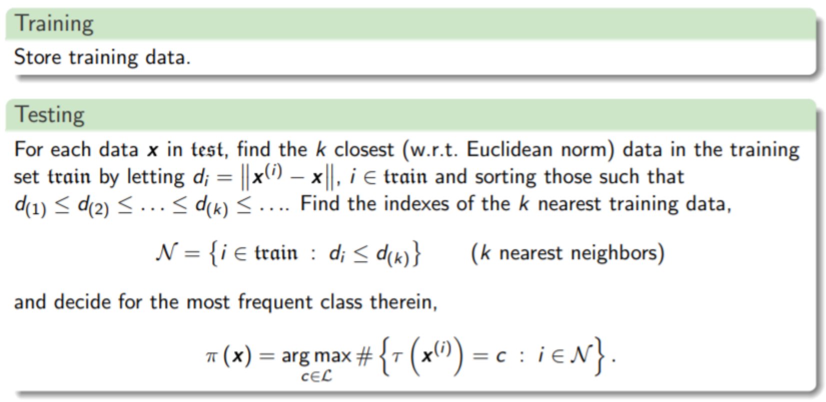
Data Science | Lab 3: Performance Evaluation

Learning Goals

- parameterize a *k*-Nearest Neighbors classifier
- understand performance measures other than accuracy:
    - precision, recall, f1-score
- read and interpret the confusion matrix
- read and interpret the classification report

k-Nearest Neighbors Classifier



Training

Store training data.

Testing

For each data $\boldsymbol{x}$ in $\mathfrak{test}$, find the k closest (w.r.t. Euclidean norm) data in the training set $\mathfrak{train}$ by letting $d_i = \left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}\right\|$, $i \in \mathfrak{train}$ and sorting those such that $d_{(1)} \le d_{(2)} \le \dots \le d_{(k)} \le \dots$. Find the indexes of the k nearest training data,

$$\mathcal{N} = \left\{ i \in \mathfrak{train} \ : \ d_i \le d_{(k)} \right\} \qquad (k \text{ nearest neighbors})$$

and decide for the most frequent class therein,

$$\pi(\boldsymbol{x}) = \underset{c \in \mathcal{L}}{\arg\max}\, \# \left\{ \tau\left(\boldsymbol{x}^{(i)}\right) = c \ : \ i \in \mathcal{N} \right\}.$$

Recall the principles of the *k*-Nearest Neighbor (*k*NN) classifier from the lecture. Take a look at scikit-learn's KNeighborsClassifier documentation and make yourself familiar with its usage. Find out about the classifiers **tie-breaking rule**, i.e. what happens if two data points have the same distance.

Tasks

Load Dataset

1. Load the breast cancer data using scikit-learn's API.
2. Read the information available in the User Guide.

Pair work: Answer the following questions

- How was the data obtained?
- How many classes are there?
- What does each row represent?
- How many data points are there?
- How many features?
- Which kind of features are there?
- Which feature(s) have/has the highest absolute values?
- Just from the information present, do you expect high or low correlation?

2. Reuse the notebook from Lab 1 to inspect the dataset and verify your assumptions.

Classification

1. Split data into a train and test set using a test set size of 15%.
2. Scale the data.
3. Train a `KNeighborsClassifier` and report the accuracy score on the test set.
    - Use $k = 7$ nearest neighbors.
    - Reuse your random seed.

```python
from sklearn.neighbors import KNeighborsClassifier
```

Confusion Matrix

1. Use scikit-learn's method to display the confusion matrix:

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test, cmap='Blues')
plt.show()
```

Pair work: Answer the following questions

- Benign data points are represented by which class number?
- Malignant data points are represented by which class number?
- How many data points are correctly classified as benign?
- How many data points are correctly classified as malignant?
- How many data points are classified as malignant, although being benign?
- How many data points are classified as benign, although being malignant? Which of the two cases is more favorable?

Performance Measures

1. Given the output from the confusion matrix, compute precision, recall and F1 score by hand for both labels.

$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$
$$\text{F1-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

2. Only then, use the methods implemented in scikit-learn for verification. Again, make sure to compute the values for both classes.

```python
from sklearn.metrics import precision_score, recall_score, f1_score
```

Classification Report

The classification report provides a very good summary of all scores. By comparing to your own computed values, make sure to be able to read and interpret the report.

```python
from sklearn.metrics import classification_report
print(classification_report(y_test, y_hat, digits=4))
```

Optional

Experiment with different settings, e.g. use a different number *k* of neighbors used for classification, use only a fraction of the input features, use the original or scaled data etc.

Homework

Redo Lab 3 for the wine dataset. Compare the *k*NN performance to that of the MinDist classifier by entering the results in your table.

Take the quiz in Moodle. Make sure to have your Python notebook open, your code up and running and be able to compute the performance metrics.

❗ This time, there are two quizzes, were the first one is only supposed to verify the correct formatting of numbers that is then needed in the second quiz (which will only be available once you finished the first one).

🕘 Deadline: 22.11.2022 (5:00 pm)

Further Reading

✨ On the importance of using good performance measures: 🌐 Accuracy Paradox (Wikipedia)