Exercise 1

CLASSIFIER 1

column order changed

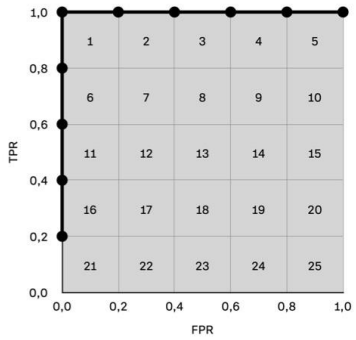
ID	Class	Prob.	TP	FP	TN	FN	FPR	TPR
1	P	0,90	1	0	5	4	0	0,2
2	P	0,80	2	0	5	3	0	0,4
3	P	0,70	3	0	5	2	0	0,6
4	P	0,60	4	0	5	1	0	0,8
5	P	0,55	5	0	5	0	0	1
6	N	0,20	5	1	4	0	0,2	1
7	N	0,20	5	2	3	0	0,4	1
8	N	0,19	5	3	2	0	0,6	1
9	N	0,19	5	4	1	0	0,8	1
10	N	0,10	5	5	0	0	1	1

Confusion Matrix	Predicted P	Predicted N	Total
Actual P	5	0	5
Actual N	0	5	5
Total	5	5	10

For the confusion matrix, assume that an element is predicted P when the probability is ≥ 0,5 and N otherwise

Calculations	Formula	Solution
Accuracy	$\frac{TP+TN}{P+N}$	1
Precision	$\frac{TP}{TP+FP}$	1
Recall	$\frac{TP}{P}$	1
F_1-measure	$\frac{2 \times precision \times recall}{precision + recall}$	1

AUC = 25 / 25 = 1



CLASSIFIER 2

column order changed

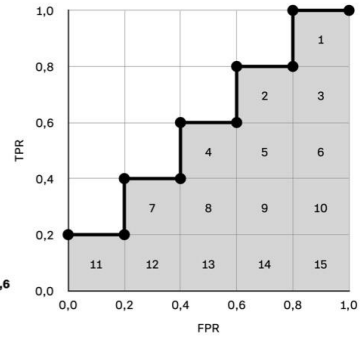
ID	Class	Prob.	TP	FP	TN	FN	FPR	TPR
1	P	0,90	1	0	5	4	0	0,2
2	N	0,85	1	1	4	4	0,2	0,2
3	P	0,70	2	1	4	3	0,2	0,4
4	N	0,60	2	2	3	3	0,4	0,4
5	P	0,55	3	2	3	2	0,4	0,6
6	N	0,48	3	3	2	2	0,6	0,6
7	P	0,40	4	3	2	1	0,6	0,8
8	N	0,35	4	4	1	1	0,8	0,8
9	P	0,30	5	4	1	0	0,8	1
10	N	0,10	5	5	0	0	1	1

Confusion Matrix	Predicted P	Predicted N	Total
Actual P	3	2	5
Actual N	2	3	5
Total	5	5	10

For the confusion matrix, assume that an element is predicted P when the probability is ≥ 0,5 and N otherwise

Calculations	Formula	Solution
Accuracy	$\frac{TP+TN}{P+N}$	0,6
Precision	$\frac{TP}{TP+FP}$	0,6
Recall	$\frac{TP}{P}$	0,6
F_1-measure	$\frac{2 \times precision \times recall}{precision + recall}$	0,6

AUC = 15 / 25 = 0,6



CLASSIFIER 3

column order changed

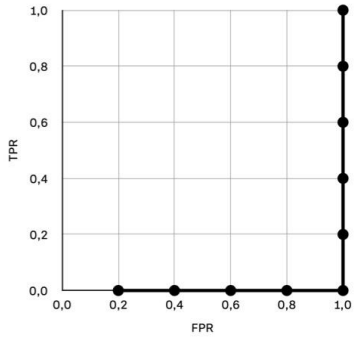
ID	Class	Prob.	TP	FP	TN	FN	FPR	TPR
1	N	0,90	0	1	4	5	0,2	0
2	N	0,85	0	2	3	5	0,4	0
3	N	0,70	0	3	2	5	0,6	0
4	N	0,60	0	4	1	5	0,8	0
5	N	0,55	0	5	0	5	1	0
6	P	0,48	1	5	0	4	1	0,2
7	P	0,40	2	5	0	3	1	0,4
8	P	0,35	3	5	0	2	1	0,6
9	P	0,30	4	5	0	1	1	0,8
10	P	0,20	5	5	0	0	1	1

Confusion Matrix	Predicted P	Predicted N	Total
Actual P	0	5	5
Actual N	5	0	5
Total	5	5	10

For the confusion matrix, assume that an element is predicted P when the probability is ≥ 0,5 and N otherwise

Calculations	Formula	Solution
Accuracy	$\frac{TP+TN}{P+N}$	0
Precision	$\frac{TP}{TP+FP}$	0
Recall	$\frac{TP}{P}$	0
F_1-measure	$\frac{2 \times precision \times recall}{precision + recall}$	*undefined*

AUC = 0 / 25 = 0

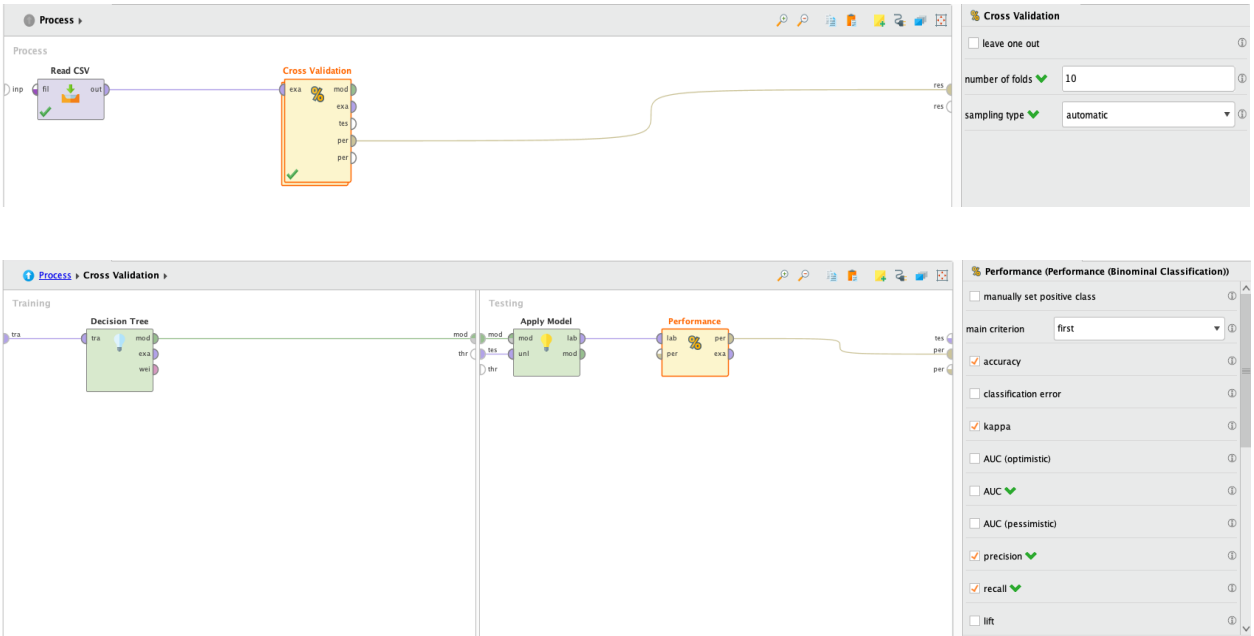


Area under the curve (AUC) calculated as the proportion of the squares it contains.

Exercise 2: Evaluation of decision trees - 1
We will use again the well-known titanic dataset from:
https://github.com/datasciencedojo/datasets/blob/master/titanic.csv
The aim is to predict whether a person will survive the accident with the algorithm decision tree and to evaluate the model.

Run the decision tree algorithm. Perform a 10-fold cross-validation. Report the confusion matrix. Use the formulas of the book / slides and calculate yourself: Accuracy, Kappa, Precision, Recall. Show your calculations.
Are your results better than a simple algorithm that would always predict “will not survive”? Justify your answer.

RapidMiner: The operator Cross Validation in RapidMiner is a nested operator, see the file “MiniTutorial” in Day 1 or the video “Validating a model | RapidMiner”:
https://www.youtube.com/watch?v=Hw3tVDZ4Pmo&list=PLssWC2d9JhOZLbQNZ80uOxLypglgWqbJA&index=15
Choose in Validation / Performance / Predictive the Performance
(Binomial Classifier) operator. In the parameters of this operator, tick the box “manually set positive class” and give “1” has positive class.



In order to execute the Cross Validation we need to set our selected class as “label” and change the type to Binominal.

Results reported from RapidMiner:

1. *Correlation Matrix:*

	true 0	true 1	class precision
pred. 0	471	108	81.35%
pred. 1	78	234	75.00%
class recall	85.79%	68.42%	

2. *Accuracy:*

   accuracy: 79.12% +/- 3.95% (micro average: 79.12%)

3. *Kappa:*

   kappa: 0.552 +/- 0.081 (micro average: 0.551)

4. *Precision:*

   precision: 76.01% +/- 8.14% (micro average: 75.00%) (positive class: 1)

5. *Recall:*

   recall: 68.45% +/- 8.06% (micro average: 68.42%) (positive class: 1)

Calculations:

1. *Correlation Matrix:*

	Pred 0	Pred 1	Total
True 0	471 (TP)	78 (FN)	549 (P)
True 1	108 (FP)	234 (TN)	342 (N)
Total	579 (P')	312 (N')	891 (P+N)

2. *Accuracy:*

   $Accuracy,\ recognition\ rate$: $\frac{TP+TN}{P+N} = \frac{471+234}{891} = 0.7912 = 79.12\,\%$

3. *Kappa:*

   $\boldsymbol{\kappa} = \frac{P_0 - P_C}{1 - P_C}$

   Number of observations: 891

   P_0: $Observed\ agreement$

   $P_0 = \frac{471+234}{891} = 0.79$

P_C: $Random\ agreement$

a. The chance that prediction and Data agree on “will not survive”:
$\frac{579}{891} * \frac{549}{891} = 0.4004$
b. The chance that prediction and Data agree on “will survive”:
$\frac{312}{891} * \frac{342}{891} = 0.1344$

$P_C = 0.4004 + 0.1344 = 0.5348$

$\boldsymbol{\kappa} = \frac{0.79-0.5348}{1-0.5348} = 0.5485$

4. *Precision:*

$Precision$: $\frac{TP}{TP+FP} = \frac{471}{471+108} = 0.8134 = 81.34\,\%$

5. *Recall:*

$Recall$: $\frac{TP}{P} = \frac{471}{549} = 0.8579 = 85.79\,\%$

Are your results better than a simple algorithm that would always predict “will not survive”? Justify your answer.

$Accuracy\ of\ "will\ not\ survive"$: $\frac{549}{891} = 0.6161 = \%61.61$

Since the accuracy of our algorithm is higher than the accuracy of a simple algorithm that would always predict “will not survive”, then we can say that our algorithm has better results and is more accurate and efficient.

Exercice 3: Evaluation of decision trees - 2

We repeat exercise 2 with another dataset: https://archive.ics.uci.edu/ml/datasets/Hepatitis .
The files are also in Moodle. The aim is to predict whether a person will die from hepatitis with the algorithm decision tree and to evaluate the model. Before running the classification task, explore the data and report exactly how you are handling missing values. This is here an important issue since the dataset does not have that many elements.
Run the decision tree algorithm. Perform a 10-fold cross-validation. Report the confusion matrix and the values for Accuracy, Kappa, Precision, Recall and the F1-Measure.
Are your results better than a simple algorithm that would always predict “LIVE”? Justify your Answer.

Data cleaning:

Missing Attribute Values: (indicated by "?")

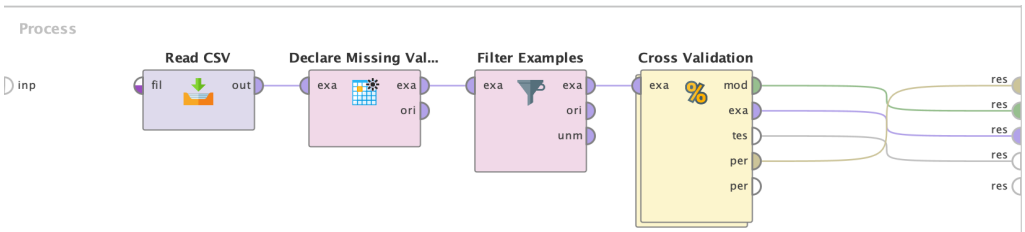
Attribute Number:	Number of Missing Values:
1:	0
2:	0
3:	0
4:	1
5:	0
6:	1
7:	1
8:	1
9:	10
10:	11
11:	5
12:	5
13:	5
14:	5
15:	6
16:	29
17:	4
18:	16
19:	67
20:	0

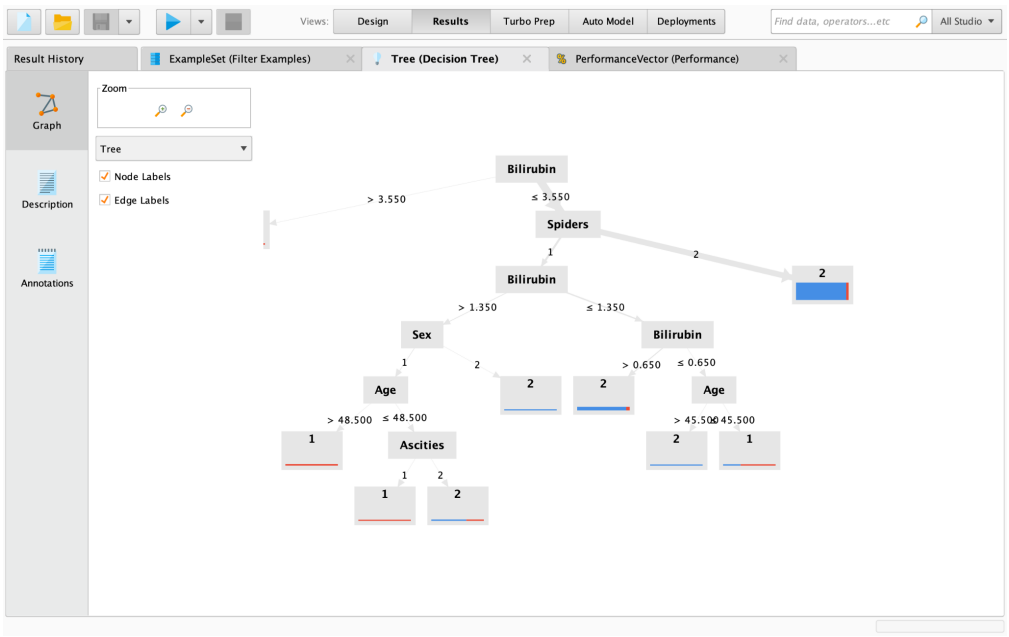
In order to process the decision tree algorithm, we need first to handle the missing values.
One possibility is to replace the missing values with mean or median.
The mean and median can only be calculated on numerical variables, therefore, these methods are suitable for continuous and discrete numerical variables only, which is not the case for all of our attributes, and has also some limitations:

- Distortion of the original variable distribution.
- Distortion of the original variance.

The best we can do in this case is to delete irrelevant attributes, where there are a lot of missing values, and then we delete some rows. The goal is to end up with a higher percentage of the dataset, in our case 80% of the original.
Missing Attribute Values: (indicated by "?")

Attribute Number:	Number of Missing Values:
9:	10
10:	11
16:	29
18:	16
19:	67





1. *Correlation Matrix:*

	true 2	true 1	class precision
pred. 2	100	11	90.09%
pred. 1	7	7	50.00%
class recall	93.46%	38.89%	

2. *Accuracy:*

   accuracy: 85.51% +/- 8.39% (micro average: 85.60%)

3. *Kappa:*

   kappa: 0.309 +/- 0.392 (micro average: 0.356)

4. *Precision:*

   precision: 50.00% (positive class: 1)

5. *Recall:*

   recall: 35.00% +/- 33.75% (micro average: 38.89%) (positive class: 1)

6. F-Measure:

   f_measure: 43.75% (positive class: 1)

Are your results better than a simple algorithm that would always predict "LIVE"? Justify your Answer.

Class Distribution:

DIE: 18
LIVE: 107

$Accuracy\ of\ "live"$: $\frac{107}{125} = 0.856 = \%85.6$

If we compare the accuracy of our decision tree's algorithm and the simple algorithm that would always predict "LIVE", we can't determine if the results of our algorithm is better, since the accuracy of both are the same.
In this case, we can have a look at the value of Kappa.
Cohen's Kappa is always less than or equal to 1. Values of 0 or less, indicate that the classifier is useless. According to Landis and Koch, a kappa value under 0 is indicating no agreement, 0..0,20 as slight, 0,01..0,40 as fair, 0,41..0,60 as moderate, 0,61..0,80 as substantial, and 0,81..1 as almost perfect agreement.
So if we take a look at our Kappa value (0,309), we can tell that our decision tree's algorithm is performing better than a simple algorithm, but maybe not better than other algorithms, e.g the neuronal network algorithm.

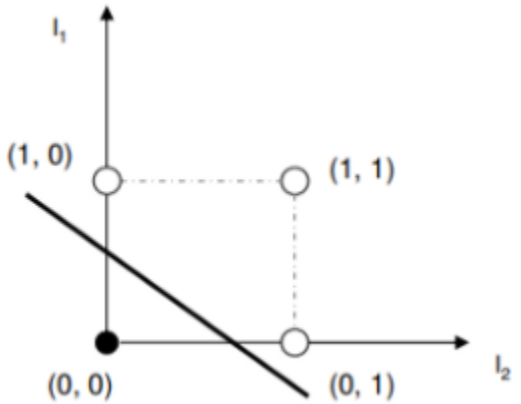
Kappa: how much better your classifier is performing over the performance of a classifier that simply guesses at random according to the frequency of each class.

Exercise 4:

Give a neural network as small as possible to calculate the Boolean OR operator.

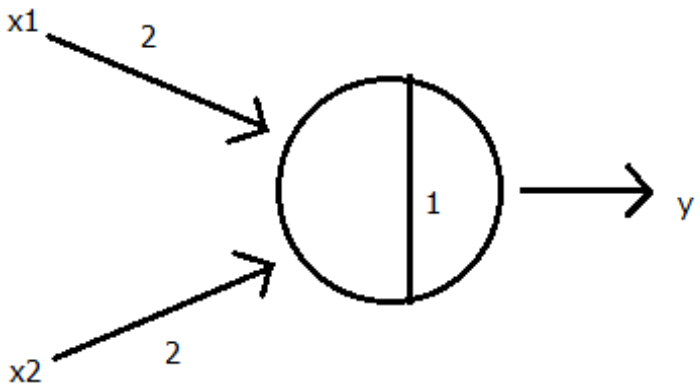
With an OR-Gate, if one or more inputs are boolean 1, then the output is a boolean 1. Only if all inputs are boolean 0, the output is a boolean 0.

The OR operator can be represented by a single neuron. The input layer has a 0 or 1 as inputs. The output layer contains the threshold. If $x_1 w_1 + x_2 w_2 > h$, the output will be 1. If $x_1 w_1 + x_2 w_2 <= h$, the output will be 0.



Source: https://medium.com/@lucaspereira0612/solving-xor-with-a-single-perceptron-34539f395182

The smallest OR-Operator Neural Network looks as follows:



Calculation for Neural Network for OR-Operator:

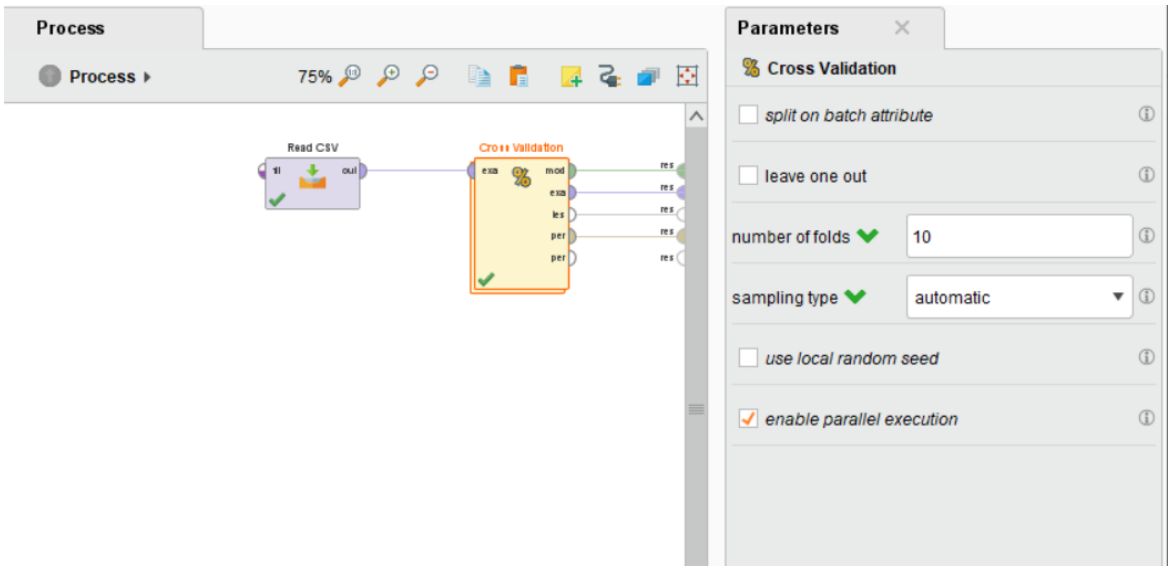
```python
import numpy as np

class FormalNeuronLayer:
  def __init__(self, w, h):
    self.w = w
    self.h = h

  def forward(self, x):
    p = np.dot(x, self.w)
    y = p > self.h
    return y.astype(np.int)

x = np.array([[0,0],
              [0,1],
              [1,0],
              [1,1]])
w = np.array([[2],
              [2]])
h = np.array([1])

formalNeuron1 = FormalNeuronLayer(w,h)
y = formalNeuron1.forward(x)
print(x)
print(y)
```

```
Inputs:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
Outputs:
[[0]
 [1]
 [1]
 [1]]
```

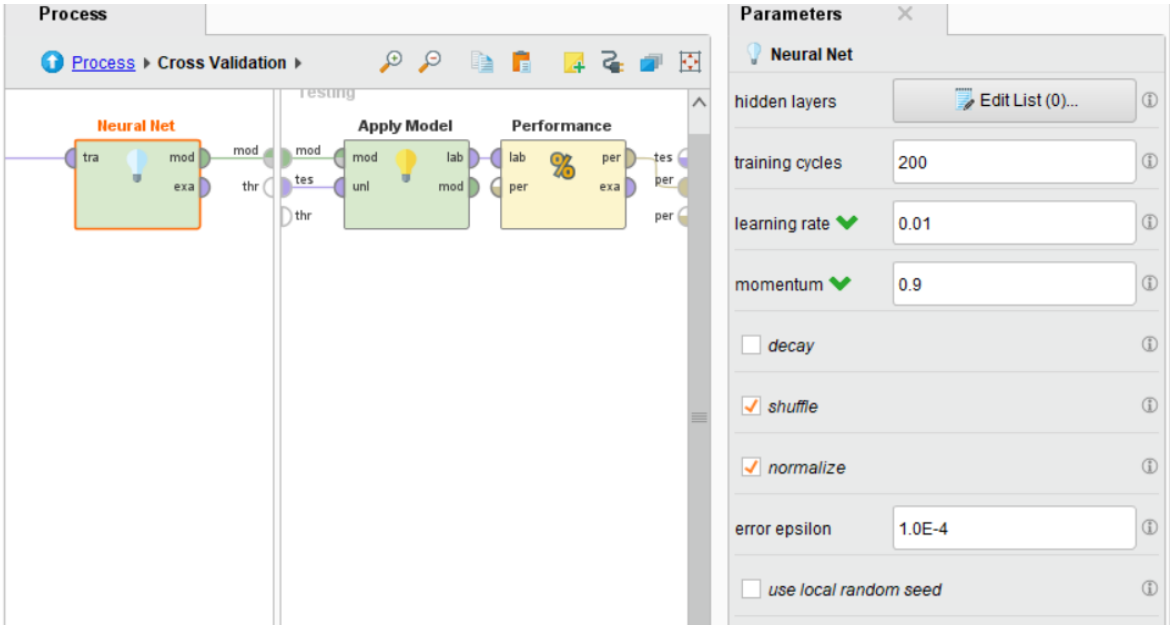
Exercise 5:

Repeat exercise 3 with a neural network instead of a decision tree. For the neural network, use the default values (learning rate, number of layers, number of neurons, activation function etc.) provided by your software.

In order to execute the cross validation we need to set our selected class as “label” and change the type to binominal. 1 is ‘no’ or ‘die’, 2 is ‘yes’ or ‘live’. For the neural network Operator on RapidMiner, all values need to be numeric.



Current default values for neural network:



Results reported from RapidMiner:

7. *Correlation Matrix:*

	true 2	true 1	class precision
pred. 2	103	8	92.79%
pred. 1	4	10	71.43%
class recall	96.26%	55.56%	

8. *Accuracy:*

   accuracy: 90.32% +/- 10.08% (micro average: 90.40%)

9. *Kappa:*

   kappa: 0.533 +/- 0.489 (micro average: 0.571)

10. *Precision:*

    precision: 65.00% +/- 47.43% (micro average: 71.43%) (positive class: 1)

11. *Recall:*

    recall: 55.00% +/- 43.78% (micro average: 55.56%) (positive class: 1)

12. F-Measure:

    f_measure: 52.94% (positive class: 1)

Conclusion:
In comparison to the decision tree, the results are clearly better with the neural network. The accuracy is 6% lower, as well as Kappa (which guesses how much better this classifier performs over a simple classifier which guesses by frequency of the class). Both precision and recall are higher and the combined f_measure has a better value.

Exercise 5:

Finally, we will automatize the search for the **best learning rate** of the neural network with a grid-search. A grid search means that one or more hyper-parameters take different values as specified in a loop (a nested loop if there is more than one parameter). For each value of the parameter (**or each set of values if there are more than one parameter), a cross validation** is performed. At the end of the grid-search, the **value of the hyper-parameter(s) with the best evaluation is returned**. The evaluation-criterion (accuracy, f-measure, recall etc.) has to be chosen. Conduct **two grid-searches**. In the first search, select **accuracy** as the evaluation criterion to maximize. This means that the measure to optimize is accuracy and the model that optimizes this measure will be returned. In the second search use **recall**, assuming that **“DIE” is the positive value**. Explain and report why recall is an important measure in this case. Report your results.

Research on neural networks learning rate

- The learning rate ranges between 0.0 and 1.0
- Too small value returns a more optimal set of weights but can take longer to train
- High value trains the model faster, but affects the accuracy of weights negatively
- Weight is not updated at full amount of error, but is scaled by the learning rate
- Amount of error estimated at backpropagation is multiplied to learning rate
- It is common to grid search learning rates on a log scale from 0.1 to 10^-5 or 10^-6.

Best learning rate of the neural network with grid-search

The RapidMiner operator for optimizing parameters will produce a loop for the learning rate between 0 and 1, incremented by 10. The learning rate with the best result for the selected performance criterion will be applied to update the weights.

1. **Accuracy**

The accuracy determines how many test set tuples were correctly classified.

accuracy: 86.35% +/- 6.51% (micro average: 86.40%)

	Pred 2	Pred 1	Total
True 2	99	8	107
True 1	9	9	18
Total	108	17	125

86,35 % of the test set tuples were correctly classified.

All iterations and learning rates between 0 and 1 ascending :

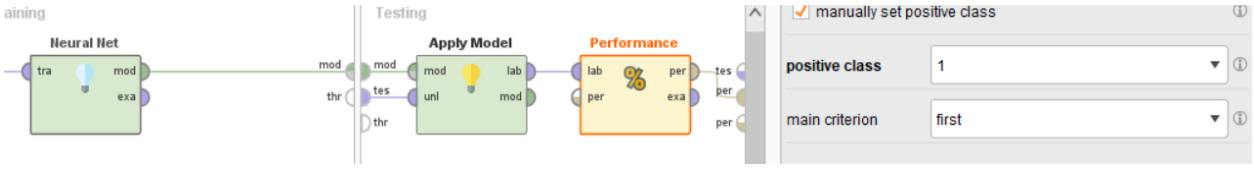
iteration	Neural Net.learning_rate ↑	accuracy
1	0	0.446
2	0.100	0.848
3	0.200	0.832
4	0.300	0.855
5	0.400	0.863
6	0.500	0.838
7	0.600	0.823
8	0.700	0.833
9	0.800	0.832
10	0.900	0.855
11	1	0.856

The accuracy at iteration 5 is the highest and best learning rate at 0.4 hence it was ultimately used for the cross validation.

```
Parameter set:

Performance:
PerformanceVector [
-----accuracy: 86.35% +/- 6.51% (micro average: 86.40%)
ConfusionMatrix:
True:    2       1
2:       99      9
1:       8       9
]
Neural Net.learning_rate        = 0.4
```

2. Recall

Set “1” or “DIE” to positive value



Result based on recall:

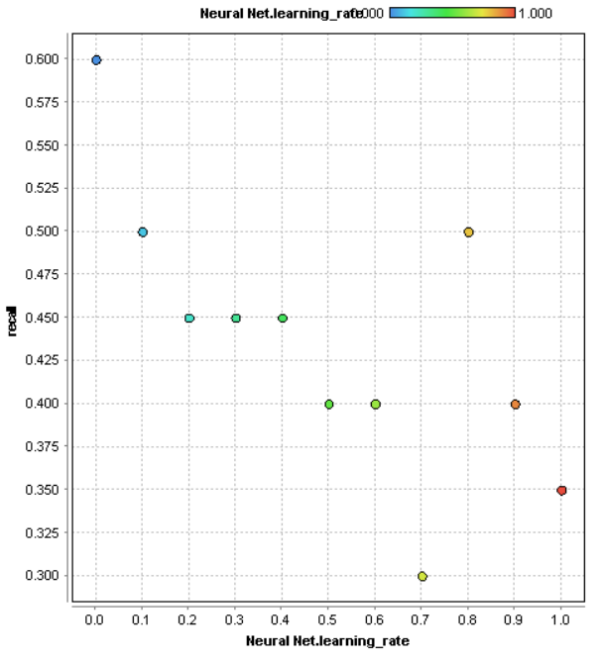
- Completeness (what percentage of positive tuples are labeled positive?)

	Pred Live	Pred Die	Total
True Live	43 (TN)	6 (FP)	49 (N)
True Die	64 (FN)	12 (TP)	76 (P)
Total	107 (N')	18 (P')	125 (P+N)

$Recall$: $\frac{TP}{P} = \frac{12}{76} = 0.1579 = 15,79\,\%$

Here the 'Die' Class is the main class of interest. 6 negative 'Live' tuples have been predicted as positive (FP). 64 are positive 'Die' tuples classified falsely as negatives (FN).

iteration	Neural Net.learning_rate ↑	recall
1	0	0.600
2	0.100	0.500
3	0.200	0.450
4	0.300	0.450
5	0.400	0.450
6	0.500	0.400
7	0.600	0.400
8	0.700	0.300
9	0.800	0.500
10	0.900	0.400
11	1	0.300



At learning rate 0 the recall is the highest hence it was ultimately used for the cross validation. The odd result might signify that the highest recall can only be achieved with the lowest possible learning rate, which can give the optimum set of weights.

ParameterSet

```
Parameter set:

Performance:
PerformanceVector [
-----recall: 60.00% +/- 51.64% (micro average: 66.67%) (positive class: 1)
ConfusionMatrix:
True:   2       1
2:      43      6
1:      64      12
]
Neural Net.learning_rate        = 4.9E-324
```

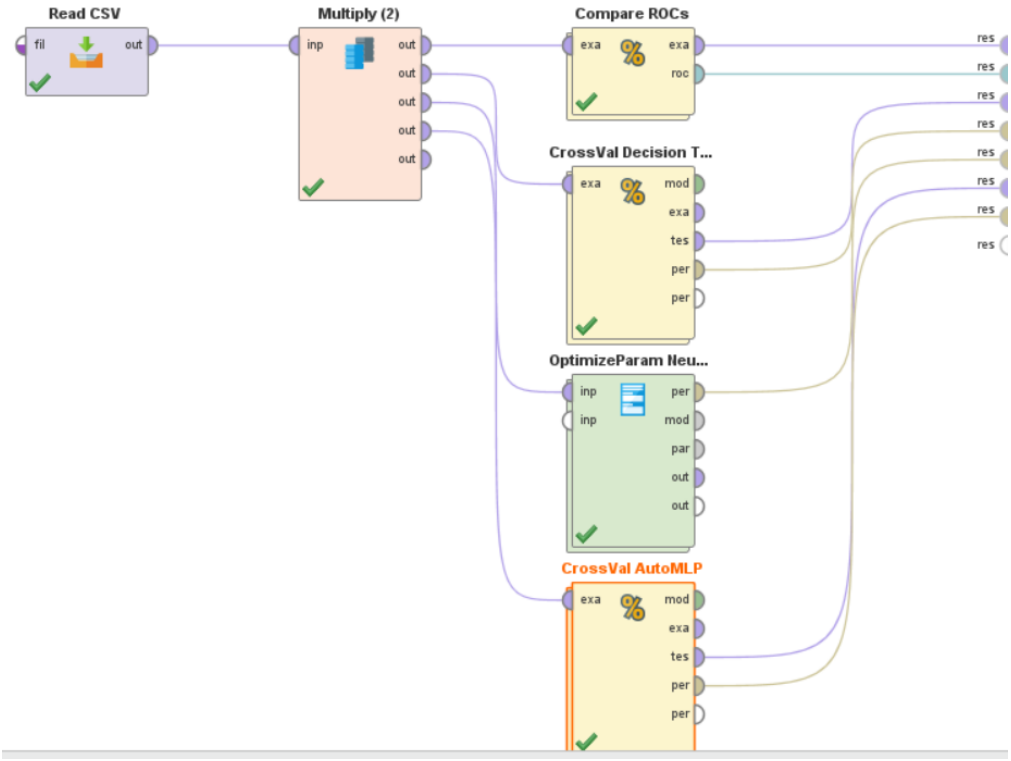
Why recall matters:

The accuracy measure predicts how accurate the prediction is in total, but it doesn't say whether the accuracy is evenly distributed on both class outcomes. If there are 85% correct outcomes, it is possible that the 85% were correct for only the more common label attribute. In this case, the missing 15% could be crucial and the result unacceptable. The Hepatitis dataset isn't evenly distributed - 'Live' is the more common outcome, whereas 'Die' is the positive class, but at the same time the rare label. Furthermore, accuracy can not be relied on if it is assumed that a data tuple has the possibility to belong to either of the 2 classes. There are more suitable measures for an unevenly distributed dataset, such as sensitivity or recall, which is used in the second grid search.

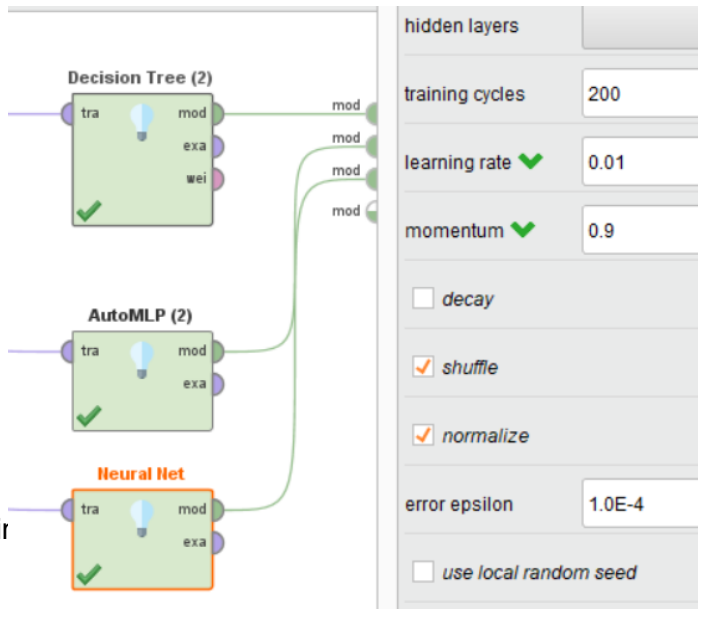
Recall gives a result of how much of the important data can be successfully classified, or how well the model classified the true positives. It is the same calculation as sensitivity.
The accuracy outcome in the test above is at 86%, but only about 15% of the positive class is correctly labeled. In conclusion, accuracy and recall can be used for different purposes. In the case of the Hepatitis dataset, recall (+ specificity / precision) are more realistic approaches.

Exercise 4:

Compare the ROC curves of the neural network and of the decision trees to argue which is the better model. For each model, take the optimized version.
Rapid Miner: Try out both the **Neural Net operator** and the **AutoMLP** operator. Once you have optimized your models, compare the neural net and the decision trees using the Compare ROCs operator.

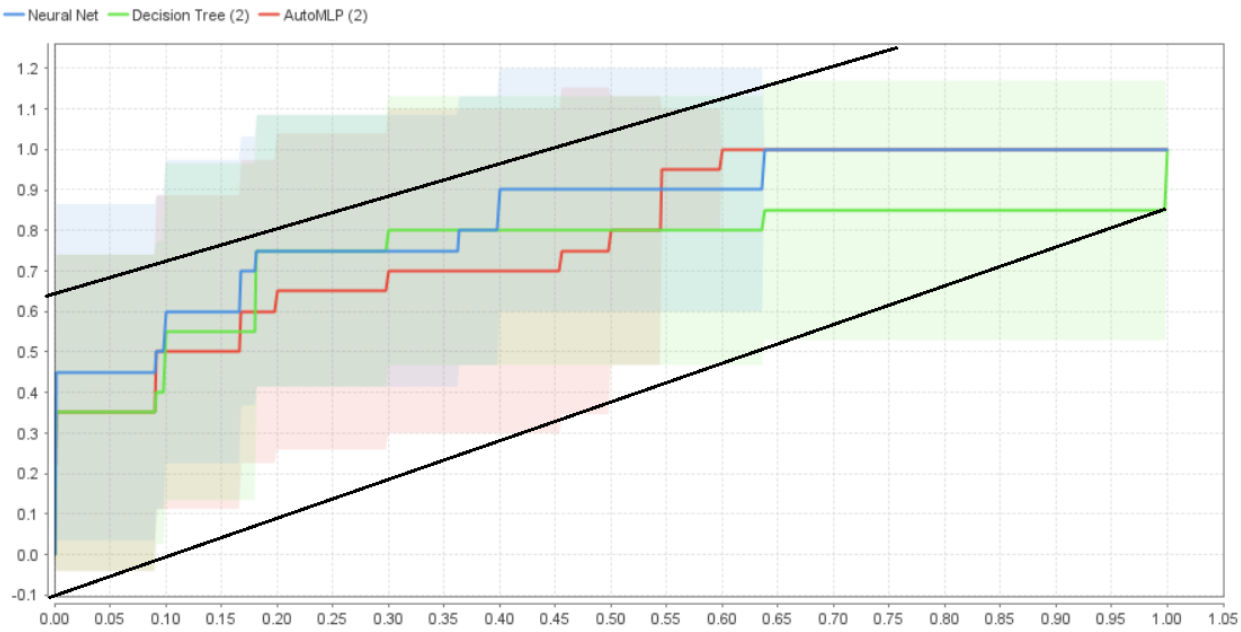


For the Neural Net, the learning rate was left at 0.01, since the recall was the best with the lowest possible learning rate value. For the measurement of the ROC-Curve, the value of true positive prediction matters, so that criterion should be enhanced.
The AutoMLP Model is also added to the evaluation.



Gruppe 1 (Jelena Mirceta, Moez Rjiba, Sir

The y-Value visualises the true positives, the x-value the false positives. The ROC-curves that are closer to the upper boundary, marked as a black line, are better models. The ones closer to the lower boundaries have a lower quality of estimating true positives.



In this case, the Neural Network is consistently closer to the upper boundary and is the best classifier for the Hepatitis Dataset. That means that the Neural Network is the most precise when choosing between a value being classified as a true positive. The confidence level retrieved from each model takes in account also the lower probabilities that a tuple should be classified as positive.
The Decision Tree is closer to the lower boundary and in this case isn't the better model in this case where it is important to classify true positives correctly.