Machine Learning - Practical Application 1

UPM Facultad de Informatica
Emre Hangül

16 12 2019

Introduction

The area or the application of the Artificial Intelligence (AI) that is used for computer systems to learn automatically from the experiences (data) is called *Machine Learning* (ML) and it is one of the hottest areas of interest in the world of data science. The experiences that the machine learning algorithms learn from are called the *data* and the general aim after the *learning* is finished is to have *inferences* or *predictions* about the data that is used for the learning algorithm.

There are several different algorithms of the ML that are being used to solve problems like e-mail filtering, medical diagnosis, image and speech recognition or online customer supports. We can further group most of these algorithms under the 2 main titles: supervised and unsupervised learning. **Supervised learning** is the machine learning algorithm that uses the existing data for *training* which is later used to *predict* or *classify* a new set of instances. For instance, in the case of supervised *classification*, the computer uses labeled instances from the input data which also carries a class variable, and later on learns from these data and tries to predict the classes of the new or unseen data based on this learning algorithm. Some of the other supervised learning methods include *k-nearest neighbors*, *support vector machines*, *regression models (linear, logistic)* or *artificial neural networks*. On the other hand, the other main subset of the ML, **unsupervised learning**, also uses the existing labeled data but to *cluster* these data according to some of the likely or unlikely features of the instances of the data. Some of the real-life applications of unsupervized learning includes grouping and segmentation of customers or classifying animals on some homogenous or heteregoneous features. Moreover, the methods of unsupervized learning includes *hierarchical clustering*, *partitional clustering* or *probabilistic clustering*.

In this study, these ML algorithms will be showcased on a chosen dataset and it will be tried to predict & cluster the class instances according to the features of those instances. Also, the *feature subset selections* such as univariate filter and multivariate (filter and wrapper) subset selections will be used to determine which features might be useful and which features might be irrelevant/redundant for predicting & clustering the instances at the hand.

Problem Description

The dataset chosen for this study is "Student Academics Performance Dataset" that is accessed through *UCI Machine Learning Repository*. This dataset can be achieved by this *link*. The name of the dataset is "Sapfile1" and the dataset is in *.arff* format.

This data is collected via a research that includes data from 3 different Indian colleges [1] that tries to *associate* the students' **end semester percentage** performances with different features of these students. There are **131** instances with **22** features of these data points in the dataset. All of the 22 features are *nominal* (or categorical) variables. The detailed description of these features and values of the features are as follows:

1. *gender* : gender of the students, {male, female}
2. *caste* : Indian caste of students, {general, SC, ST, OBC, MOBC}
3. *class_10_percentage* : students' grade percentiles in the *HSLC Exams of India* in class 10 of the schools, {Best, Very Good (Vg), Good, Pass, Fail} (corresponding value intervals of these grades (ordered): >80-60-45-30<)
4. *class_12_percentage* : students' grade percentiles in the *HSLC Exams of India* in class 12 of the schools, {Best, Very Good (Vg), Good, Pass, Fail} (corresponding value intervals of these grades (ordered): >80-60-45-30<)
5. *internal_assessment_percentage* : students' grade percentiles in several tests, assignments, quizzes etc., {Best Very Good (Vg), Good, Pass, Fail} (corresponding value intervals of these grades (ordered): >80-60-45-30<)
6. *end_semester_percentage* : students' grade percentiles in the end of semester examinations., {Best Very Good (Vg), Good, Pass, Fail} (corresponding value intervals of these grades (ordered): >80-60-45-30<)
7. *previous_failed_paper* : whether or not student has a failed paper in the previous classes, {Yes, No}
8. *marital_status* : marital status of the students, {Married, Unmarried}
9. *lives_in* : students' living location, {Town (T), Village (V)}
10. *admission* : students' admission to schools, {Free, Paid}
11. *family_monthly_income* : monthly income of the families of the students, {Very High (Vh), High, Above Medium (Am), Medium, Low} (corresponding value intervals of these grades (ordered) : >30,000-20,000-10,000-5,000<)
12. *family_size* : size of the families of students, {Large, Average, Small} (corresponding value intervals of these grades (ordered) : >12-6<)

13. *father_qualification* : literacy or education level of the father of the students, {Il (illiterate), Um (under class 10), 10 (passed class 10), 12 (passed class 12), Degree (passed bachelor of arts or science or commerce exams)}
14. *mother_qualification* : literacy or education level of the mother of the students, {Il (illiterate), Um (under class 10), 10 (passed class 10), 12 (passed class 12), Degree (passed bachelor of arts or science or commerce exams)}
15. *father_occupation* : occupation of the father of students, {Service, Business, Retired, Farmer, Others}
16. *mother_occupation* : occupation of the mother of students, {Service, Business, Retired, Farmer, Others}
17. *number_of_friends* : number of friends of the students, {Large, Average, Small} (corresponding value intervals of these grades (ordered) : >12-6<)
18. *hours_study* : study hours of the students, {Good, Average, Poor} (corresponding value intervals of these grades (ordered) : >6-<4)
19. *school_attended_at_class_10* : school type at the class 10 of students, {Government (Govt), Private (Private)}
20. *language_student* : language taught for the students in the schools, {English (Eng), Assamese (Asm), Hindi (Hin), Bengali (Ben)}
21. *travel_time_to_college* : time required for a student to travel from home to school, {Large, Average, Small} (corresponding value intervals of these grades (ordered) : >2-1<)
22. *class_attendance* : class attendance percentage, {Good, Average, Poor} (corresponding value intervals of these grades (ordered) : >80-60<)

Furthermore, the response or the target variable that will be used to predict its class values is **end_semester_percentage**. It is priorily suspected that some of the features of the students might have an association with the target variable, such that changing the values of these features might be directly affecting the *class* of the outcome variable end_semester_percentage. Furthermore, it will be tried to show that there exists some grouping of the instances (or in this case, students) such that the groups have *high difference between* them and *low difference within* them.

Methodology

In the last chapter, it is said that it will be tried to construct ML algorithms that will be used to predict the classes of the target feature. These algorithms comprise of *supervized learning* methods, and such methods that will be used in this study can be both *non-probabilistic*and *probabilistic*.

The software to implement these methods on the chosen dataset is ***Weka*** which is an open source Machine Learning software that is built by the University of Waikato, New Zealand. The documentation of these software can be reached with this *link*.

The examples of non-probabilistic supervized classification algorithms that will be used in this study are:

- *K-Nearest Neighbor* : used for classifying a class based on *k* instances that are closer to the unknown instance at hand, based on the majority rule. The Weka classifier name for this algorithm is named *IBk* (Instance-based k)
- *Rule Induction* : used for classifying an output based on an "if-else-then" type of rule applied to an input. The Weka classifier name for the RIPPER algorithm (uses pruning) is *JRip*.

- *Classification Trees* : used for classifying an output in a tree like diagram such that the unseen instances are classified at the *leaf* node of the tree, which is constructed going from the *root* node to down and testing an attribute at each node. The Weka classifier name for this algorithm is *J48* (Corresponds for C4.5 method which is an extension of ID3 method)
- *Support Vector Machines* : used for classifying classes based on a separation of them by a hyperplane. The Weka classifier name for this algorithm is *SMO*.
- *Artificial Neural Networks* : used for classifying classes in a neural-network-alike system with input, nodes (classifiers), hidden layers, kernels (edges) and outputs. The Weka classifier name for one of the neural networks algorithm is *MultilayerPerceptron*.

Furthermore, the examples of probabilistic supervized classification algorithms that will be used in this study are:

- *Logistic Regression* : used for classifying classes based on a non-linear function (sigmoid) of the features that produces log-odds of those features. The Weka classifier name for this algorithm is *Logistic*.
- *Bayesian Classifiers* : used for classifying classes based on Bayesian Networks (constructed using Bayes' theorem). The Weka classifier names for the Bayesian Classifiers is *NaiveBayes* and *BayesNet* (for Bayesian Networks).

Also, another probabilistic supervized learning algorithm is discriminant analysis (linear or quadratic), but since it requires the response (class) variable to be binary (end_semester_percentage has more than 2 classes), this algorithm will not be used in this study.

Moreover, the learning algorithms that will be constructed will use several *feature subsetting techniques* to achieve results. First, the models will be constructed using *all* of the variables, then the models will be constructed after doing *univariate filter feature subset selection* and finally the models will be constructed using *multivariate filter and wrapper feature subset selection* methods. The univariate filter subset selection methods comprise of *gain-ratio*, *symmetrical-uncertainty* and *information-gain*, while the multivariate filter feature selection methods include *reliefF* method, *correlation-based* evaluation method and *conditional mutual information*. All of these methods try to filter the attributes by removing redundant features *before* the learning model is applied. On the other hand, the *wrapper* feature subset selection methods try to filter the redundant attributes by *using* them simultaneously with a learner (classifier) and then removes them in terms of their performances in those classifiers. Some of the wrapper search methods include *best first* and *greedy stepwise* search algorithms.

Results

Supervized Learning:

First, the probabilistic and non-probabilistic supervized learning algorithms will be constructed. The models and the results using *Weka Software* are as follows:

Using all of the variables:

Non-Probabilistic Classifiers:

- **K-Nearest Neighbor:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *lazy*, the classifier *IBk* is chosen. 10-fold cross validation is used. Several k values like 1,2,3,4,5 and 6 is used. The best accuracy and the best Kappa statistic is attained using $k = 4$. The confusion matrix and the performance measure statistics are below.

a	b	c	d	e	<– classified as
3	4	1	0	0	a = Best
2	31	8	1	0	b = Vg
0	14	34	6	0	c = Good
0	2	8	17	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	85	64.8855 %
Kappa statistic	0.4795	
Mean absolute error	0.1947	
Root mean squared error	0.3192	
Relative absolute error	70.932 %	
Root relative squared error	86.416 %	

- **Rule Induction:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *rules*, the classifier *JRip* is chosen. 10-fold cross validation is used. The number of rules used for the RIPPER algorithm is 10. The confusion matrix and the performance measure statistics are below.

a	b	c	d	e	<– classified as
5	2	1	0	0	a = Best
2	24	16	0	0	b = Vg
0	9	37	8	0	c = Good
0	1	6	20	0	d = Pass
0	0	0	0	0	e = Fail

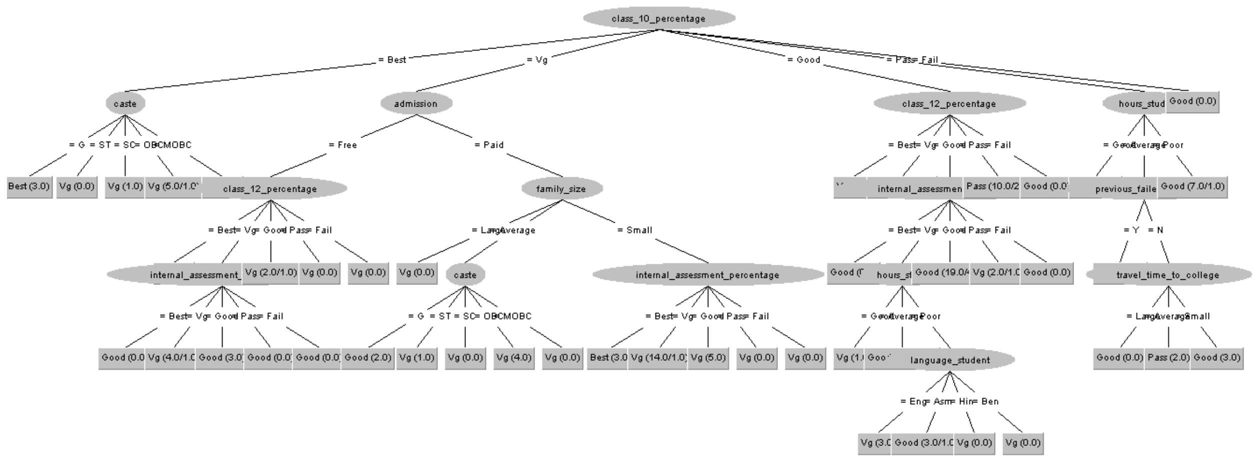
Correctly Classified Instances	86	65.6489 %
Kappa statistic	0.4917	
Mean absolute error	0.1815	
Root mean squared error	0.3297	
Relative absolute error	66.1245 %	
Root relative squared error	89.2637 %	

- **Classification Trees:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *trees*, the classifier *J48* is chosen. 10-fold cross validation is used. The confusion matrix and the performance measure statistics are below. A depiction of the C4.5 tree is also provided.

a	b	c	d	e	<– classified as
3	4	1	0	0	a = Best
0	29	12	1	0	b = Vg
0	10	37	7	0	c = Good
0	3	10	14	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	83	63.3588 %
Kappa statistic	0.447	
Mean absolute error	0.1743	
Root mean squared error	0.3471	
Relative absolute error	63.4748 %	
Root relative squared error	93.9734 %	



- **Support Vector Machines:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *functions*, the classifier *SMO* is chosen. 10-fold cross validation is used. The confusion matrix and the performance measure statistics are below.

a	b	c	d	e	<– classified as
4	3	1	0	0	a = Best
3	26	12	1	0	b = Vg
1	11	32	10	0	c = Good
0	2	9	16	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	78	59.542 %
Kappa statistic	0.406	
Mean absolute error	0.2614	
Root mean squared error	0.3481	
Relative absolute error	95.2003 %	
Root relative squared error	94.2307 %	

- **Artificial Neural Networks:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *functions*, the classifier *MultilayerPerceptron* is chosen. 10-fold cross validation is used. The confusion matrix and the performance measure statistics are below.

a	b	c	d	e	<– classified as
3	3	2	0	0	a = Best
2	25	12	3	0	b = Vg
2	11	32	9	0	c = Good
0	2	10	15	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	75	57.2519 %
Kappa statistic	0.3692	
Mean absolute error	0.1777	
Root mean squared error	0.391	
Relative absolute error	64.7199 %	
Root relative squared error	105.8486 %	

Probabilistic Classifiers:

- **Logistic Regression:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *functions*, the classifier *Logistic* is chosen. 10-fold cross validation is used. The confusion matrix and the performance measure statistics are below.

a	b	c	d	e	<– classified as
4	3	1	0	0	a = Best
2	20	10	10	0	b = Vg
5	15	22	12	0	c = Good
1	3	11	12	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	58	44.2748 %
Kappa statistic	0.2063	
Mean absolute error	0.2221	
Root mean squared error	0.4683	
Relative absolute error	80.88 %	
Root relative squared error	126.7697 %	

- **Bayesian Classifiers:**

Model Creation: The dataset is opened in *Explorer*. From the *Classify* section, under the subsection *bayes*, the classifiers *NaiveBayes* and *BayesNet* are chosen. For the bayesian network algorithm *BayesNet*, several different search algorithms like K2, True-Augmented-Naive (TAN) or Markov-blanked-based classifier are used. 10-fold cross validation is used. The best BayesNet model is with K2 search algorithm. The confusion matrices and the performance measure statistics of NaiveBayes and BayesNet with K2 are given in order below . .

a	b	c	d	e	<– classified as
4	3	1	0	0	a = Best
4	26	11	1	0	b = Vg
0	10	35	9	0	c = Good
0	1	7	19	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	84	64.1221 %
Kappa statistic	0.4746	
Mean absolute error	0.1561	
Root mean squared error	0.3189	
Relative absolute error	56.8525 %	
Root relative squared error	86.3364 %	

a	b	c	d	e	<– classified as
4	3	1	0	0	a = Best
4	25	12	1	0	b = Vg
0	10	35	9	0	c = Good
0	1	7	19	0	d = Pass
0	0	0	0	0	e = Fail

Correctly Classified Instances	83	63.3588 %
Kappa statistic	0.4628	
Mean absolute error	0.1566	
Root mean squared error	0.325	
Relative absolute error	57.046 %	
Root relative squared error	87.9825 %	

Using filter and wrapper feature subset selections:

Non-Probabilistic Classifiers:

- **K-Nearest Neighbor:**

Model Creation: The dataset is opened in *Explorer*. From the *meta* section, the classifier *AttributeSelectedClassifier* is chosen. This classifier first reduces the dimensions of the *training* data by a selected filter&wrapper, then applies the selected features to the *test* data, therefore it prevents the user from a possible *overfit*. In the AttributeSelectedClassifier, the classifier used is *IBk*, the **univariate filter** attribute evaluators used are *InfoGainAttributeEval*, *GainRatioAttributeEval* and *SymmetricalUncertAttributeEval*, the **multivariate filter** attribute evaluators used are *ReliefFAttributeEval* (reliefF selection) and *CfsSubsetEval* (correlation-based selection). Moreover, the **wrapper** evaluator used is *WrapperSubsetEval* (which uses the same classifier as it is being used on, e.g. IBk wrapper classifier for IBk algorithm).The search method for univariate filters and ReliefFAttributeEval is *Ranker*. The search methods for CfsSubsetEval and WrapperSubsetEval are *best-first* and *GreedyStepwise*. 10-fold cross validation is used. For the evaluators with ranker search, user selects different subset of attributes in terms of their rankings from top to bottom to get the ideal amount of attributes. For the evaluators with best-first and GreedyStepwise searches, the AttributeSelectedClassifier automatically chooses the ideal amount of attributes and uses them on the test data. The resulting attributes found after the evaluators are applied are shown below (the numbers correspond to the attributes, e.g. "1" is gender, "2" is caste etc. This info is given in the Problem Description section earlier). Also for this IBk algorithm, several *k* values like 1,2,3,4,5 and 6 are tried.The best accuracy and the best Kappa statistic is attained using $k = 4$. After several different models are applied, **CfsSubsetEval** multivariate filter selection evaluator with best-first search method is found to be the most accurate model with **70.99 %** accuracy rate. Compared to the IBk model seen earlier that uses all of the features (64.88 % accuracy), the model with reduced features has about *6 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5,22,20,18,13,7	66.41 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10,22,18,17	67.17 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22,7,18,10,13	68.70 %
ReliefFAttributeEval + Ranker	3,5,4,20,10,17,7,22,18,13	66.41 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	70.99%
CfsSubsetEval + GreedyStepwise	3,4,5,7,10,12,18,20,22	69.46 %
WrapperSubsetEval + BestFirst	3,4,12,15,16,18	57.25 %
WrapperSubsetEval + GreedyStepwise	4	55.72

- **Rule Induction:**

Model Creation: The process using "AttributeSelectedClassifier" that is explained above in the k-NN algorithm is also applied for the Rule Induction. The classifier *JRip* is chosen. 10-fold cross validation is used. The resulting attributes found after the evaluators are applied are shown below. After several different models are applied, **InfoGainAttributeEval** multivariate filter selection evaluator with ranker search method is found to be the most accurate model with **68.70 %** accuracy rate. Compared to the Rule Induction model seen earlier that uses all of the features (65.64 % accuracy), the model with reduced features has about *3 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5,22,20,18,13,7,10	68.70 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10,22,18	67.17 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22,7,18,10,13,17	67.17 %
ReliefFAttributeEval + Ranker	3,5,4,20,10,17,7,22,18,13	65.64 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	64.88%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	64.88 %
WrapperSubsetEval + BestFirst	3,4,5,9,10,11,12,16,18,19,21	60.30 %
WrapperSubsetEval + GreedyStepwise	4,10	58.01

- **Classification Trees:**

Model Creation: The process using "AttributeSelectedClassifier" that is explained above in the k-NN algorithm is also applied for the Classification Trees. The classifier *J48* is chosen. 10-fold cross validation is used. The resulting attributes found after the evaluators are applied are shown below. After several different models are applied, **InfoGainAttributeEval** multivariate filter selection evaluator with ranker search method is found to be the most accurate model with **64.88 %** accuracy rate. Compared to the Classification Trees(J48) model seen earlier that uses all of the features (63.354 % accuracy), the model with reduced features has about *1.5 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5,22,20,18,13	64.88 %
GainRatioAttributeEval + Ranker	3,4,5,7,20	61.06 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22,7,18	63.35 %
ReliefFAttributeEval + Ranker	3,5,4,20,10,17	61.06 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	61.83%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	61.83 %
WrapperSubsetEval + BestFirst	4,10	62.59 %
WrapperSubsetEval + GreedyStepwise	4,10	58.77

- **Support Vector Machines:**

Model Creation: The process using "AttributeSelectedClassifier" that is explained above in the k-NN algorithm is also applied for the Support Vector Machines. The classifier *SMO* is chosen. 10-fold cross validation is used. The resulting attributes found after the evaluators are applied are shown below. After several different models are applied, **GainRatioAttributeEval** multivariate filter selection evaluator with ranker search method is found to be the most accurate model with **70.99 %** accuracy rate. Compared to the Support Vector Machines (SMO) model seen earlier that uses all of the features (59.54 % accuracy), the model with reduced features has about *11 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5	67.93 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10	70.99 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22,7,18	68.70 %
ReliefFAttributeEval + Ranker	3,5,4,20,10,17	70.99 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	64.88%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	65.64 %
WrapperSubsetEval + BestFirst	4,10	56.48 %
WrapperSubsetEval + GreedyStepwise	4,10	57.25

- **Artificial Neural Networks:**

Model Creation: The process using "AttributeSelectedClassifier" that is explained above in the k-NN algorithm is also applied for the Artificial Neural Networks. The classifier *MultilayerPerceptron* is chosen.

10-fold cross validation is used. The resulting attributes found after the evaluators are applied are shown below. After several different models are applied, **WrapperSubsetEval** wrapper selection evaluator with BestFirst search method is found to be the most accurate model with **68.70 %** accuracy rate. Compared to the Artificial Neural Networks (muultilayer-perceptron) model seen earlier that uses all of the features (57.25 % accuracy), the model with reduced features has about *11 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5,22,20,18,13,7,10,17	66.41 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10,22,18,17	67.93 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22,7,18,10	65.64 %
ReliefFAttributeEval + Ranker	3,5,4,20,10,17	66.41 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	60.30%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	59.54 %
WrapperSubsetEval + BestFirst	3,5	68.70 %
WrapperSubsetEval + GreedyStepwise	4,10	57.25

Probabilistic Classifiers:

- **Logistic Regression:**

Model Creation: The process using "AttributeSelectedClassifier" that is explained above in the k-NN algorithm is also applied for the Logistic Regression. The classifier *Logistic* is chosen. 10-fold cross validation is used. The resulting attributes found after the evaluators are applied are shown below. After several different models are applied, **WrapperSubsetEval** wrapper selection evaluator with BestFirst search method is found to be the most accurate model with **68.70 %** accuracy rate. Compared to the Logistic Regression model seen earlier that uses all of the features (44.27 % accuracy), the model with reduced features has about *24 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4	65.64 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10	65.64 %
SymmetricalUncertAttributeEval + Ranker	3,4	64.12 %
ReliefFAttributeEval + Ranker	3,5	67.93 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	56.48%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	56.48 %
WrapperSubsetEval + BestFirst	3,4,18	68.70 %
WrapperSubsetEval + GreedyStepwise	3,4,18	68.70

- **Bayesian Classifiers:**

Model Creation: The process using "AttributeSelectedClassifier" that is explained above in the k-NN algorithm is also applied for the Logistic Regression. The classifiers *NaiveBayes* and *BayesNet* are chosen. For the bayesian network algorithm *BayesNet*, several different search algorithms like K2, True-Augmented-Naive (TAN) or Markov-blanked-based classifier are used. 10-fold cross validation is used. The resulting attributes found after the evaluators are applied are shown below. After several different models are applied, for *NaiveBayes*, **SymmetricalUncertAttributeEval** univariate filter selection evaluator with ranker search method is found to be the most accurate model with **70.99 %** accuracy rate. Compared to the NaiveBayes model seen earlier that uses all of the features (64.12 % accuracy), the model with reduced features has about *7 %* increase in accuracy. Furthermore, for *BayesNet*, **SymmetricalUncertAttributeEval** univariate filter selection evaluator with ranker search method is found to be the most accurate model with **70.99 %** accuracy rate. Compared to the NaiveBayes model seen earlier that uses all of the features (64.12 % accuracy), the model with reduced features has about *7 %* increase in accuracy.

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5,22,20,18,13,7,10,17,14,15	66.41 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10,22,18	68.70 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22	70.99 %
ReliefFAttributeEval + Ranker	3,5,4,20,10	70.99 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	67.17%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	67.17 %
WrapperSubsetEval + BestFirst	1,3,5,7,8,10,15,18	63.35 %
WrapperSubsetEval + GreedyStepwise	3,5,7,18	61.06

Evaluator + Search Method	Attributes Used	Accuracy
InfoGainAttributeEval + Ranker	3,4,5,22,20,18,13,7,10	65.64 %
GainRatioAttributeEval + Ranker	3,4,5,7,20,10	70.22 %
SymmetricalUncertAttributeEval + Ranker	3,4,5,20,22,7,18	68.70 %
ReliefFAttributeEval + Ranker	3,5,4,20,10,17	69.46 %
CfsSubsetEval + BestFirst	3,4,5,7,10,12,18,20,22	69.46%
CfsSubsetEval + GreedyStepwise	3,4,5,7,12,18,20,22	68.70 %
WrapperSubsetEval + BestFirst	3,5,7,10	61.06 %
WrapperSubsetEval + GreedyStepwise	3,5,7,10	63.35

Unsupervized Learning

There are several clustering methods that will be applied on Weka software in this study. These clustering techniques are *Hierarchical Clustering*, *Partitional Clustering* and *Probabilistic Clustering*. For the Hiearchical Clustering, the Weka clusterer names are *CobbWeb* and *HiearchicalClusterer*. For the Partitional Clustering, the Weka clusterer names are *SimpleKMeans* and *FarthestFirst*. For the Probabilistic Clustering, the Weka clusterer name is *EM* (expectation-maximization algorithm). These models are as follows:

Hiearchical Clustering:

The first Weka clusterer used is *HierarchicalClusterer*. First, the class variable *end_semester_percentage* is removed in the Preprocess tab. Then from the Cluster tab, HierarchicalClusterer is selected. Number of clusters are set to 5. *MEAN*, *AVERAGE* and *WARD* are selected as different link types. Euclidean distance is used. The resulting clusters and a depiction of the clustering tree are as follows:

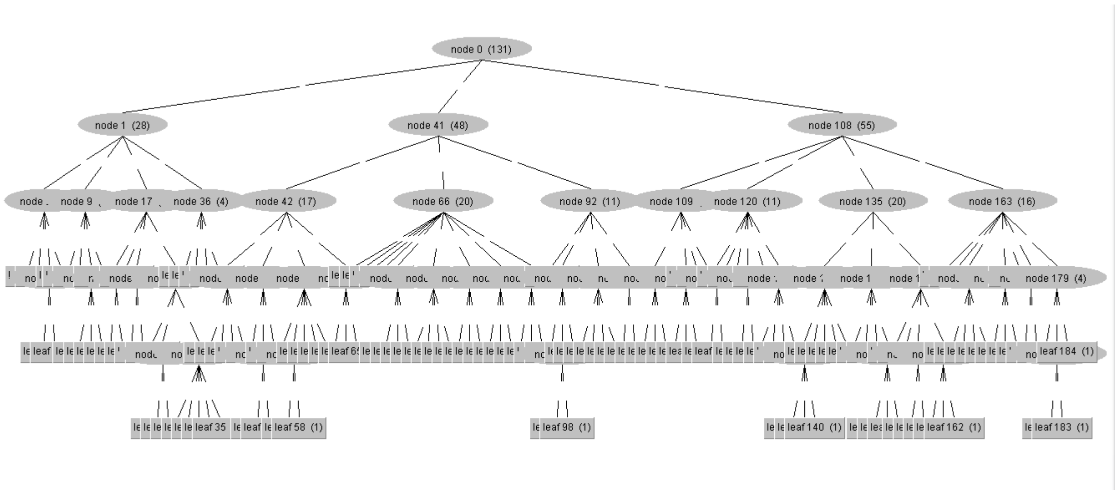
(Tables show results with *mean*, *average* and *ward* in that order, left to right)

Cluster No	Number of Instances
0	29
1	51
2	13
3	24
4	14

Cluster No	Number of Instances
0	54
1	37
2	26
3	10
4	4

Cluster No	Number of Instances
0	18
1	17
2	57
3	21
4	18

Further, the *CobWeb* hiearchical clusterer method is used. First, the class variable *end_semester_percentage* is removed in the Preprocess tab. Then from the Cluster tab, CobWeb is selected. Different cutoff points like 0.3 or 0.1 are tried. A depiction of the clustering tree from CobWeb is as follows:



Partitional Clustering:

The Weka clusterer used for partitioning are *FarthestFirst* and *SimpleKMeans*, which are both related with k-means algorithm. First, the class variable *end_semester_percentage* is removed in the Preprocess tab. Then the 2 different clusterers are selected successively. Number of clusters are set to 5. There are 73 instances (55.72 %) that are *incorrectly* classified in FarthestFirst algorithm and there are 69 instances (52.67 %) that are incorrectly classified in SimpleKMeans algorithm. Overall, these numbers slightly differ with every different *seed* defined for their respective algorithms. The resulting clusters are as follows:

Cluster No	Number of Instances
0	38
1	29
2	29
3	22
4	13

Cluster No	Number of Instances
0	42
1	11
2	35
3	28
4	15

Probabilistic Clustering:

The Weka clusterer used for probabilistic clustering is *EM* which stands for "Expectation-Maximization". It is an algorithm related with finding maximum likelihood of the estimates of parameters with iterations in a statistical model. This clusterer is chosen from the cluster tab. Number of clusters is set to 5. The resulting clusters are as follows:

Cluster No	Number of Instances
0	24
1	31
2	16
3	28
4	32

Conclusion

In this study, the academic performances of students' from 3 different Indian colleges are acquired in a dataset from the UCI Machine Learning Repository to do the Machine Learning tasks of supervized learning and unsupervized learning. The dataset included 22 features with 131 instances and the class (target) variable was the students' end of the year performances. The analysis of supervized learning is done such that first, all of the features are included in several probabilistic and non-probabilistic methods, and then the feature subset selection methods of univariate and multivariate filter and wrapper methods are applied before applying those probabilistic and non-probabilistic methods. The results showed that in each of the probabilistic and non-probabilistic methods, the accuracy have increased after the feature subset selection is done, compared to using all of the variables. Furthermore, the unsupervized learning methods such that hieararchical, partitional and probabilistic clusterings are applied and visualized. In general, the results clearly indicated several associations and groupings between the features of this dataset and an individual can further implement the steps of this study on other datasets or datasets combined with the one used in this study.

References

[1] Hussain, S., Abdulaziz Dahan, N., Ba-Alwi, F. and Ribata, N. (2018). Educational Data Mining and Analysis of Students' Academic Performance Using WEKA. Indonesian Journal of Electrical Engineering and Computer Science, [online] 9(2), p.447. Available at: https://pdfs.semanticscholar.org/46b5/436be736e5a48ab127b5a856e73e46487cc4.pdf?_ga=2.83460053.2579395.1576276487-364639372.1576276487 [Accessed 14 Dec. 2019].