MEDNARODNA
PODIPLOMSKA ŠOLA
JOŽEFA STEFANA

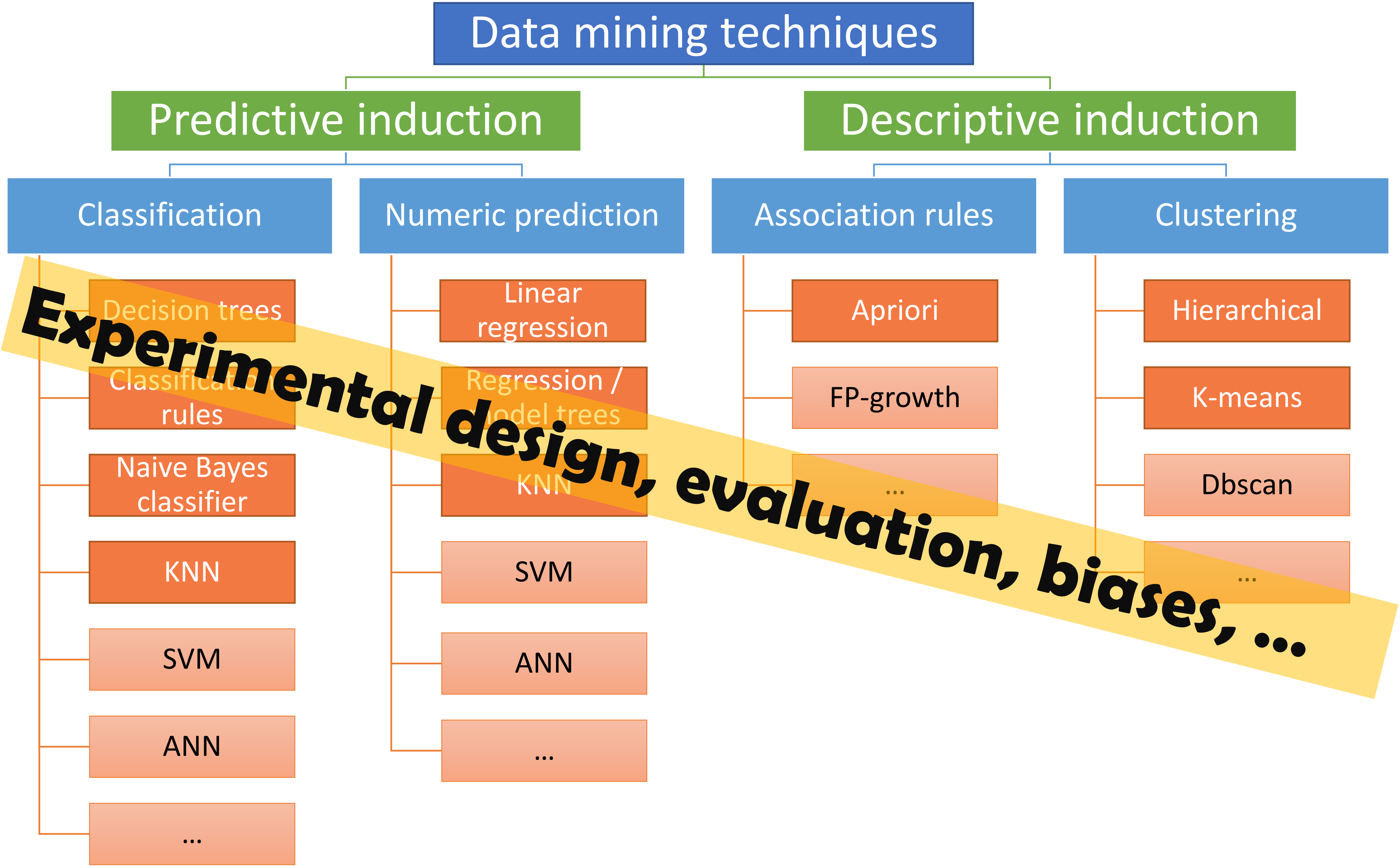
INFORMATION AND COMMUNICATION TECHNOLOGIES
PhD study programme

Data Mining and Knowledge Discovery

Petra Kralj Novak

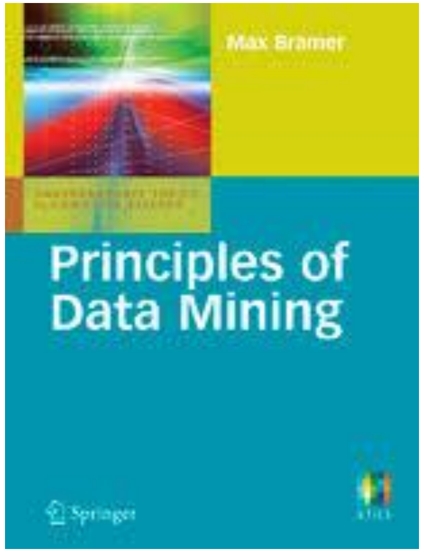
November 17, 2020

http://kt.ijs.si/petra_kralj/dmkd3.html



Data mining techniques
Predictive induction
Descriptive induction
Classification
Numeric prediction
Association rules
Clustering
Decision trees
Classification rules
Naive Bayes classifier
KNN
SVM
ANN
…
Linear regression
Regression / model trees
KNN
SVM
ANN
…
Apriori
FP-growth
…
Hierarchical
K-means
Dbscan
…
Experimental design, evaluation, biases, …

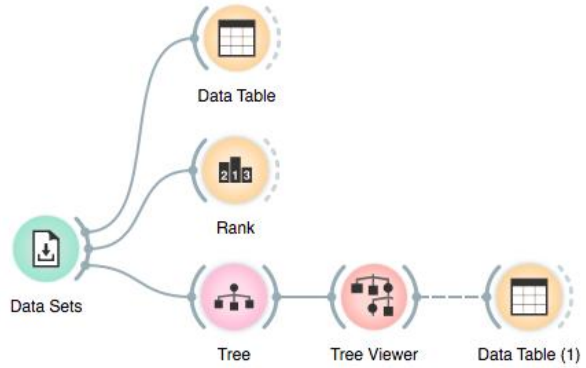
Bramer, Max. (2007). Principles of Data Mining. 10.1007/978-1-84628-766-4.



1. Data for Data Mining
2. Introduction to Classification: Naïve Bayes and Nearest Neighbour
3. Using Decision Trees for Classification
4. Decision Tree Induction: Using Entropy for
5. Decision Tree Induction: Using Frequency
7. Continuous Attributes
8. Avoiding Overfitting of Decision Trees
9. More About Entropy
10. Inducing Modular Rules for Classification
11. Measuring the Performance of a Classifier
12. Association Rule Mining I
13. Association Rule Mining II
14. Clustering
15. Text Mining

- Basic chapters about classification: 1, 2, 3, 4, 6, 8, 11
- Necessary prerequisite also for the course by prof. dr. Sašo Džeroski, doc. dr. Panče Panov: Computational Scientific Discovery from Structured, Spatial and Temporal Data

Hands-on



orange

- Open source machine learning and data visualization
- Interactive data analysis workflows with a large toolbox
- Visual programming
- *Demsar J, Curk T, Erjavec A, Gorup C, Hocevar T, Milutinovic M, Mozina M, Polajnar M, Toplak M, Staric A, Stajdohar M, Umek L, Zagar L, Zbontar J, Zitnik M, Zupan B (2013) Orange: Data Mining Toolbox in Python, JMLR 14(Aug): 2349−2353.*

scikit learn

- **scikit-learn** is Gold standard of Python machine learning
- Simple and efficient tools for data mining and data analysis
- Well documented
- *Pedregosa et al. (2011) [Scikit-learn: Machine Learning in Python](), JMLR 12, pp. 2825-2830.*

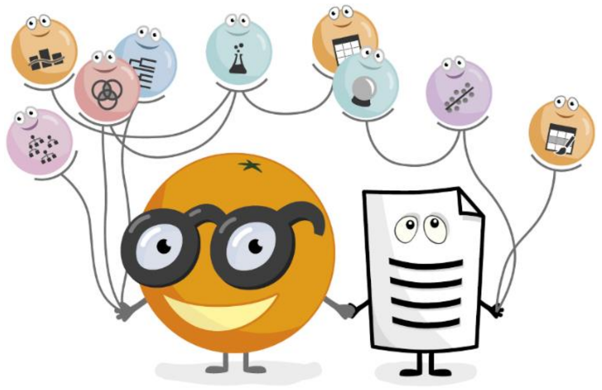
K Keras

- Neural-network library written in Python.
- *Chollet, F. et al. (2015) "Keras"*

```
# ------------------------------------------------------------------------------------------
print("Train and test classification models")
classifiers = [
    # ("Naive Bayes", naive_bayes.MultinomialNB()),
    ("Logistic regression", linear_model.LogisticRegression(C=1e5, solver='lbfgs', multi_class='multinomial', max_iter=600)),
    ("MultinomialNB", MultinomialNB()),
    ("SVC", svm.LinearSVC()),
    ("SVC-RBF", svm.SVC(gamma='scale', decision_function_shape='ovo'))]


for name, classifier in classifiers:
    classifier.fit(train_data, y_train)
    predictions = classifier.predict(test_data)
    classifier.confusion_matrix = metrics.confusion_matrix(predictions, y_test, labels=["negative", "neutral", "positive"])
    classifier.accuracy = metrics.accuracy_score(predictions, y_test)
    print(name, classifier.accuracy, "\n Confusion matrix: \n", classifier.confusion_matrix)
    pickle_clf(classifier, path="./models/"+name+".pkl")
```

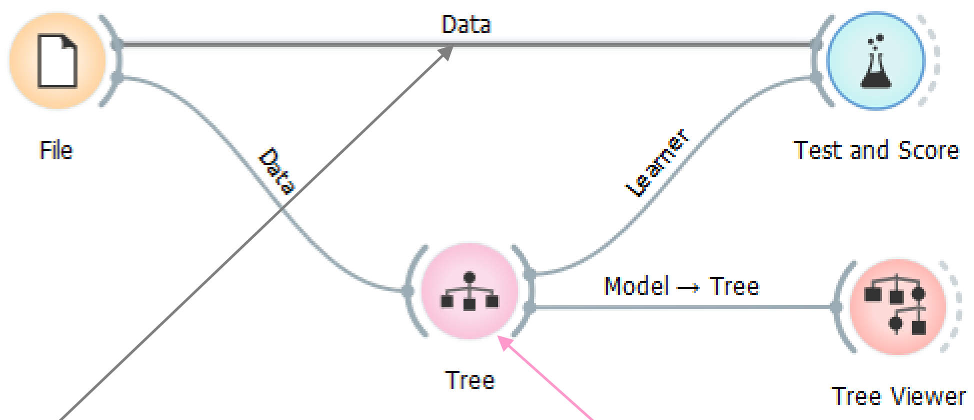
orange

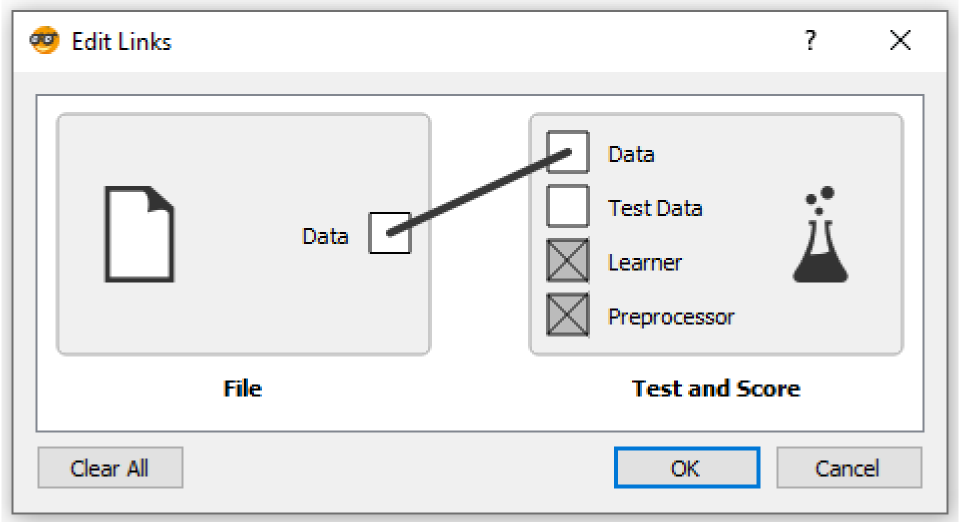


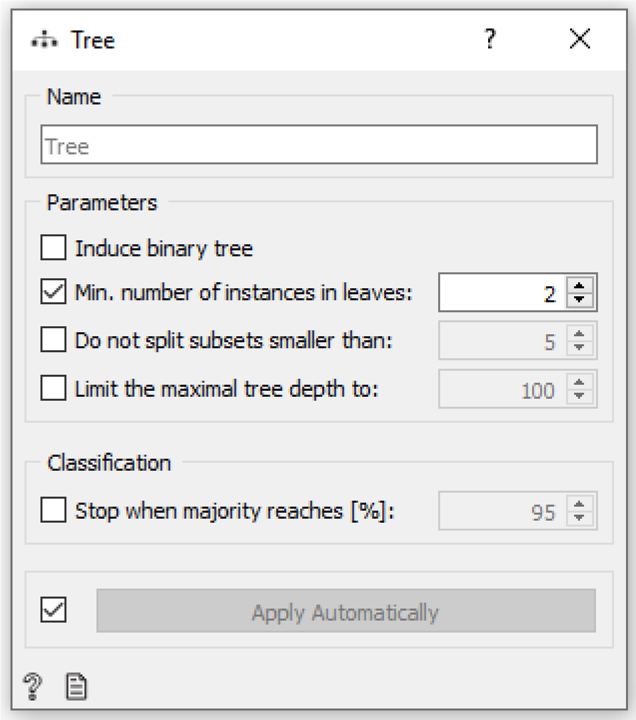
- Open source machine learning and data visualization
  - Software: https://orange.biolab.si/
  - Datasets: http://file.biolab.si/datasets/
  - Tutorials: https://www.youtube.com/channel/UClKKWBe2SCAEyv7ZNGhIe4g
- Interactive data analysis workflows
- Visual programming
- Based on numpy, scipy and **scikit-learn**, GUI: Qt framework

orange workflow

- Widgets: building blocks of data analysis workflows that are assembled in Orange's visual programming environment.
- A typical workflow may mix widgets for **data manipulation**, **visualization**, **modeling**, **evaluation**, …
- Widgets have inputs and outputs (typically *data objects, learner objects, classifier objects*, …) and parameters
- Interactive







Classification

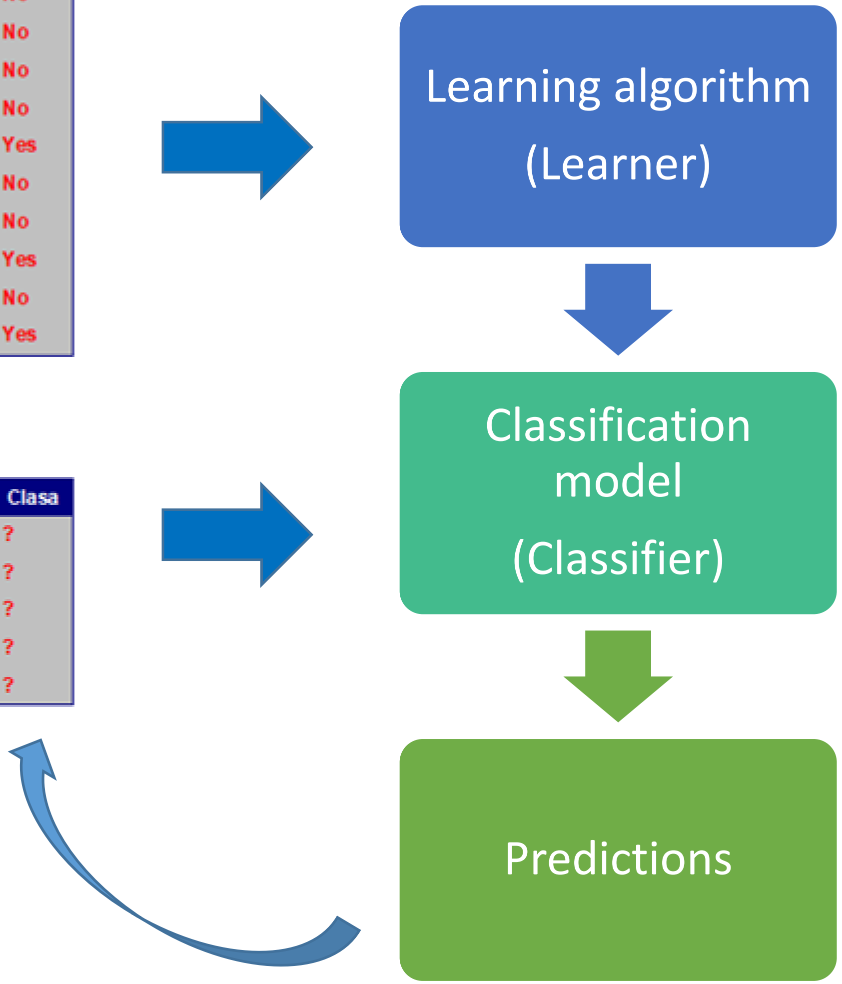
The basic classification schema

Št	Atrib1	Atrib2	Atrib3	Clasa
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training set

Št	Atrib1	Atrib2	Atrib3	Clasa
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

New data



- A classifier is a function that maps from the attributes to the classes
  - Classifier(attributes) = Classes
  - *f(X) = y*
- In training, the attributes and the classes are known (training examples) and we are learning a mapping function *f* (the classifier)
  - *?(X) = y*
- When predicting, both the attributes and the classifier are known, and we are assigning the classes
  - *f(X) = ?*
- What about evaluation?

The basic classification schema - evaluation

X_{train} y_{train}

Šr	Atrib1	Atrib2	Atrib3	Clasa
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training set

X_{test} y_{test} y_{pred}

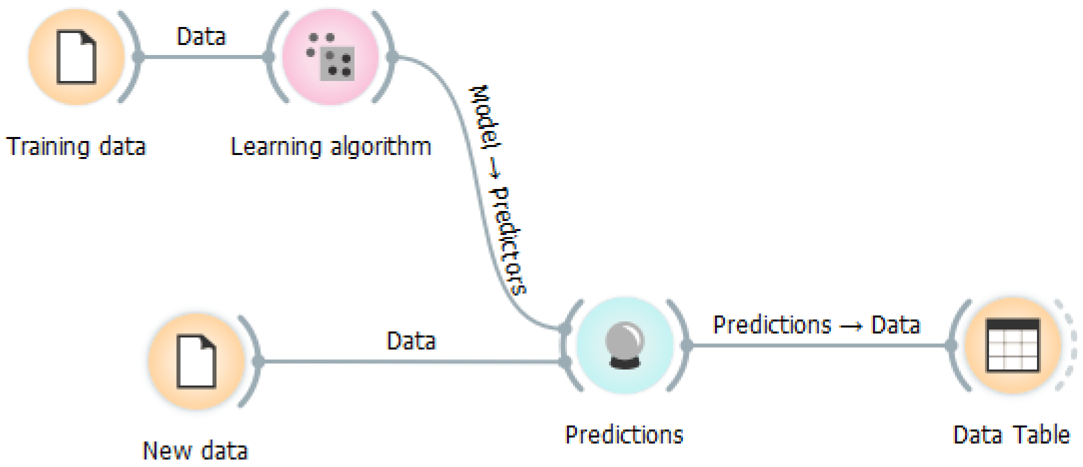
Šr	Atrib1	Atrib2	Atrib3	Clasa	Clasa
11	No	Small	55K	No	No
12	Yes	Medium	80K	No	Yes
13	Yes	Large	110K	No	No
14	No	Small	95K	No	No
15	No	Large	67K	Yes	No

Testing data

- When evaluating, *f*, *X* and y are known. We compute the predictions $y_p = f(X)$ and evaluate the difference between *Y* and Y_p.
- *Train and test data:*

  X_{train}, X_{test}, y_{train}, y_{test}

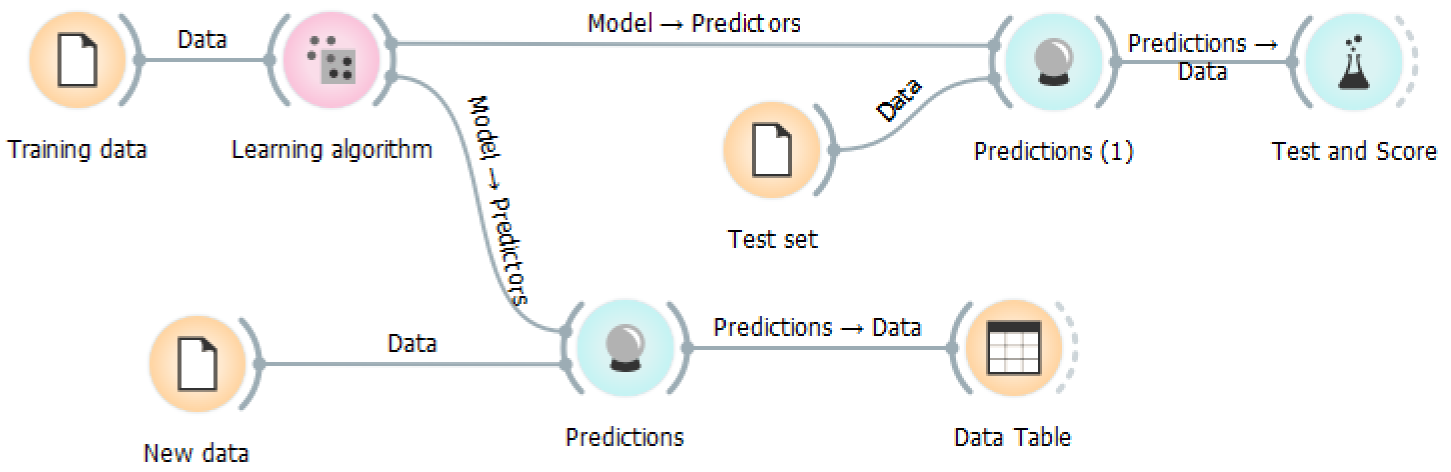
Basic classification schema in Orange



- We train the model on the train set
- We predict the target for the new instances
- There are several classification algorithms:
  - Decision trees
  - Naive Bayes classifier
  - K nearest neighbors (KNN)
  - Artificial neural networks (ANN)
  - ….

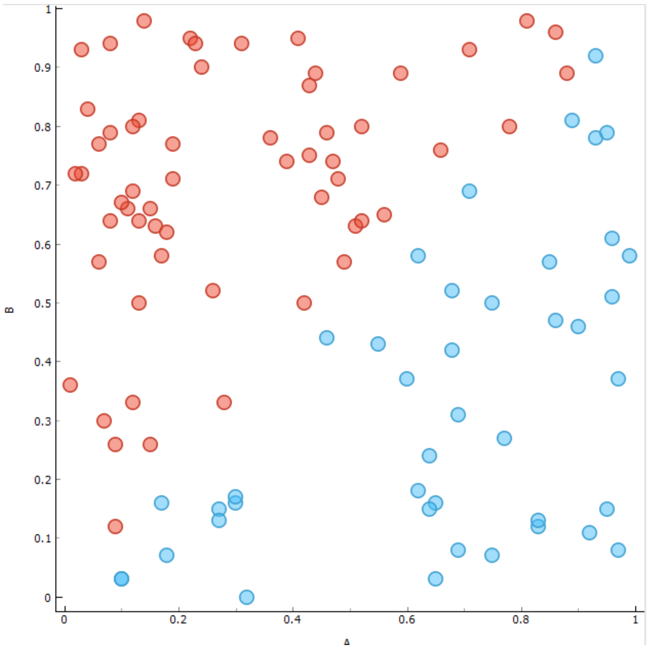
Classification with evaluation

- We train the model on the train set
- We evaluate on the test set
- We classify the new instances

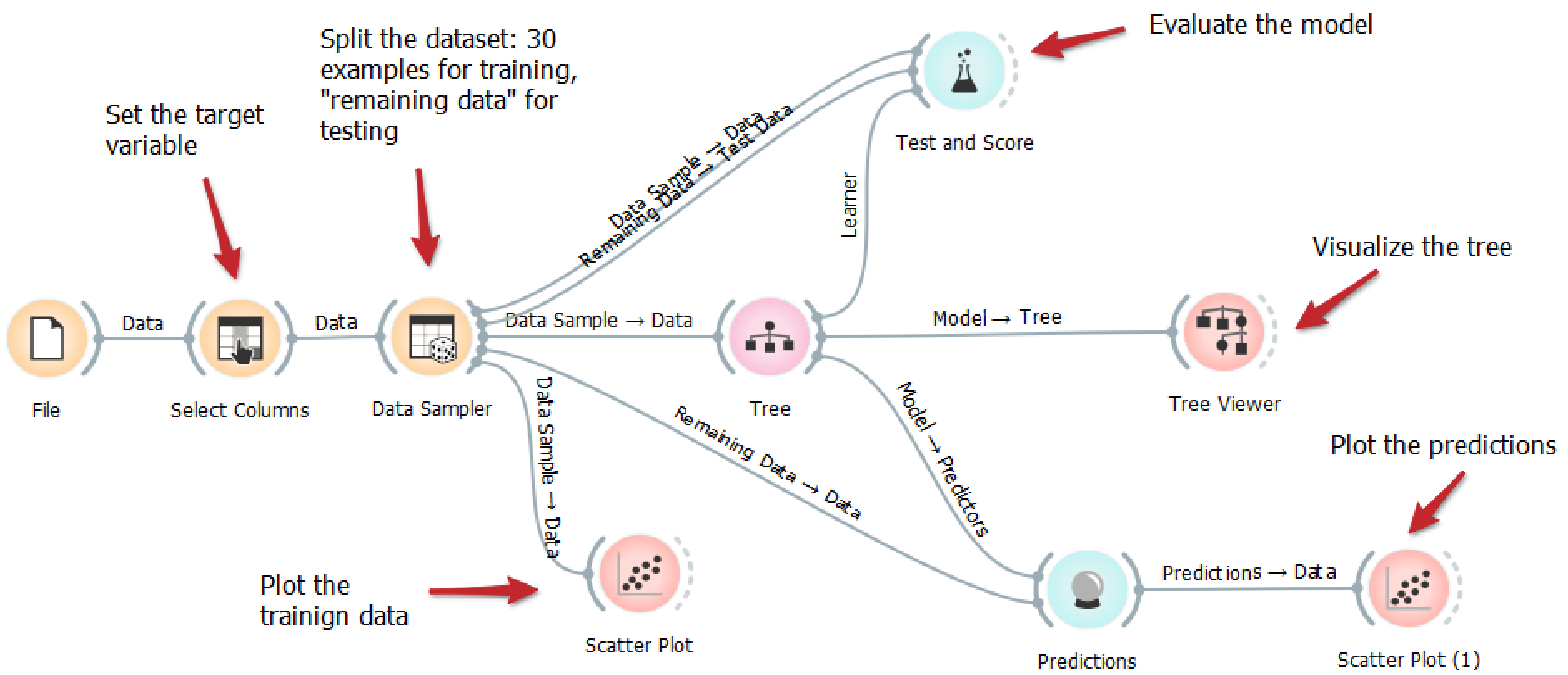


Lab exercise: Decision trees & Language bias

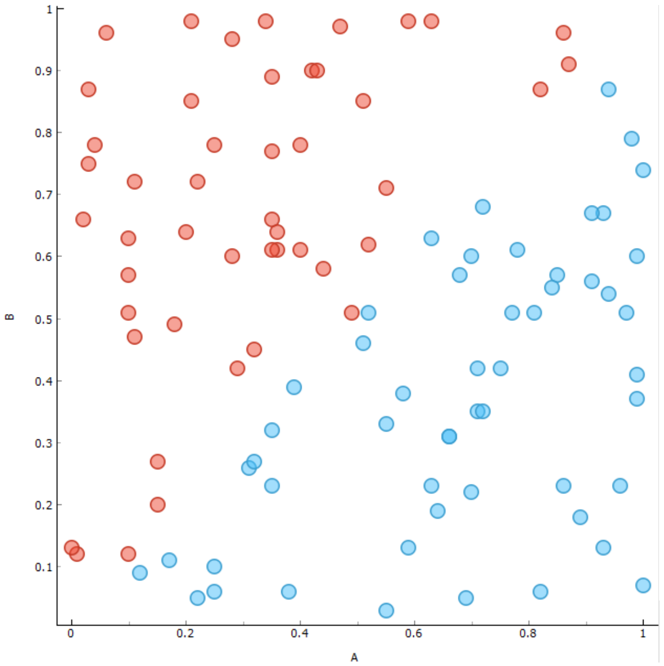
- Use the dataset A-greater-then-B.csv from http://source.ijs.si/pkraljnovak/DM_course
    - Attributes A, B and C have random values
    - Target variable „A>B“, has value „true“ if A>B else “false”
- Use Orange trees to predict „A>B“ from the attributes A, B in C
    - Use separate test set for validation (Widget Data Sampler)
    - Plot the training and classified data in “Scatter Plot”
- How good is your model?
- How does the training set size influence the model performance?



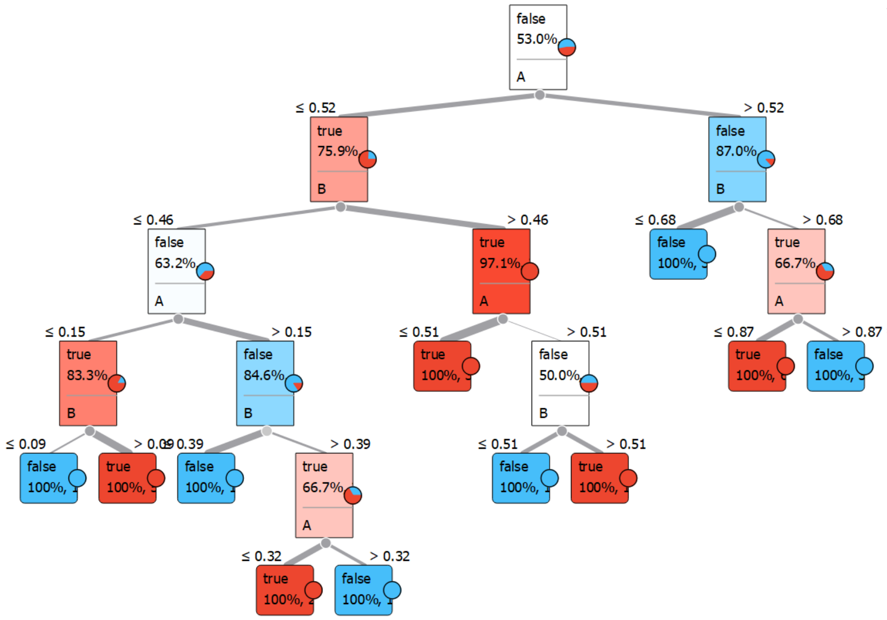
Lab exercise: Decision trees & Language bias



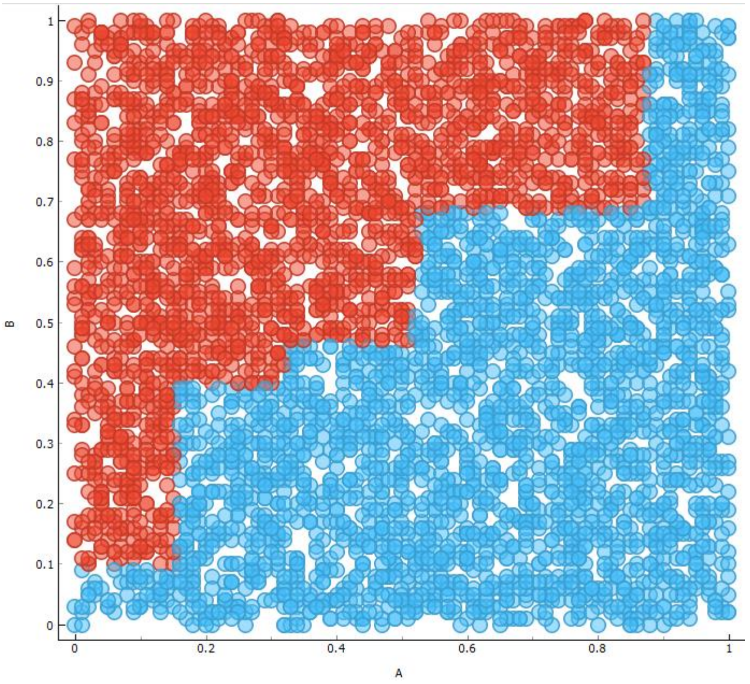
Lab exercise: Decision trees & Language bias



Training set

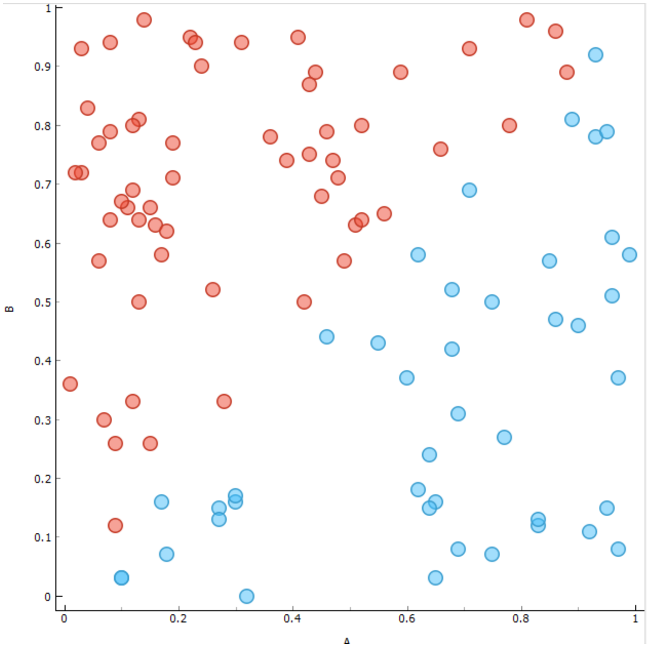


Decision tree

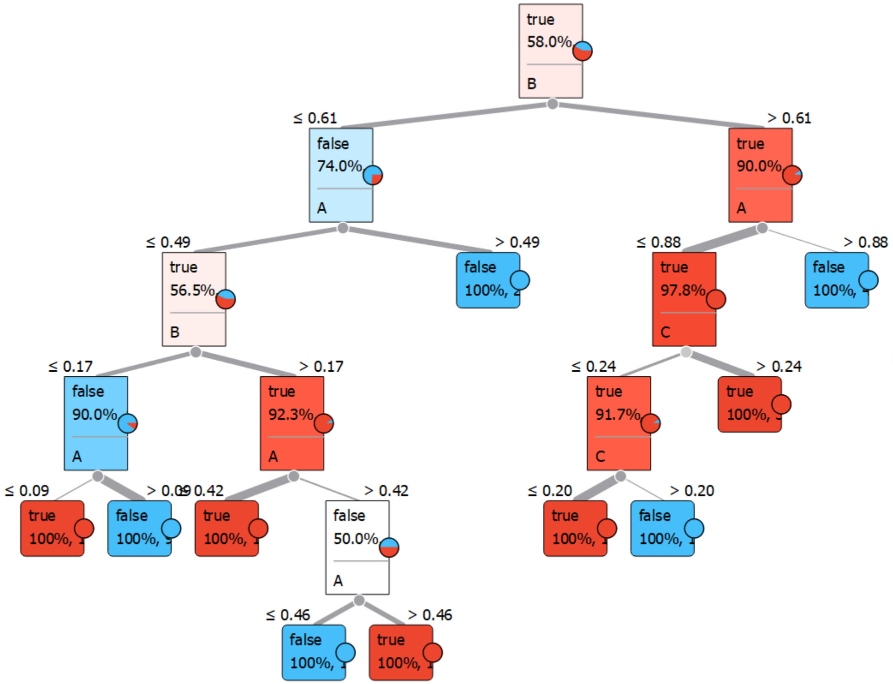


Test set

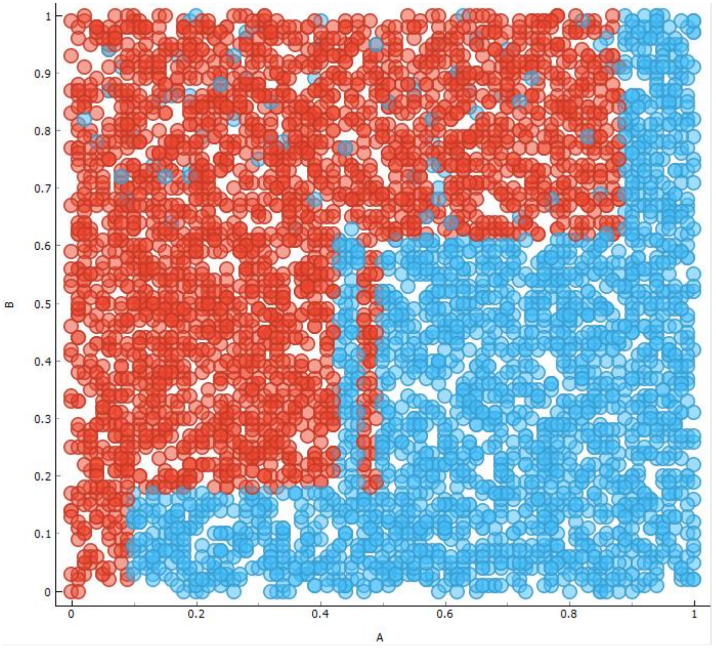
Same program, different random seed



Training set



Decision tree



Test set

How to overcome this

- Feature engineering
  - Create a new feature A>B
  - Examples
    - We have a person's height and body mass
      → Create a new attribute BMI (bod mass index)

$$BMI = \frac{Weight\ (kg)}{[Height(m)]^2}$$

    - We have income and outcome data
      → Create a new attribute "profit"
- Ensemble
  - We build more models that vote for the final classification
  - Random forest: Several trees built on different subsets od the training set
  - On the "A>B" example, decision trees achieve CA 88,2% while random forest 90,8%
  - As a general rule, classifier ensembles always outperform single classifiers
- Use other classifiers
  - Linear classifier, SVM with linear kernel….

Basic classification in scikit

```
csvFileName = r".\Datasets\A-greater-then-B.csv"
df = pd.read_csv(csvFileName)

feature_cols = ['A', 'B', 'C']
target_var = 'A>B'

X = df[feature_cols].values
y = df[target_var].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

decision_tree = tree.DecisionTreeClassifier()

decision_tree.fit(X_train, y_train)

y_pred = decision_tree.predict(X_test)

accuracy = metrics.accuracy_score(y_test, y_pred)
```

2_classification.py on http://source.ijs.si/pkraljnovak/DM_course

Basic classification in scikit

A	B	C	A>B
0.953725	0.544997	0.854959	TRUE
0.490541	0.953735	0.200973	FALSE
0.987391	0.524999	0.092299	TRUE
0.074883	0.145092	0.158558	FALSE
0.215517	0.003417	0.441095	TRUE
0.993418	0.69765	0.535384	TRUE
0.90678	0.787445	0.043996	TRUE
0.22488	0.316079	0.542245	FALSE
0.478895	0.262404	0.505151	TRUE
0.348876	0.77149	0.946645	FALSE
0.092137	0.563398	0.245223	FALSE
0.177709	0.068787	0.88188	TRUE
0.794501	0.546356	0.087682	TRUE
0.535652	0.797198	0.449511	FALSE
0.998743	0.904471	0.609526	TRUE
0.378494	0.969959	0.421158	FALSE

```
csvFileName = r".\Datasets\A-greater-then-B.csv"
df = pd.read_csv(csvFileName)

feature_cols = ['A', 'B', 'C']
target_var = 'A>B'

X = df[feature_cols].values
y = df[target_var].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

decision_tree = tree.DecisionTreeClassifier()

decision_tree.fit(X_train, y_train)

y_pred = decision_tree.predict(X_test)

accuracy = metrics.accuracy_score(y_test, y_pred)
```

2_classification.py on http://source.ijs.si/pkraljnovak/DM_course

Scikit documentation

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.`**`DecisionTreeClassifier`**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

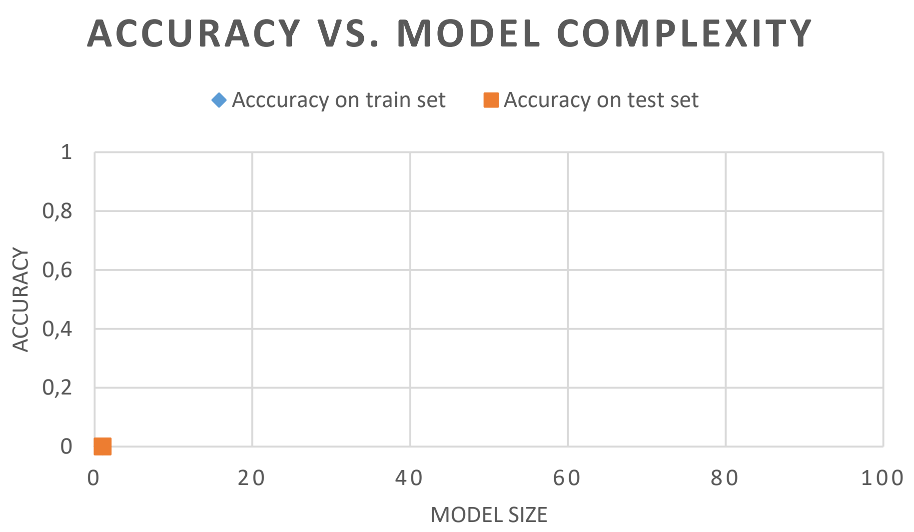
Home assignment

Model complexity (e.g. number of leafs) vs. accuracy on train and test set

Datasets:

- A-greater-then-B.csv
- Another reasonably sized classification dataset from http://file.biolab.si/datasets/

You can start from the samples of code from the gitlab repository http://source.ijs.si/pkraljnovak/DM_course



Evaluation

How good is the model

!!!

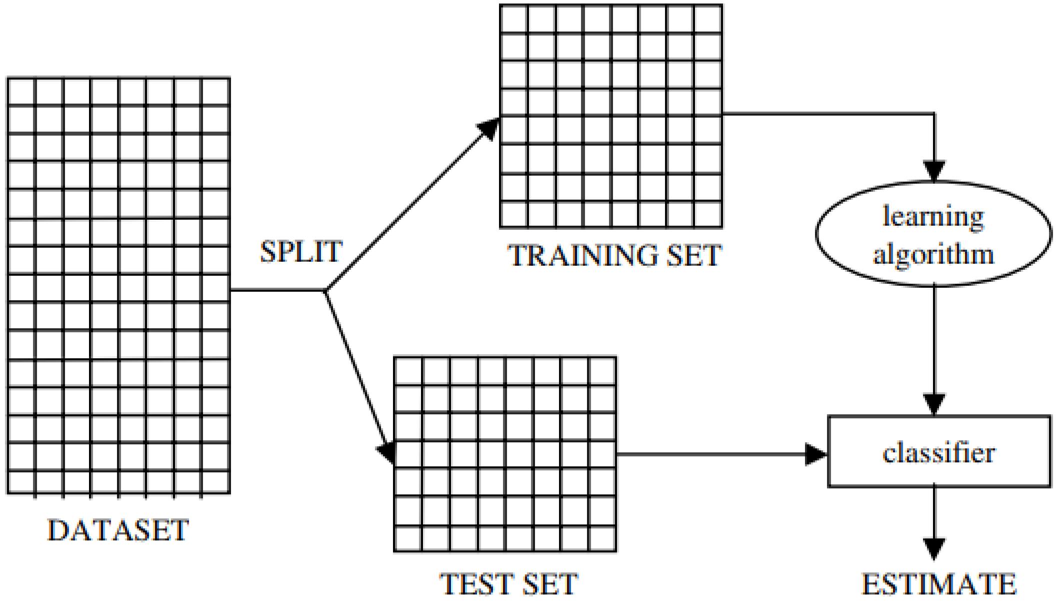
Evaluation goal

- How good is the model

- Method
  - HOW we measure?

- Metric
  - WHAT we measure?

Method: Test on a separate test set



Stratified sampling

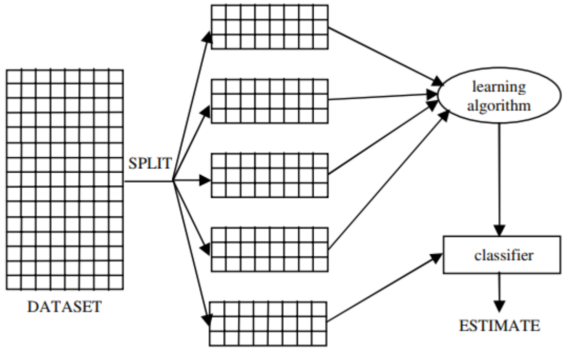
- Stratified sampling aims at splitting one data set so that each split are similar with respect to the target variable distribution.

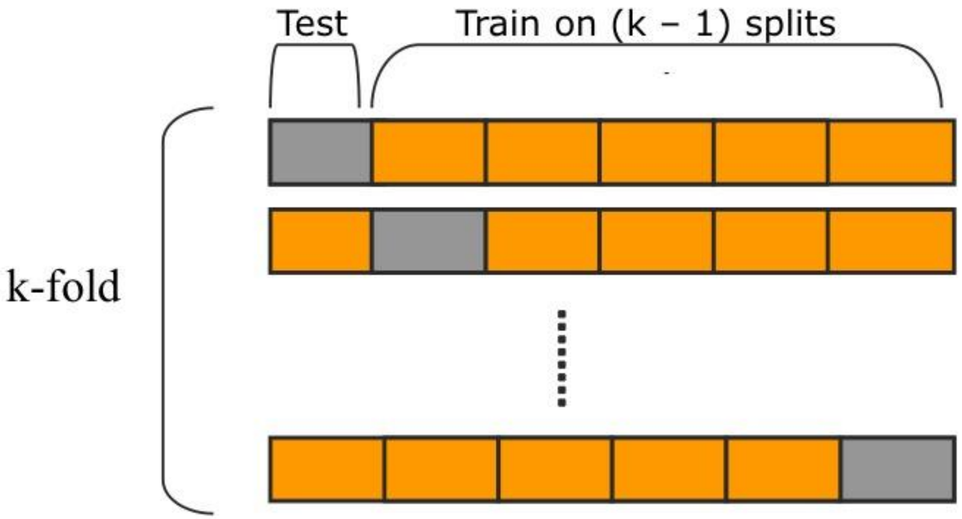
Method: Random sampling

- Repeat several times „Test on a separate test set“ with different test set selections
- Compute the mean, variance on the results …
- The evaluation is more robust as it does not depend so much on a single random split

Method: *K*-fold cross validation

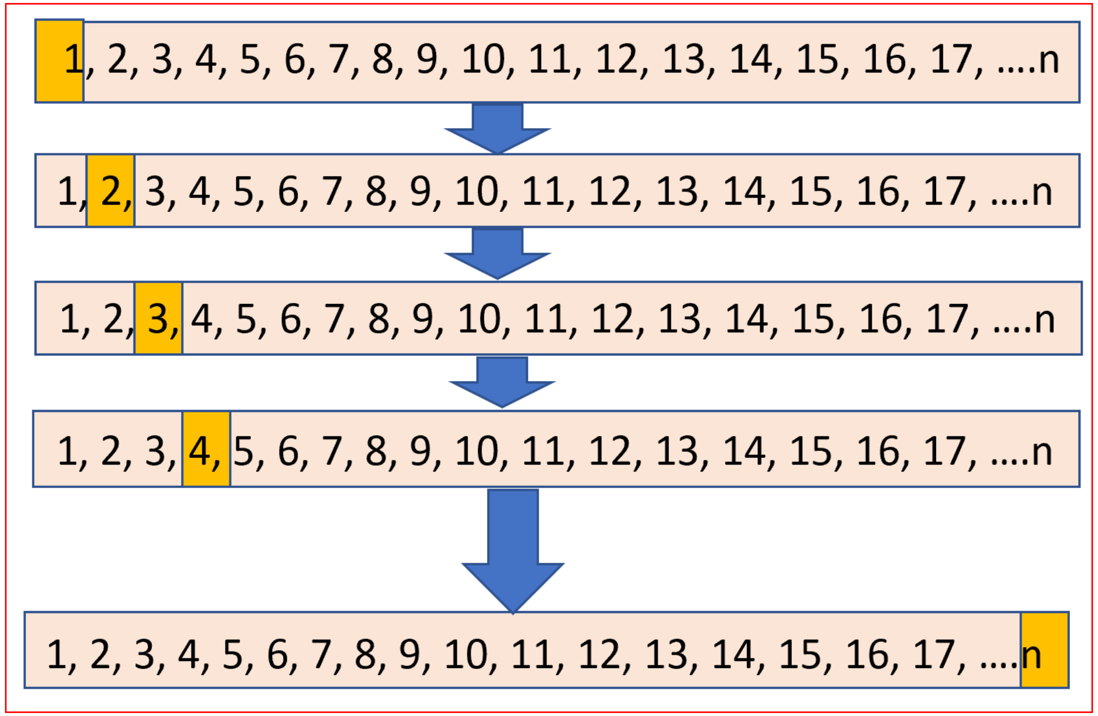
- Most commonly used in machine learning
- Split the dataset into *k* (disjunctive) subsets
- Repeat *k*-times:
    - Use a different subset for testing
    - Use all the other data for training
- Each example is in the test set just once





Method: Leave one out (N-fold cross-validation)

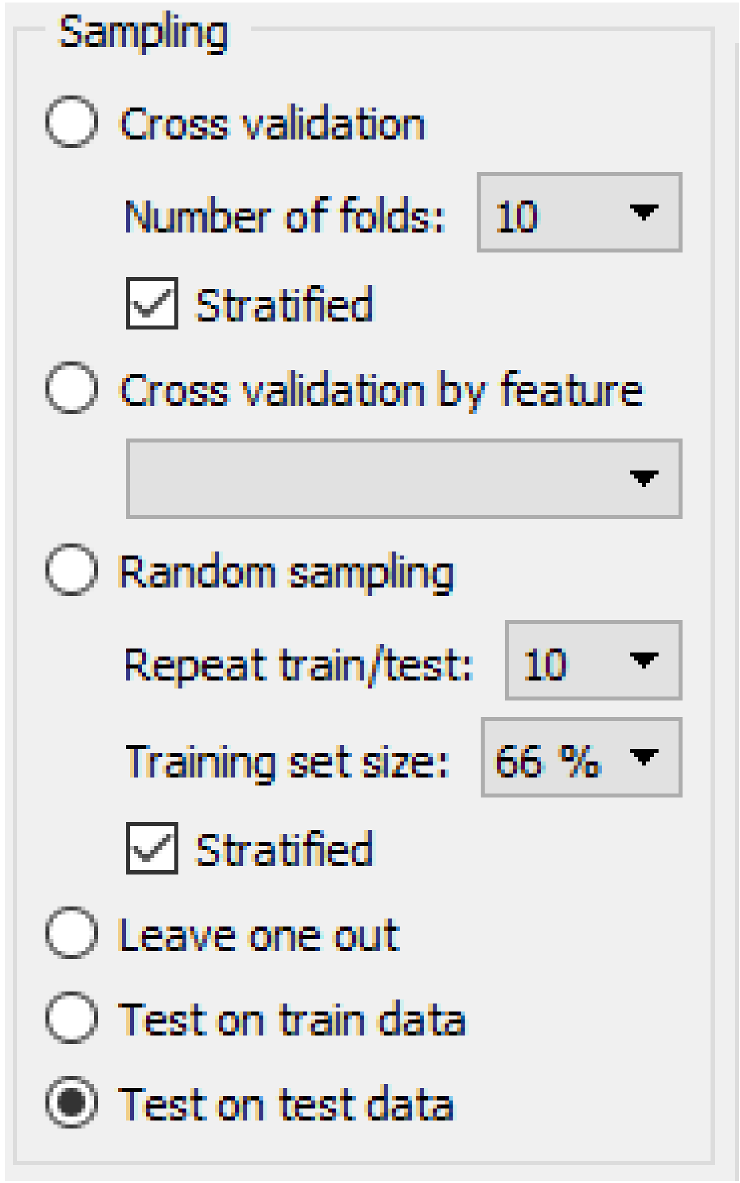
- For small datasets
- Similar to cross validation with test set size =1
- Repeat the training *N*-times if there is *N* examples in the dataset



Evaluation methods in Orange

Test & Score

- Cross validation
- Random sampling
- Leave one out
- Test on train data
- Test on test data



Questions

- What are properties of the results when testing on the training set?

Classification quality measures

Confusion matrix (error matrix)

Breakdown of the classifier's performance, i.e. how frequently instances of class X were correctly classified as class X or misclassified as some other class.

Primer: car

Actual \ Predicted	unacc	acc	good	v-good	Σ
unacc	1154	54	2	0	1210
acc	94	276	14	0	384
good	0	44	22	3	69
v-good	0	25	0	40	65
Σ	1248	399	38	43	1728

Primer: titanic

Actual \ Predicted	no	yes	Σ
no	1364	126	1490
yes	362	349	711
Σ	1726	475	2201

Confusion matrix

- Matrix of correct and incorrect classifications
  - Rows are actual values
  - Columns are predicted values
  - Correct classifications are on the diagonal

Actual \ Predicted	unacc	acc	good	v-good	Σ
unacc	1154	54	2	0	1210
acc	94	276	14	0	384
good	0	44	22	3	69
v-good	0	25	0	40	65
Σ	1248	399	38	43	1728

Confusion matrix for two classes

Predicted

Correct classification	Classified as	
	+	−
+	true positives	false negatives
−	false positives	true negatives

Actual

TP: true positives
The number of positive instances that are classified as positive

FP: false positives
The number of negative instances that are classified as positive

FN: false negatives
The number of positive instances that are classified as negative

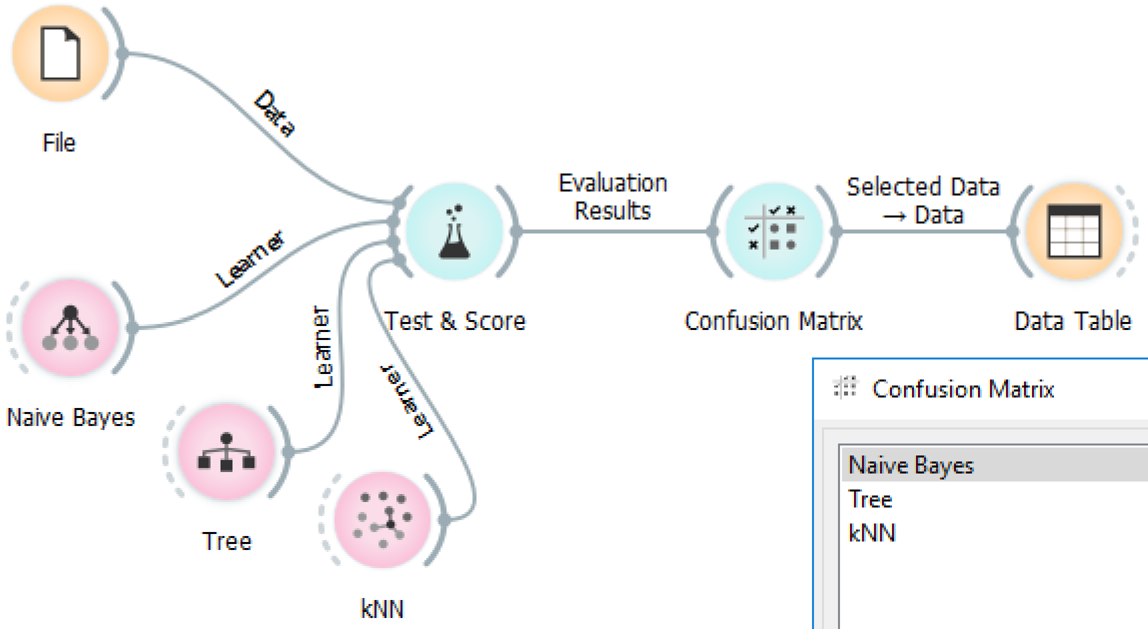
TN: true negatives
The number of negative instances that are classified as negative

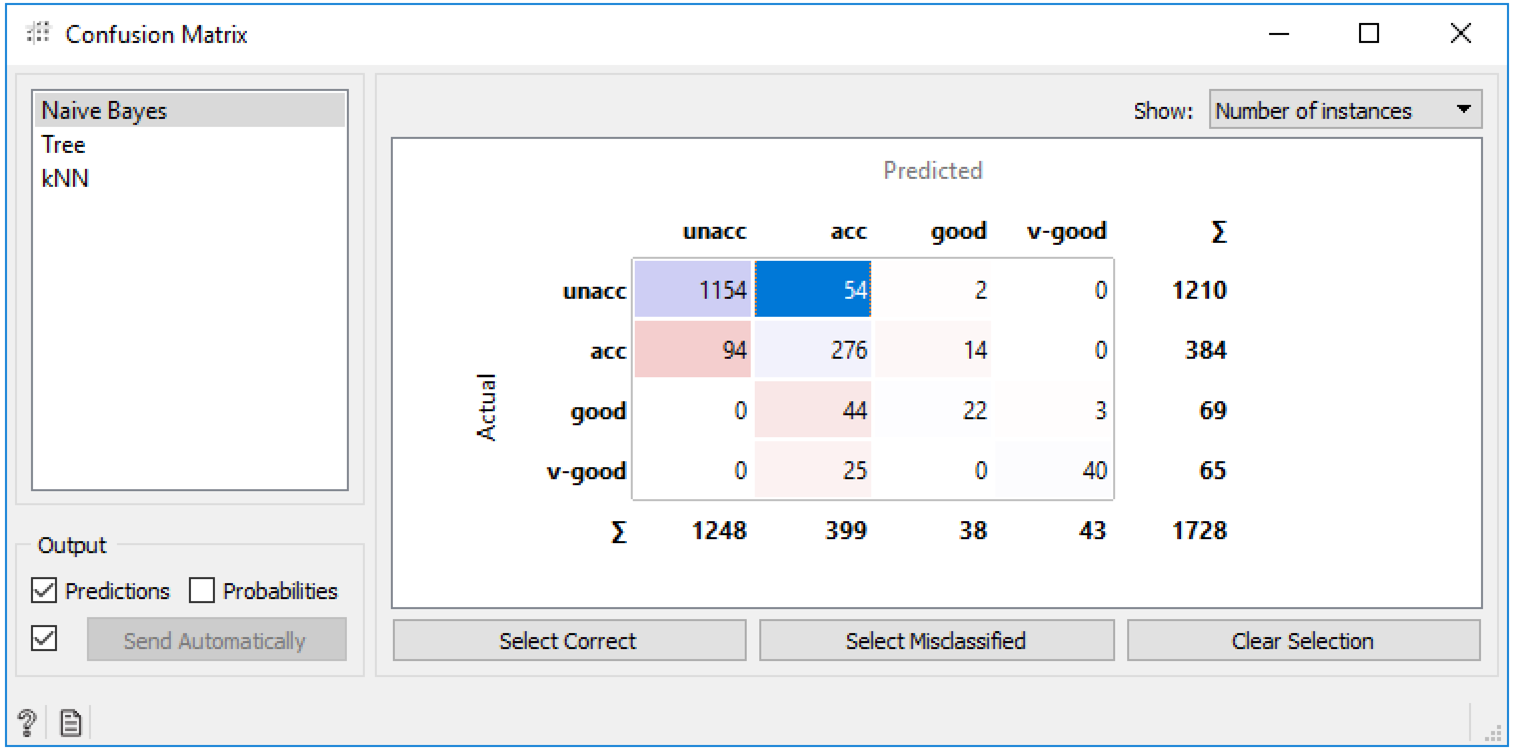
- Diagonal: correct classifications
- Outside: misclassifications
- Classification accuracy =

= |correct classifications| / |all examples|

= |correct classifications| / (|correct classifications| + |misclassifications|)

In Orange, the confusion matrix is interactive





Classification accuracy

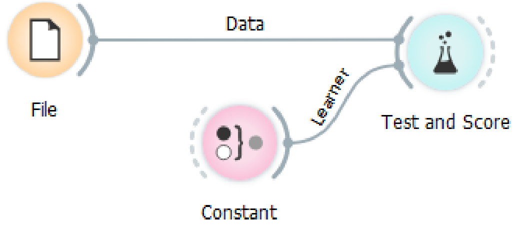
- Percentage of correctly classified examples

Classification accuracy =
= |correct classifications| / |all examples|
= |correct classifications| / (|correct classifications| + |misclassifications|)

Question

- When is classification accuracy “good”?

Majority class classifier (Constant)



Predicted

Actual		unacc	acc	good	v-good	Σ
	unacc	1154	54	2	0	1210
	acc	94	276	14	0	384
	good	0	44	22	3	69
	v-good	0	25	0	40	65
	Σ	1248	399	38	43	1728

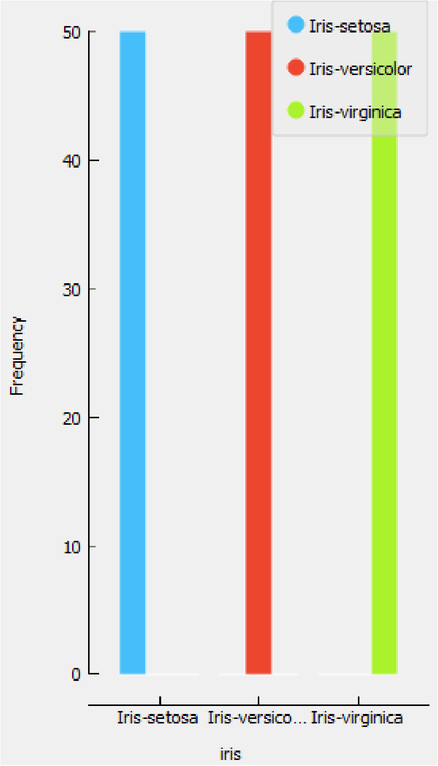
Predicted

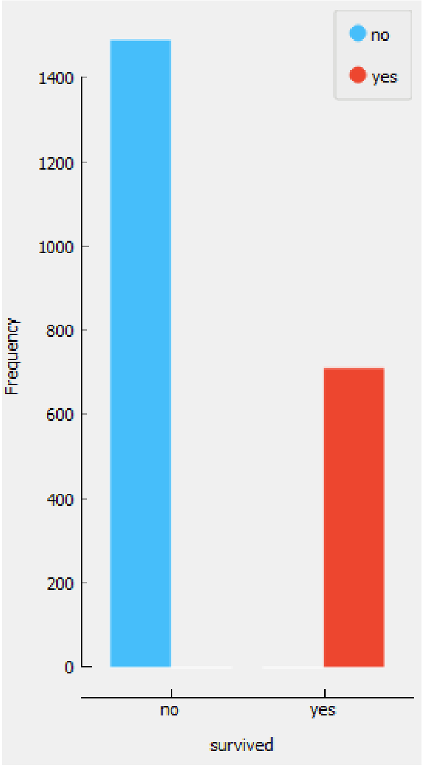
Actual		no	yes	Σ
	no	1364	126	1490
	yes	362	349	711
	Σ	1726	475	2201

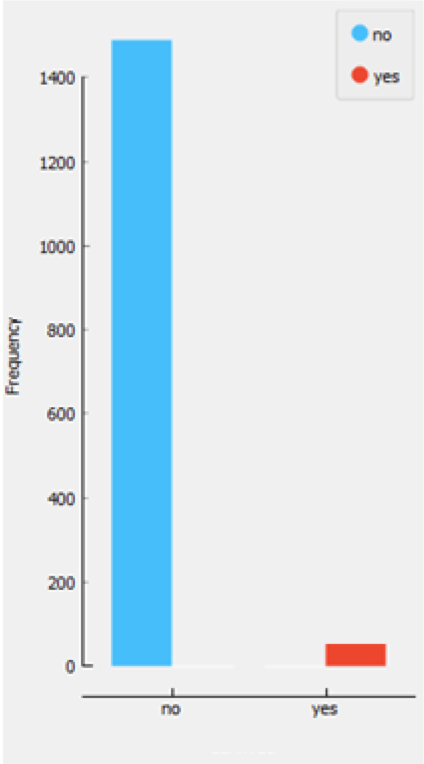
- What is the classification accuracy of a classifier that classifies all the examples in the majority class?
- Car: 70% Titanic: 68%

Imbalanced Data and Unequal Misclassification Costs

- Imbalanced dataset: One class is minority compared to the other(s)
  - The minority class is usually the one of interest







Imbalanced Data and Unequal Misclassification Costs

- Imbalanced dataset: One class is minority compared to the other(s)
  - The minority class is usually the one of interest
- Unequal misclassification costs:
  - Some errors are more costly (have more severe consequences)
- Examples:
  - Screening tests (nuchal scan, Zora, Dora, Svit, …)
  - Intrusion detection
  - Credit card fraud

Exercise: Credit card fraud

„FED report notes the fraud rate for debit and prepaid signature transactions in 2012 was approximately 4.04 basis points (bps), or about ***four per every 10,000 transactions****."*

- What is the classification accuracy of a classifier that classifies all the examples a „not fraudulent"?
  - Answer: 99.96%
- Can a classifier with a 97% accuracy "better" then the one with classification accuracy 99.96%?

https://www.pymnts.com/in-depth/2014/a-tale-of-two-fraud-stats/

Exercise: Credit card fraud

Two confusion matrices for two classifiers

		Predicted		
		Fraud	Not fraud	
Actual	Fraud	0	4	4
	Not fraud	0	9996	9996
		0	10000	10000

		Predicted		
		Fraud	Not fraud	
Actual	Fraud	4	0	4
	Not fraud	300	9696	9996
		304	9696	10000

Classification accuracy

- CA = (0 + 9996)/10000 = 99.96%

- CA = (4 + 9696)/10000 = 97.00%

A model with a worse classification accuracy compared to the majority class is better.

Precision, Recall & F1

- Class-specific metrics
  - Precision (Positive Predictive Value)
    - Proportion of instances classified as positive that are really positive
  - Recall (True Positive Rate, TP Rate, Hit Rate, Sensitivity)
    - The proportion of positive instances that are correctly classified as positive
  - F1
    - Harmonic mean of precision and recall

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

- We can average the metrics over the classes (macro average) or weigh them by the number of examples (micro average)

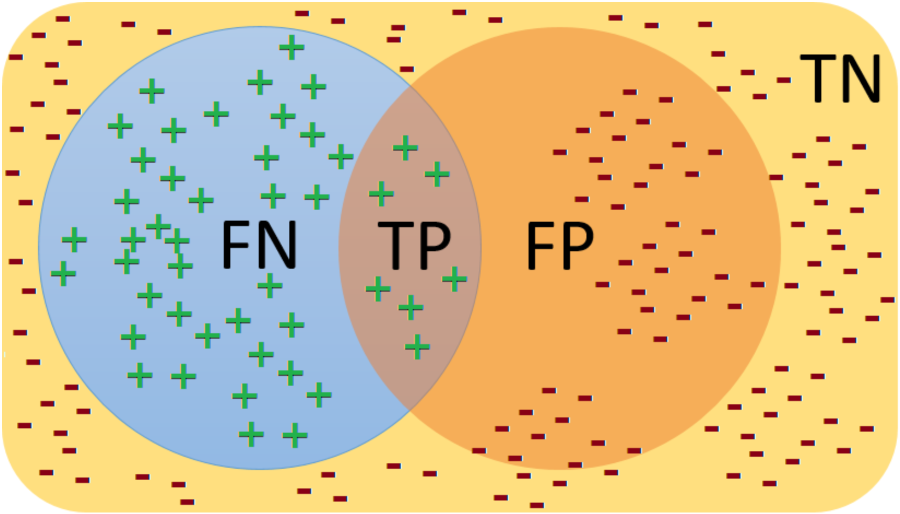
Precision and Recall

PRECISION

- Out of all the examples the classifier labeled as positive, what fraction were correct?

RECALL

- Out of all the positive examples there were, what fraction did the classifier pick up?



Precision, recall, F1

		Predicted class +	Predicted class −	Total instances
Actual class	+	TP	FN	P
	−	FP	TN	N

True Positive Rate or Hit Rate or Recall or Sensitivity or TP Rate	TP/P	The proportion of positive instances that are correctly classified as positive
Precision or Positive Predictive Value	TP/(TP+FP)	Proportion of instances classified as positive that are really positive
F1 Score	$(2 \times \text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$	A measure that combines Precision and Recall
Accuracy or Predictive Accuracy	$(TP + TN)/(P + N)$	The proportion of instances that are correctly classified

- Priklic
- Natančnost
- Mera F1
- Klasifikacijska točnost

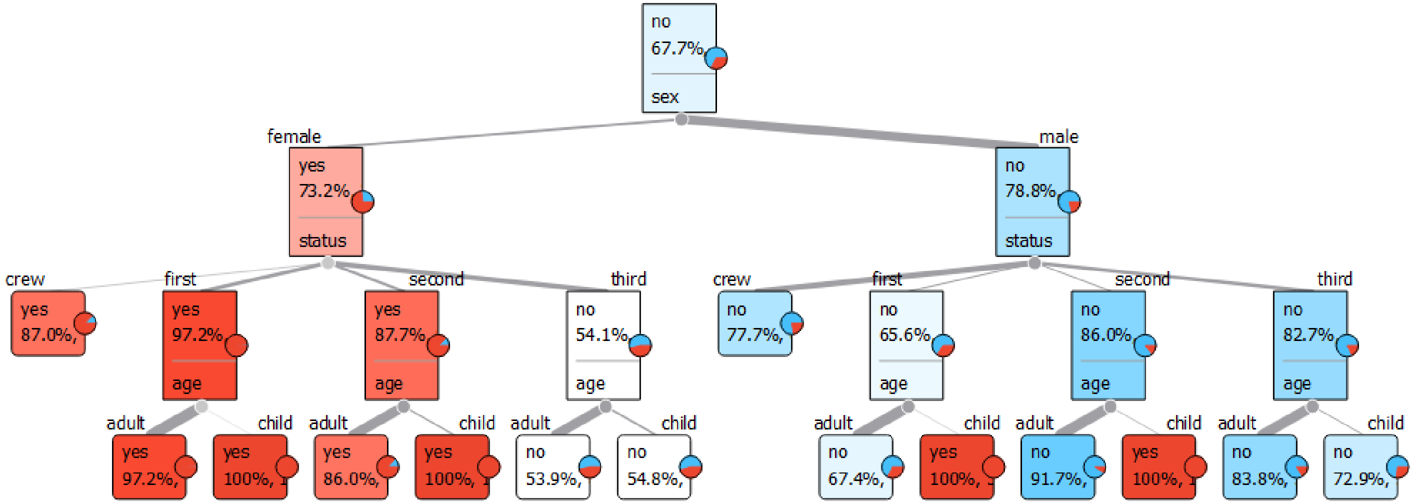
ROC

Bramer (2007), chapter 11: **Measuring the Performance of a Classifier**

Fawcett, Tom. "**An introduction to ROC analysis**." Pattern recognition letters 27.8 (2006): 861-874.

High precision and/or high recall?

- Can we make a model more precise (increase precision)?
- How sure is the model about a certain prediction?
- We can set different thresholds and get different binary classifiers.
- Find a trade-off between precision and recall appropriate for a problem at hand.



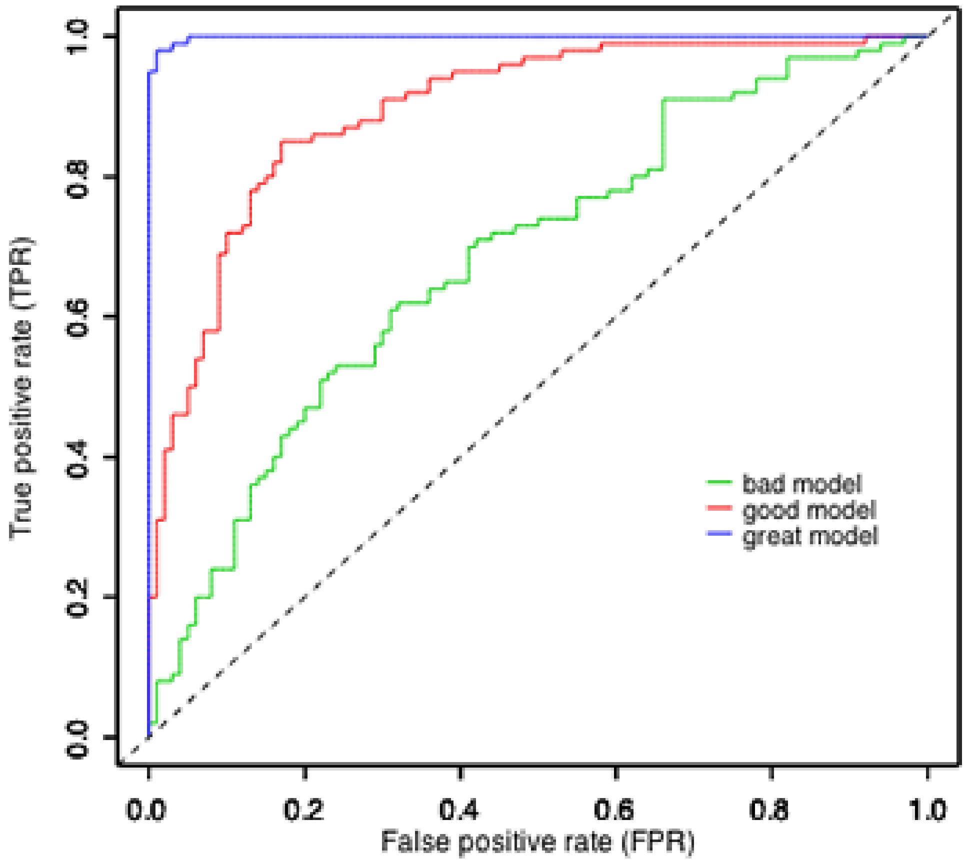
Probabilistic classification

- A **probabilistic** classifier is a classifier that is able to predict, given an observation of an input, a **probability** distribution over a set of classes, rather than only outputting the most likely class that the observation should belong to.
- Ranking
- Tresholds/cutpoints

	Actual class	Confidence classifier forclass Y
P1	Y	1
P2	Y	1
P3	Y	0.95
P4	Y	0.9
P5	Y	0.9
P6	N	0.85
P7	Y	0.8
P8	Y	0.6
P9	Y	0.55
P10	Y	0.55
P11	N	0.3
P12	N	0.25
P13	Y	0.25
P14	N	0.2
P15	N	0.1
P16	N	0.1
P17	N	0.1
P18	N	0
P19	N	0
P20	N	0

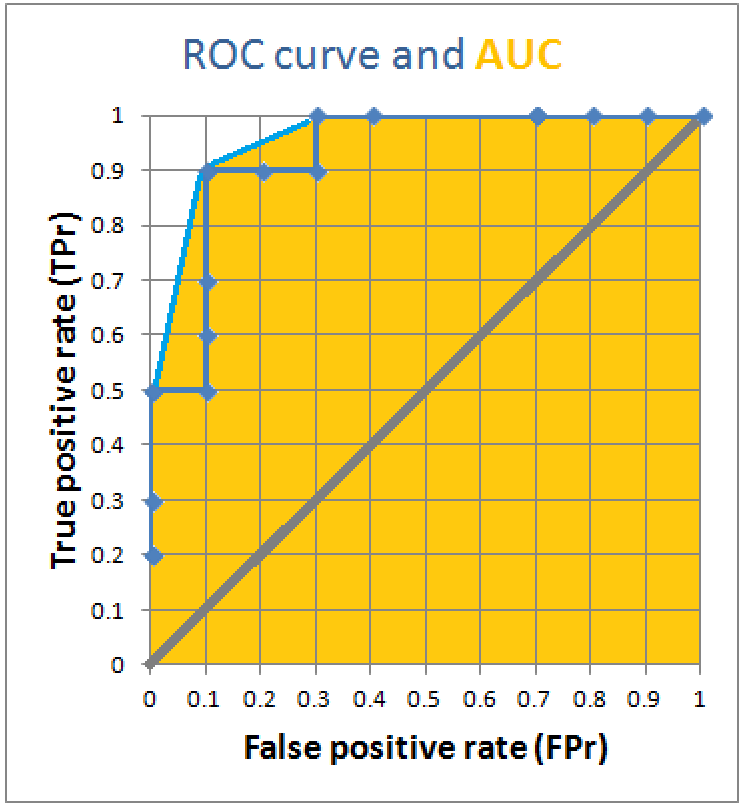
ROC curve and AUC

- **Receiver Operating Characteristic curve** (or ROC curve) is a plot of the true positive rate (TPr=Sensitivity=Recall) against the false positive rate (FPr) for different possible cut-points.
- It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
- The closer the curve to the top left corner, the "better" the classifier.
- The diagonal represents the random classifiers (predicting the positive class with some probability regardless the data).



AUC - Area Under (ROC) Curve

- Performance is measured by the area under the ROC curve (AUC). An area of 1 represents a perfect classifier; an area of 0.5 represents a worthless classifier.
- The area under the curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive example higher than a randomly chosen negative example.

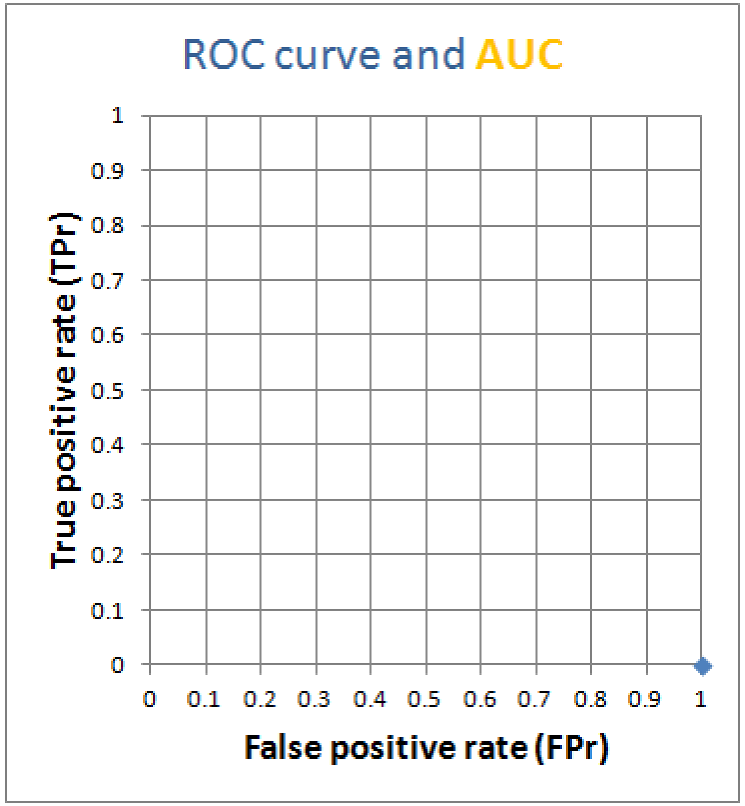


Exercise: ROC curve and AUC

	Actual class	Confidence classifier forclass Y	FP	TP	FPr	TPr
P1	Y	1				
P2	Y	1				
P3	Y	0.95				
P4	Y	0.9				
P5	Y	0.9				
P6	N	0.85				
P7	Y	0.8				
P8	Y	0.6				
P9	Y	0.55				
P10	Y	0.55				
P11	N	0.3				
P12	N	0.25				
P13	Y	0.25				
P14	N	0.2				
P15	N	0.1				
P16	N	0.1				
P17	N	0.1				
P18	N	0				
P19	N	0				
P20	N	0				

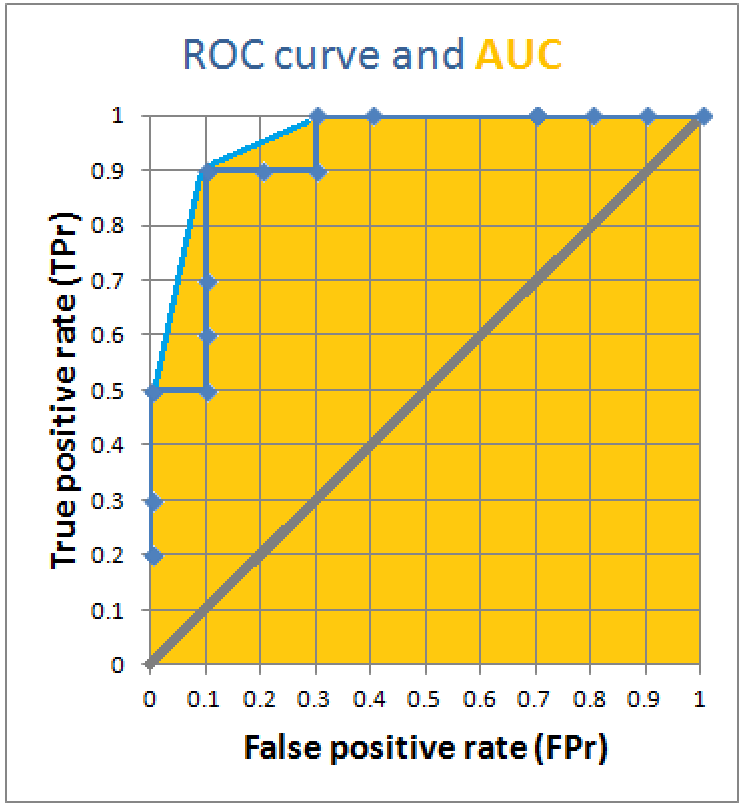
ROC curve and AUC

	Actual class	Classifier confidence forclass Y	FP	TP	FPr	TPr
P1	Y	1	0	2	0	0.2
P2	Y	1	0	2	0	0.2
P3	Y	0.95	0	3	0	0.3
P4	Y	0.9	0	5	0	0.5
P5	Y	0.9	0	5	0	0.5
P6	N	0.85	1	5	0.1	0.5
P7	Y	0.8	1	6	0.1	0.6
P8	Y	0.6	1	7	0.1	0.7
P9	Y	0.55	1	9	0.1	0.9
P10	Y	0.55	1	9	0.1	0.9
P11	N	0.3	2	9	0.2	0.9
P12	N	0.25	3	9	0.3	0.9
P13	Y	0.25	3	10	0.3	1
P14	N	0.2	4	10	0.4	1
P15	N	0.1	7	10	0.7	1
P16	N	0.1	7	10	0.7	1
P17	N	0.1	7	10	0.7	1
P18	N	0	8	10	0.8	1
P19	N	0	9	10	0.9	1
P20	N	0	10	10	1	1



ROC curve and **AUC**

	Actual class	Classifier confidence forclass Y	FP	TP	FPr	TPr
P1	Y	1	0	2	0	0.2
P2	Y	1	0	2	0	0.2
P3	Y	0.95	0	3	0	0.3
P4	Y	0.9	0	5	0	0.5
P5	Y	0.9	0	5	0	0.5
P6	N	0.85	1	5	0.1	0.5
P7	Y	0.8	1	6	0.1	0.6
P8	Y	0.6	1	7	0.1	0.7
P9	Y	0.55	1	9	0.1	0.9
P10	Y	0.55	1	9	0.1	0.9
P11	N	0.3	2	9	0.2	0.9
P12	N	0.25	3	9	0.3	0.9
P13	Y	0.25	3	10	0.3	1
P14	N	0.2	4	10	0.4	1
P15	N	0.1	7	10	0.7	1
P16	N	0.1	7	10	0.7	1
P17	N	0.1	7	10	0.7	1
P18	N	0	8	10	0.8	1
P19	N	0	9	10	0.9	1
P20	N	0	10	10	1	1



Area Under (the convex) Curve
AUC = 0.96

ROC curve properies

- **Universal baselines**: the major diagonal of an ROC plot depicts the line of random performance which can be achieved without training.
- **Linear interpolation**: any point on a straight line between two points representing the performance of two thresholds A and B can be achieved by making a suitably biased random choice between A and B
- **Optimality**: a point D dominates another point E if D’s tpr and fpr are not worse than E’s and at least one of them is strictly better.
- **Area**: the area under the ROC curve (AUROC) estimates the probability that a randomly chosen positive is ranked higher by the model than a randomly chosen negative

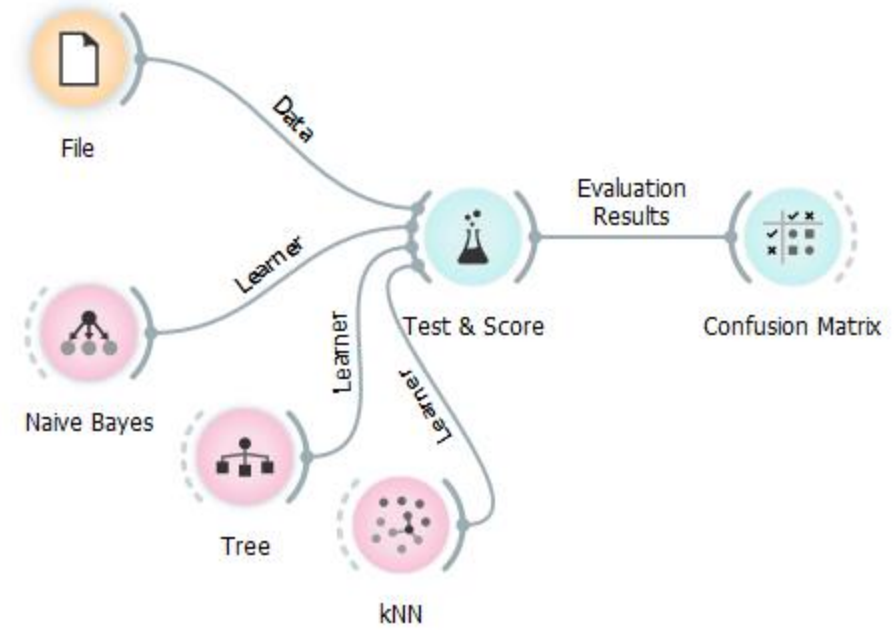
Flach, Peter, and Meelis Kull. "Precision-recall-gain curves: PR analysis done right." *Advances in neural information processing systems*. 2015.

Classification evaluation in Orange

- AUC
  - Area under curve
  - AUROC
  - Površina pod ROC krivuljo
- CA – classification accuracy
  - Klasifikacijska točnost
- F1 – harmonično povprečje priklica in natančnosti
- Precision – natančnost
- Recall - priklic

Evaluation Results

Method	AUC	CA	F1	Precision	Recall
kNN	0.951	0.845	0.823	0.835	0.845
Naive Bayes	0.971	0.863	0.858	0.859	0.863
Tree	0.991	0.951	0.951	0.951	0.951



https://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

sklearn.metrics: Metrics ¶

metrics.accuracy_score(y_true, y_pred[, …])	Accuracy classification score.
metrics.auc(x, y)	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score(y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.balanced_accuracy_score(y_true, y_pred)	Compute the balanced accuracy
metrics.brier_score_loss(y_true, y_prob[, …])	Compute the Brier score.
metrics.classification_report(y_true, y_pred)	Build a text report showing the main classification metrics
metrics.cohen_kappa_score(y1, y2[, labels, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix(y_true, y_pred[, …])	Compute confusion matrix to evaluate the accuracy of a classification.
metrics.dcg_score(y_true, y_score[, k, …])	Compute Discounted Cumulative Gain.
metrics.f1_score(y_true, y_pred[, labels, …])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score(y_true, y_pred, beta[, …])	Compute the F-beta score
metrics.hamming_loss(y_true, y_pred[, …])	Compute the average Hamming loss.
metrics.hinge_loss(y_true, pred_decision[, …])	Average hinge loss (non-regularized)
metrics.jaccard_score(y_true, y_pred[, …])	Jaccard similarity coefficient score
metrics.log_loss(y_true, y_pred[, eps, …])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef(y_true, y_pred[, …])	Compute the Matthews correlation coefficient (MCC)
metrics.multilabel_confusion_matrix(y_true, …)	Compute a confusion matrix for each class or sample
metrics.ndcg_score(y_true, y_score[, k, …])	Compute Normalized Discounted Cumulative Gain.
metrics.precision_recall_curve(y_true, …)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support(…)	Compute precision, recall, F-measure and support for each class
metrics.precision_score(y_true, y_pred[, …])	Compute the precision
metrics.recall_score(y_true, y_pred[, …])	Compute the recall
metrics.roc_auc_score(y_true, y_score[, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve(y_true, y_score[, …])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_loss(y_true, y_pred[, …])	Zero-one classification loss.

21 measures of accuracy from scikit-learn documentation for Classification problems

Some of these are restricted to the binary classification case:

precision_recall_curve (y_true, probas_pred)	Compute precision-recall pairs for different probability thresholds
roc_curve (y_true, y_score[, pos_label, …])	Compute Receiver operating characteristic (ROC)
balanced_accuracy_score (y_true, y_pred[, …])	Compute the balanced accuracy

Others also work in the multiclass case:

cohen_kappa_score (y1, y2[, labels, weights, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.
confusion_matrix (y_true, y_pred[, labels, …])	Compute confusion matrix to evaluate the accuracy of a classification
hinge_loss (y_true, pred_decision[, labels, …])	Average hinge loss (non-regularized)
matthews_corrcoef (y_true, y_pred[, …])	Compute the Matthews correlation coefficient (MCC)

Some also work in the multilabel case:

accuracy_score (y_true, y_pred[, normalize, …])	Accuracy classification score.
classification_report (y_true, y_pred[, …])	Build a text report showing the main classification metrics
f1_score (y_true, y_pred[, labels, …])	Compute the F1 score, also known as balanced F-score or F-measure
fbeta_score (y_true, y_pred, beta[, labels, …])	Compute the F-beta score
hamming_loss (y_true, y_pred[, labels, …])	Compute the average Hamming loss.
jaccard_score (y_true, y_pred[, labels, …])	Jaccard similarity coefficient score
log_loss (y_true, y_pred[, eps, normalize, …])	Log loss, aka logistic loss or cross-entropy loss.
multilabel_confusion_matrix (y_true, y_pred)	Compute a confusion matrix for each class or sample
precision_recall_fscore_support (y_true, y_pred)	Compute precision, recall, F-measure and support for each class
precision_score (y_true, y_pred[, labels, …])	Compute the precision
recall_score (y_true, y_pred[, labels, …])	Compute the recall
zero_one_loss (y_true, y_pred[, normalize, …])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

average_precision_score (y_true, y_score[, …])	Compute average precision (AP) from prediction scores
roc_auc_score (y_true, y_score[, average, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

21 measures of accuracy from scikit-learn documentation for Classification problems

Félix Revert: The proper way to use Machine Learning metrics
https://towardsdatascience.com/the-proper-way-to-use-machine-learning-metrics-4803247a2578

Home assignment

Model complexity (e.g. number of leafs) vs. accuracy on train and test set

Datasets:

- A-greater-then-B.csv
- Another reasonably sized classification dataset from http://file.biolab.si/datasets/

You can start from the samples of code from the gitlab repository http://source.ijs.si/pkraljnovak/DM_course