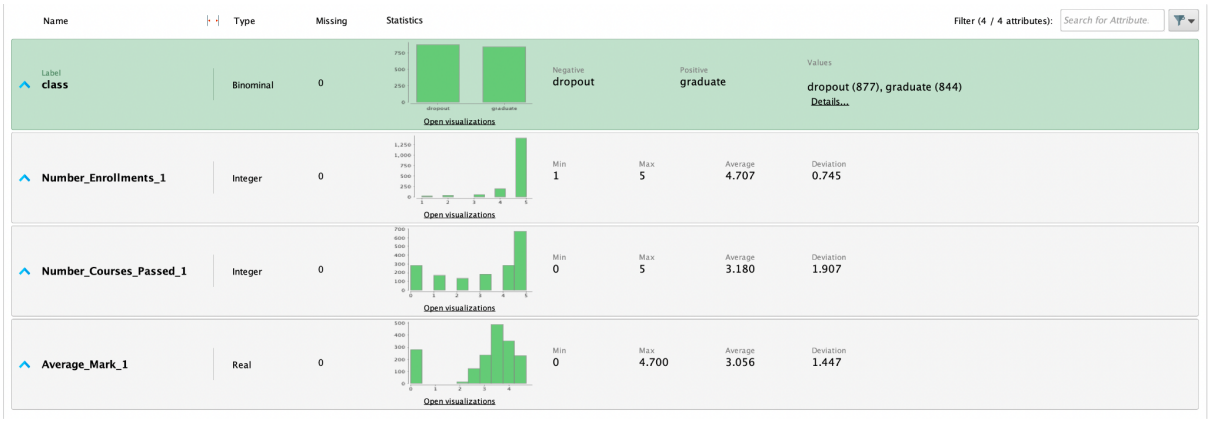
Exercise 1:

Explore the data with the summary statistics and different visualisations. Decide and justify which attributes you will keep to predict whether a student will drop out.



We see from the summary statistics that the data is very balanced regarding the class attribute (almost as many graduates as dropouts). “Dropout” will be our positive class.
The values in the Average_Mark column are encoded as follows:

6 - (average mark as per the semester report) = (value in Average_Mark)

We assume based on examination of individual samples that only marks of passed courses were included in the calculation of the average mark. But a student was assigned a 6 in the semester report if she or he did not pass any course. Hence 282 students have the value 0 for both attributes Average_Mark and Number_Courses_Passed. So although an Average_Mark value of 0 is technically a missing value, it still holds valuable information. As the examples match 1:1 regarding the feature Number_Courses_Passed we can assume that the data is clean in that aspect. As we need a numerical value for the Neural Networks later anyway, we will keep everything as is. We did not find any other indications for erroneous data as well. Average_Mark seems normally distributed as expected (except 0 as explained). Mark 1 (Best) on the semester report would equal a 5 and mark 4 (Worst, but still passed) would equal a 2 in this dataset. For the following computations the values will be kept, but when analysing the results they have to be transformed back as follows:

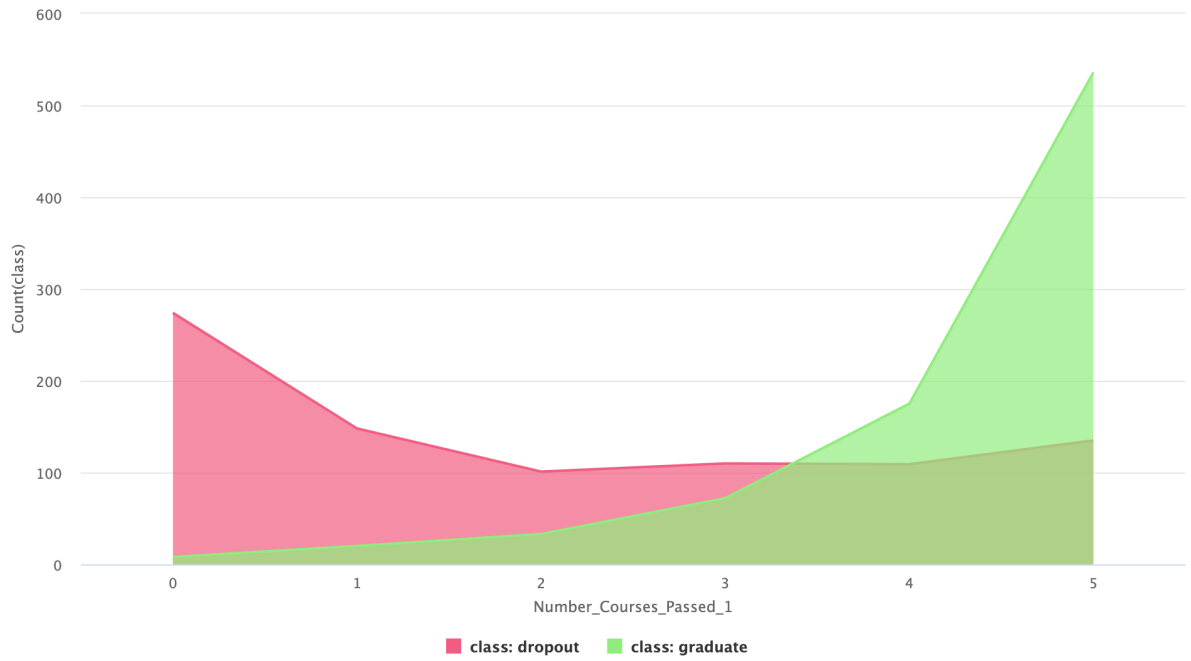
6 - (value in Average_Mark) = (average mark as per the semester report)

Number_Enrollments and Number_Courses_Passed are quite unbalanced, but we assume that this is reflects reality (Most students enrol in all five courses in the first semester and many will pass all of them)

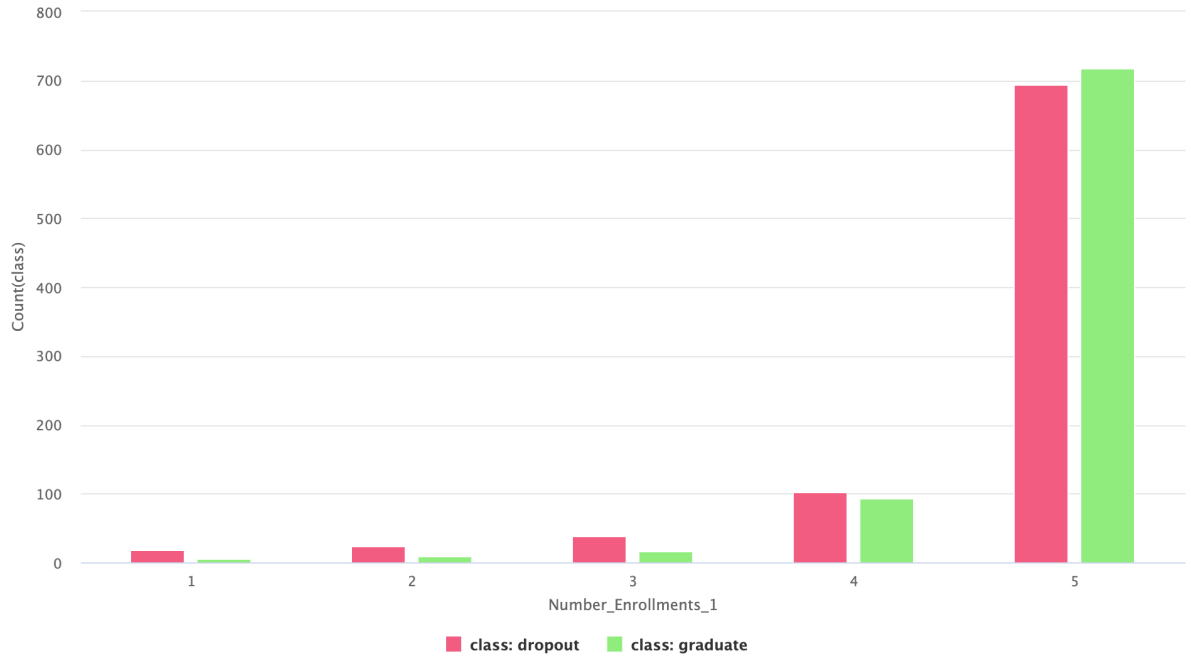
Before we looked at the visualisations we computed a correlation matrix for all features:

Attributes	Number_Enrollments_1	Number_Courses_Passed_1	Average_Mark_1	class
Number_Enrollments_1	1	0.354	0.201	0.102
Number_Courses_Passed_1	0.354	1	0.768	0.608
Average_Mark_1	0.201	0.768	1	0.489
class	0.102	0.608	0.489	1

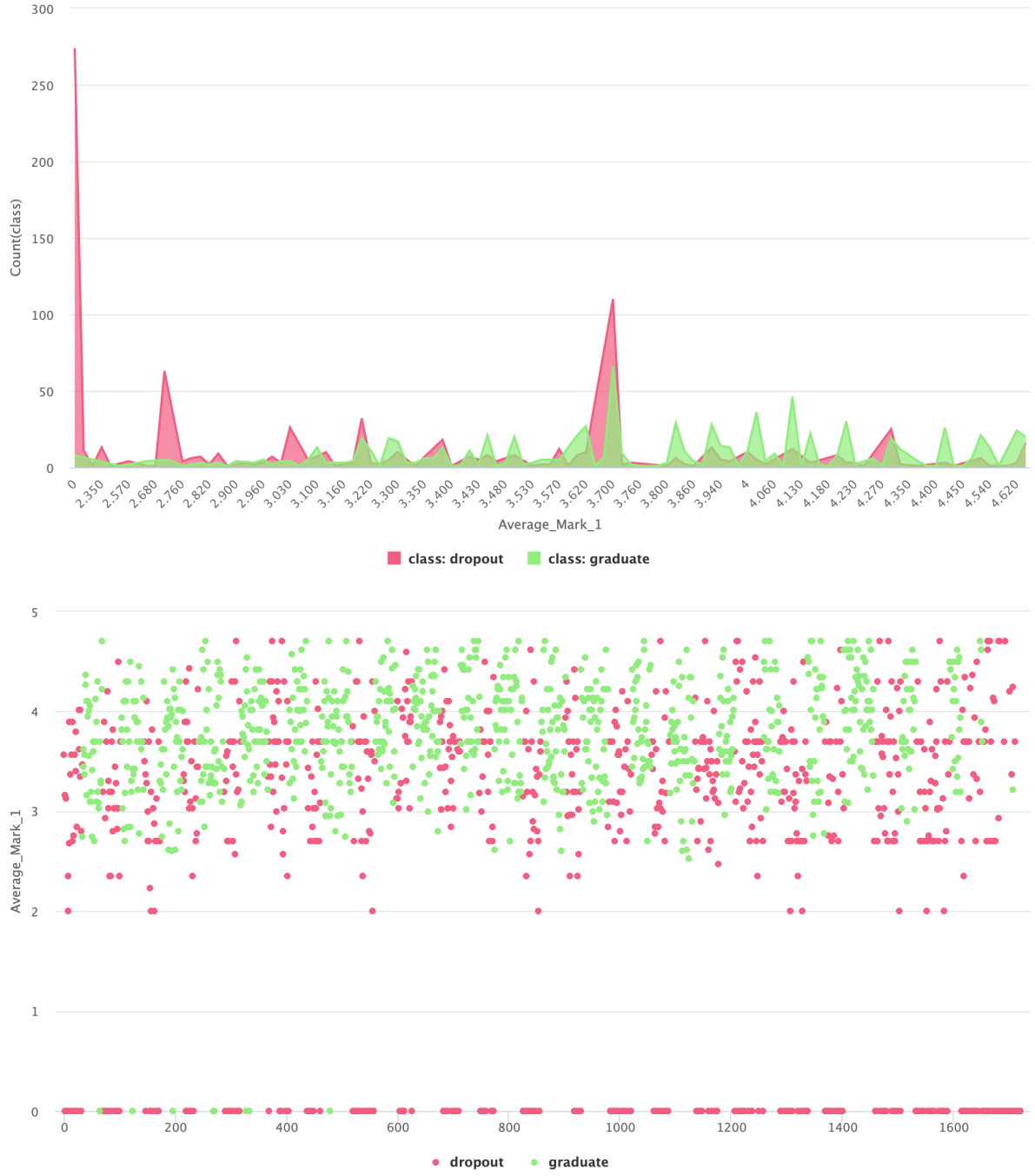
This suggests a strong correlation between Average_Mark and Number_Courses_Passed, which might indicate an exclusion of one of these two attributes. With the visualisation of Number_Courses_Passed (below) we see however how relevant this feature might be in our future predictions. Way more graduates passed 4 or 5 courses in the first semester than dropouts and if a student only passed 3 or less courses he or she was way more likely to drop out later. This circumstance might be reflected in the mild correlation between this feature and the class attribute.

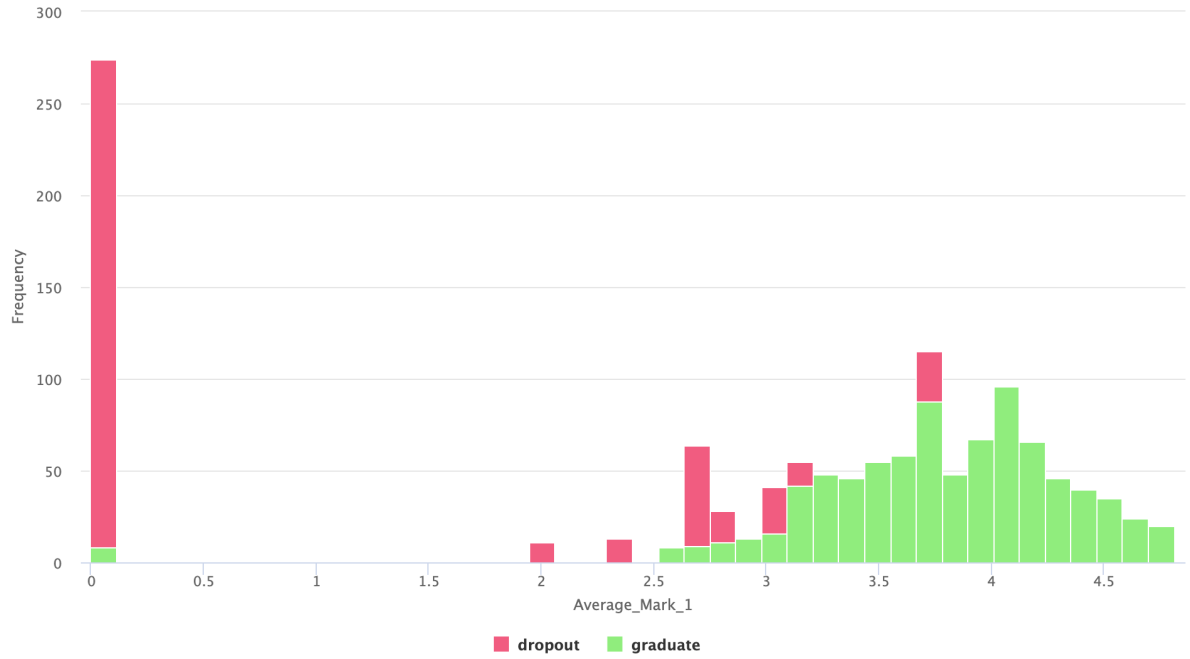


A similar although much fainter pattern is visible in the following visualisation of Number_Enrollments in relation to the class attribute.

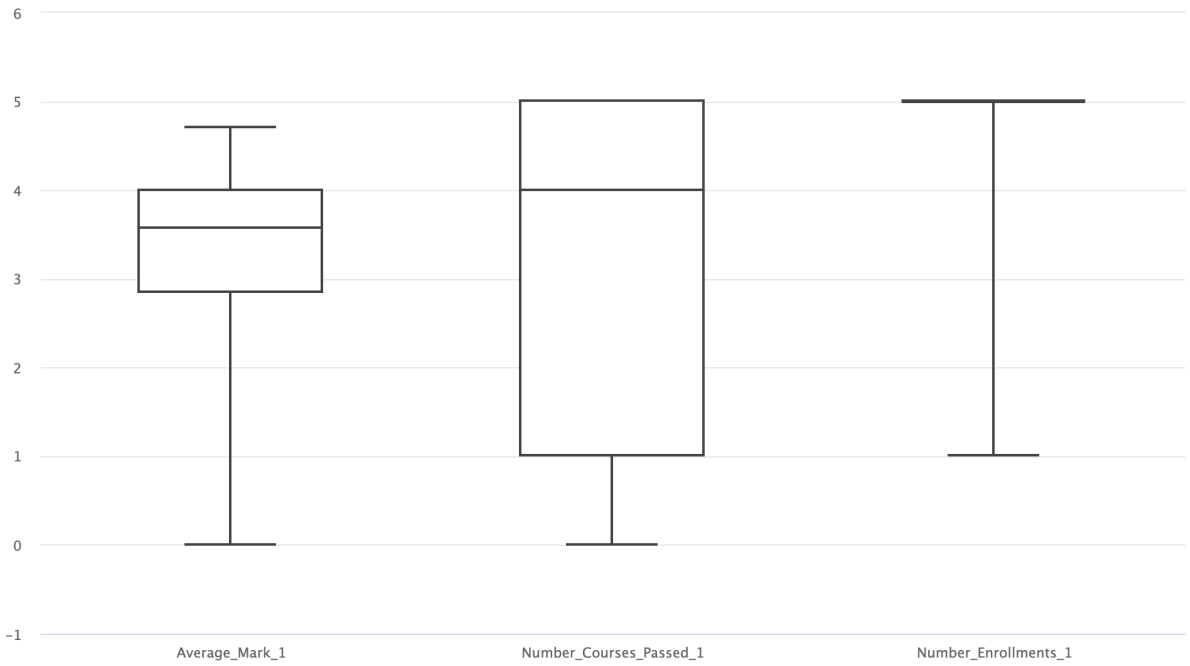


The plots of the Average_Mark feature (below) do show some worrysome patterns at first sight. The red (dropout) spike (equivalent to a higher densitiy in the scatter plot) at zero was explained above, the other spikes are explained with the discretization of marks (1.0, 1.3, 1.7, 2.0 and so on) that lead to a rather discretised distribution of Average_Mark values as well. The higher spikes at 3.7 (equivalent to real mark 2.3) and 2.7 (equivalent to real mark 3.3) are unusual, but at least in the first case not problematic as there is a higher density at this value in both classes. In the second case there are more dropouts with that mark, but since the mark is quite bad this is expected. All in all we still believe that the dataset is clean and its values are correct. (Please note that values < 2 and ≥ 0 are not shown in the first plot. The histogram on the next page shows a more complete although not as detailed picture of the distribution of Average_Mark in relation to the class attribute)

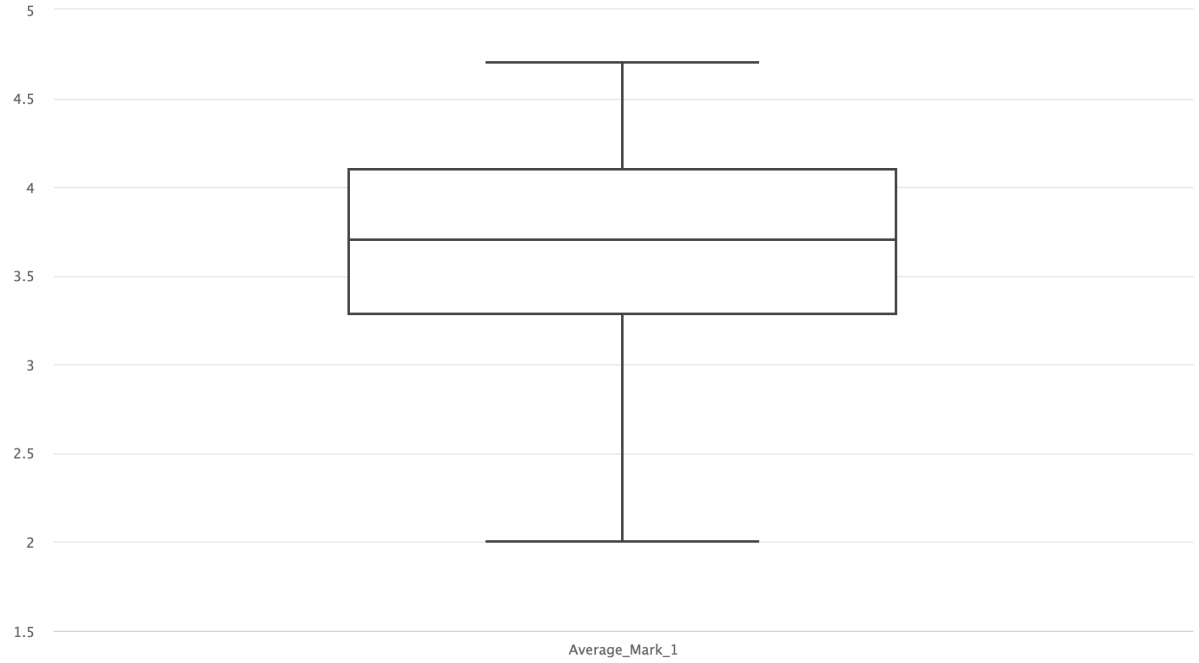




The boxplots of the three features that are not the class attribute confirm the observations above. In addition to the information the summary statistics gave us (min, max, average, etc.) we learn that the median of Average_Mark is 3.57 (equivalent to real mark 2.43) and the medians of Number_Courses_Passed and Number_Enrollments are 4 and 5 respectively. The interquartile range of Average_Mark is 1.15 as the middle half of students had a real average mark between 2 and 3.15. All of the metrics of the feature Average_Mark are heavily influenced by the many Zero-outliers that did not pass any course as described above. The middle half of students did however pass at least one course. A boxplot of Average_Mark of a “cleaned” subset can be seen on the next page.



Without the students that did not pass any course, the median Average_Mark lies at 3.7 (equivalent to real mark 2.3), the lower quartile at 3.28 (equivalent to real mark 2.72), the upper quartile at 4.1 (equivalent to real mark 1.9), meaning the middle half of students that did pass at least one course in the first semester had an average mark in the range of 1.9 and 2.72 (IRQ = 0.82).



In conclusion we could not definitively identify any feature that should be discarded for the following tasks. The data is clean and since there is such a limited number of features to begin with, we will make our predictions with all of them and try out different combinations of attributes. We suspect that Number_Enrollments might not be as significant as the rest.

If a predictor always predicts "dropout", what will be its probability of being correct? Explain your answer.

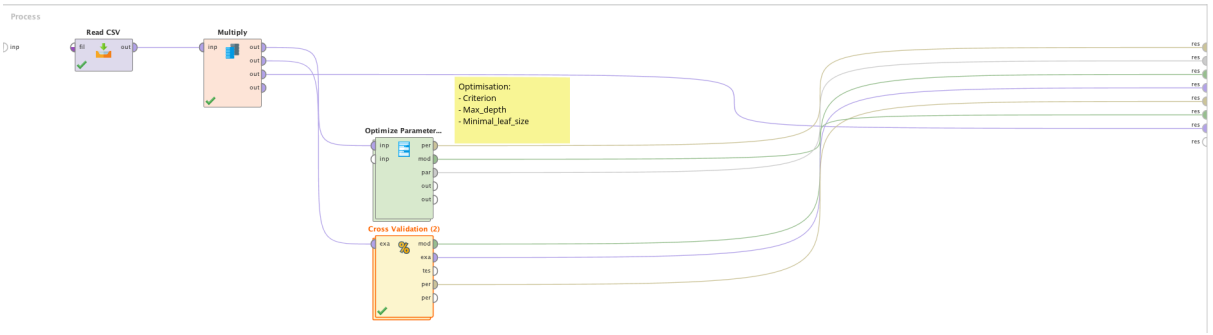
Since the predictor always predicts "dropout", P'=0 and N'=1721 (all students). Out of those, 877 truly did not graduate (TN) and 844 actually did graduate and are falsely predicted to be dropouts (FN). Since P'=0, TP and FP also equal 0. The probability to be correct equals the overall chance to have not survived (around 50,96%), as the predictor will only be correct in those cases and in no other case. The confusion matrix is as follows:

Confusion Matrix	**Predicted P**	**Predicted N**	**Total**
Actual P	0	844	844
Actual N	0	877	877
Total	0	1721	1721

Probability of being correct	**Formula**	**Solution** (approx.)
	$\frac{TN}{P+N}$	50,96 %

Exercice 2:

Use decision trees. Describe your best model (criteria, pruning etc.) and its evaluation results. Report which performance measure (accuracy, AUC, precision, recall, F1 etc.) you have chosen to optimise the hyper-parameters and explain shortly your choice.



maximal depth	20
apply pruning	
confidence	0.1
apply prepruning	
minimal gain	0.01
minimal leaf size	2
minimal size for split	4
number of prepruning alternatives	3

Since we are trying to predict whether a student will drop out, we chose our positive class to be “dropout”.

Without optimisations:

Criterion	Accuracy	Kappa	AUC	Precision	Recall	F-measure	Sensitivity	Specificity
Gain-Ratio	accuracy: 78.09% +/- 3.21% (micro average: 78.09%)	kappa: 0.563 +/- 0.064 (micro average: 0.563)	AUC: 0.834 +/- 0.028 (micro average: 0.834) (positive class: dropout)	precision: 82.90% +/- 4.96% (micro average: 82.64%) (positive class: dropout)	recall: 72.17% +/- 5.04% (micro average: 72.18%) (positive class: dropout)	f_measure: 77.02% +/- 3.43% (micro average: 77.05%) (positive class: dropout)	sensitivity: 72.17% +/- 5.04% (micro average: 72.18%) (positive class: dropout)	specificity: 84.23% +/- 5.38% (micro average: 84.24%) (positive class: dropout)
Information Gain	accuracy: 78.79% +/- 3.07% (micro average: 78.79%)	kappa: 0.577 +/- 0.061 (micro average: 0.577)	AUC: 0.844 +/- 0.031 (micro average: 0.844) (positive class: dropout)	precision: 84.41% +/- 3.29% (micro average: 84.32%) (positive class: dropout)	recall: 71.72% +/- 5.66% (micro average: 71.72%) (positive class: dropout)	f_measure: 77.42% +/- 3.73% (micro average: 77.51%) (positive class: dropout)	sensitivity: 71.72% +/- 5.66% (micro average: 71.72%) (positive class: dropout)	specificity: 86.14% +/- 3.50% (micro average: 86.14%) (positive class: dropout)
Gini-Index	accuracy: 79.37% +/- 2.91% (micro average: 79.37%)	kappa: 0.588 +/- 0.058 (micro average: 0.588)	AUC: 0.843 +/- 0.028 (micro average: 0.843) (positive class: dropout)	precision: 85.02% +/- 3.38% (micro average: 84.89%) (positive class: dropout)	recall: 72.40% +/- 5.59% (micro average: 72.41%) (positive class: dropout)	f_measure: 78.07% +/- 3.56% (micro average: 78.15%) (positive class: dropout)	sensitivity: 72.40% +/- 5.59% (micro average: 72.41%) (positive class: dropout)	specificity: 86.61% +/- 3.63% (micro average: 86.61%) (positive class: dropout)
Accuracy	accuracy: 78.79% +/- 4.98% (micro average: 78.79%)	kappa: 0.576 +/- 0.100 (micro average: 0.576)	AUC: 0.793 +/- 0.047 (micro average: 0.793) (positive class: dropout)	precision: 81.84% +/- 5.49% (micro average: 81.76%) (positive class: dropout)	recall: 75.13% +/- 5.69% (micro average: 75.14%) (positive class: dropout)	f_measure: 78.28% +/- 5.14% (micro average: 78.31%) (positive class: dropout)	sensitivity: 75.13% +/- 5.69% (micro average: 75.14%) (positive class: dropout)	specificity: 82.58% +/- 5.66% (micro average: 82.58%) (positive class: dropout)

Using cross validation we can have a first look on how our decision tree algorithm performs, based on which criteria we choose.
So if we have a look at the results, we can see that Gini-Index is performing better than Gain-Ratio or Information Gain with every hyperparameter that we selected, but not better than Accuracy when it comes to accuracy and F-measure.

To determine which efficient model to use, we will optimise the parameters using Grid Search maximal_depth, minimal_leaf_size and confidence relative to f_measure.

Main criterion: F-measure and positive class *"dropout"*

Optimisations:

1. Decision Tree.criterion
2. Decision Tree.maximal_depth
3. Decision Tree.minimal_leaf_size
4. Decision Tree.confidence

Best model with main criterion F-measure:

1. Criteria: Information Gain
2. Maximal depth: 19
3. Confidence: 0.3
4. Minimal leaf size: 1

Confusion Matrix:

	True dropout	True graduate
pred.dropout	618	107
pred.graduate	235	737

accuracy: 80.13% +/- 3.53% (micro average: 80.13%)
kappa: 0.604 +/- 0.071 (micro average: 0.604)
AUC: 0.832 +/- 0.030 (micro average: 0.832) (positive class: dropout)
precision: 85.92% +/- 5.37% (micro average: 85.71%) (positive class: dropout)
recall: 73.20% +/- 3.75% (micro average: 73.20%) (positive class: dropout)
f_measure: 78.98% +/- 3.61% (micro average: 78.97%) (positive class: dropout)
sensitivity: 73.20% +/- 3.75% (micro average: 73.20%) (positive class: dropout)
specificity: 87.33% +/- 5.21% (micro average: 87.32%) (positive class: dropout)

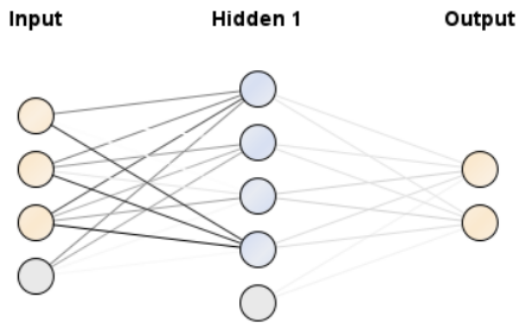
iteration	Decision Tree.criterion	Decision Tree.maximal_depth	Decision Tree.minimal_leaf_size	Decision Tree.confidence	f_measure ↓
2914	information_gain	19	1	0.300	0.790
3040	accuracy	-1	31	0.300	0.789
2100	accuracy	70	31	0.200	0.788
4860	accuracy	39	1	0.500	0.788
28	accuracy	60	1	0.000	0.788
1476	accuracy	50	1	0.150	0.788

In conclusion, we can note that the Grid Search is a very important tool to determine a certain optimisation to the hyperparameters so we can obtain a better model than the original one.
Note that the optimization of all the hyper parameters requires a lot of computer resources!
We can see a slight improvement on the Accuracy, Kappa, Precision and also the F-measure.
Indeed, even if the improvement of the hyperparameters is of the order of 1%, this is a significant gain for a large data set.

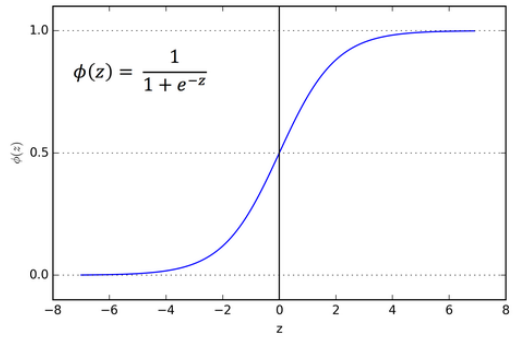
Exercise 3:

Use neural networks. Describe your best model (learning rate, momentum etc.) and its evaluation results. Choose the same performance measure to optimise the hyper-parameters as in 2.

The Neural Network operator takes the input and feeds it forward to train it with a backpropagation algorithm n times, defined with the parameter training cycles. The output values are compared with the correct answer with an error function in the backpropagation process of error. The error is fed backwards and the weights updated by also considering the learning rate and momentum. The learning rate defines by what rate the weights are optimised and the momentum adds a fraction of the last update to the current one.



Our Neural Network Algorithm has 1 hidden layer and will go through 300 training cycles. It uses by default the Sigmoid Function as the activation function. The activation function activates the nodes by a statistical rate. The Sigmoid Function can learn complex non-linear problems, where data can't be separated by one clear line. Any input is transformed to a range from 0 to 1.



Source: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Neural Network default settings:

Training cycles: 300
Number of cycles the feed forward and backpropagation passes through. Set to 300.
Learning rate: 0.01 (default, will be optimized by grid-search)
Parameter that decides by what percentage the weights are changed.

Momentum: 0.9
The fraction of the previous weight added to updated the current weight.
Hidden Layers: 1 Hidden Layer, bias added
Activation Function: Sigmoid Function
The function that the weighted sum of inputs passes through. Output gives the input for the next layer.
Normalize: For Sigmoid Function, attributes need to be normalized to be between -1 and 1, so it's set to true.

Neural Network with Cross Validation, without optimised parameters and local seed

First we proceeded with an assessment whether measure results differed with different positive classes. The accuracy of 79,54 % is the same with either of the classes being positive. WIth parameter optimization, the rate of correct classification can possibly be higher. It changes when looking at precision and recall. The precision for dropout as positive label is higher than for graduate as positive label. When it predicts that the student is a dropout, it is correct 85.10% of the time. But it is only correct 75.66% of the time when the student is a graduate and graduate is a positive class.

Positive class: graduate	Positive class: dropout
accuracy: 79.54% +/- 2.94% (micro average: 79.55%) True: dropout graduate dropout: 640 115 graduate: 237 729 kappa: 0.592 +/- 0.059 (micro average: 0.592) precision: 75.66% +/- 3.55% (micro average: 75.47%) (positive class: graduate) recall: 86.38% +/- 5.43% (micro average: 86.37%) (positive class: graduate) f_measure: 80.53% +/- 2.76% (micro average: 80.55%) (positive class: graduate) sensitivity: 86.38% +/- 5.43% (micro average: 86.37%) (positive class: graduate) specificity: 72.96% +/- 5.78% (micro average: 72.98%) (positive class: graduate)	accuracy: 79.54% +/- 2.94% (micro average: 79.55%) True: dropout graduate dropout: 640 115 graduate: 237 729 kappa: 0.592 +/- 0.059 (micro average: 0.592) precision: 85.10% +/- 5.01% (micro average: 84.77%) (positive class: dropout) recall: 72.96% +/- 5.78% (micro average: 72.98%) (positive class: dropout) f_measure: 78.36% +/- 3.48% (micro average: 78.43%) (positive class: dropout) sensitivity: 72.96% +/- 5.78% (micro average: 72.98%) (positive class: dropout) specificity: 86.38% +/- 5.43% (micro average: 86.37%) (positive class: dropout)

To measure f-measure, recall and precision it was important to decide whether dropout or graduate should be the positive class. The subject for this project is to find out whether the

students will drop out if their grades are not high enough, if they failed the course or if they never finished it. So ‘dropout’ is decided to be the positive class.
Now that dropout is the positive class, we want to have the highest rate of dropout results therefore tolerate a decreased precision to have a higher recall, while keeping the accuracy high, too. Since the dataset is balanced, the distinction in positive and negative class is only interesting to a certain extent, namely to have a good predictor. Measuring accuracy is also sufficient for this balanced dataset, so we will observe the accuracy for the neural network too.

Optimized Neural Network Parameters (positive class: dropout)

At first we were using 10-fold Cross Validation and optimising the parameters with Grid Search **Learning Rate** and **Momentum** with respect to the measure **recall**. The grid-search will go through 10 different, incremented iterations in order to finetune a chosen parameter.

First try:
As a result, the accuracy dropped to 50,53 %, precision to 51,54% and the recall increased to 91,34 %. The predictor now takes every single possible dropout outcome into the calculation, which decreases accuracy and precision. The predictor will be overfitting, thus the parameter Momentum was taken out of the optimisation operator. The results now are more adequate:

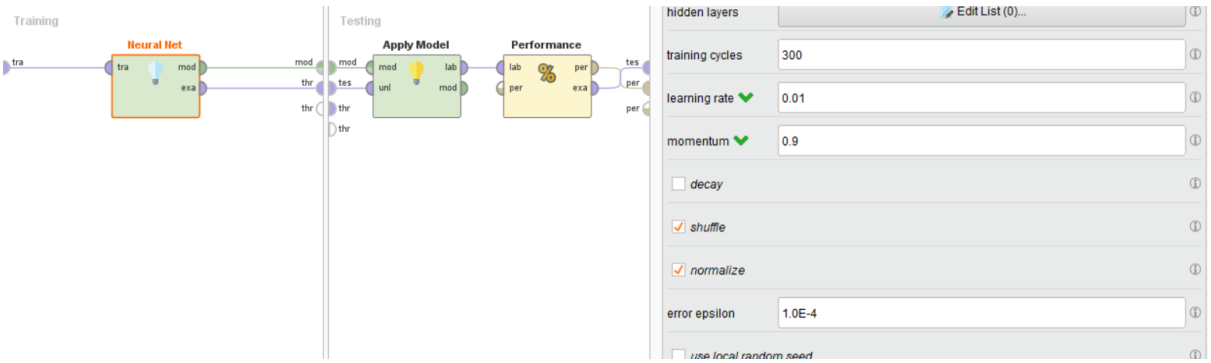
Second try:
With the momentum taken out of the equation, the results are as follows:

accuracy: 76.29% +/- 9.41% (micro average: 76.29%)
precision: 81.61% +/- 12.24% (micro average: 77.56%) (positive class: dropout)
recall: 75.26% +/- 11.54% (micro average: 75.26%) (positive class: dropout)
f_measure: 76.79% +/- 4.97% (micro average: 76.39%) (positive class: dropout)

True:	dropout	graduate
dropout:	660	191
graduate:	217	653

The accuracy fell a little from the the predictor model without optimised parameters. The recall is higher now, with 75.26 %.

Further investigation into the matter which measure to choose and to align it with other predictors used in this project, the decision was made to use **f-measure** as the measure for all models instead of solely looking at the recall.
F-measure is a performance measure that shows the harmonic mean between precision and recall. Measuring only recall or precision signifies heavy weight on the dropout class. Once the recall gets better, the precision gets lower, which also makes the graduate label look more insignificant. In real life, both class labels matter and only as the subject for the project, we are more interested in predicting dropouts.
Best model:



With regard to using f-measure as the main criterion, the results are as follows:

PerformanceVector:

ConfusionMatrix:
True: dropout graduate
dropout: 637 108
graduate: 240 736

accuracy: 79.78% +/- 3.55% (micro average: 79.78%)
kappa: 0.597 +/- 0.071 (micro average: 0.597)
precision: 85.92% +/- 6.29% (micro average: 85.50%) (positive class: dropout)
recall: 72.63% +/- 4.42% (micro average: 72.63%) (positive class: dropout)
f_measure: 78.55% +/- 3.60% (micro average: 78.55%) (positive class: dropout)
sensitivity: 72.63% +/- 4.42% (micro average: 72.63%) (positive class: dropout)
specificity: 87.22% +/- 6.46% (micro average: 87.20%) (positive class: dropout)

The best model so far has not only increased the f-measure, but also the accuracy from the unoptimised predictor. It also figured that the best Neural Network learning rate is 0.3. This Neural Network is going to be used in the comparison with other models.

Optimize Parameters (Grid) (2) (11 rows, 3 columns)

iteration	Neural Net.learning_rate	f_measure ↓
4	0.300	0.786
6	0.500	0.784
2	0.100	0.781
3	0.200	0.780
8	0.700	0.778
7	0.600	0.773
11	1	0.772
10	0.900	0.768
5	0.400	0.768
9	0.800	0.750
1	0	0.424

Exercice 4:

Explore at least three other classification algorithms. You may choose an algorithm that we did not see in class, and that you would like to explore. Describe briefly how the algorithms work. Do not forget to cite your sources! Report your best models after optimization of the hyperparameters and report their evaluation results. Choose the same measure to optimise the hyperparameters as in 2.

1. Logistic Regression:

Logistic Regression is a Machine Learning algorithm which is used for classification problems, it is a predictive analysis algorithm based on the concept of probability.
It is the go-to method for binary classification problems (problems with two class values).
Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values to predict an output value (y). A key difference from linear regression is that the output value being modelled is a binary value (0 or 1) rather than a numeric value.

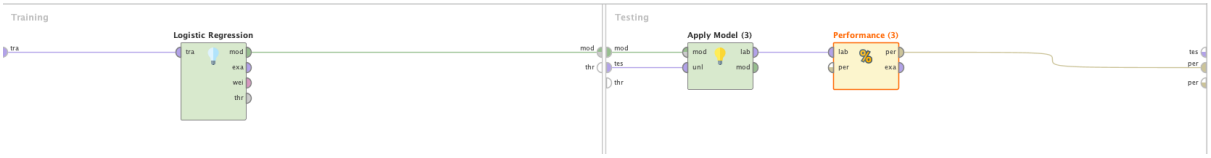
Below is an example logistic regression equation:

$$y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

(*Source: https://machinelearningmastery.com/logistic-regression-for-machine-learning/)*



iteration	Logistic Regression.solver	Logistic Regression.max_iterations	f_measure ↓
43	L_BFGS	80	0.789
54	COORDINATE_DESCENT_NAIVE (experimental)	100	0.789
34	COORDINATE_DESCENT_NAIVE (experimental)	60	0.788
36	AUTO	70	0.788
15	COORDINATE_DESCENT (experimental)	20	0.788

Note that the result may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision.

Main criterion: F-measure and positive class *“dropout”*

Logistic regression does not really have any critical hyperparameters to tune.

Sometimes, you can see useful differences in performance or convergence with different solvers (*solver*).

(Source: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/)

Optimisations:

1. Logistic Regression.solver
2. Logistic Regression.max_iterations

Best model with main criterion F-measure:

1. Iteration: 91
2. Logistic Regression.solver

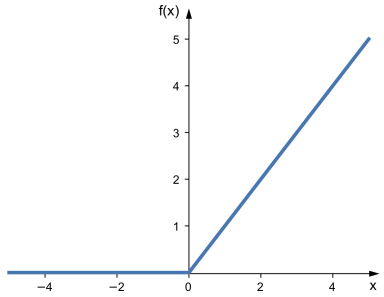
Confusion Matrix:

	True dropout	True graduate
pred.dropout	637	100
pred.graduate	240	744

accuracy: 80.24% +/- 3.61% (micro average: 80.24%)
kappa: 0.606 +/- 0.072 (micro average: 0.606)
AUC: 0.865 +/- 0.024 (micro average: 0.865) (positive class: dropout)
precision: 86.58% +/- 4.77% (micro average: 86.43%) (positive class: dropout)
recall: 72.64% +/- 4.36% (micro average: 72.63%) (positive class: dropout)
f_measure: 78.93% +/- 3.84% (micro average: 78.93%) (positive class: dropout)
sensitivity: 72.64% +/- 4.36% (micro average: 72.63%) (positive class: dropout)
specificity: 88.15% +/- 4.88% (micro average: 88.15%) (positive class: dropout)

2. Deep Learning

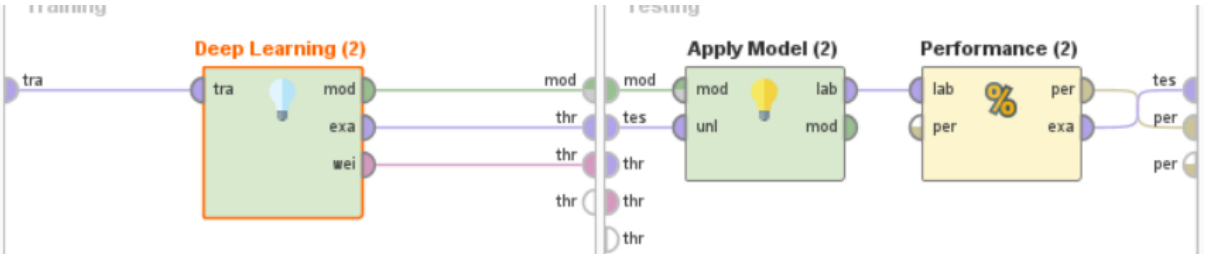
Corresponding to the Neural Network algorithm used in Exercise 3, the Deep Learning operator in RapidMiner offers more powerful optimization possibilities to create a better classifier. Deep Learning is an extended Neural Network that works on the same principle but can contain a larger number of neurons and different activation functions. While the Neural Network operator offers to use the Sigmoid Function, Deep Learning can use ReLu or Tanh activation functions which are in some cases better alternatives. It also uses the adaptive learning rate algorithm (ADADELTA) which combines the learning rate and momentum, making the hyperparameter search simpler.
Deep Learning has a preset of 2 hidden Layers with 50 Neurons each. The recommended activation function is Rectifier (ReLu). It's apparently



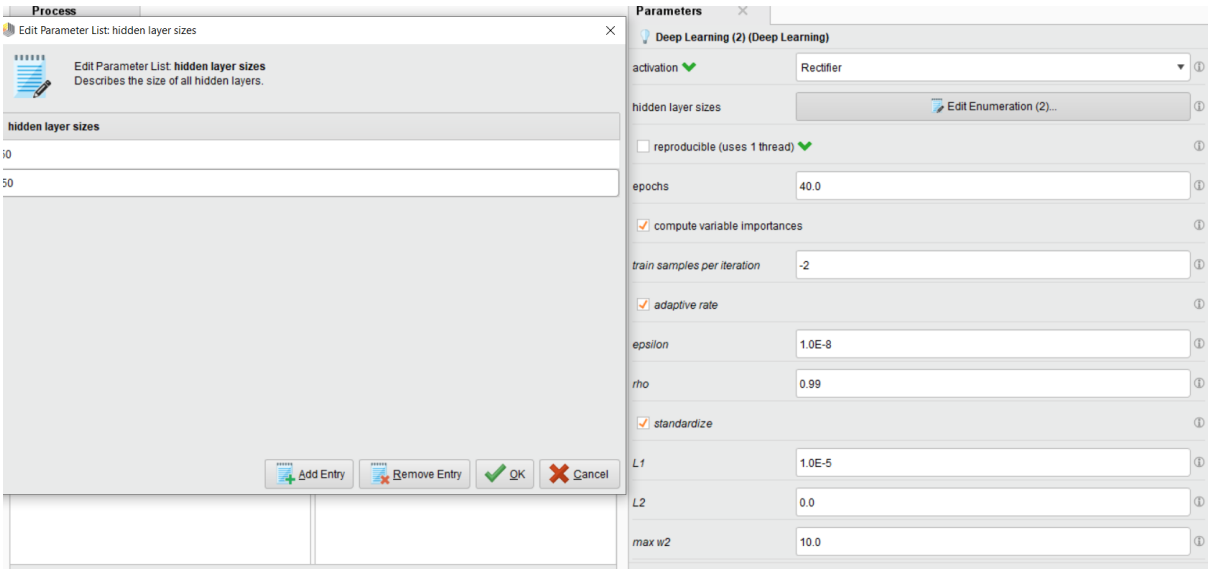
$$f(x) = \max(0, x)$$ or $$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Source: https://sebastianraschka.com/faq/docs/relu-derivative.html

The Deep Learning operator picks automatically the best distribution function as well as loss function depending whether the values are nominal or numbers. For our dataset, it picked the Bernoulli distribution as the distribution function. Cross Entropy was used as the loss function which is used in backpropagation.



Optimizations:
The number of epochs was manually set to 40. Running it through the Grid Search required high CPU resources.



Results for Deep Learning parameters:

Layer	Units	Type	Dropout	L1	L2	Mean Rate	Rate RMS	Momentum	Mean Weight	Weight RMS	Mean Bias	Bias RMS
1	3	Input	0.00 %									
2	50	Rectifier	0	0.000010	0.000000	0.003758	0.002894	0.000000	-0.047888	0.177146	0.243932	0.143874
3	50	Rectifier	0	0.000010	0.000000	0.142306	0.242870	0.000000	-0.041331	0.138517	0.835515	0.115257
4	2	Softmax		0.000010	0.000000	0.008887	0.015279	0.000000	0.009526	0.401118	0.000800	0.034873

PerformanceVector:
accuracy: 80.35% +/- 2.81% (micro average: 80.34%)
ConfusionMatrix:

True:	dropout	graduate
dropout:	635	98
graduate:	242	746

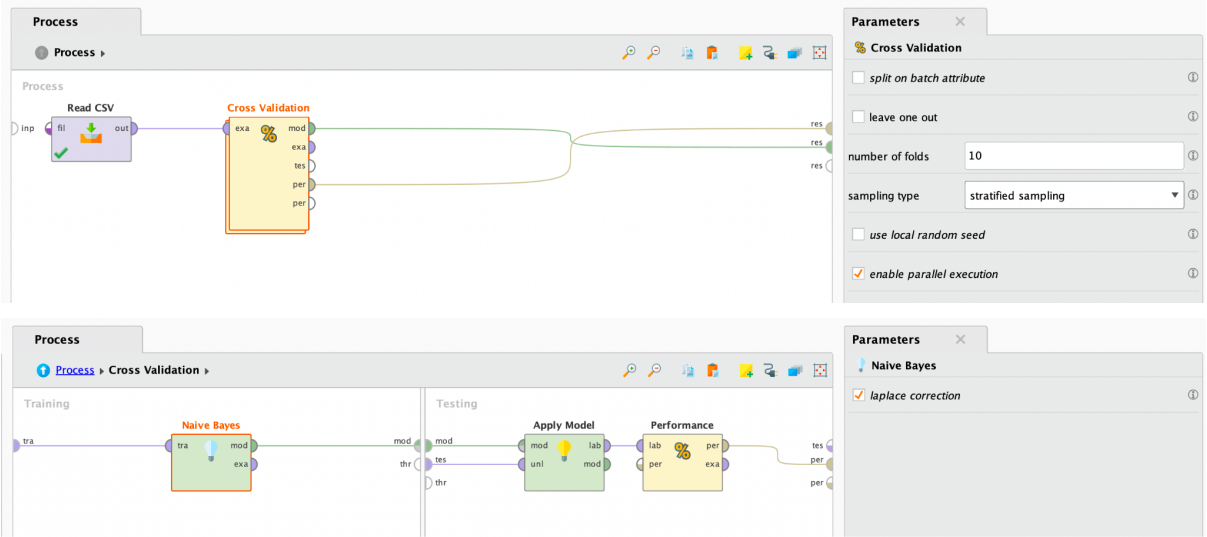
kappa: 0.606 +/- 0.056 (micro average: 0.606)
precision: 86.70% +/- 3.40% (micro average: 86.63%) (positive class: dropout)
recall: 72.40% +/- 4.61% (micro average: 72.41%) (positive class: dropout)
f_measure: 78.83% +/- 3.40% (micro average: 78.88%) (positive class: dropout)
sensitivity: 72.40% +/- 4.61% (micro average: 72.41%) (positive class: dropout)
specificity: 88.39% +/- 3.19% (micro average: 88.39%) (positive class: dropout)

Both accuracy and the f_measure have increased slightly from using the Neural Network operator.

3. Naïve Bayes

Bayes' theorem lets us calculate the (posterior) probability of a hypothesis conditioned on observed data **P(H|X)**, meaning the chance of this hypothesis being correct, given the data provided by one sample. In our case we want to know the probability of belonging to one of our two classes, a student dropping out or graduating, provided her or his number of enrollments, number of courses passed and average mark in the first semester.
This probability is calculated using posterior probability of him/her having specific values in these three features given that we know whether she/he will drop out or graduate **P(X|H)** (the probability of observed data conditioned on the hypothesis), the prior probability of being a dropout or graduate in this dataset regardless of any observed data **P(H)** ($P(C_{dropout})$ = 877/1721 ≈ 50,1%) and the prior probability of having specific values in the three features based on the whole dataset, meaning regardless of class **P(X).**
The formula is as follows: $P(H|X) = \frac{P(X|H)P(H)}{P(X)}$
The naïve Bayesian classifier will predict a sample to belong to the class with the highest posterior probability P(H|X), and therefore try to maximise this value. In fact only the numerator of the fraction needs to be maximised as P(X) is constant for all classes.
The fact that makes this classifier naïve is the assumption that there are no dependencies among the attributes (class-conditional independence). Although we know that in our case this is not true (E.g. when a student did not pass any course his or her average mark will be 0), this premise greatly reduces the computation time while still leading to good results, as empirical studies have shown — as do our results.
To get P(X|H) the probabilities of each attribute value is calculated and multiplied with another. With $C_{dropout}$ and $C_{graduate}$ being our hypotheses H, then a student will be a predicted to be a dropout if $P(X|C_{dropout})P(C_{dropout})$ is greater than $P(X|C_{graduate})P(C_{graduate})$.
Source: Jiawei Han, Micheline Kamber, Jian Pei: Data Mining – Concept and Techniques

Below is the application of the classifier in RapidMiner. The only hyperparameter is the use of laplace correction which prevents the occurrence of zero values when multiplying the probabilities, which would cause misleading results. Enabling or disabling this parameter did not lead to different test results.



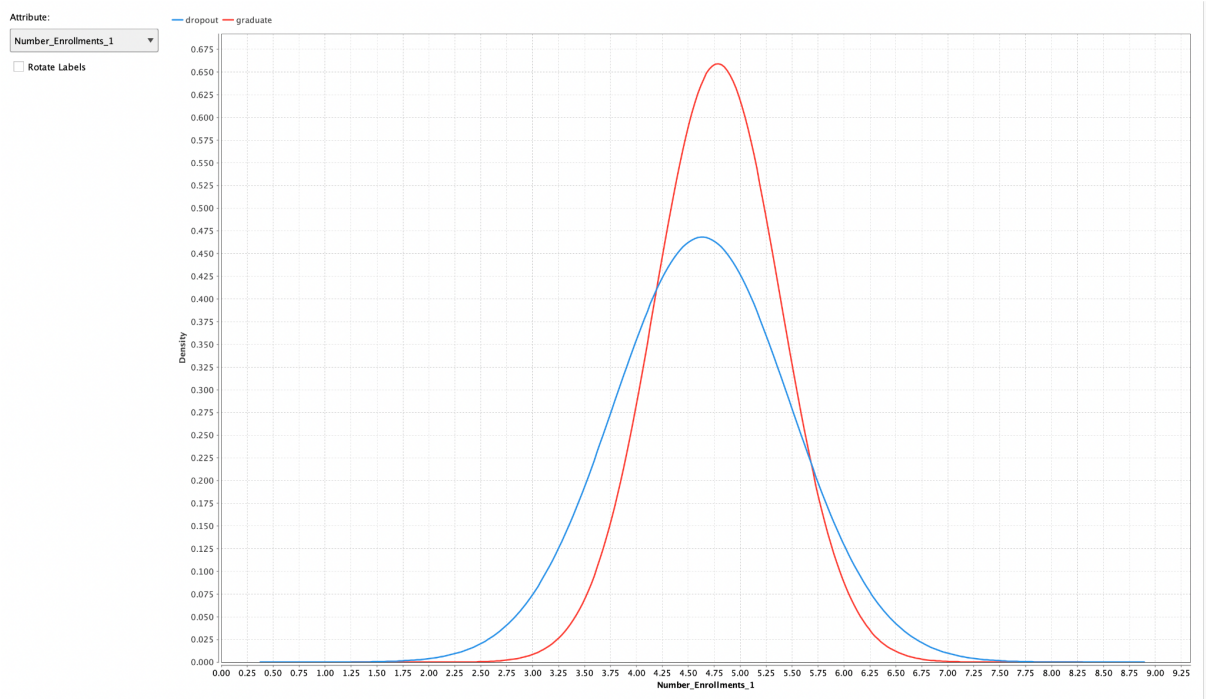
The results are shown on the following pages.

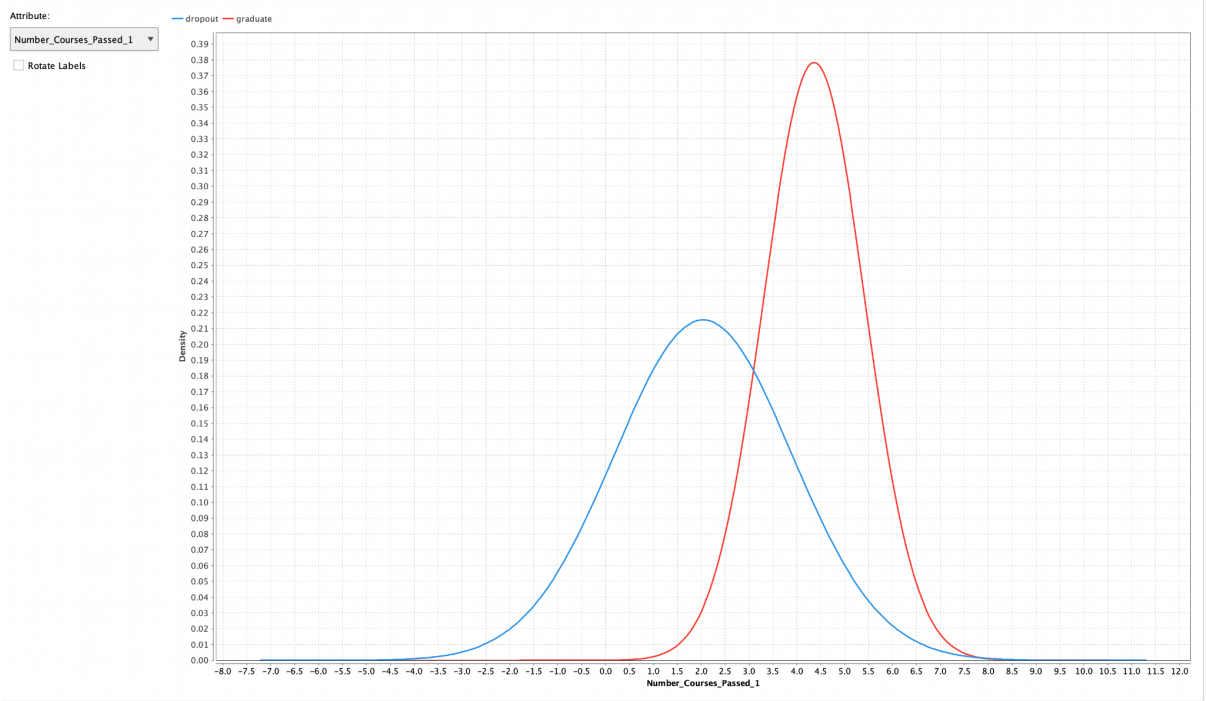
f_measure: 72.78% +/- 4.66% (micro average: 72.84%) (positive class: dropout)

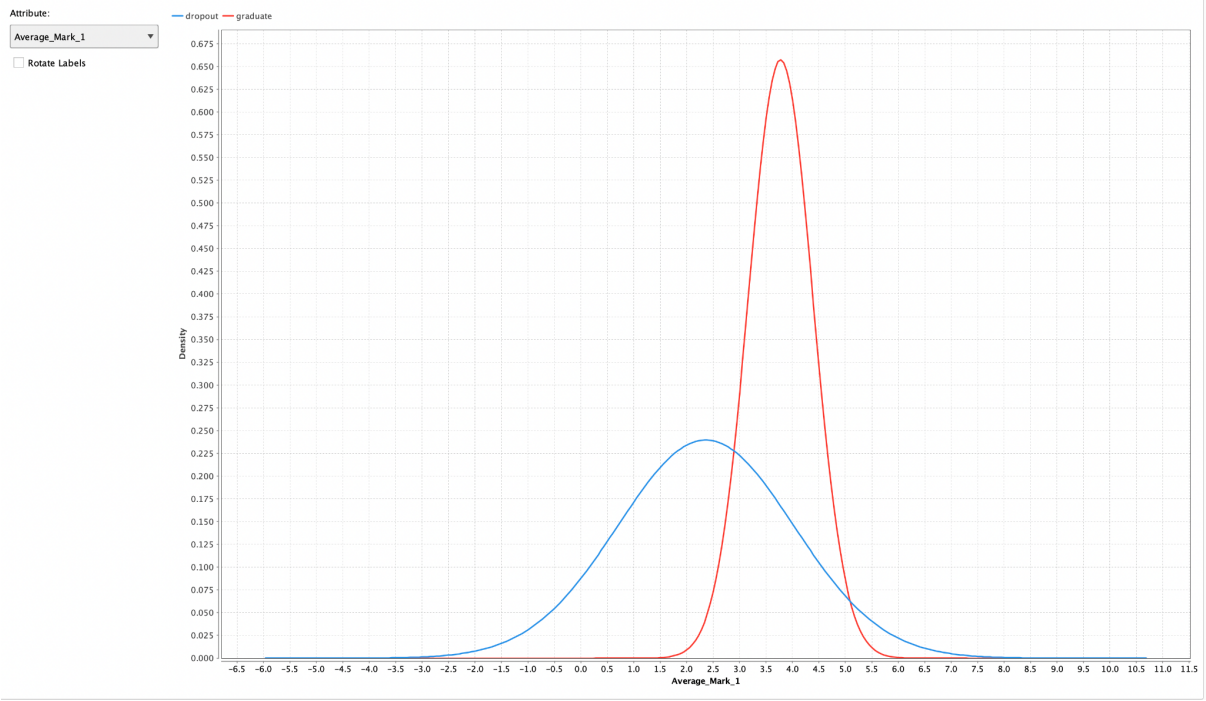
	true graduate	true dropout	class precision
pred. graduate	766	330	69.89%
pred. dropout	78	547	87.52%
class recall	90.76%	62.37%	

The most relevant part of the model, the probability measure in regard to the class P(X|H), that we try to maximise — together with P(H) —, is shown in the following table and plots. This clearly shows how important the features Number_Courses_Passed and Average_Mark are for the final classification, as the respective means for each class (or tip of each curve) are more clearly separated. Looking at Number_Enrollments, the curves are almost on top of each other. However, the probability of $P(H|X_{Number\_Enrollments})$ can still be maximised easily, as their standard deviation (or spread of the curves) and density at the peak is different. This gives us an indication on the definiteness of the prediction for that class in regards to specific values, as we effectively maximise and compare a (narrow) area under each curve around the value of one feature of our (test) sample.

Attribute	Parameter	dropout	graduate
Number_Enrollments_1	mean	4.633	4.784
Number_Enrollments_1	standard deviation	0.852	0.605
Number_Courses_Passed_1	mean	2.042	4.363
Number_Courses_Passed_1	standard deviation	1.852	1.054
Average_Mark_1	mean	2.362	3.776
Average_Mark_1	standard deviation	1.665	0.607





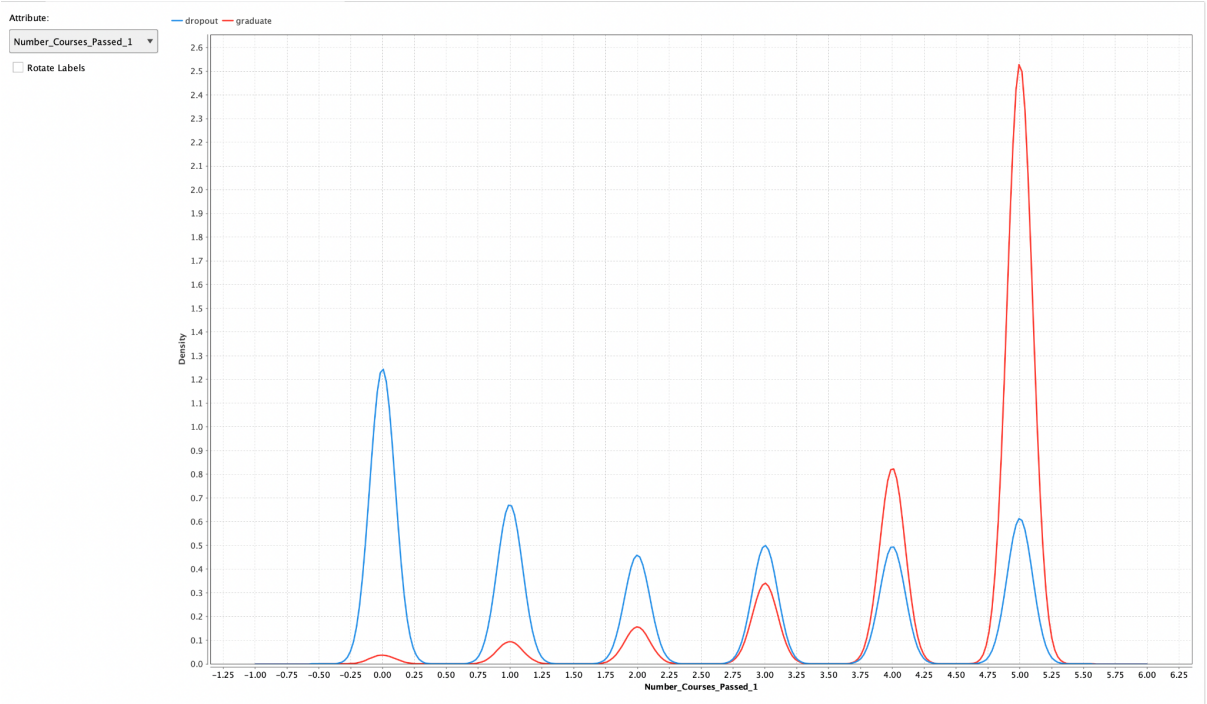


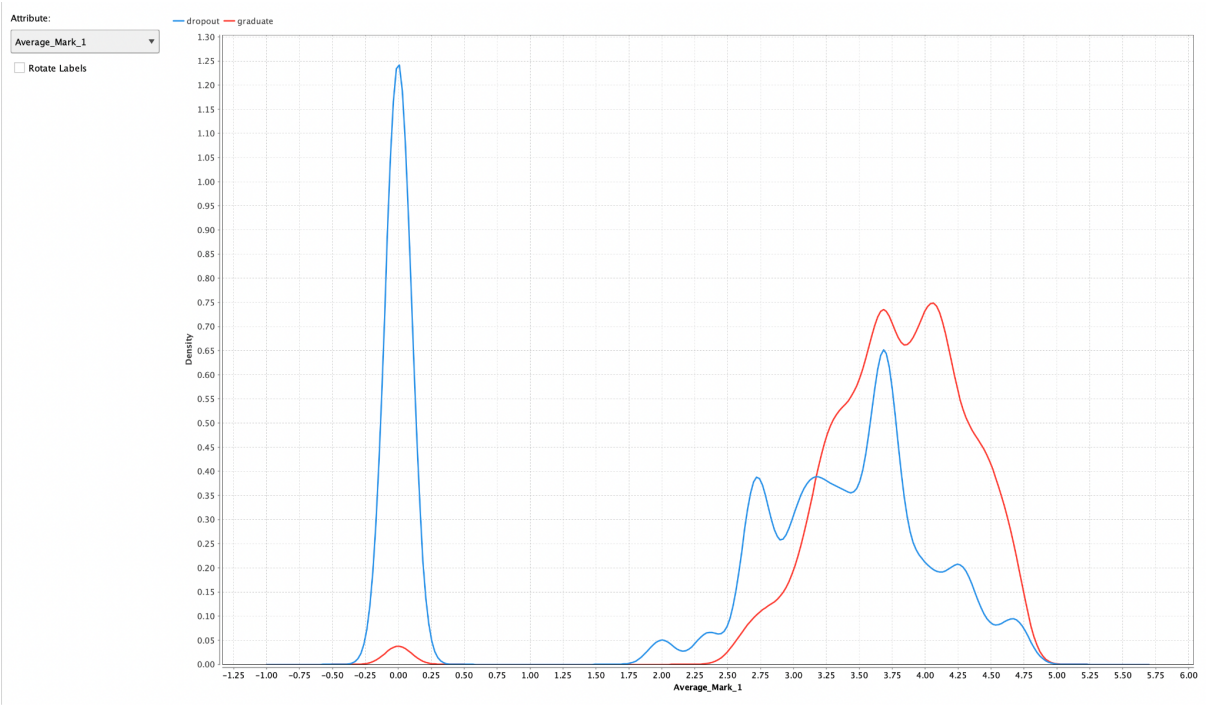
Unsatisfied with the result we looked at a related classifier in RapidMiner: “Naive Bayes (Kernel)”, which combines multiple Gaussian probability densities. As we see in the plots on the next page this better depicts not only the discrete values in the features regarding course counts, but also the extreme value of 0 within Average_Mark for students that did not pass any course in the first semester. Here we can optimise a number of parameters, with the number of kernels being the most important one. With automatically optimising the hyperparameters as before, we tested the number of kernels from 1 – 100 (among others). Although high kernel counts led to the best models regarding our performance measure, we assume these models to be quite overfitted. As we can see in the following table showing the 10 best hyperparameter configurations, 2 kernels seem to be almost as good.

Optimize Parameters (Grid) (8800 rows, 7 columns)

..	Naive Bayes (Kernel) (2).estimation_mode	Naive Bayes (Kernel) (2).bandwidth_selection	Naive Bayes (Kernel) (2).number_of_kernels	Naive Bayes (Kernel) (2).use_application_grid	Naive Bayes (Kernel) (2).application_grid_size	f_measure ↓
.	greedy	heuristic	28	true	19	0.786
.	greedy	fix	2	true	55	0.785
.	greedy	fix	47	true	19	0.784
.	greedy	fix	2	true	28	0.784
.	greedy	heuristic	18	true	19	0.784
.	greedy	heuristic	2	true	28	0.784
.	greedy	fix	69	true	19	0.783
.	greedy	fix	56	true	19	0.783
.	greedy	fix	31	true	19	0.783
.	greedy	heuristic	26	true	19	0.783

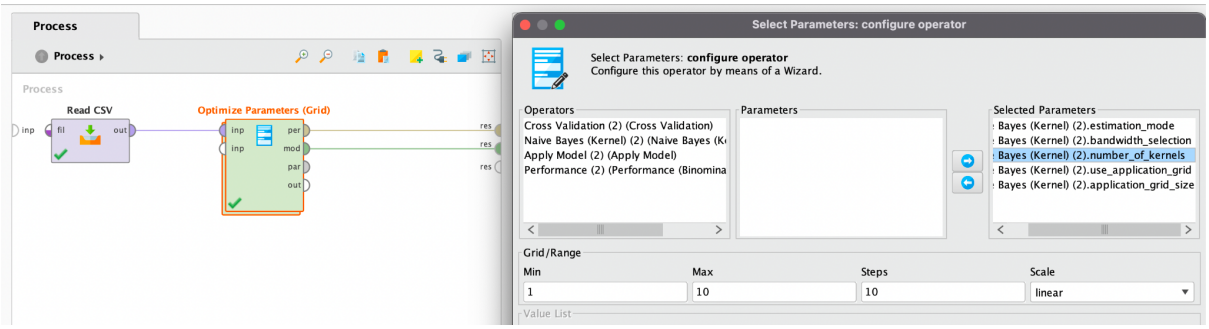
Plots for the overfitted model with 28 kernels:





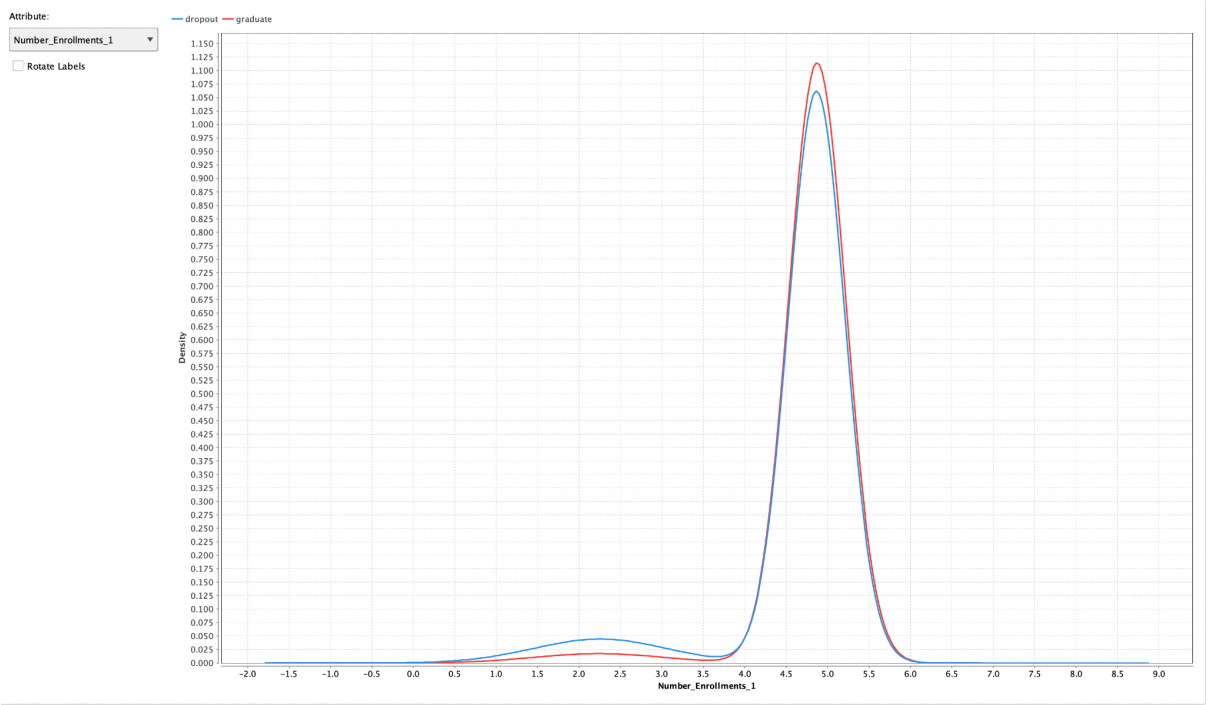
In the end we decided on the following parameters: estimation_mode: **greedy**, bandwith_selection: **heuristic**, number_of_kernels: **2**, application_grid_size: **28**

The results for this model is as follows (slightly different F-measure, because we ran the automatic parameter optimization again with a maximum of 10 kernels, to avoid the overfitted models)

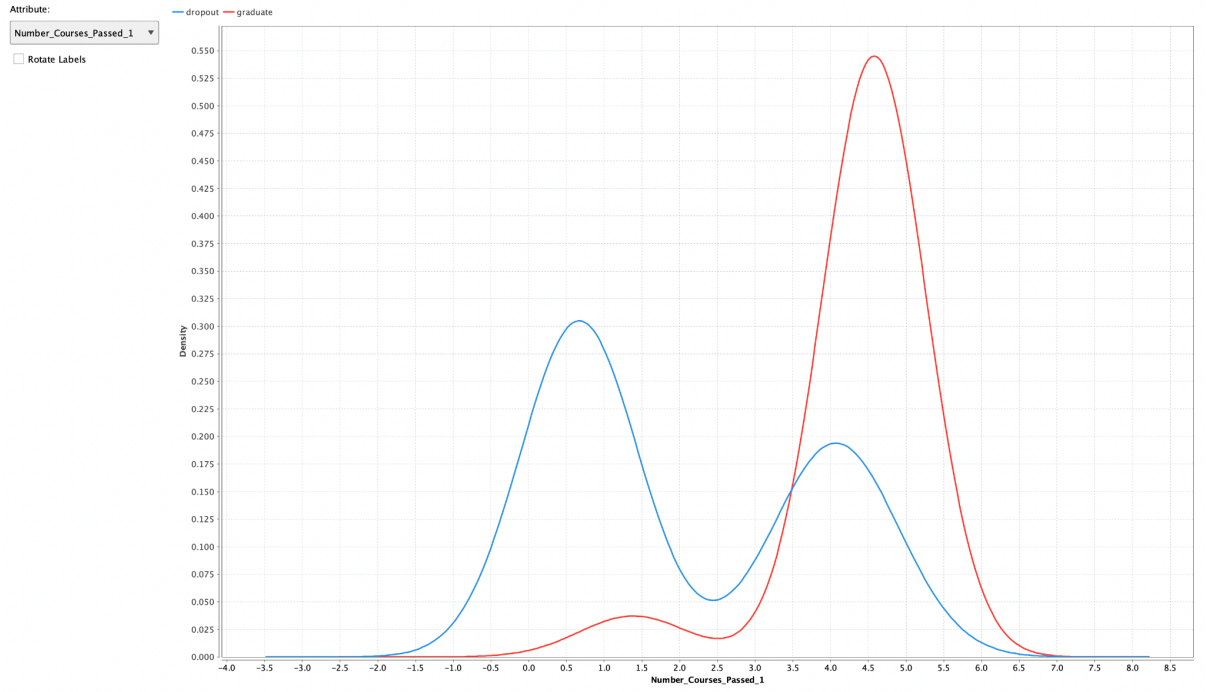


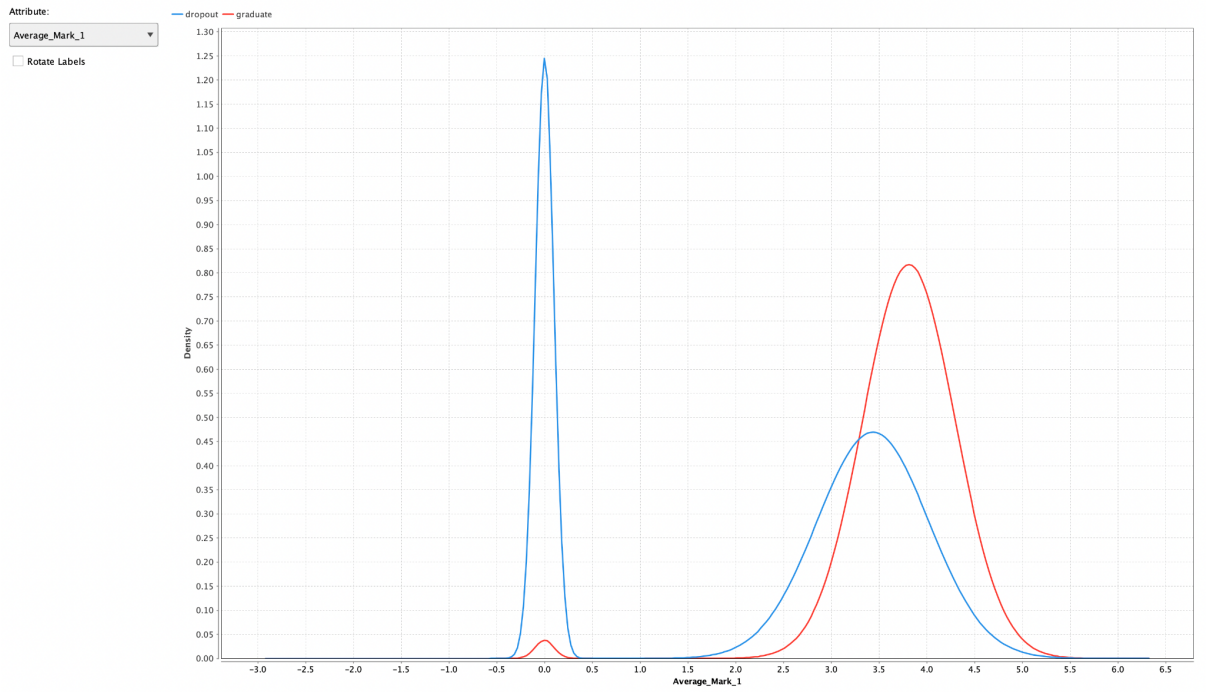
f_measure: 78.96% +/- 2.86% (micro average: 79.01%) (positive class: dropout)

	true graduate	true dropout	class precision
pred. graduate	698	209	76.96%
pred. dropout	146	668	82.06%
class recall	82.70%	76.17%	



The following plots regarding the probability measures for Number_Courses_Passed as well as Average_Mark clearly reflect the fact that students that did not pass any course in the first semester are far more likely to drop out, as the blue peaks at/around 0 indicate.
A related, although way less decisive observation can be made in the plot on the previous page regarding the feature Number_Enrollments.





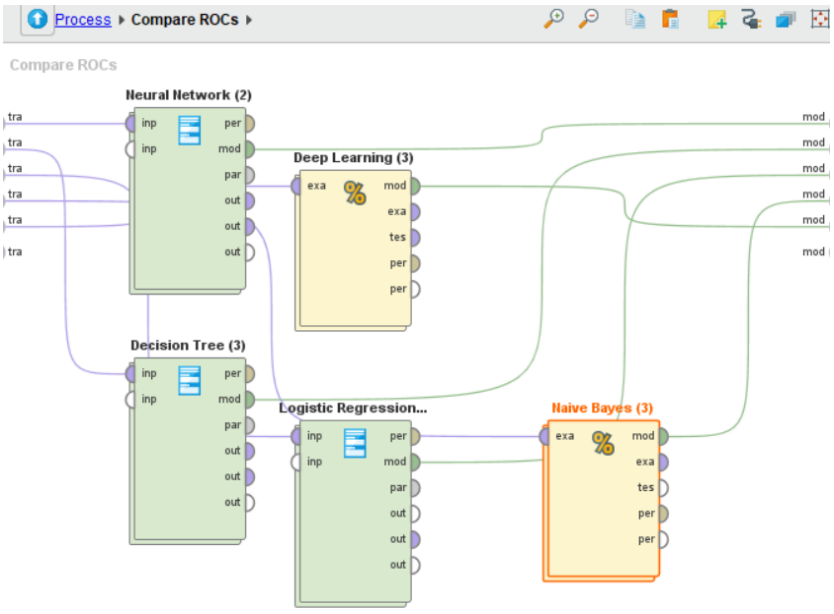
Exercice 5:

Which model (exercise 2 to 4) gives the best results? Justify your answer shortly.

	Decision Tree	Neural Network	Logistic Regression	Deep Learning	Naïve Bayes	Naïve Bayes (Kernel)	Best
F-Measure	**78.98%**	78.55%	78.93%	78.83%	72.78%	78.96%	Decision Tree
Accuracy	80.13%	79.78%	80.24%	**80.35%**	76.29%	79.37%	Deep Learning
Recall	73.20%	72.63%	72.64%	72.40%	62.36%	**76.17%**	Naïve Bayes (K.)
Precision	78.98%	85.92%	86.58%	86.70%	**87.64%**	82.20%	Naïve Bayes
Sensitivity	73.20%	72.63%	72.64%	72.40%	62.36%	**76.17%**	Naïve Bayes (K.)
Specificity	87.33%	87.22%	88.15%	88.39%	**90.76%**	82.71%	Naïve Bayes

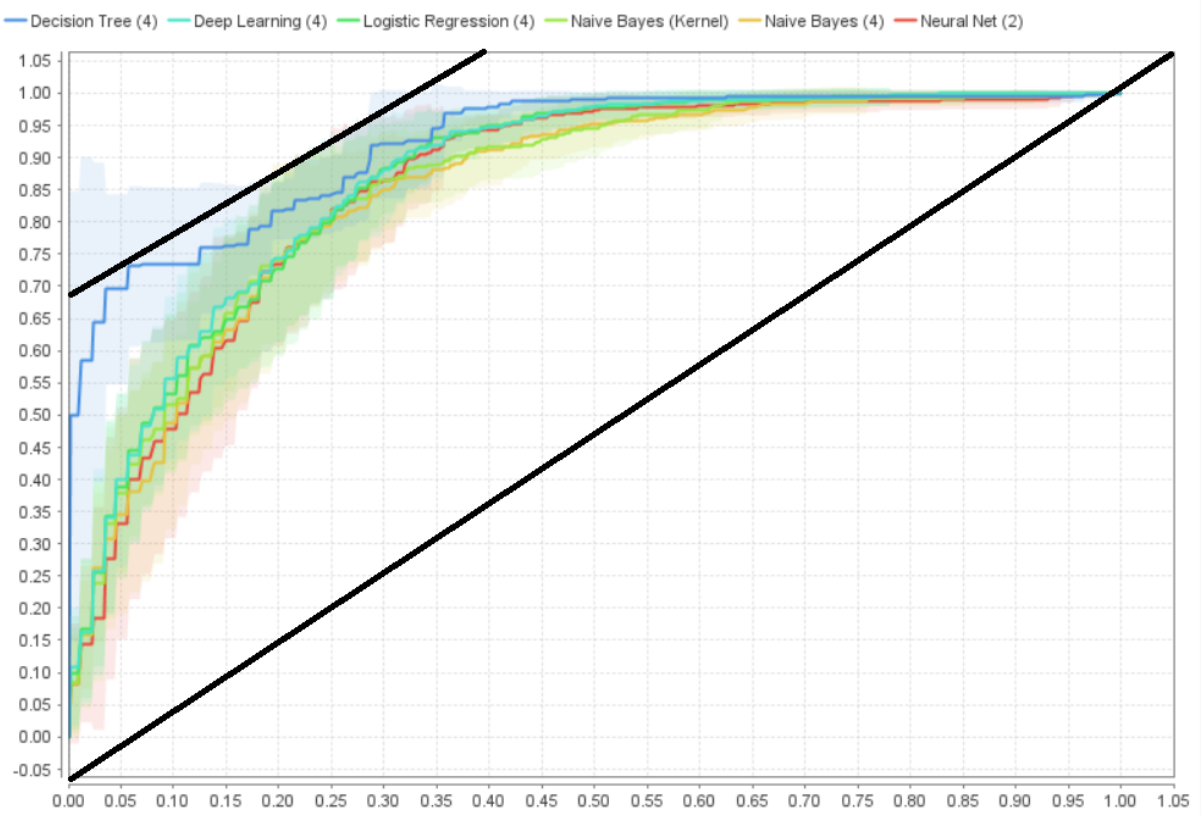
In regards to the F-Measure, the best result was achieved by the Decision Tree by an almost negligible margin to all the others. All models except the traditional Naïve Bayes are within 78.55% and 78.98% and can be considered equally good.
The contender in accuracy (by a small margin) is Deep Learning, but one of the biggest problems about Deep Learning is that because of the feed-forward and backpropagation process as well as the large number of parameters, the training is very slow and very CPU-intensive. For that reason, the Decision Tree or Naïve Bayes (Kernel) classifier are the better predictors.
In all other evaluation measures either the standard Naïve Bayes or Naïve Bayes (Kernel) had the advantage.

Comparison of ROC curves:



When comparing the ROC curves, Decision Tree also turns out to be the best classifier that has the best quality to estimate true 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.



Exercice 6:

Calculate the three features of the dataset with your own results of the first semester. What do all the models predict with your own data? Can some models provide some explanations for their prediction and which are these explanations? Do you find the predictions made by the different models trustworthy?

Number_Enrollments_1	Number_Courses_Passed_1	Average_Mark_1
5	5	3.38
5	5	4.3
5	5	5.0

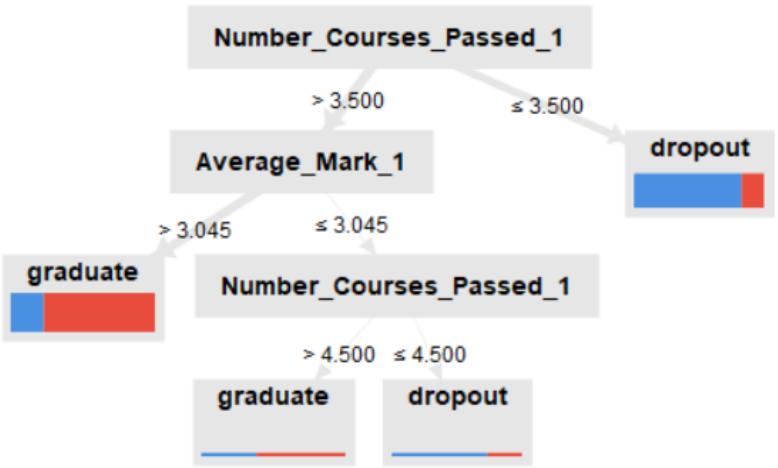
All 3 project team members have passed all 5 enrollments.

Neural Network

Row No.	prediction(cl...	confidence(...	confidence(...	Number_Enr...	Number_Co...	Average_Ma...
1	graduate	0.160	0.840	5	5	3.380
2	graduate	0.226	0.774	5	5	4.300
3	graduate	0.229	0.771	5	5	5

Decision Tree

Row No.	prediction(cl...	confidence(...	confidence(...	Number_Enr...	Number_Co...	Average_Ma...
1	graduate	0.229	0.771	5	5	3.380
2	graduate	0.229	0.771	5	5	4.300
3	graduate	0.229	0.771	5	5	5

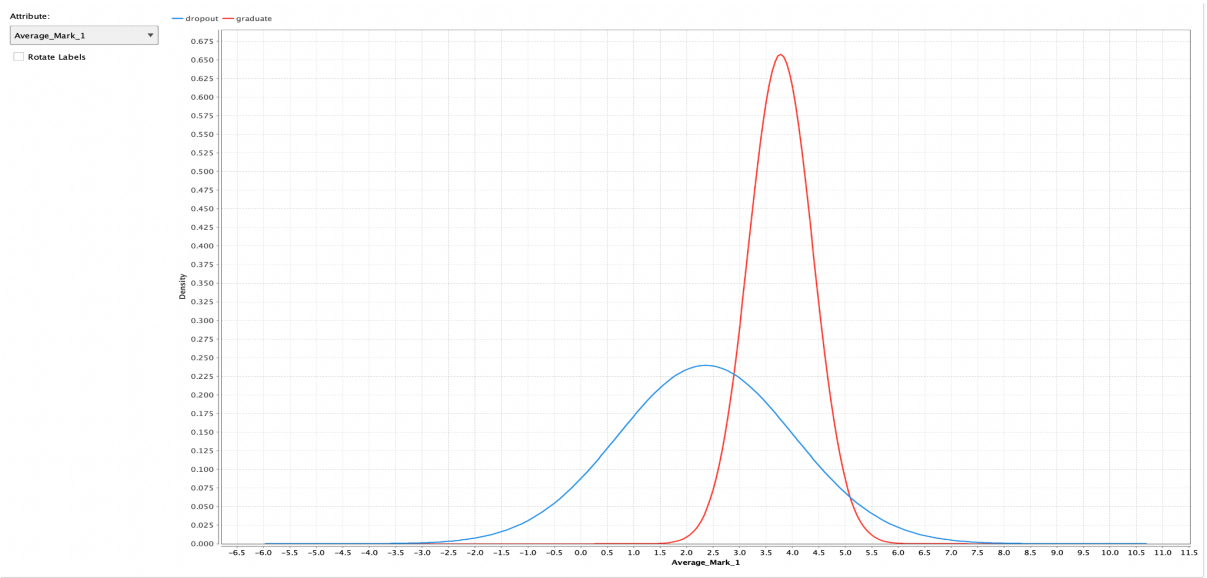


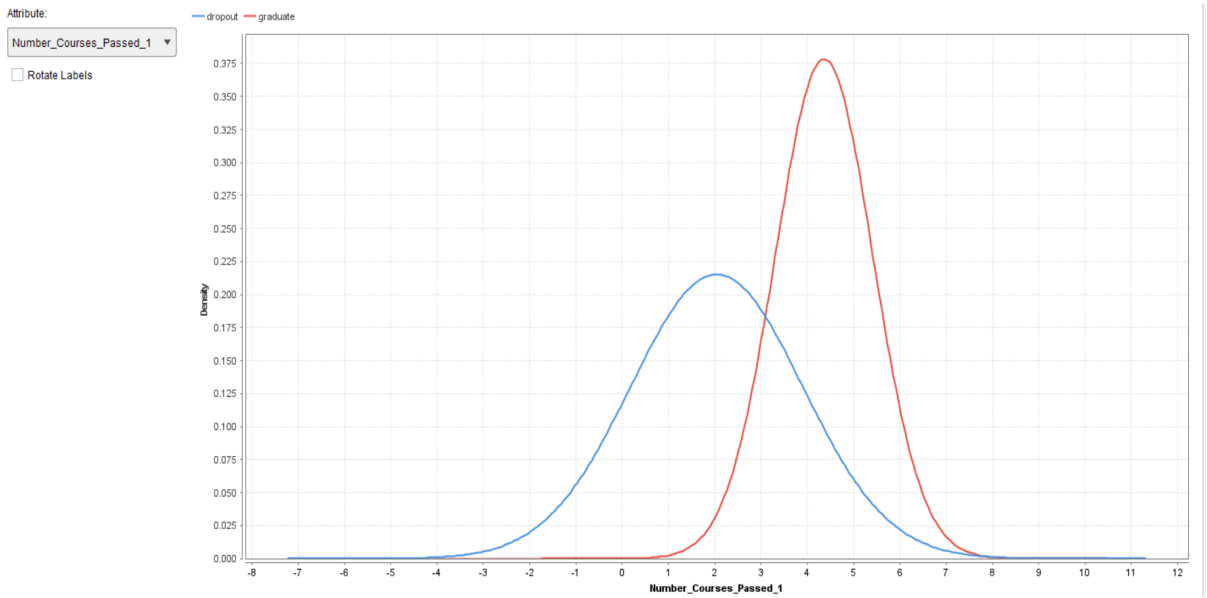
Logistic Regression

Row No.	prediction(cl...	confidence(...	confidence(...	Number_Enr...	Number_Co...	Average_Ma...
1	graduate	0.251	0.749	5	5	3.380
2	graduate	0.158	0.842	5	5	4.300
3	graduate	0.108	0.892	5	5	5

Naïve Bayes

Row No.	prediction(cl...	confidence(...	confidence(...	Number_Enr...	Number_Co...	Average_Ma...
1	graduate	0.049	0.951	5	5	3.380
2	graduate	0.035	0.965	5	5	4.300
3	graduate	0.098	0.902	5	5	5





Deep Learning

Row No.	prediction(cl...	confidence(...	confidence(...	Number_Enr...	Number_Co...	Average_Ma...
1	graduate	0.193	0.807	5	5	3.380
2	graduate	0.076	0.924	5	5	4.300
3	graduate	0.070	0.930	5	5	5

All students have been predicted as possible graduates by every single model.

From all the models, only Decision Tree and Naive Bayes give an explanation or a visualisation of the process.
Following along the path of the Decision Tree, it is explained how the outcome is decided. For example, all the students in the test data set have passed all 5 courses, and the trained model has set the threshold to <= 3.5 if the student will be a dropout. It is interesting to observe that the attribute ‘Number of enrollments’ was not used in this classification process.

Regarding the Naive Bayes classifier, since we only have three features (out of which actually only two are really of interest, as explained before), we can look at the probability measures for the attributes Number_Courses_Passed and Average_Mark and see immediately that the probability for our values in these categories are always higher for the class “graduate” (red curve).
As these probabilities are multiplied, the maximum of all probabilities combined would still be with class “graduate”. That is to say, supposing that we would graduate, we can roughly verify that hypothesis only by looking at these curves and finding the maximum (Since in our case P(H) is almost 0.5 for both classes, as our dataset is so well balanced). As in the end, this is what Bayes’ Theorem tells us in a nutshell: Maximising P(X|H) gives us the class prediction P(H|X).

Assessing the Decision Tree for the three students from above, the prediction seems rather optimistic than realistic. The results don’t necessarily represent these students since there can be occurrences in later semesters that prove the results to be falsely predicted. Later semesters tend to be harder, higher in workload, or students can choose to change their subject. This unfolds yet another topic whether students who “drop out” are actually dropouts or are just changing their paths.
Seeing it from the perspective that all models predicted the same, there seems to be a pattern and therefore a tendency. But for each model, the accuracy of the prediction is around 80%. Every 5th student gets an inaccurate prediction. Considering that, the predictor is only trustworthy with reservations.

Exercice 7:

Suppose your best model has an accuracy of 77.5% for male students and an accuracy of 76.7% for female students. Would you find this model to be fair? If not, who are disadvantaged, male students or female students?

Assuming a 50/50 split of male and female students, we look at our dataset to estimate the impact of the difference in accuracy along the genders. We find this model to be fair as it would predict approx. 267 male and 264 female students accurately and the difference is only 3 students (about 0.35% of one gender). Female students have a slight disadvantage.

Same questions if your best model has an accuracy of 75.5% for male students and an accuracy of 83.7% for female students.

Using the same reasoning as above, this model would predict approx. 260 male and 288 female students correctly. The difference of 28 students is equal to about 8.13% of one gender, one order of magnitude more than with the other model. Therefore we find this model to be unfair. Male students have a disadvantage.

Exercice 8:

In this investigation, could some bias occur at some step? Refer to all the steps described in Srinivasan, R., Chander, A.: Biases in AI Systems - A survey for practitioners. Communications of the ACM August 2021 (Vol. 64, No. 8).Pages 44-49 https://doi.org/10.1145/3464903 and argue why or why not some type of bias could occur at that step.

The different steps described in the Survey are:

- Data collection
- Data annotation and labelling
- Data preparation and processing

Based on our investigation, there is a high probability that a bias will occur at some step, in our case from the first step.

Indeed, we are trying to predict the sex of the students with the wrong collected data.
Based on the dataset that we have, we can't rely on the grades to determine if a student is a male or a female.

There is no direct or indirect relation between the grades and the sex.
We can have similar grades' combinations that can give different targets.
For example, a male and a female can have the same combination of grades (1,2,3).
=> Biases related to problem formulation

What would have been more appropriate in this data set is to use, for example, an identity photo to determine the traits that make up each sex and that can predict whether the student is male or female, a deep learning algorithm based on a neural network.
Or another example, but outside the university context, would be to identify the sex through the level of hormones present in a blood sample.
Oestrogen and testosterone for females and male respectively.

Since the bias occurred from the first step, which is the most precious one, it doesn't matter if there are biases in the other steps, since we don't have the right data.

Exercice 9:

~~If you know of universities/institutions that use such a warning system to warn students who are at risk of dropping-out, report what you know about the usefulness and acceptance of such warning systems.~~
If you don't know such universities/institutions, what do you think about the usefulness / acceptance of such warning systems?

We don't know any universities/institutions that use such a system, but try to give our thoughts regarding its potential usefulness and acceptance.
Firstly, we think that such systems cannot replace better and more detailed information and longer trial periods for freshmen and -women regarding course of studies, course selection and the courses themselves.
There are also many other factors impacting an individual's course of studies (especially its duration) like family and side jobs. We find it difficult to give a student advice regarding his or her studies without inclusion of these aspects.
Furthermore we are very worried regarding added stress such a system could impose on students. We think it's good that the system does not provide a grade prediction, because this could add pressure to actually achieve or surpass that grade in the following semester(s).
The dropout warning could trigger stress, too, however, which could lead to worse grades and maybe even a dropout that would not have happened without the warning.
Predictions should be positively worded and students should be encouraged to pursue a study of their interest, not one that has a "guaranteed" chance of graduation.
That being said, such a system could also have the potential to motivate one to go through demanding periods and/or add proportion to one's performance. They could definitely give the deciding signal when one's thinking about dropping out of a study that does not really suit her or him, too.
Ultimately we were quite undecided within the group whether we would personally use such a system. One student would use it hoping it would actually reduce their pressure to perform, another would use it only under the condition it would not include grades and a third would not use it all as they don't believe in the applicability of such a prediction on their individual situation.