Université de Lille

Introduction to scikit-learn and classification

Machine Learning
N. Veerapen
SIAD M2 DS 2024–2025

Let us first create a synthetic dataset that will allow us to test ML algorithms under controlled conditions and practice using some basic numpy and matplotlib/seaborn operations.

Question 1 – Generating synthetic data – twoClasses

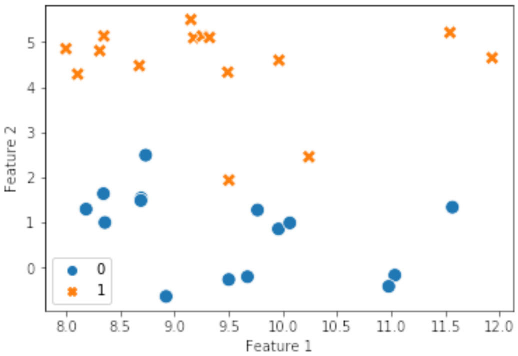
Create the `twoClasses()` function that returns a custom dataset for classification.

1. Start with 30 points (samples) grouped into 2 *blobs* (import `make_blobs` found in `sklearn.datasets`, check the documentation) and a random state set to 4. Edit and insert the following code into your function and display the content of the objects obtained when the function is executed.

```
X, y = make_blobs(centers, random_state, n_samples)
```

2. Visualize the scatter plot corresponding to the data generated by the function. The values in X are the points, while the values in y are the color and shape of the points.

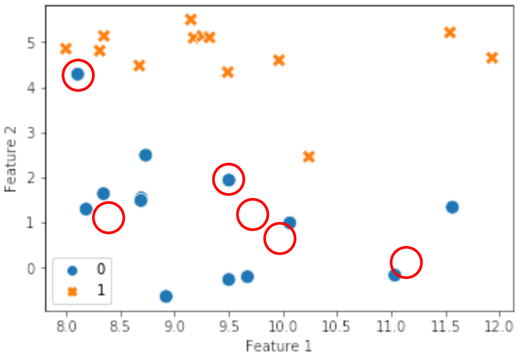
```
import seaborn as sns
g = sns.scatterplot(x, y, hue, style) # hue(color), style(shape)
```



3. Assign *class 0* to values of y at positions 7 and 27 in the array.
4. Delete the observations at positions 0, 1, 5, 26 in X and y.

```
import numpy as np
new_array = np.delete(array, indices, axis) # axis=0 (row)
```

Visualize the dataset following these modifications (the differences with the original are circled in red in the plot below).



Question 2 – k-NN – Generalization

Let us use the k-NN algorithm on the breast cancer detection dataset already found in scikit-learn.

1. Load the dataset as shown below.

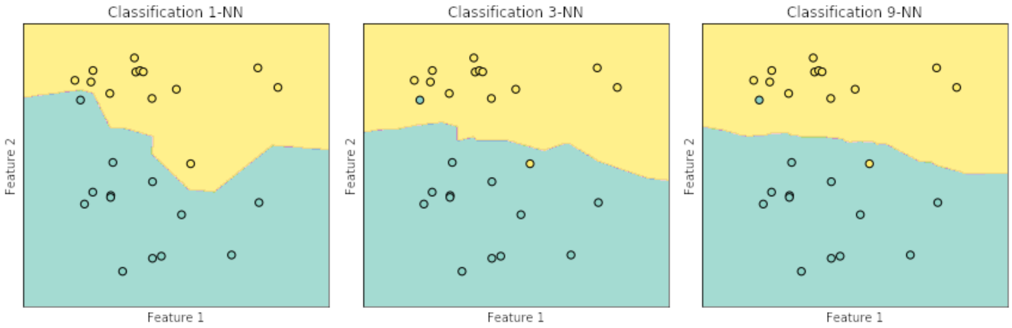
```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

2. Inspect the data.
3. Import `train_test_split` from `sklearn.model_selection` and split the data into training and test sets a random state equal to 66 (`random_state = 66`). What is the random state used for? What are the `shuffle` and `stratify` parameters for?
4. Import `KNeighborsClassifier` from `sklearn.neighbors` and create a classifier with k=1.
5. Learn the model (use the `fit()` method) on the training set.
6. Display the values of the predictions on the training and test sets and compute the score (accuracy).
7. Record the score on the training and test sets for classifiers with parameter values k=1 to 21.
8. Plot a graph comparing the two curves corresponding to the numbers you just recorded. What can you observe?
9. Ideally, is there something that we should have done to our dataset before creating the model?

Question 3 – k-NN – Decision boundaries

Let us use the k-NN algorithm on the twoClasses dataset in order to visualize *decision boundaries*. Open the Lab1-Question3 notebook that contains code for this question.

1. Import `train_test_split` from `sklearn.model_selection` and apply it to twoClasses with a `random_state` of 0. Check the behavior of `train_test_split` when using the same `random_state` and when changing the value of this parameter.
2. Import `KNeighborsClassifier` from `sklearn.neighbors` and create a classifier with k=3.
3. Learn the model (use the `fit()` method) on the training set.
4. Display the values of the predictions on the test set and compute the score.
5. Complete the existing code in the notebook (look for the TODOs in the code) in order to visualize the decision boundary:



6. Edit the code to visualize what happens with 5 different values of k. What do you observe as the value of k increases?
7. In `DecisionBoundaryDisplay.from_estimator()`, change the `response_method` to `'predict_proba'`. What is the meaning of the different colors?

8. Switch back to `response_method='predict'` but add `weights='distance'` to your k-NN classifier. What do you observe and why?

Question 4 – Linear Models – Logistic Regression

Let us use logistic regression on the breast cancer dataset. Again, use stratified sampling.

1. Apply logistic regression sklearn.linear_model.LogisticRegression().
2. Compute scores on the training and test sets.
3. By default, scikit-learn applies L2 regularization to logistic regression. The default value for the regularization parameter is C=1. Repeat the last 2 steps for C=0,001 and C=100. What do you observe? Why?
4. Let us observe the coefficients of the 3 models with the following code.

```
import matplotlib.pyplot as plt
plt.plot(<model1>.coef_.T, 'o', label="C=1")
plt.plot(<model2>.coef_.T, '^', label="C=0.001")
plt.plot(<model3>.coef_.T, 'v', label="C=100")
plt.xticks(range(cancer.data.shape[1]), cancer.feature_names,
           rotation=90)
plt.hlines(0,0, cancer.data.shape[1])
plt.ylim(-5,5)
plt.xlabel("Feature")
plt.ylabel("Coefficients")
plt.legend()
plt.show()
```

5. Try again with L1 regularization, `sklearn.linear_model.LogisticRegression(penalty='l1')`, that limits the model to using fewer features. Is this what we can observe by examining the coefficients?
6. Ideally, is there something that we should have done to our dataset before creating the model?