IWiSUM ćwiczenia 1.

Karol Capała, Mateusz Jarosz, Bartłomiej Śnieżyński

8-11.10.2024

Zajęcia mają cel przypomnieć podstawowe informacje i narzędzia z uczenia maszynowego

1. Pobrać i wczytać do notebooka WDBC dataset
2. Używając funkcji KFold z SciKit przygotować dane do sprawdzianu krzyżowego
3. Wytrenować przy użyciu sprawdzianu krzyżowego drzewo decyzyjne, las losowy, regresje logistyczna, support vector machine i mulit-layer perceptron, a następnie przeanalizuj i porównaj wyniki. Na jakie problemy trzeba zwrócić uwagę podczas treningu?
4. Pobrać i wczytać do notebooka Ionosphere dataset i powtórzyć punkty 2-3
5. Pobrać ze strony kursu i wczytać do notebooka space.csv
6. Bez podziału na zbiór treningowy i testowy wytrenować modele w oparciuo metody z 3.
7. Zaimplementować (i jeśli potrzeba zdebugować) funkcje do rysowania z Listing 1.
8. Wyrysować i porównać domeny otrzymane przy pomocy poszczególnych metod.
9. Dla losowej dystrybucji danych 2D wygenerować klasteryzację danych używając metody K-mean i zwizualizować. Kod można napisać samemu lub skorzystać z istniejących narzędzi (np. link)

Listing 1: Rysowanie domen

```
from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('o', 's', '^', 'v', '<')
    colors = ('blue','red', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
    x2_min, x2_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    lab = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    lab = lab.reshape(xx1.shape)
    plt.contourf(xx1, xx2, lab, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class examples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    c=colors[idx],
                    marker=markers[idx],
                    label=f'Class {cl}',
                    edgecolor='black')
```