
Epoch=99983,     E=0.02487,      w=[-1.43986537 -5.95546741 -6.883574   -6.18899034  0.18115556]
Epoch=99984,     E=0.02487,      w=[-1.43987264 -5.95548168 -6.88359257 -6.18900693  0.18115503]
Epoch=99985,     E=0.02487,      w=[-1.4398799  -5.95549595 -6.88361113 -6.18902351  0.1811545 ]
Epoch=99986,     E=0.02487,      w=[-1.43988717 -5.95551022 -6.8836297  -6.18904009  0.18115397]
Epoch=99987,     E=0.02487,      w=[-1.43989443 -5.95552448 -6.88364827 -6.18905668  0.18115343]
Epoch=99988,     E=0.02487,      w=[-1.4399017  -5.95553875 -6.88366683 -6.18907326  0.1811529 ]
Epoch=99989,     E=0.02486,      w=[-1.43990896 -5.95555302 -6.8836854  -6.18908984  0.18115237]
Epoch=99990,     E=0.02486,      w=[-1.43991623 -5.95556729 -6.88370396 -6.18910642  0.18115184]
Epoch=99991,     E=0.02486,      w=[-1.43992349 -5.95558155 -6.88372253 -6.18912301  0.18115131]
Epoch=99992,     E=0.02486,      w=[-1.43993076 -5.95559582 -6.8837411  -6.18913959  0.18115078]
Epoch=99993,     E=0.02486,      w=[-1.43993802 -5.95561009 -6.88375966 -6.18915617  0.18115025]
Epoch=99994,     E=0.02486,      w=[-1.43994529 -5.95562435 -6.88377823 -6.18917275  0.18114972]
Epoch=99995,     E=0.02486,      w=[-1.43995255 -5.95563862 -6.88379679 -6.18918934  0.18114919]
Epoch=99996,     E=0.02486,      w=[-1.43995982 -5.95565289 -6.88381536 -6.18920592  0.18114866]
Epoch=99997,     E=0.02486,      w=[-1.43996708 -5.95566715 -6.88383392 -6.1892225   0.18114813]
Epoch=99998,     E=0.02486,      w=[-1.43997435 -5.95568142 -6.88385248 -6.18923908  0.1811476 ]
Epoch=99999,     E=0.02486,      w=[-1.43998161 -5.95569569 -6.88387105 -6.18925566  0.18114707]
```

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

```
[<matplotlib.lines.Line2D at 0x79723031b0a0>]
```




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

```
0.9927272727272727
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.