Perceptron + letters

1. Przygotować kod klasy **SLP** (Single Layer Perceptron). Klasa powinna posiadać metody:

def __init__(self, eta=0.05, n_iter=10, random_state=1):
def fit(self, X, y):
def predict(self, X):
def misclassified(self, X):
def show(self, X):

Obiekt tej klasy w momencie wywołania metody fit automatycznie dopasowuje architekturę sieci do zbioru X (tworzy obiekty wykonanej w trakcie zajęć klasy Perceptron w ramach pojedynczej warstwy) oraz przeprowadza uczenie.

2. Utworzyć obiek klasy **SLP** o nazwie **net**. Pozostawić domyślne wartości parametrów **eta** i **n_iter**.

net = SLP()

3. Wczytać dane z pliku letters.data i wybrać z nich **tylko te przypadki (i powiązane z nimi odpowiedzi) wskazane w indywidualnym zestawie danych *set*.**

4. Podzielić indywidualne dane na zbiory X oraz y. Pełny zbiór letters.data składa się z przykładów podanych w wierszach. Pierwsze 35 wartości każdego wiersza stanowią wartości 35 pikseli kodujących literę. Litery zakodowano na siatce o szerokości 5 pikseli i wysokości 7 pikseli. Kolejne 26 wartości koduje wektor odpowiedzi oczekiwanej na wyjściu sieci.
Poniżej przykład zbiorów X i y dla zbioru indywidualnego set=[10,11,12,13,14,15,16,17,18,19]:

Zawartość X:

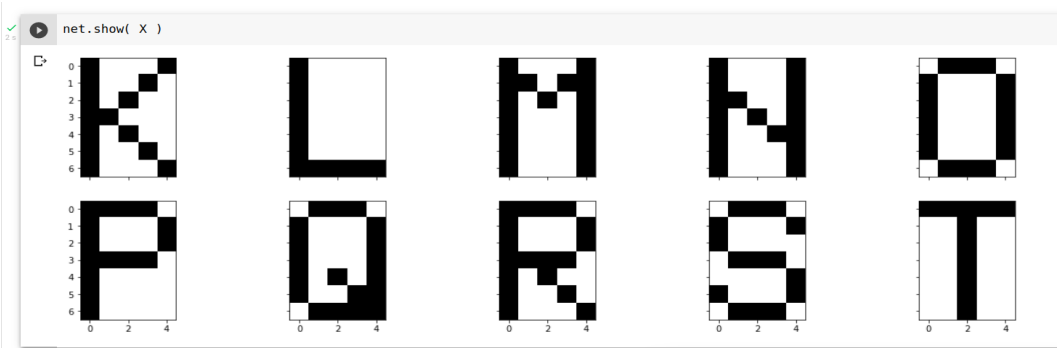
```
X
```

```
array([[ 1, -1, -1, -1,  1,  1, -1, -1,  1, -1,  1, -1,  1, -1, -1,  1,
         1, -1, -1, -1,  1, -1,  1, -1, -1,  1, -1, -1,  1, -1,  1, -1,
        -1, -1,  1],
       [ 1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1,
        -1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1,  1,
         1,  1,  1],
       [ 1, -1, -1, -1,  1,  1,  1, -1,  1,  1,  1, -1,  1, -1,  1,  1,
        -1, -1, -1,  1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1, -1,
        -1, -1,  1],
       [ 1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,  1, -1, -1,  1,  1,
        -1,  1, -1,  1,  1, -1, -1,  1,  1,  1, -1, -1, -1,  1,  1, -1,
        -1, -1,  1],
       [-1,  1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,
        -1, -1, -1,  1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1, -1,  1,
         1,  1, -1],
       [ 1,  1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,
         1,  1,  1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1,
        -1, -1, -1],
       [-1,  1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,
        -1, -1, -1,  1,  1, -1,  1, -1,  1,  1, -1, -1,  1,  1, -1,  1,
         1,  1,  1],
       [ 1,  1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1, -1, -1,  1,  1,
         1,  1,  1, -1,  1, -1,  1, -1, -1,  1, -1, -1,  1, -1,  1, -1,
        -1, -1,  1],
       [-1,  1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1, -1, -1, -1, -1,
         1,  1,  1, -1, -1, -1, -1, -1,  1,  1, -1, -1, -1,  1, -1,  1,
         1,  1, -1],
       [ 1,  1,  1,  1,  1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1,
        -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1, -1, -1,
         1, -1, -1]])
```

Zawartość y:

```
y

array([[ 1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
       [-1,  1, -1, -1, -1, -1, -1, -1, -1, -1],
       [-1, -1,  1, -1, -1, -1, -1, -1, -1, -1],
       [-1, -1, -1,  1, -1, -1, -1, -1, -1, -1],
       [-1, -1, -1, -1,  1, -1, -1, -1, -1, -1],
       [-1, -1, -1, -1, -1,  1, -1, -1, -1, -1],
       [-1, -1, -1, -1, -1, -1,  1, -1, -1, -1],
       [-1, -1, -1, -1, -1, -1, -1,  1, -1, -1],
       [-1, -1, -1, -1, -1, -1, -1, -1,  1, -1],
       [-1, -1, -1, -1, -1, -1, -1, -1, -1,  1]])
```

5. Wyświetlić graficznie dane z indywidualnego zbioru X korzystając z zaimplementowanej metody show.
Poniżej przykład dla zbioru indywidualnego set=[10,11,12,13,14,15,16,17,18,19]:



6. Przeprowadzić uczenie modelu wywołując ***fit(X,y)***.

net.fit(X,y)

7. Wyświetlić wynik predict na zbiorze uczącym.
Poniżej przykład dla zbioru indywidualnego set=[10,11,12,13,14,15,16,17,18,19]:

```
net.predict(X)

[array([ 1, -1, -1, -1, -1, -1, -1, -1, -1, -1]),
 array([-1,  1, -1, -1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1,  1, -1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1,  1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1, -1,  1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1,  1, -1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1,  1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1,  1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1, -1,  1])]
```

8. Wyświetlić zawartość errors_.
Poniżej przykład zbiorów X i y dla zbioru indywidualnego set=[10,11,12,13,14,15,16,17,18,19]:

```
net.errors_
```

```
array([35., 13.,  9.,  4.,  1.,  0.,  0.,  0.,  0.,  0.])
```

9. Wyświetlić wynik misclassified na zbiorze uczącym.
Poniżej przykład zbiorów X i y dla zbioru indywidualnego set=[10,11,12,13,14,15,16,17,18,19,20]:

```
net.misclassified(X,y)
```

```
0
```

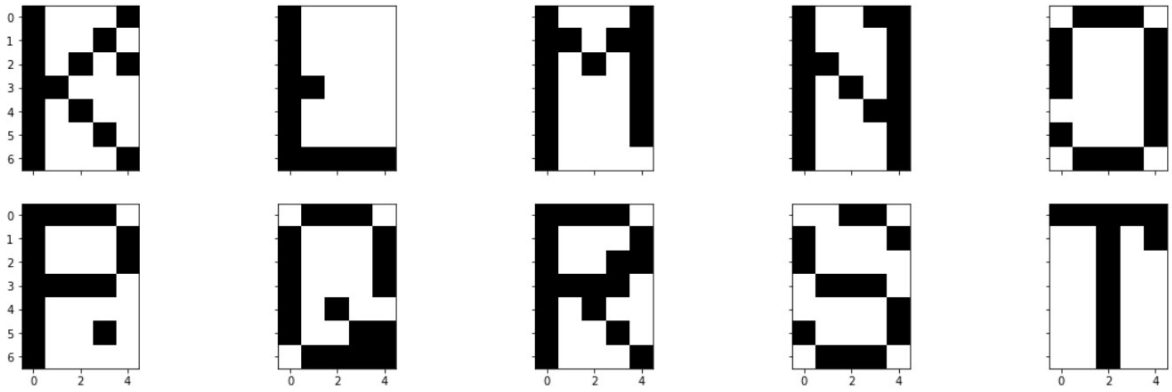
10. Korzystając z funkcji damage uszkodzić kolejno 5%, 15%, 40% każdego przypadku z indywidualnego zbioru X.

```
def damage(X,percent,seed=1):
    rgen = np.random.RandomState(seed)
    result = np.array( X )
    count = int( X.shape[1]*percent/100 )

    for indeks_example in range( len(X) ):
        order = np.sort( rgen.choice( X.shape[1], count , replace=False) )
        for indeks_pixel in order:
            result[indeks_example][indeks_pixel]*=-1

    return result
```

```
[603] damaged5 = damage(X,5)
      damaged15 = damage(X,15)
      damaged40 = damage(X,40)
```

11. Wyświetlić graficznie dane z każdego uszkodzonego zbioru oraz wyniki predict i misclassified.

```
net.show( damaged5 )
```



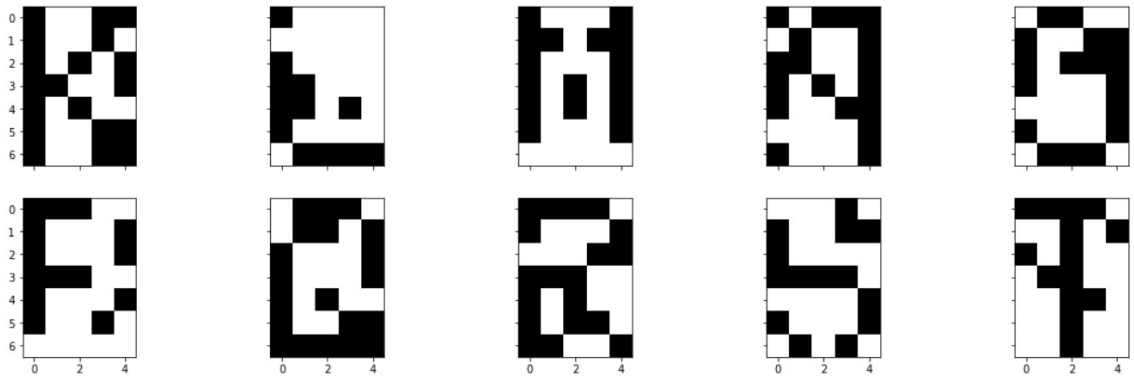
```
[611] net.predict( damaged5 )
```

```
[array([ 1, -1, -1, -1, -1, -1, -1, -1, -1, -1]),
 array([-1,  1, -1, -1, -1, -1, -1, -1,  1, -1]),
 array([-1, -1,  1, -1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1,  1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1, -1,  1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1,  1, -1,  1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1,  1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1,  1, -1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1, -1,  1])]
```

```
net.misclassified( damaged5 ,y)
```

```
3
```

```
[605] net.show( damaged15 )
```



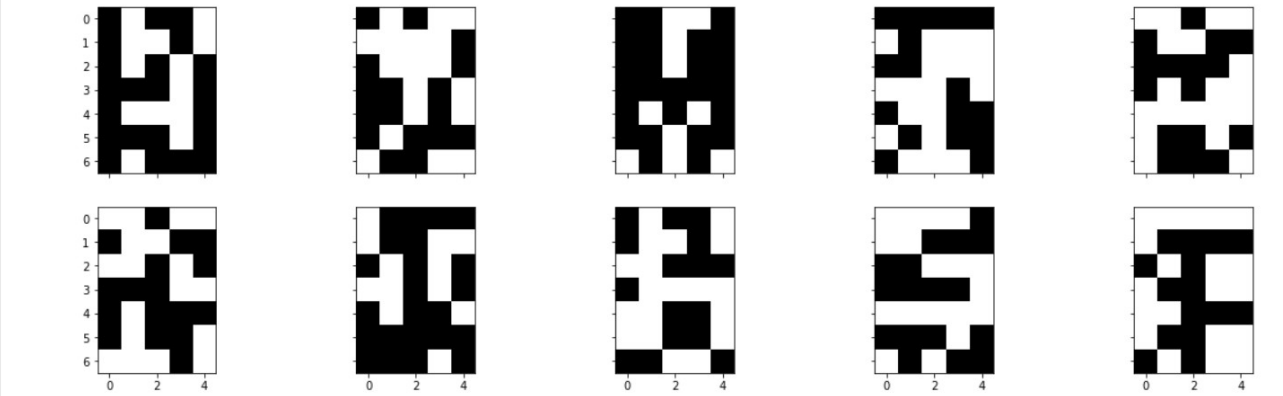
```
net.predict( damaged15 )
```

```
[array([ 1, -1, -1, -1, -1, -1, -1, -1, -1, -1]),
 array([-1,  1, -1, -1, -1, -1, -1, -1,  1, -1]),
 array([-1, -1,  1,  1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1,  1, -1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1,  1,  1, -1, -1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1,  1, -1, -1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1,  1, -1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1,  1, -1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1, -1,  1])]
```

```
[615] net.misclassified( damaged15 ,y)
```

```
4
```

```
[606] net.show( damaged40 )
```



```
[613] net.predict( damaged40 )
```

```
[array([-1, -1,  1, -1, -1,  1, -1,  1,  1,  1]),
 array([-1,  1, -1, -1,  1, -1, -1,  1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1, -1, -1, -1,  1]),
 array([-1, -1, -1,  1, -1, -1, -1, -1,  1,  1]),
 array([-1, -1, -1, -1,  1, -1, -1, -1, -1, -1]),
 array([-1,  1, -1, -1,  1, -1, -1, -1, -1,  1]),
 array([-1, -1, -1, -1, -1, -1,  1,  1, -1, -1]),
 array([-1,  1,  1,  1, -1, -1, -1,  1, -1, -1]),
 array([-1,  1,  1, -1,  1, -1, -1, -1,  1, -1]),
 array([-1, -1, -1, -1, -1, -1,  1, -1, -1,  1])]
```

```
net.misclassified( damaged40 ,y)
```

```
25
```

12. Kod wynikowy zapisać w formie pliku ***perce_own.ipynb*** (zeszytu Colab).
13. Dokumentację przebiegu zadania i otrzymanych wyników zapisać w formie pliku ***perce_own.pdf***. Elementy konieczne dokumentacji to te, których dotyczą zrzuty ekranu. Pamiętajmy o odpowiednich tytułach/podpisach.
14. Ponowić operacje korzystając z implementacji perceptronu biblioteki ***scikit-learn***.
15. Kod wynikowy zapisać w formie pliku ***perce_scikit.ipynb*** (zeszytu Colab).
16. Dokumentację przebiegu zadania i otrzymanych wyników zapisać w formie pliku ***perce_scikit.pdf***.

17. Wskazane powyżej pliki zapakować do formy archiwum zip (bez hasła) i przesłać na skrzynkę na kampusie. Nazwa archiwum składać się musi z nazwiska oraz imienia autora (bez polskich znaków) oddzielonych podkreślnikiem.

Przykład:
olszewski_pawel.zip