Wykorzystywane w badań oprogramowanie zostało zaimplementowane w języki **Python 3.6** z wykorzystaniem biblioteki **NumPy**, wykorzystaniej do przeprowadzenia operacji na macierzach i wektorach. Dodatkowo wykorzystany został moduł **csv** w celu zautomatyzowania przeprowadzanych testów. W procesie tworzenia wykorzystane zostało środowisko **Visual Studio Code**.

Implementacje obu rodzajów perceptronów zostały zawarte w dwóch plikach *simpleperceptron.py* oraz *adaline.py*. Cała mechanika testowania została zaimplementowana natomiast w pliku *testing.py*. W wyniku testów wygenerowany został plik *results.csv* zawierający wyniki przeprowadzonych badań.

Konstruktory klas reprezentujących poszczególne rodzaje perceptronów przedstawiają się następująco.

```
SimplePerceptron(numOfInputs, (minWeight, maxWeight),
                 activation, activationThreshold)


Adaline(numofinputs, (minweight, maxweight), activationthreshold)
```

Wyjaśnienia poszczególnych parametrów przedstawiają się następująco:

- numOfInputs - liczba danych treningowych.
- minWeight - minimalna wartość wag początkowych.
- maxWeight - maksymalna wartość wag początkowych.
- activation - funkcja aktywująca (bipolarna lub unipolarna).
- activationThreshold - wartość graniczna funkcji aktywującej.

Funkcje rozpoczynające proces treningu natomiast prezentują się następująco:

```
SimplePerceptron.train(trainingData, labels, iterations, learning_rate)


Adaline.train(trainingData, labels, iterations, errorThreshold, learningRate)
```

Wyjaśnienia poszczególnych parametrów przedstawiają się następująco:

- trainingData - wektor danych wejściowych.
- labels - wektor danych oczekiwanych.
- iterations - maksymalna, nieprzekraczalna liczba epok.
- errorThreshold - minimalna wartość błędu średniokwadratowego (dla perceptronu Adaline).
- learningRate - współczynnik uczenia.