Artificial Intelligence and Knowledge Engineering
Assignment 4

Katarzyna Fojcik, Joanna Szołomicka, Teddy Ferdinan

March 7, 2025

1 Excercise goal

The purpose of the exercise is to provide a practical introduction to simple machine learning algorithms and the basic steps of implementing a machine learning project: data mining and problem definition, data preparation, selection of algorithms and hyperparameters, evaluation of algorithms, improvement of results, presentation of results.

2 Theoretical introduction

In this section, you will find information useful during solving the stated problems.

2.1 Machine Learning

Machine Learning is one of the fields of artificial intelligence, a broad category of algorithms that aim to perform specific tasks automatically. Algorithms learn from data and then apply that knowledge to infer and solve the task.

Machine learning types:

- **Supervised learning** – a method of learning in which the training set, on which the algorithm learns, contains the solution to the problem in the form of labels or classes. The two main areas that use supervised learning are **regression** (value prediction) and **classification** (class prediction).
- **Unsupervised learning** – a method of learning in which the training set is not labeled. The input to the model is raw data, and the model has to find connections between the data (so-called learning without a teacher). Unsupervised algorithms are **clustering** and various algorithms for **data visualization** and **dimensionality reduction**.
- **Reinforcement learning** – its task is to interact with the environment based on the information collected. In contrast to methods mentioned earlier, in reinforcement learning, one does not prepare a set of training data, but an environment from which the model will collect data automatically. The aim of the model is to maximize the reward the environment returns. The environment may depend on the learning objective. In the case of teaching a program that plays games, the environment will be a game with its rules, and in the case of a program that learns to control a rover it will be the real world.

2.2 Division of data into training and validation sets

Training set — a set of data used to train a model. Using this data, the model learns to classify properly and builds all dependencies. It predicts possible outcomes and makes decisions based on the data given to it.

Validation set – a dataset used to run an unbiased evaluation of a model that was learning on training data. The evaluation is performed during the selection of the model or hyperparameters. It is important that the data in the validation set has not been used for the model training, as it will be unsuitable for objective, unbiased evaluation.

Test set – when the selected model is trained, it is time to test it on data from test set. It is very important that this data has not been previously used to train or validate the model. The purpose of testing the model is to see how well the chosen algorithm performs on data that it has never dealt with before.

Cross-validation is an alternative method of preparing training and testing sets for the model. It is **more efficient in assessing the actual accuracy of the model**, but it is less efficient than the naive method. This method consists of **dividing the dataset into subsets** and creating **validation and training datasets** out of them.

There are many types of cross-validation. Baseline K-fold cross-validation assumes that the data is independent and identically distributed (IID). It divides all samples into k groups of equal size if possible. The ML algorithm is trained using $k-1$ groups, and the skipped group is used for validation. Thus, k training sessions are conducted and the results from k validations are averaged.

The ratio of splitting the data into training, validation and test sets should depend on the sample size. There are different splitting practices, but it is generally accepted that for smaller sets we prioritize the need to train the algorithm on as many samples as possible. Then the validation (and test) sets are relatively smaller (in the case of cross-validation, we choose a larger k, e.g., 10). However, having several thousand samples at our disposal, we can decide to increase the number of samples in the validation set (alternatively, reduce the number of groups to $k = 5$).

2.3 Overfitting and Underfitting

Overfitting occurs when a model learns patterns that are too specific to the training data, potentially capturing both meaningful details and noise, making it unreliable for predicting unseen data. An overfitted model focuses too much on training data details instead of learning generalizable patterns. Overfitting can result from: (1) training data that does not sufficiently represent real-world data; and/or (2) an overly complex model (i.e., the number of parameters in the model is too large relative to the dataset size). To mitigate overfitting, techniques like regularization, dropout, early stopping of the training, parameter pruning, and Bayesian priors are used.

Underfitting occurs when a model is too simplistic to capture patterns, leading to poor performance on both seen and unseen data. Underfitting may happen because of (1) an overly simple model (i.e., the number of parameters in the model is too small relative to the dataset size); (2) insufficient training data; (3) inadequate input features; and/or (4) excessive regularization which hinders learning.

2.4 Naive Bayes Classifier

The Naive Bayes Classifier is a simple probabilistic classifier that naively assumes that features (predictors) for a fixed class label are independent and are equally significant. It is based on Bayes Theorem.

Definition (Bayes Theorem) *Let A i B be events, $P(B) > 0$, $P(A|B)$ is conditional probability, then:*

$$\begin{aligned} P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A) \\ P(B|A) = \frac{P(A|B)P(B)}{P(A)} \end{aligned} \tag{1}$$

Definition (Naive Bayes Classifier) *Let $X_1, \ldots, X_n$ be features, then:*

$$\begin{aligned} P(X_1|X_2, Y) = P(X_1|Y) \\ P(X_1, X_2, Y) = P(X_1|X_2, Y)P(X_2, Y) = P(X_1|X_2, Y)P(X_2|Y)P(Y) \\ P(X_1, X_2|Y) = P(X_1|Y)P(X_2|Y) \\ P(X_1, X_2, \ldots, X_n|Y) = \prod_{i=1}^{n} P(X_i|Y) \end{aligned} \tag{2}$$

2.5 Decision tree

A decision tree in machine learning is a representation of a classifier in the form of a tree that supports the decision process. Building such a tree is based on various algorithms and their parameters, which are calculated for each new node of the decision tree. The example parameters are entropy and information gain.

Definition (Entropy) *Entropy is a measure of the variability of the data given by formula 3.*

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{3}$$

where S is a current dataset for which entropy is calculated (for each node of the tree it will be a different - suitably smaller dataset), X is a set of classes in the set S, $p(x)$ is the ratio of the number of elements in the class x to the number of elements in the set S.

Definition (Information Gain) *Information Gain $G(A)$ is a measure of the difference in entropy before and after splitting dataset using a given attribute (feature).*

$$G(A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v) \tag{4}$$

where $H(S)$ is the entropy of the set S, $Values(A)$ is the set of all values of attribute A, S_v is a subset of S such that $S_v = \{s \in S : A(s) = v\}$, $H(S_v)$ is the entropy of S_v.

Decision trees are very susceptible to **overfitting**. As the depth increases, the trees are able to adapt very well to the training data, but at the same time their ability to generalize deteriorates (the error on the test set increases).

2.6 PCA (Principal Component Analysis)

PCA is a statistical method of factor analysis: a dataset consisting of N observations, each containing K variables, can be interpreted as a cloud of N points in K-dimensional space. The aim of PCA is to rotate the coordinate system in such a way that the variance of the first coordinate is maximized first, then the second and so on. The transformed coordinate values are the principal components. In machine learning, it is used to reduce the dimension of the data.

2.7 Metrics

To evaluate the classification, the results of various metrics are used (usually only for the validation data):

2.7.1 Confusion matrix

Depending on the performance of the classifier, four cases are distinguished (Table 1):

- **TP (True Positive)** – number of truly recognized positive cases
- **FP (False Positive)** – number of falsely recognized positive cases
- **FN (False Negative)** – number of falsely recognized negative cases
- **TN (True Negative)** – number of truly recognized negative cases

		Real class	
		Positive class	Negative class
Prediction	Positive class	TP	FP
	Negative class	FN	TN

Table 1: Confusion matrix.

2.7.2 Accuracy

Accuracy ACC is the most common indicator that allows us to assess the quality of data classification. It specifies which part of all classified items has been correctly classified. That is, we divide the sum of the correct classifications from a given category (TP) and the correct ones from other categories (TN) by the number of all classified cases.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} \tag{5}$$

2.7.3 Recall/sensitivity/true positive rate

Recall TPR tells us what is the proportion of correctly predicted positive cases (TP) among all positive cases (including those misclassified as negative - FN).

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6}$$

2.7.4 Precision/positive predictive rate

We divide the number of correctly predicted positive values (TP) by the sum of all positively predicted values (including those incorrectly predicted), i.e. TP+FP. As a result, we find how many positively predicted examples are actually positive.

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{7}$$

2.7.5 F1-score

F1-score is the harmonic mean between precision and recall. The closer it is to one, the better the classification model. In the best case, F1-score is equal to 1, when we are dealing with perfect sensitivity and precision.

$$\mathrm{F1} - \mathrm{score} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{8}$$

3 Task

Face the problem of identifying the type of fetal morphologic pattern based on cardiotocogram (CTG) diagnostic features. To do this, use the provided `cardiotocography_v2.csv` file. This data was adapted from the original Cardiotocography dataset[1] for the purpose of this laboratorium exercise.

This file contains 2126 rows of fetal cardiotocograms (CTGs) data. The goal is to perform a classification of 10 types of fetal morphologic patterns based on 21 diagnostic features. There are some missing values in the features. The columns in the file are:

- LB - [feature] FHR baseline (beats per minute)
- AC - [feature] number of accelerations per second
- FM - [feature] number of fetal movements per second
- UC - [feature] number of uterine contractions per second

[1] D. Campos and J. Bernardes. "Cardiotocography," UCI Machine Learning Repository, 2000. [Online]. URL: https://doi.org/10.24432/C51S4N

- DL - [feature] number of light decelerations per second
- DS - [feature] number of severe decelerations per second
- DP - [feature] number of prolonged decelerations per second
- ASTV - [feature] percentage of time with abnormal short term variability
- MSTV - [feature] mean value of short term variability
- ALTV - [feature] percentage of time with abnormal long term variability
- MLTV - [feature] mean value of long term variability
- Width - [feature] width of FHR histogram
- Min - [feature] minimum of FHR histogram
- Max - [feature] maximum of FHR histogram
- Nmax - [feature] number of histogram peaks
- Nzeros - [feature] number of histogram zeros
- Mode - [feature] histogram mode
- Mean - [feature] histogram mean
- Median - [feature] histogram median
- Variance - [feature] histogram variance
- Tendency - [feature] histogram tendency
- CLASS - [target] fetal morphologic pattern class code (1 to 10)

The tasks should be done using Python and/or WEKA.

Scoring:

1. data mining – provide basic statistics and comments on the features and labels in the dataset. (10 points)
2. data preparation – split the data into training and validation sets (alternatively use cross-validation). Prepare the data using a method for dealing with missing values (suggested: removal, imputation, interpolation). Then, use 2 different methods (separately) for data processing (suggested: normalization, standardization, discretization, feature selection, PCA). Compare the classification results without data processing against the results from different data processing methods. (30 points)
3. classification – run classification experiments using (1) naive Bayes classifier and (2) decision tree, each using 3 different sets of hyperparameters. Study the impact on the results. (40 points)

   Bonus - Test more advanced algorithms (with understanding!), such as random forest or Support Vector Machine (SVM). (5 points)

   Bonus - For one chosen classifier, propose one approach for mitigating overfitting. Run the experiment using this approach and compare the results against the experiment without it. (5 point)
4. classification evaluation – use the metrics for classification evaluation to compare the results of different data preparation methods and classifier. Interpret the results. (20 points)

Prepare a report for the task, including a brief description of all the steps performed and the task results with an interpretation (preferably collected in a table). In your report, indicate the source materials used and briefly describe the libraries used in the implementation. Reports must be sent to the tutor at least 24h before the tasks' submission.

4 Appendix - filters and classifiers in Weka Explorer

4.1 Filters

- **attribute discretization:** filters → supervised → attribute → Discretize lub filters → unsupervised → attribute → Discretize
- **attribute normalization:** filters → unsupervised → attribute → Normalize
- **PCA:** filters → unsupervised → attribute → PrincipalComponents
- **deleting an attribute:** filters → unsupervised → attribute → Remove
- **completing the attribute value:** filters → unsupervised → attribute → ReplaceMissingValues
- **attribute standardization:** filters → unsupervised → attribute → Standardize
- **deleting instances with given attribute values:** filters → unsupervised → instance → RemoveWithValues

4.2 Attribute selection

Card *Select attributes* - selection by the button *Attribute Evaluator*:

- GainRatioAttributeEval
- InfoGainAttributeEval
- PrincipalComponents

4.3 Classifiers

Card *Classify*:

- **naive Bayes classifier:** classifiers → bayes → NaiveBayes or NaiveBayesMultinomial (depending on the needs)
- **decision tree C4.5:** classifiers → trees → J48
- **group of classifiers:**
  - **boosting:** classifiers → meta → AdaBoostM1
  - **bagging:** classifiers → meta → Bagging
  - **different types of classifiers:** classifiers → meta → Stacking

5 Appendix - useful libraries for Python

- from sklearn.preprocessing import Normalizer, StandardScaler
- from sklearn.decomposition import PCA
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.naive_bayes import GaussianNB