1 Classification, Regression and Clustering

1.1 What are the differences between classification, regression and clustering? Which are supervised and which are unsupervised learning task?

Regression and classification are supervised learning tasks and clustering is an unsupervised learning task. Classification revolves around predicting class memberships (a discrete response variable) of instances based on a feature vector, while regression aspires to predict continuous response variables based on a feature vector. In both cases the underlying predictor or estimator is trained on a set of training instances consisting of predictor variables (features) and a response variable (e.g. class label, or continuous regressand). In contrast to the two aforementioned tasks, clustering concerns itself with grouping instances into a usually unknown number of groups based on some notion or measure of similarity or dissimilarity.

1.2 Describe a real-world big data classification example, and discuss why it is a classification problem and why it is a big data problem.

In the field of precision medicine using the history of medical records, pathology images, genomics data, past treatments, etc to determine clinical phenotypes and based on that predicting the best treatment or combination of treatments constitutes a big data classification problem in my mind. It is

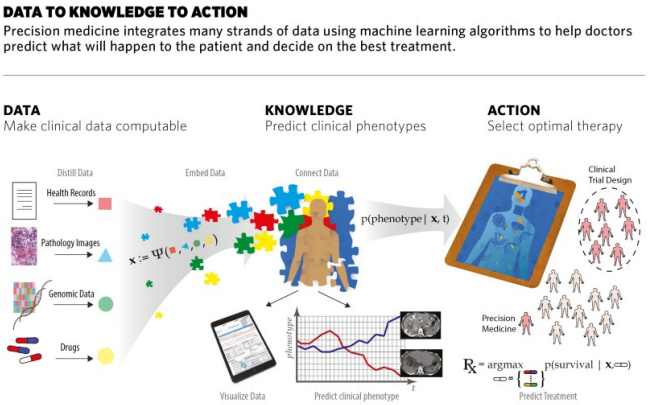


Figure 1: Source: `https://www.nature.com/articles/d42473-019-00035-5`

a classification problem due to the fact that treatment options are a discrete variable and the system would have to be trained on labeled data first. I consider it a Big Data problem due to the volume, variety and velocity of the data.

Volume One would have to include a rather large number of training samples to capture the variety of human beings (e.g. in terms of medical history or in terms of genetic variations). But even just for one individual one would also have to include a fair amount of actual data given that a full history of medical records, pathology images is amongst the predictor variables as well as genetic information. Multiplied by millions of training instances, the volume of the data set would certainly fall within the scope of what one calls big data.

Variety As eluded to already in the previous paragraph, there is a large variety of data involved in the task: the full history of medical records, pathology images is amongst the predictor variables as well as genetic information and pharmacological history. The term "history" is not just symbolizing "a complete set of what is on file" but refers as well to the time dimension of what happened when, which complicates the task even further and increases the variety of the data set.

Velocity With a sufficiently large group of individuals there would be frequent updates to medical records and new information coming in all the time. It might not be at the speed and volume of the amount of twitter messages generated every single second, but it would certainly present a challenge how to deal with everything being fluid as the life of patience goes on. As the environment and the world around us changes, so might the data set and the findings (concept drift). E.g. one could assume that a pandemic like COVID-19 might leave some artifacts behind in medical data and health outcomes. Something one would have to deal with and account for.

1.3 Describe a real-world big data regression example, and discuss why it is a regression problem and why it is a big data problem.

Trying to predict stock prices for the very near future (e.g. milliseconds, seconds, minutes) based on a multitude of market data and other data feeds (e.g. news tickers, twitter, etc) constitutes a Big Data regression task in my view. It's a regression problem as stock prices are continuous variables (with

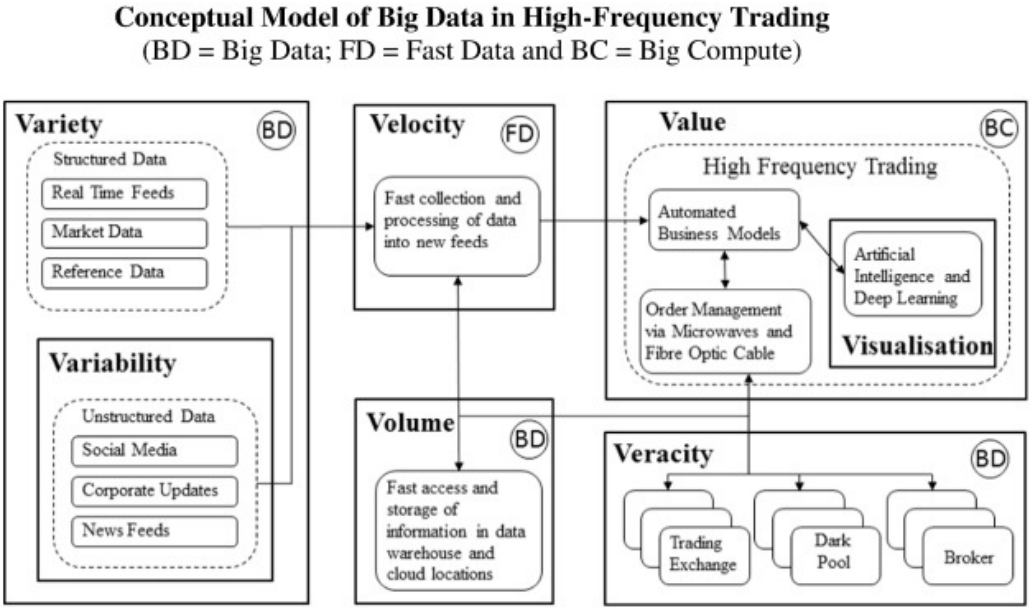


Figure 2: Source: [5]

limited precision) and such a system would have to be trained on training data with known values for the dependent variable. It is a Big Data problem because volume and variety of data is potentially "unlimited" depending on what is available and what has predictive power. As described above, real time market data, news ticker, real time satellite feeds, etc etc is all potentially within scope. High velocity is equally inherent to high frequency trading with most data sources ideally having a time resolutions in the seconds or milliseconds.

1.4 Describe a real-world big data clustering example, and discuss why it is a clustering problem and why it is a big data problem.

Coming back to the example given for a Big Data classification problem in section 1.2, one can consider grouping patience into clinical phenotypes a clustering problem if there is no up front taxonomies of these phenotypes (which could turn the task into a classification problem). If on the other hand it is unclear how many of what particular clinical phenotypes or historical medical trajectories exist, then this would constitute a clustering problem. The justification of why that is a Big Data problem is the same as per section 1.2.

2 Different Types of Feature Selection/Construction Methods

2.1 What are the major differences between feature ranking/feature weighting and feature subset selection methods?

Single feature ranking sorts n features according to some notion of individual importance or relevance in a descending order. One might then select the top m features out of the pool of n features to build a feature set. The underlying assumption of the approach is that a combination of strong individual features will lead to a strong feature subset. This assumption is problematic due to feature interactions. Weakly relevant features can become highly relevant in combination with another feature and vice versa. In statistics this problem is well known as multicollinearity as well as the notion of suppressor and enhancer variables. Possible metrics for feature ranking are correlation or mutual information as well as the results of a χ^2 test. In contrast, feature subset selection evaluates joint feature sets instead of just singular features. This allows for global feature interaction to be taken into account, as whole subsets are evaluated (including the feature interactions therein). One potential drawback of this approach is the curse of dimensionality as all possible feature combinations would have to be tried out if one wants to find the best combination. Therefore one important design decision is the strategy for exploring the feature subset search space. Two widely used ones are sequential forward selection (SFS) [6] and sequential backward elimination (SBE) [3]. SFS starts with the single best performing feature and increases the number of features by one at a time. For each increase in the number of features the algorithms tests through all remaining features to determine which additional features provides the highest marginal benefit when added to the existing feature set. SBE does the opposite by starting with a full feature set and then shrinking the featured set by one feature at a time. Determining which particular feature to drop at each shrinkage step is the computationally intensive part. Examples of feature subset selection is e.g. the CFS described in section 2.4 as well as any subset selection performed in a wrapper method (something described in the following section).

2.2 Define filter, wrapper, and embedded feature selection/construction approaches.

The filter approach is a feature selection system that does not involve any learning (e.g. classification) algorithm during the feature selection process but uses some mathematical heuristic instead. Examples are the correlation based feature selection method described in 2.4. Another example is the Minimum Redundancy-Maximum Relevance filter, that uses the sum of the mutual information between class(es) and features as well as the sum of the mutual information between features as a measure of relevance and redundancy. Both metrics are then combined to one single metric either as a quotient or a difference. In the wrapper approach [2] the feature selection mechanism uses a classification or regression algorithm as an evaluation module. For each feature subset candidate the classifier or model has to be retrained and one then compares accuracy results for the different feature subsets. This is potentially very costly as one has to retrain the model from scratch for every single feature subset. Any classifier can be used as part of the wrapper approach. Probably the most interesting ones would be those most prone to irrelevant or noisy features, like NB or instance-based learning algorithms (e.g. KNN). Less interesting would be classifiers that fall under the embedded methods. The search strategy for the feature subset space is an important aspect of these methods. Those were discussed already in section 2.1. Embedded feature selection takes place when classifier or regression methods implicitly perform feature (subset) selection as part of how they work. An example is a simple decision tree. Decision trees even take local feature importance into account as part of their implicit feature selection process. Another example could be sparse logistic regression, where the number of variables is penalized as part of the parameter estimation process.

2.3 Compare the performance of filter, wrapper and embedded methods in terms of the classification accuracy, the computation cost, and the generality to different classification methods, and explain why.

With reference to the lecture slides and figure 3 therein, one can describe the differences of the 3 methods as follows. The wrapper approach adapts the selection of the features to the particular

algorithmic bias of whatever classification algorithm is used inside the wrapper. As a result of it, this method tends to have the best classification accuracy but due to the bias also the worst generalisation performance of the selected features. Apart from that the computational cost tends to be high, because for the exploration of the feature subset search space each time the classifier has to be retrained. The filter method on the other hand tends to be computationally less demanding as no classifier needs to be trained. As oftentimes more heuristic relevance or importance metrics are used, the classification accuracy tends to be lower than for the other methods as no algorithmic bias of a particular classifier is taken into account as part of the feature selection process. The upside of this is that the generalisation performance tends to be the highest form all three methods. The embedded methods sit in the middle for all three performance dimensions. The computational cost is higher than for the filter, as after-all at least one classifier has to be trained, but by far not as many as for the wrapper method. Particularly if a rather simple classifier is used as part of the embedded method (e.g. a decision tree) medium classification performance and medium generalisation performance is achieved. As with all general statements in life or any attempt to "put things into semantic boxes", I would argue that one can always find certain counter examples to the above statements. So these are probably more to be seen as broad stroke "stylized facts" rather ultimate facts without exceptions.

	Classification Accuracy	Computational Cost	Generality (to different classifiers)
Filter	Low	Low	High
Embedded	Medium	Medium	Medium
Wrapper	High	High	Low

Figure 3: Source: AIML 427 Lecture script, Week 2 1/2, page 14

2.4 Describe the main idea of Correlation-based feature Selection method (CFS), including the two search methods that can be used with it. Is CFS a feature ranking or feature subset selection method? Is it a filter, wrapper, or embedded method? And explain why.

The CFS as described in [1] is a correlation based heuristic to evaluate the worth of feature **subsets**. It is therefore a subset selection method and not "just" a feature ranking algorithm. The main aim of CFS is to select a set of features that are highly correlated with the class but share little correlation with each other [1, p. 2]. Mathematically the average feature-class correlation gets normalized by the average feature-feature correlation (as measure of redundancy). CFS is a filter approach, as a mathematical heuristic gets employed for the evaluation of feature subsets, rather than the direct performance of a classifier trained on these subsets. One potential draw back of CFS is, that it is a global feature selector and might eliminate candidates that are only locally relevant for some areas of the instance space (e.g. in a small sub-branch of a decision tree). As potential search strategy for feature selection [1] mentions two strategies, namely greedy hill climbing and best first search. Greedy hill climbing comprises well known approaches like sequential forward selection (SFS) [6] and sequential backward elimination (SBE) [3]. This strategy considers only local changes to the current feature subset (e.g.the addition or deletion of a single feature at a time). It is a greedy search strategy as it doesn't backtrack on decisions (addition or exclusion of features) once they are taken, even if that leads down a path that is globally sub-optimal. An alternative is the so called best first search [4]. This algorithms adds one feature at a time, but it is allowed to backtrack when the current branch of the feature search tree isn't leading to improvements anymore (or is exhausted). The algorithm reverts back to feature subsets that were previously considered to be inferior candidates and continues its exploration from thereon down a new path in the search tree. Given enough time the algorithm will end up exploring the whole search space. It is therefore common to stop it after it reverted back X times in a row (e.g X=5) without improving the current best result. This search strategy is what the inventor of CFS ultimately chose as it lead to slightly better results than SFS.

3 Use KNIME to perform feature Selection [45 marks]

(i) Use KNIME to:

i(a) calculate the accuracy of the Naive Bayes (NB) classification algorithm by using the given training set (i.e. Ionospheretrain.csv) to train the algorithm and test the learnt classifier on the given test set (i.e. Ionospheretest.csv). Report both the training and testing accuracy,

As can be seen in the screenshot in figure 4 the accuracy on the training set is 0.829 and the on test set 0.802.

i(b) split the whole dataset (i.e. Ionosphere.csv), using a random seed of '202203', to 70% as the training set and the other 30% as the test set, and calculate the accuracy using NB. Report both the training and testing accuracy,

As can be seen as well in the screenshot in figure 4, when using the random split with the above seed, the accuracy on the training set is 0.861 and on the test set 0.858

i(c) and compare the results of (a) and (b), and discuss your findings.

The accuracy for the data sets resulting from the random split is better (for the training as well as the test set) compared to the "pre-split" data sets that were provided as part of the assignment. This can be explained due to the fact different partitioning of a data set invariably introduces different sample bias and different patterns of idiosyncratic noise. So between a) and b), each partition of the data contains it's own unique bias (or idiosyncratic noise) which might let the classification algorithm perform better or worse. In the case of the Naive Bayes classifier, we have to remember that no implicit feature selection takes places. Each feature gets to have it's say in a multiplicative fashion. A feature that is completely uncorrelated with the class membership, might suddenly become correlated or inversely correlated with the class if the partitioning of the data over-samples or under-samples certain types of instances by chance. Suddenly one might have a collection of data points where certain features end up corrupting other features inside the NB equation, while a different partitioning of the data might lead to a combination of data points where this effect is not or less present. So performance varies depending on the unique set of data points that end up in the training and test set.
Despite the above differences between a) and b), both result sets have in common, that the accuracy on the training data set it better than on the test data set. This is usually the case, (though not guaranteed) as during the training of the classifier the decision boundaries get partly influenced by the unique sample bias (idiosyncratic noise) present in the training data set (in comparison to the full data set or the real underlying concept). These trained decision boundaries might then not match 100% with the idiosyncratic noise present in the test data set, leading to a lower accuracy on the test data set. The overall take away here is that a single random split is better than not splitting the data at all, but performing several random splits or using cross validation techniques gives a better idea about the range the accuracy values can take when one operationalizes a classifier in production / in the real world.

(ii) If a student uses the Forward Feature Selection meta node in KNIME to select features on the whole dataset (i.e. Ionosphere.csv) with the default settings. S/he uses the forward feature selection strategy, NB as the wrapped classifier, and select features which lead to the best classification performance. Report the selected features. Is there anything wrong during this feature selection process? If so, briefly discuss the reasons.

As can be seen as well in the screenshot in figure 5 and 6 the following features were selected in my case: F3, F4, F15, F25, F26
F1 was included in the second screenshot part of the KNIME setting to include static columns (like class and ID). As the feature is always 0 it was considered a static column by KNIME. So F1 was

not included as a feature but as a static column. We can see as well in the first screenshot that F1 wasn't considered a feature as the forward selection starts with F4 and F1 doesn't show up as an added feature until we reach the optimum at F15.
As there is a random split of the data within the feature selection module (to get out-of-sample accuracy estimates), the above results might differ every single time the algorithm is run. I did not set a random seed as the instructions stated to leave default settings in place.
The approach taken above (as per instructions) is problematic due to the fact that the whole data set is used for the purpose of feature selection. This introduces a feature selection bias into the process in the sense that all data was used during the training process (for the purpose of feature selection). There is no "untainted" test set available anymore. If one were to use the supplied Ionospheretest.csv test set, any accuracy score would not be a true reflection anymore of accuracy on completely unseen data. So the generalisation performance would remain unclear or would likely be overestimated by the tainted test set.

(iii) Use KNIME to process data: transform the given training set (i.e. Ionospheretrain. csv) and the given test set (i.e. Ionospheretest.csv) by keeping only the features selected in Question (ii). Calculate the accuracy of NB on the transformed training and test sets. Report the accuracy, compare the results with that calculated in Question i(b), analyse the differences and provide discussions.

As can be seen as well in the screenshot in figure 7 the classification accuracy for the above approach is 0.918 for the training set and 0.887 for the test set. Both values are higher (better) than their corresponding values from section i(a) (0.829 on Ionospheretrain.csv and 0.802 on Ionospheretest.csv) and section i(b) (0.861 for the training and 0.858 for the test set when a random split was applied on Ionosphere.csv). The results indicate that feature selection leads to an improvement of classification accuracy in our case. This suggests that the data set contains noisy features that are detrimental to the performance of NB. As NB doesn't perform any implicit feature selection, it is forced to take into account all features (good or bad) in a multiplicative fashion and is therefore prone to be negatively affected by noisy features. One could argue that the accuracy of 0.887 on the test set is potentially over-optimistic due the feature selection bias as discussed in the previous section.

(iv) Use KNIME to perform wrapper feature selection: Use the Forward Feature Selection meta node with NB as the wrapped classifier to perform feature selection on the given training set (i.e. Ionospheretrain.csv), and test the selected features on the given test set (which is to use the selected features to transform Ionospheretrain.csv and Ionospheretest.csv, then calculate the classification accuracy using NB) then:

iv(a) report the selected features, and the NB classification testing accuracy,

As can be seen in figure 8, the training accuracy is 0.902 and the test accuracy is 0.887. The selected features are F4, F6, F7, F13, F18, F23 and F26 (not visible in figure 8).

iv(b) compare the obtained accuracy with that obtained in Question i(a), analyse the differences and provide discussions,

Both values are higher (better) than their corresponding values from section i(a) (0.829 on Ionospheretrain.csv and 0.802 on Ionospheretest.csv). The results indicates that feature selection leads to an improvement of classification accuracy in our case. This suggests that the data set contained noisy features that are detrimental to the performance of NB. As NB doesn't perform any implicit feature selection, it is forced to take into account all features (good or bad) and is therefore prone to be negatively affected by noisy features.

iv(c) compare the results (i.e. the selected features and the classification accuracy) with that obtained in Question iii, analyse the differences and provide discussions.

The selected features are F4, F6, F7, F13, F18, F23, F26, compared to F3, F4, F15, F25, F26 from section (iii) (noting that the feature selection actually took place in section (ii)). F4 and F26 are the only features present in both feature sets. The classification accuracy for the above approach (as already mentioned) is 0.902 for the training data set and the test accuracy is 0.887. The accuracy in section (iii) is 0.918 for the training set and 0.887 for the test set. It is interesting to see that the accuracy on the test set is virtually identical, but that the accuracy of the training data set is below that of section (iii). That is slightly counter-intuitive as a feature selection that was performed exclusively on the training data set (particularly when a wrapper approach is used) should lead to a higher training data accuracy than a feature selection that was performed on the whole data set. It can maybe be explained through the additional random split inside the feature forward selection meta node. This node splits the training set into a training-training and a training-test set for the purpose of feature selection. This (again) creates unique idiosyncratic noise patterns inside these data partitions. This can lead to a choice of features that perform well in the training-test set inside the forward feature selection meta node, but not overly well on the whole training set.

(v) Use the C4.5 decision tree to perform classification on the given training set (i.e. Ionospheretrain.csv) and the given test set (i.e. Ionospheretest.csv), report the generated tree (provide the screenshot of "Decision Tree View (simple)"), the training accuracy, the test accuracy, and analysis how decision tree can be used for feature selection.

The screenshot of the decision tree is provided in figure 9. As can be seen from figure 10, the training accuracy reached an impressive 0.992, while the test accuracy reached 0.840. Given the considerable gap between the two figures one could assume some slight over-fitting on the training data. If one activates MDL pruning of the tree (c.f. figure 11), the training accuracy drops down to 0.922 but the test accuracy raises to 0.868. Given that the gap between training accuracy and test accuracy narrowed from both sides, it's fair to say that the amount of over-fitting was reduced. From figure 9 we can see how the decision tree performs implicitly a feature selection. The only features being used by the tree are the following 11 features: F3, F4, F5, F7, F16, F20, F21, F26, F30, F32, F33. Given that the original feature set contained 33 features (32 if one removed the static column of F1), this constitutes a considerable reduction in the number of features. The decision tree does this selection implicitly by how it works, as for each additional branch (starting from a leaf) the usefulness of all features are assessed for the purpose of creating a new branch. If a feature isn't de-facto useful in that context, it simply isn't used. One advantage of this type of feature selection is that locally relevant features can "make the cut". E.g. a feature might be not very relevant in the bigger scheme of things, but only be relevant in one single branch. The decision tree would still use this feature if it is useful even for one branch, while global feature selection algorithms might discard such a feature because it was only relevant for a very small partition of the instance space.

(vi) Use PCA (the PCA Compute node) in KNIME to perform dimensionality reduction: transform data using PCA on the given training set (i.e. Ionospheretrain.csv), then

vi(a) report the best 5 principle components,

The 5 principle components with the largest eigenvalues are shown in table 1 in the Annex.

vi(b) use the best 5 principle components to transform Ionospheretrain.csv and the given test set (i.e. Ionospheretest.csv), calculate and report the accuracy of C4.5 on the transformed training and test sets,

The results of this step are shown in figure 12. The training accuracy is a rather impressive 0.9755, but that drops to 0.840 for the test set (for an unpruned tree).

vi(c) compare the training accuracy and testing accuracy with those obtained in Question v, analyse the differences and provide discussions.

These results are somehow comparable to those of section (v). The training accuracy is lower when using the PCA transformed data, while the test accuracy remains the same (at least for an unpruned tree). Performing a dimensionality reduction might deprive the decision tree from the oportunity to make use of locally relevant features as discussed in section (v). That would explain why the training accuracy is lower. It is interesting though that the test accuracy is almost identical between the two approaches. That said, we know that the test accuracy of the approach without PCA can be improved by pruning the tree to remove overfitting. In that case the PCA accuracy of the test set would be below that accuracy of an approach without PCA. That would make sense due to the potential "removal" of locally relevant features due to dimension reduction.

(vii) Use KNIME to perform Correlation-based Feature Selection method (CFS): Use Rank Correlation node with the Spearman's rank correlation to measure the feature-class and feature-feature correlations. You can create a new meta node (following Forward Feature Selection meta node), for CFS on the given training set (i.e. Ionospheretrain.csv). Test the selected features on the given test set (i.e. Ionospheretest.csv) using NB. Report the selected features and the classification accuracy, compare them with those obtained in Question iv, analyse the differences and provide discussions.

Figure 13 and 14 show the implementation of CFS with sequential forward selection. Figure 14 shows the inner workings of the forward feature selection meta node. I'm sure this could have been solved more elegantly but I think it does what it needs to do. The features selected by CFS are F2, F3, F4, F6, F7, F13, F30 and F32. I tried sequential forward selection as well as sequential backward elimination and received the same results. The training accuracy is 0.918 and the test accuracy 0.915. These are rather impressive results in comparison to many of the other approaches (particularly for the test data set) but particularly the small difference in accuracy between the training and the test data set is impressive and illustrates the good generalisation properties of the filter approach.
The features selected in section (iv) by means of a wrapper method were: F4, F6, F7, F13, F18, F23, F26. That means features F4, F6 and F13 are shared between the two approaches (wrapper with NB vs. CFS). The training accuracy from section (iv) was 0.902 and the test accuracy was 0.887. That means that both training and test accuracy of the CFS filter approach were better than the ones from the wrapper approach in section (iv). That is an interesting finding and to a degree unexpected, as wrapper approaches are supposed to have superior classification accuracy. One explanation might be that the forward feature selection procedure got stuck in a local maximum (due to the greedy search strategy).

References

[1] Mark A. Hall. "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 359–366. ISBN: 1558607072.

[2] Ron Kohavi and George H. John. "Wrappers for feature subset selection". In: *Artif. Intell.* 97.1-2 (1997), pp. 273–324.

[3] T. Marill and D. Green. "On the effectiveness of receptors in recognition systems". In: *IEEE Transactions on Information Theory* 9.1 (1963), pp. 11–17.

[4] Elaine Rich and Kevin Knight. *Artificial intelligence (2. ed.)*. McGraw-Hill, 1991, pp. I–XVII, 1–621.

[5] Jonathan J.J.M. Seddon and Wendy L. Currie. "A model for unpacking big data analytics in high-frequency trading". In: *Journal of Business Research* 70 (2017), pp. 300–307.

[6] A.W. Whitney. "A Direct Method of Nonparametric Measurement Selection". In: *IEEE Transactions on Computers* C-20.9 (1971), pp. 1100–1103.

Annex

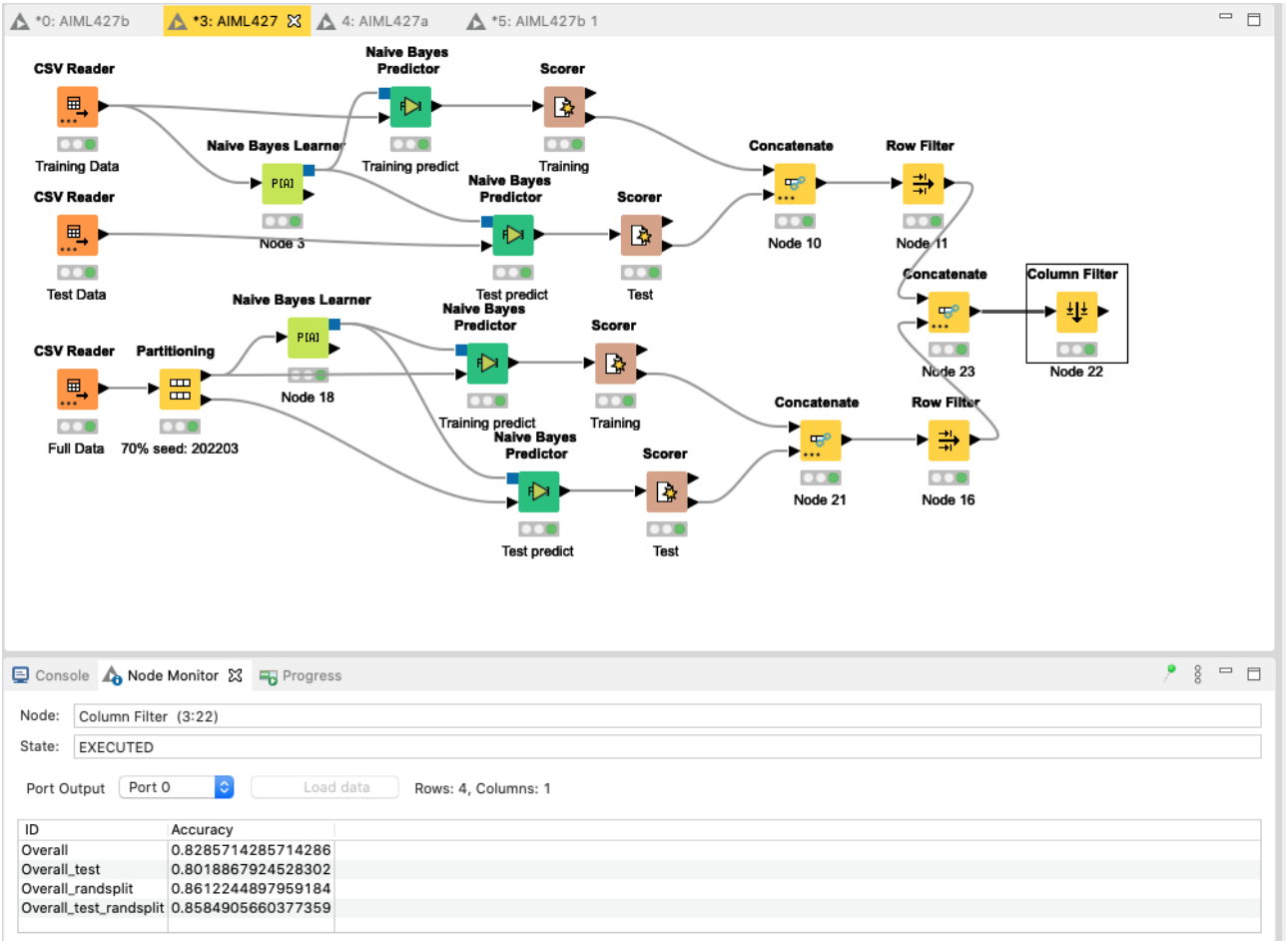


Figure 4: KNIME results for question 3.i.a, .b and .c

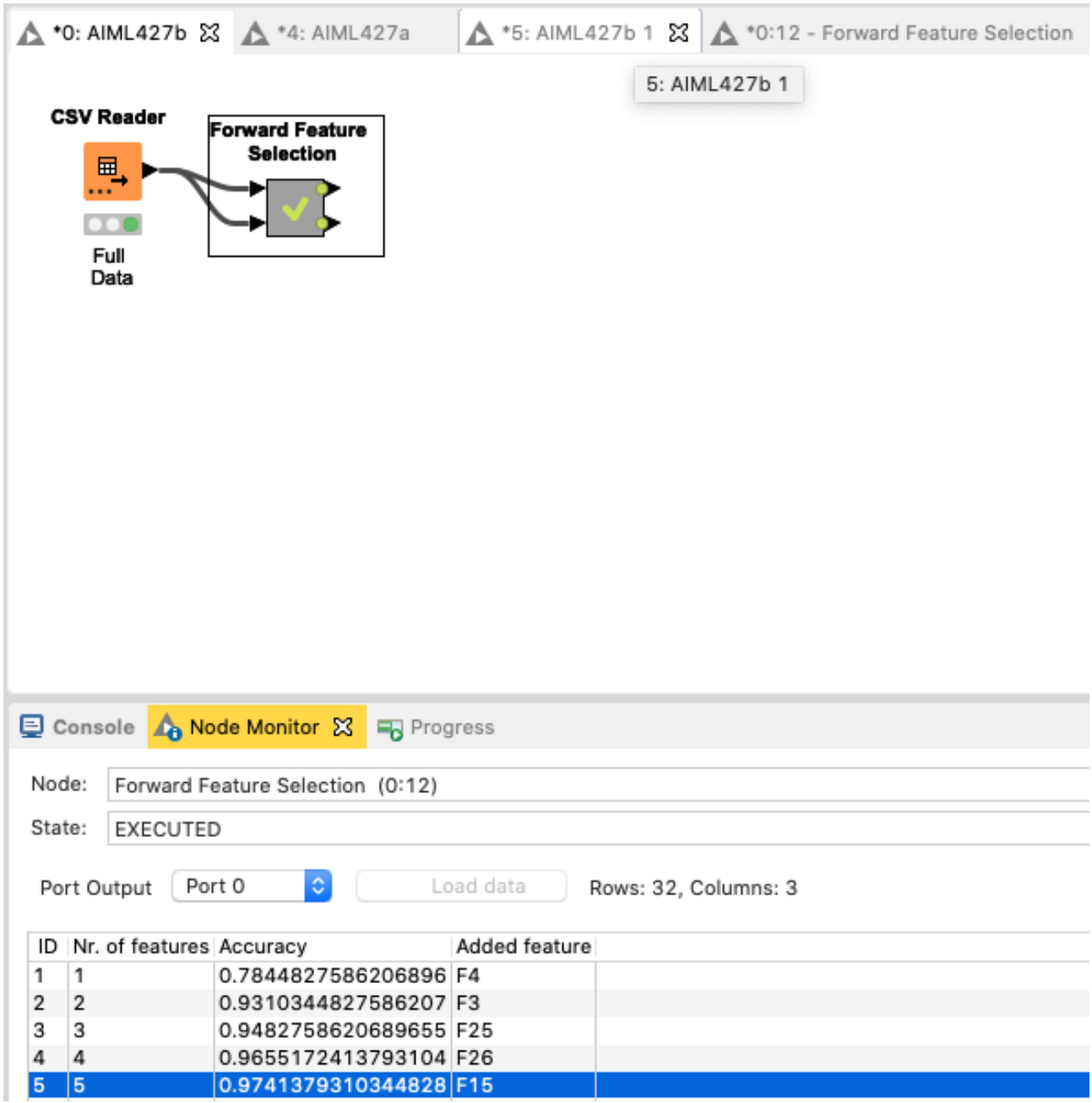
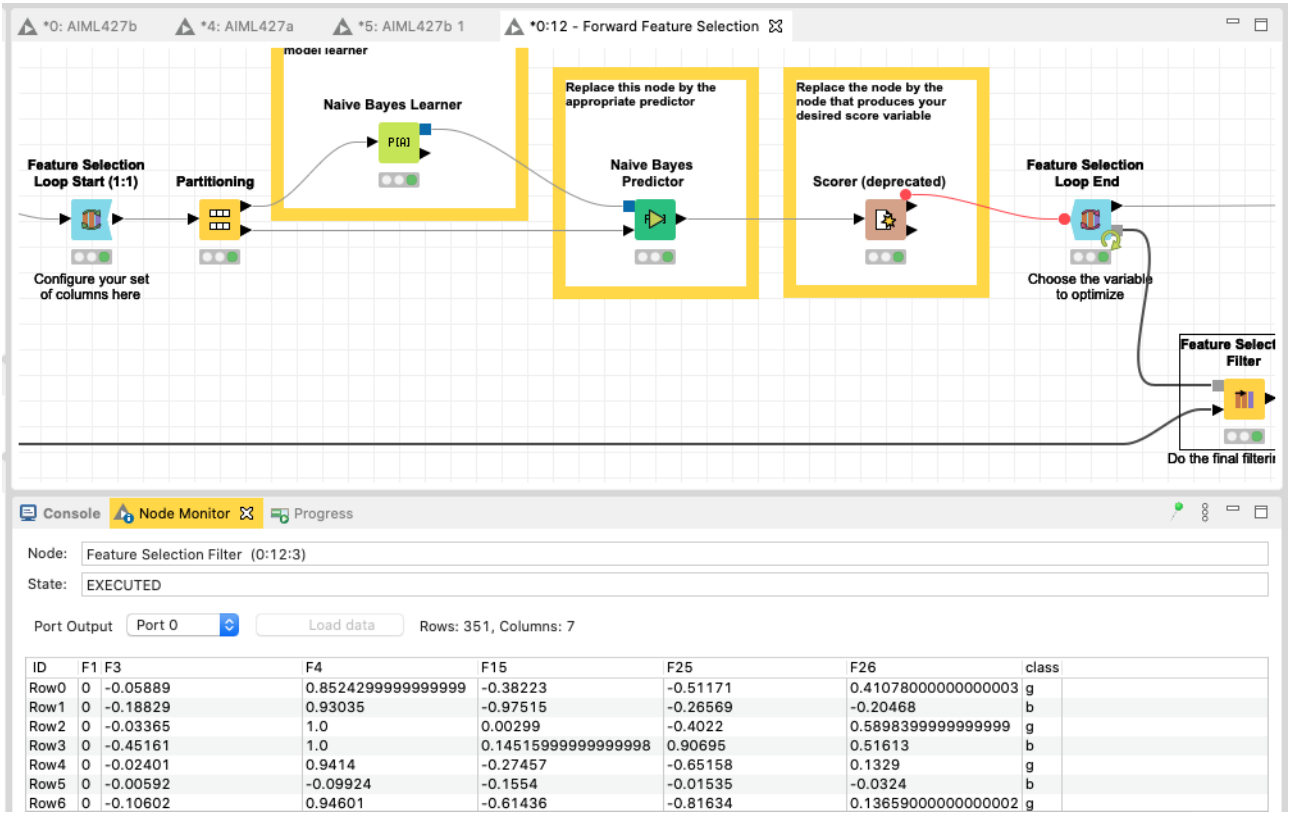
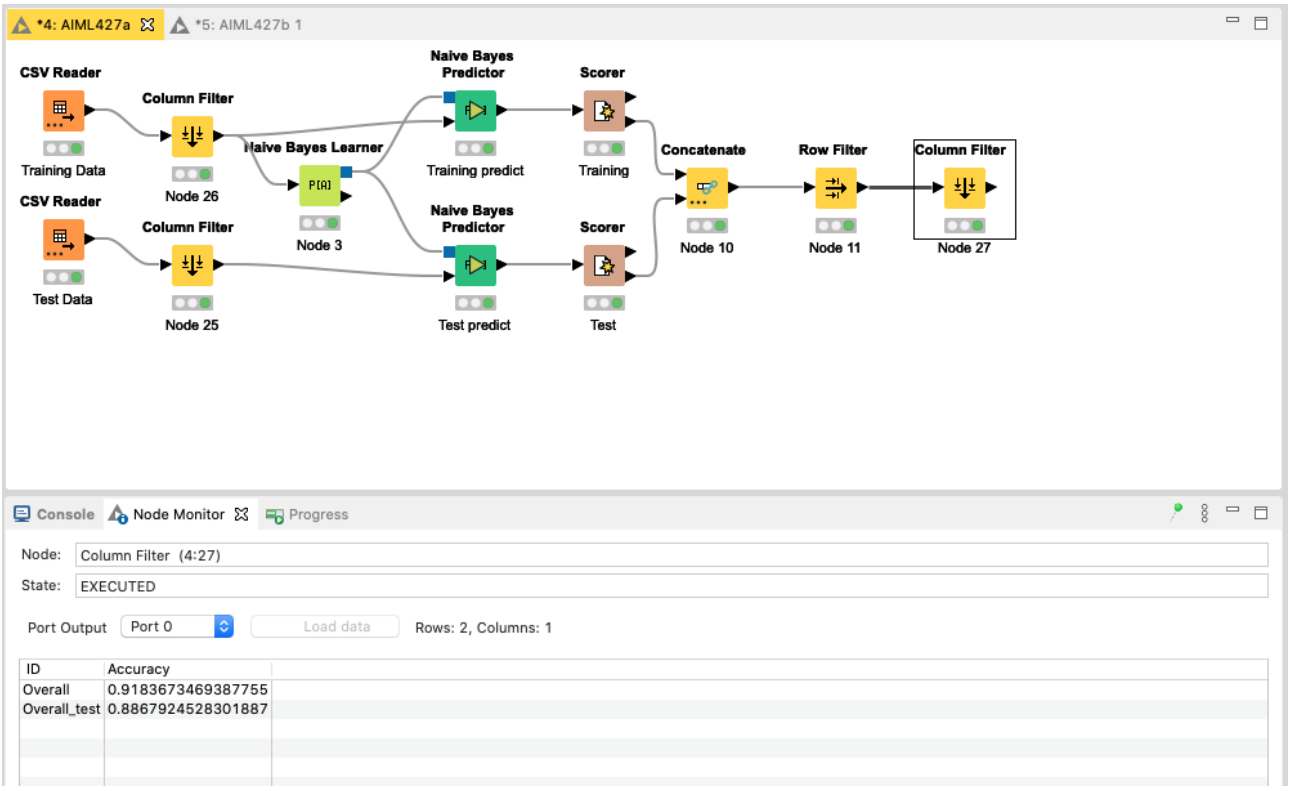


Figure 5: KNIME results for question 3.ii



ID	F1	F3	F4	F15	F25	F26	class
Row0	0	-0.05889	0.8524299999999999	-0.38223	-0.51171	0.41078000000000003	g
Row1	0	-0.18829	0.93035	-0.97515	-0.26569	-0.20468	b
Row2	0	-0.03365	1.0	0.00299	-0.4022	0.5898399999999999	g
Row3	0	-0.45161	1.0	0.14515999999999998	0.90695	0.51613	b
Row4	0	-0.02401	0.9414	-0.27457	-0.65158	0.1329	g
Row5	0	-0.00592	-0.09924	-0.1554	-0.01535	-0.0324	b
Row6	0	-0.10602	0.94601	-0.61436	-0.81634	0.13659000000000002	g

Figure 6: KNIME results for question 3.ii



ID	Accuracy
Overall	0.9183673469387755
Overall_test	0.8867924528301887

Figure 7: KNIME results for question 3.iii

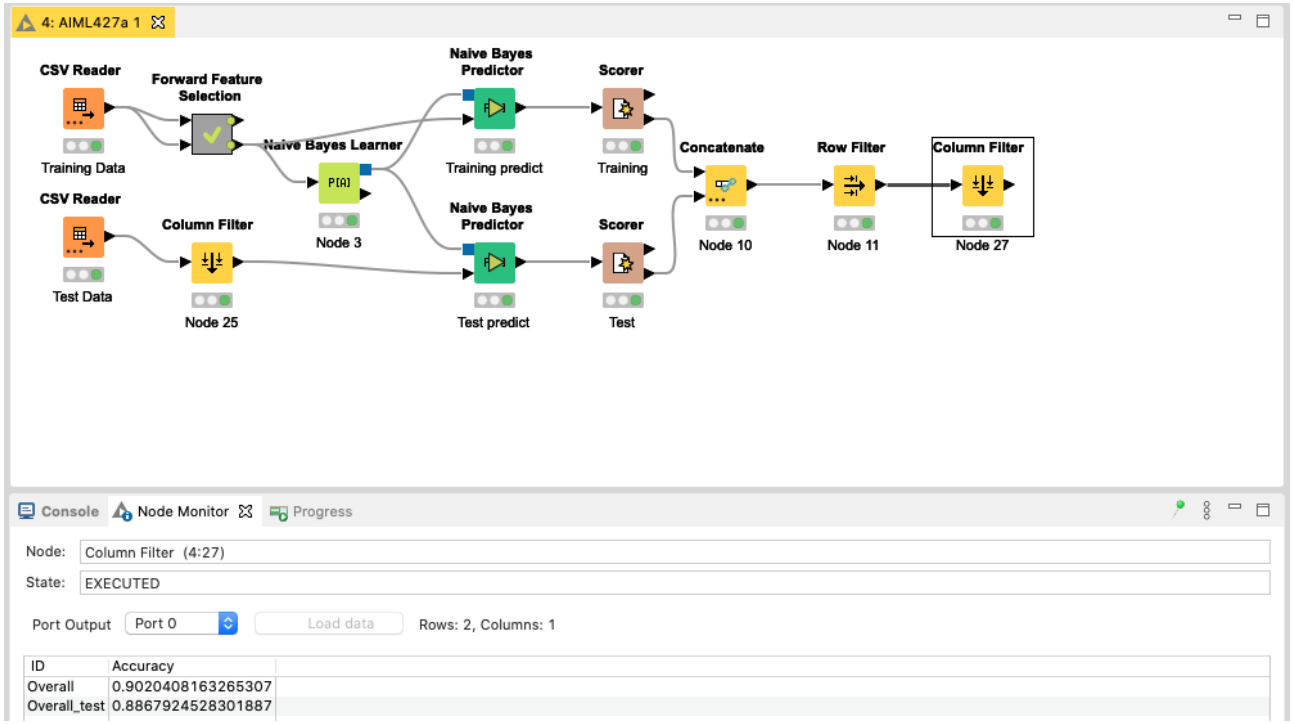


Figure 8: KNIME results for question 3.iv

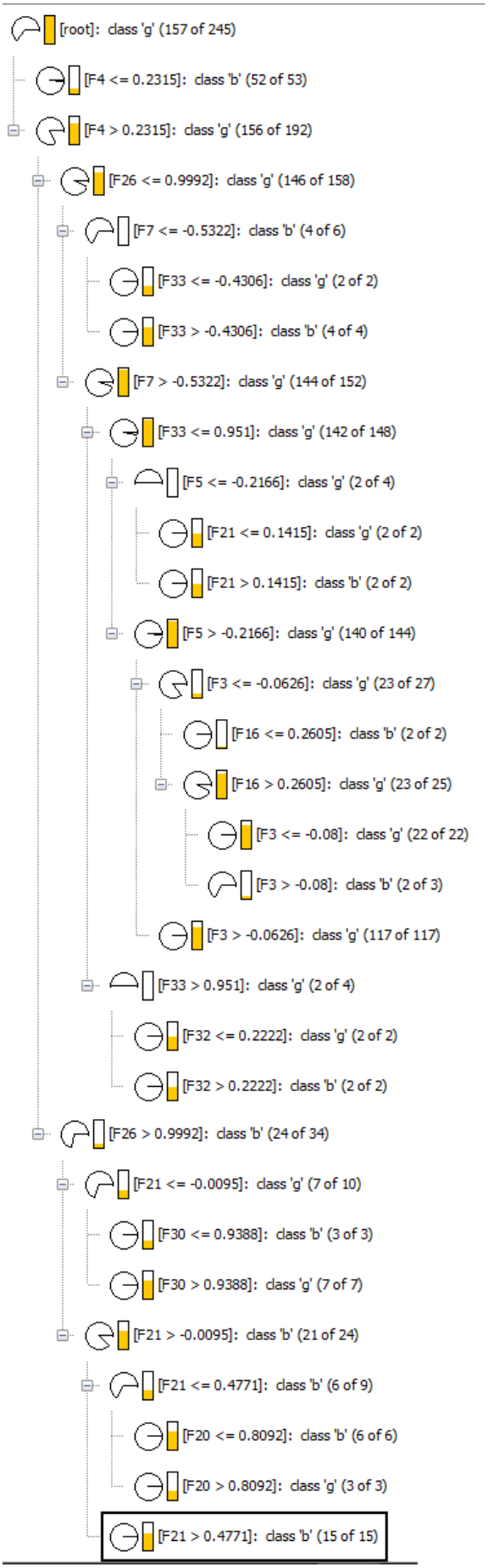


Figure 9: KNIME results for question 3.v (C4.5)

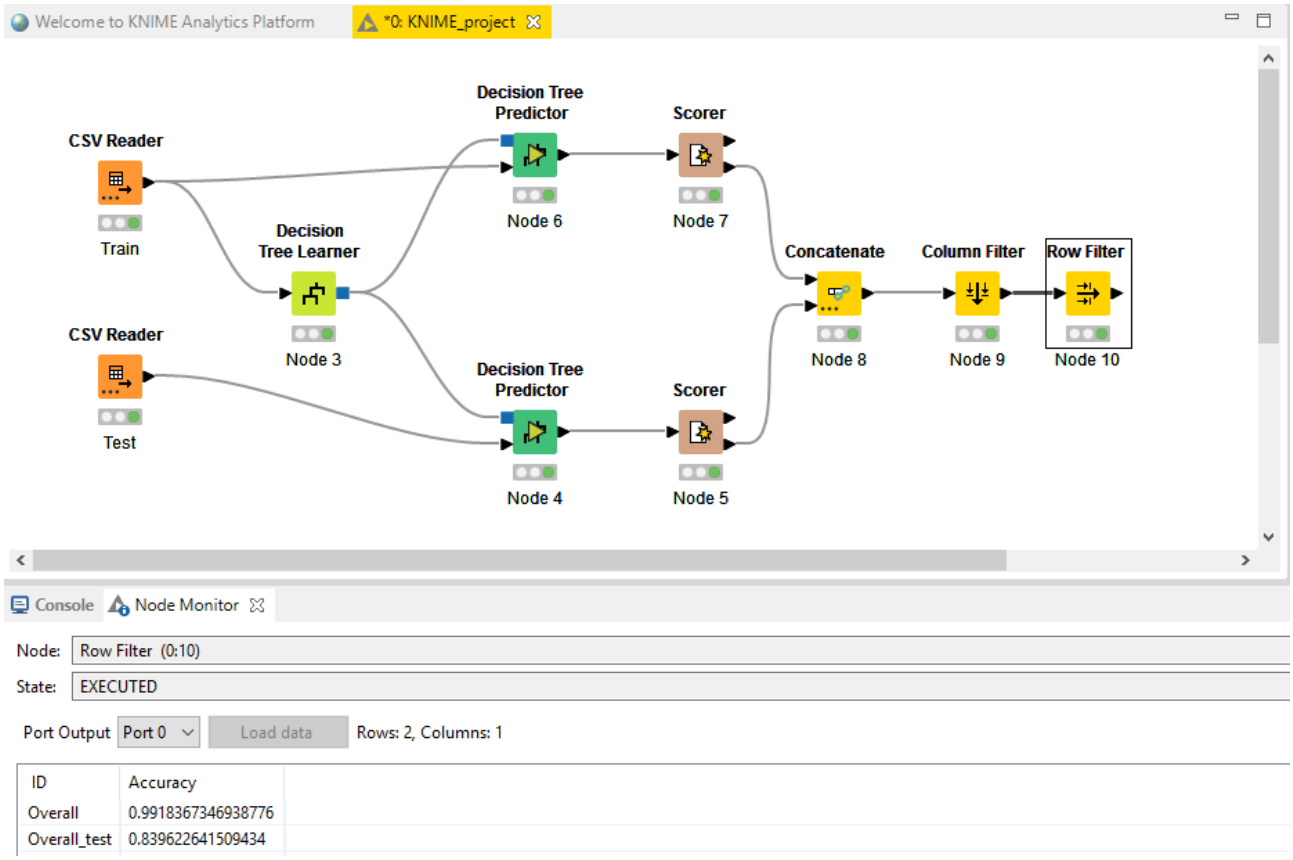


Figure 10: KNIME results for question 3.v (C4.5)

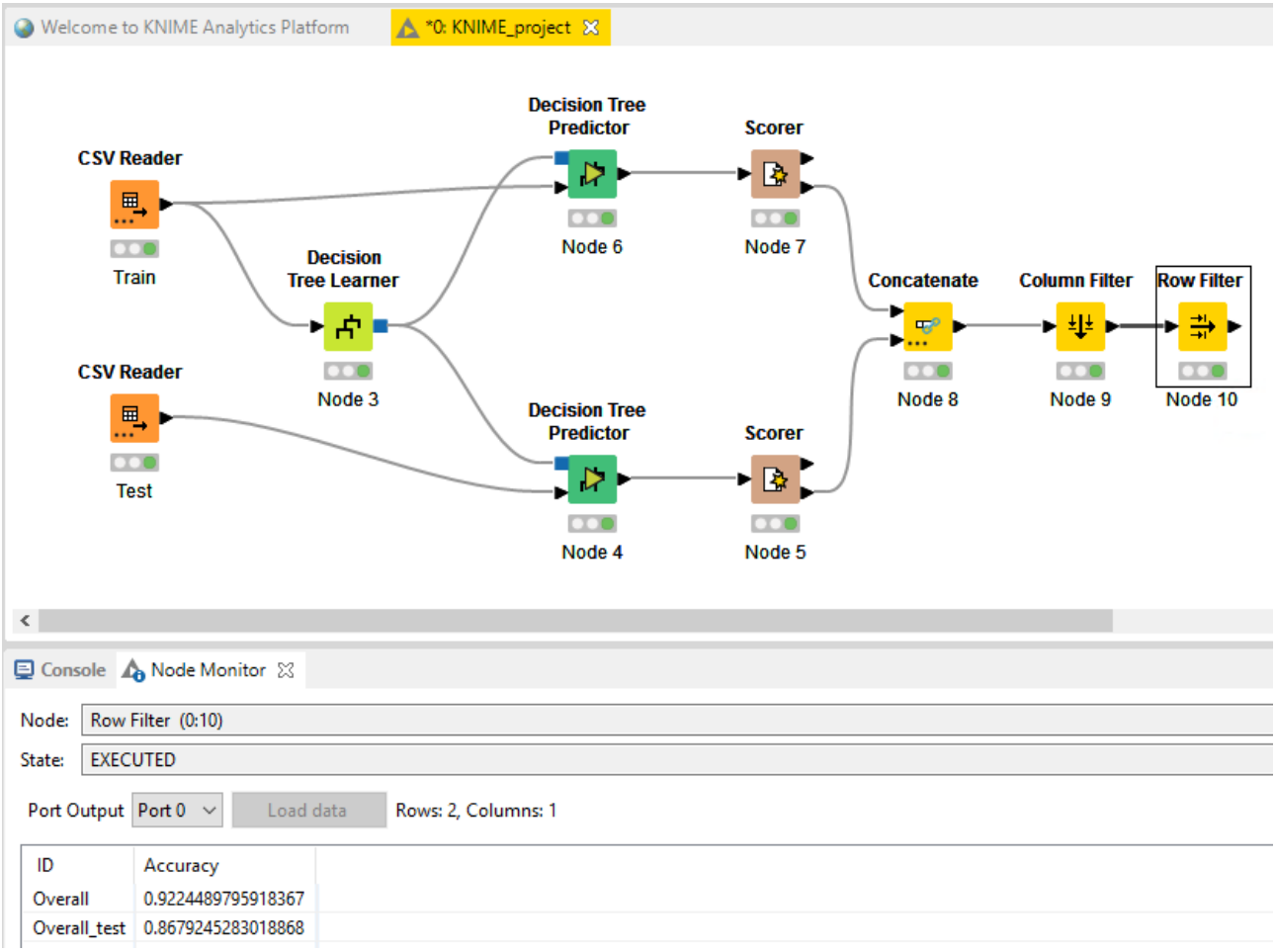


Figure 11: KNIME results for question 3.v (C4.5 with pruning)

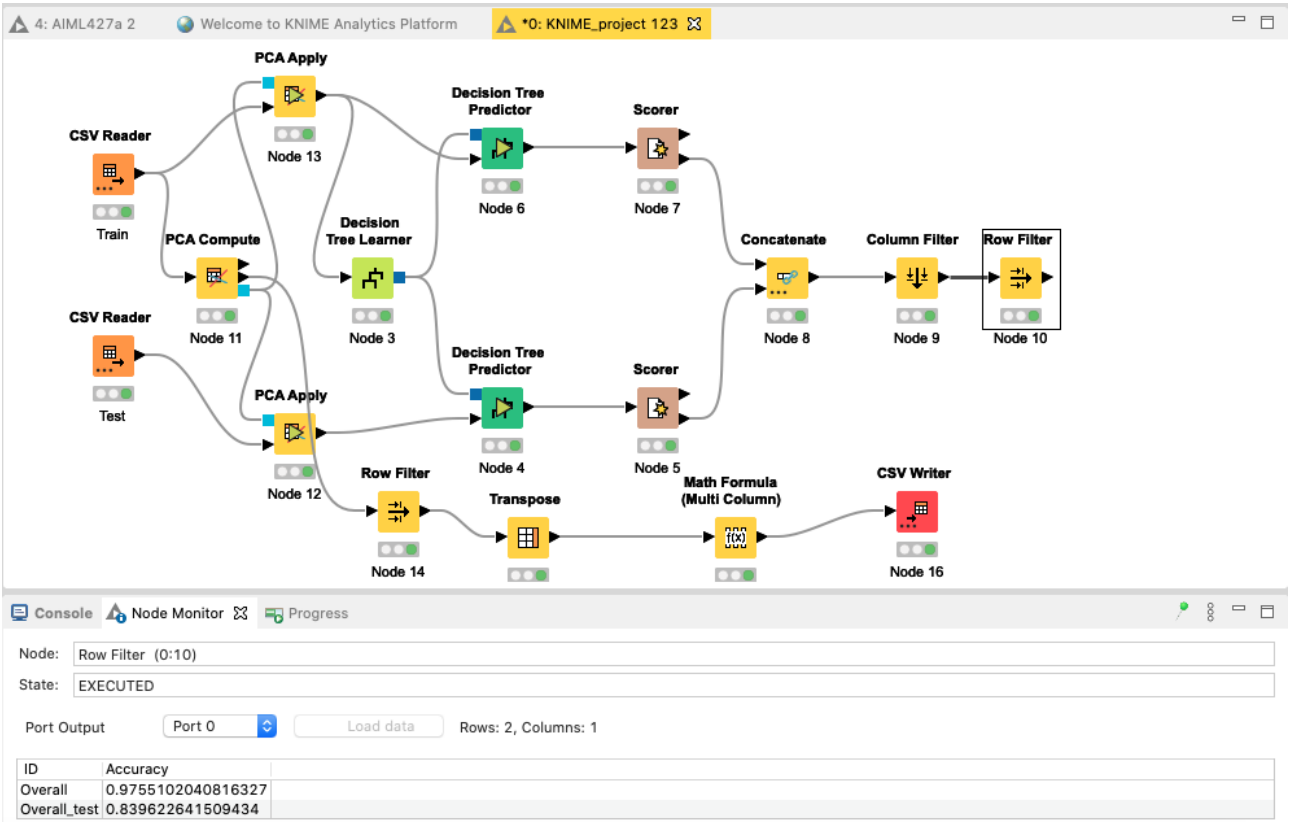


Figure 12: KNIME results for question 3.vi (PCA)

"row ID"	"0. eigenvector"	"1. eigenvector"	"2. eigenvector"	"3. eigenvector"	"4. eigenvector"
"eigenvalue"	2.906	1.069	0.737	0.635	0.479
"F1"	0	0	0	0	0
"F2"	0.095	-0.059	0.3	0.204	0.05
"F3"	-0.079	-0.066	0.182	0.041	-0.418
"F4"	0.15	-0.096	0.271	0.197	-0.026
"F5"	-0.086	0.061	0.2	-0.076	0.407
"F6"	0.178	-0.11	0.179	0.207	0.107
"F7"	-0.144	-0.041	0.358	-0.029	-0.036
"F8"	0.205	-0.09	-0.008	0.198	0.109
"F9"	-0.132	-0.045	0.223	0.091	-0.032
"F10"	0.24	-0.116	-0.088	0.073	-0.074
"F11"	-0.109	-0.124	0.296	0.143	-0.122
"F12"	0.306	-0.177	-0.088	0.115	-0.005
"F13"	-0.062	-0.193	0.244	0.229	0.02
"F14"	0.334	-0.152	-0.059	0.101	-0.038
"F15"	-0.08	-0.186	0.147	0.026	0.047
"F16"	0.302	-0.044	-0.16	-0.032	-0.006
"F17"	0.001	-0.25	0.031	-0.004	0.375
"F18"	0.308	-0.001	-0.076	-0.083	0.062
"F19"	0.019	-0.391	0.028	0.03	0.064
"F20"	0.297	0.07	0.007	-0.053	0.108
"F21"	0.072	-0.341	-0.073	-0.018	-0.011
"F22"	0.279	0.085	0.076	-0.136	-0.168
"F23"	0.026	-0.3	-0.073	-0.003	-0.352
"F24"	0.224	0.193	0.211	-0.129	-0.066
"F25"	0.048	-0.258	-0.085	-0.251	0.258
"F26"	0.151	0.142	0.044	-0.254	-0.218
"F27"	0.078	-0.257	0.166	-0.323	-0.019
"F28"	0.198	0.229	0.266	-0.074	-0.045
"F29"	0.014	-0.21	0.093	-0.275	-0.345
"F30"	0.196	0.216	0.217	0.078	0.031
"F31"	0.021	-0.064	0.184	-0.47	0.201
"F32"	0.2	0.156	0.182	0.115	0.106
"F33"	-0.025	-0.051	0.203	-0.343	0.044

Table 1: KNIME results for question 3.vi (first 5 principle components)

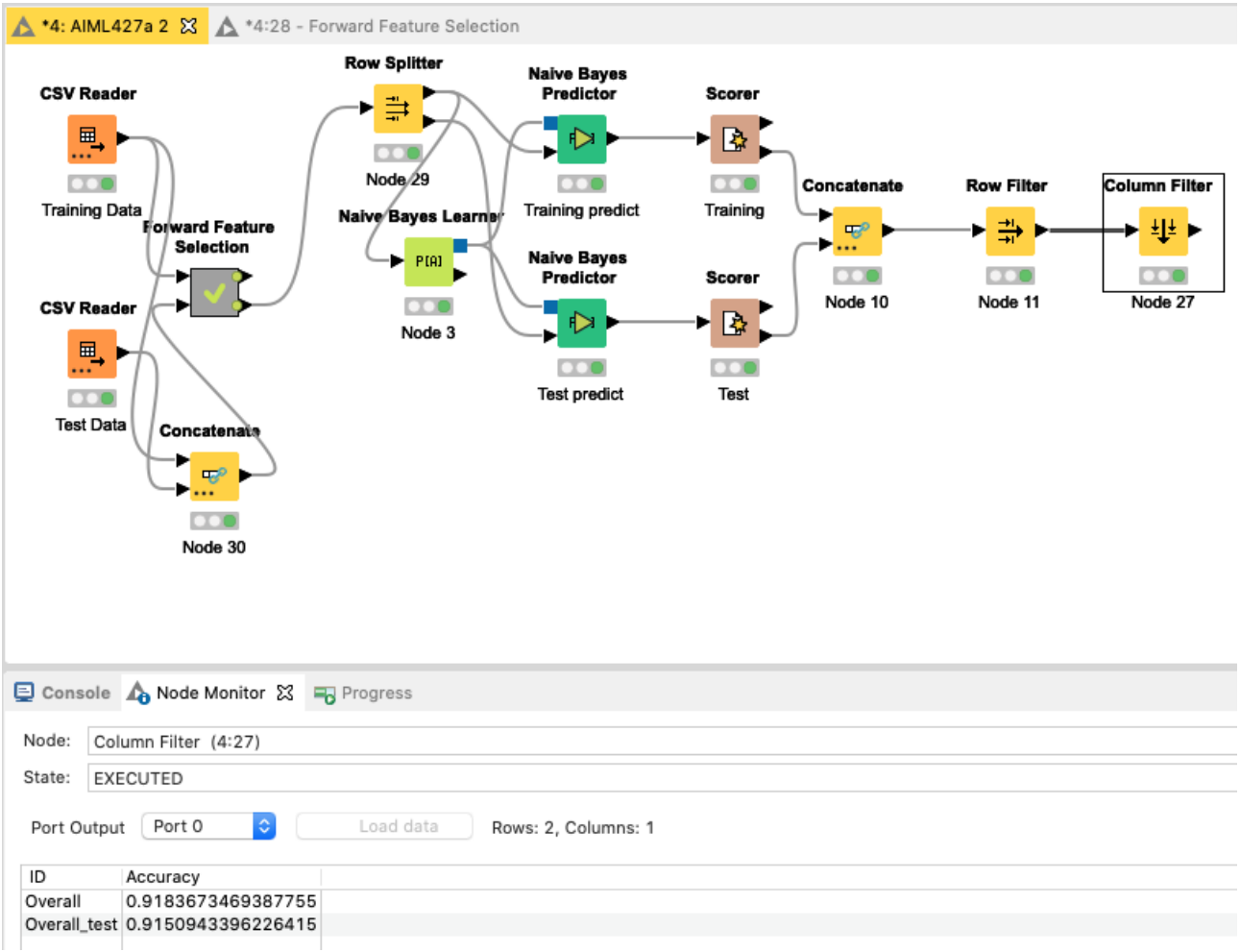
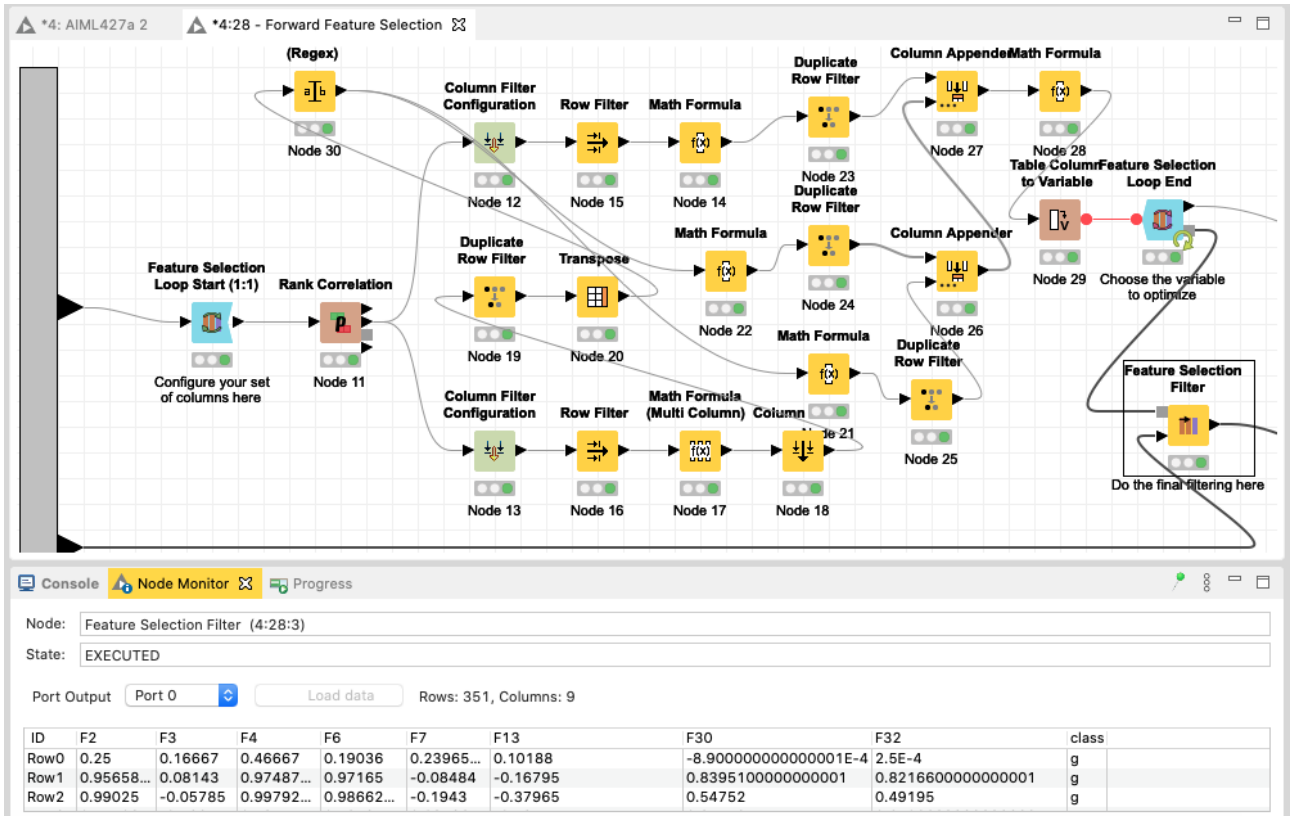


Figure 13: KNIME results for question 3.vii (CFS)



ID	F2	F3	F4	F6	F7	F13	F30	F32	class
Row0	0.25	0.16667	0.46667	0.19036	0.23965...	0.10188	-8.9000000000000001E-4	2.5E-4	g
Row1	0.95658...	0.08143	0.97487...	0.97165	-0.08484	-0.16795	0.8395100000000001	0.8216600000000001	g
Row2	0.99025	-0.05785	0.99792...	0.98662...	-0.1943	-0.37965	0.54752	0.49195	g

Figure 14: KNIME results for question 3.vii (CFS)