CIS435: Data Mining & Data Warehousing

Final Project

Caroline Artz

December 2012

1 Introduction

Following a brief description of the data, I will provide an overview including the advantages and disadvantages and rationale for choosing of the main algorithms reported on here. Next I will outline my methods and results, including parameter tuning for each of the learning schemes, with respect to the different WEKA[1] modules. Finally, I will compare and discuss my outcomes.

2 Data

The Wine dataset[2] describes the chemical analysis of three different cultivars of wine grown in Italy and consists of 13 constituents as continuous attributes plus the categorical class label, Type. The raw dataset has not been standardized and presents highly variables attribute scales. Thus, appropriate measures will be taken during data preprocessing and/or classifier training, as I will be evaluating non-scale-invariant classifiers performance. With 13 attributes and 178 instances, this is a highly dimensional dataset and visualization will be important to analyzing the underlying data structure and attribute dependencies in an effort to optimize classification tasks. Additionally, feature selection and data reduction will be important to consider for optimal classification accuracy.[3]

3 Approaches & Rationale

Because we are provided the class label for the wine data sets, I focus my analysis on supervised learning and specifically, data preprocessing and classification. Given the nature of this dataset, knowledge gained from past experience with the data, and an exhaustive but informal exploration of the algorithms covered this quarter, and ones that were not, I narrowed the focus to three algorithms that consistently perform well with this data. This data set as been described as "easy"[2], and I did find that other learning schemes produced good models, including logistic regression and JRip, but I chose the top three based on the average results from my exploratory analysis in WEKA Experimenter (described in section 4):

3.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) neural network, *MultilayerPerceptron*, determines a set of weights connecting the input space (the attributes), nodes of the hidden layer(s) and output space (classes) that minimize the classification error. This is learned through backpropagation algorithm has advantages in that it shows good predictive performance across many domains and specifically for our purposes, is robust to attributes that prove redundant and irrelevant[4]. Unfortunately, MLP models are difficult to dissect and parameter tuning is hard to fine-tune. Additionally, the MLP has been shown to overfit relatively easily. While I employ 10-fold cross-validation, metalearners for proper (i.e., less overly optimistic) attribute selection as part of each classifier training process and proper supervised discretization of the data, without a separate set of fresh, unseen data from within this domain, I cannot with the highest possible degree of confidence determine whether the model produced these algorithms, including the MLP, as overfit the training data. Another potential problem to consider with the MLP is that it may be sensitive to noise in the data. However, knowledge of this dataset leads me to believe that there is likely not very much noise.[2]

3.2 Naïve Bayes

The Naïve Bayes classifiers assume conditional independence between the attributes when estimating the probability that a given instances belongs each class and makes classification decisions based on the most likely class membership[5]. I chose a Naïve Bayes (NB) algorithm, *NaiveBayesSimple*, for training a classifier because of the success observed previously on this dataset even in the absence of any data preprocessing or attribute selection. Although the full dataset contains a number of correlated attributes which inherently contradict the NB assumption of conditional independence among the features, researchers have shown optimal NB performance on the highly feature dependent end of the spectrum in addition to when its assumption is met.[6] Further supporting its potential for high performance with this dataset is that the Naïve Bayes algorithm learns well without needing a huge dataset (as a result of its assumption), generalizes to unseen data, and is easily interpretable. The Naïve Bayes classifier is also robust to noise in a data set.

3.3 k-Nearest Neighbor

Another previously seen contender in this data set is the instance based lazy learner, k-nearest neighbor algorithm which classifies each instance based on the class label of proximally close training instances as measured in d-dimensional (d=# of features) space. Although a surface-level evaluation might conclude that a nearest-neighbor learner is unfit for our potentially redundant and highly correlated data set, the kNN

algorithm has shown good overall predictive performance[4]. So, although initially counter-intuitive we may see that the kNN learners benefit from the underlying structure of the Wine data set.

4 Methods & Results

4.1 PCA and Visualization

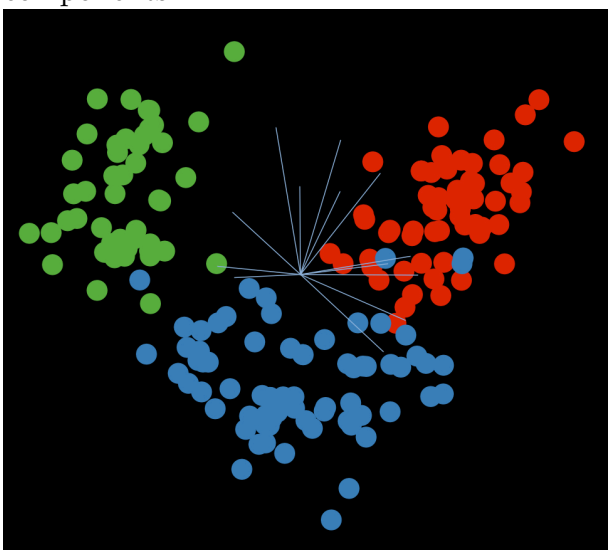
Pattern Matrix[a]

	Component		
	1	2	3
Flavanoids	.913		
OD280_OD315	.895		
Total_Phenols	.825		
Hue	.749		
Proanthocyanins	.676		
Malic_Acid	-.543		
Nonflavanoid_Phenols	-.529		
Alcohol		.871	
Color_Intensity	-.465	.794	
Proline	.357	.733	
Magnesium		.456	
Ash			.866
Ash_Alcalinity		-.392	.787

Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.
a. Rotation converged in 8 iterations.

SPSS Principal Component Analysis Pattern Matrix with attribute load on principal components

To confirm and further enrich my understanding of the structure of the wine data set, I re-preformed variations of PCA using SPSS and studied the attributes transformed in the PC space and their respective load on the principal components. This shows that first two principal components are well defined and appropriately reflected by the attribute subset data sets provided. These two principal components account for a large portion of the variance in the data set (Note: my SPSS trial is up and I don't have the exact value anymore). I performed some additional instance and attribute visualization using an updated WEKA 7.7 that allows data space exploration a projection plot. Below is an example of the projection plot with the data projected over the first two principal components.



4.2 Analyses

4.2.1 Initial Exploratory Analysis: Experimenter

After an informal exploration of the data and all candidate algorithms (classification, clustering, association) covered throughout the quarter, I narrowed my focus to those classification learners listed in Section 3 (Naïve Bayes, Multilayer Perceptron, and k-Nearest Neighbor).

In order to test many configurations over multiple datasets in an efficient and comparative manor, I used WEKA's Experimenter to take a closer look. I explored the algorithms over standardized versions of the datasets provided but without any additional preprocessing (e.g., no discretization for kNN as there is no internal option as with MLP and the non-simple NB). That is, I used the unsupervised filter → Standardize in WEKA Explorer and saved to new .arff files as opposed to using filtered classifiers, simply for ease as they are more time consuming to configure and reconfigure.

In light of the fact that many of the algorithms (including many not included in my main analyses reported here—jRip, Logistic, etc.) have accuracy performance scores that rank as statistically indistinguishable using paired t-tests with a .05 confidence level, I viewed my Experimenter analyses as preliminary groundwork to see if any of the schemes showed higher accuracy percentages on any of the subset data sets in order to shape additional adjustments in the Explorer and Knowledge Flow. I did not use the Experimenter to rank the 'best' model, as there isn't one classifier schema I could find that was statistically better with any reasonable confidence level and the ranking of classifiers based on classification accuracy value alone varied heavily from the Explorer/Knowledge Flow results for similarly accurate models. Any significance testing under .6 confidence level could not discriminate among 6 or 7 algorithms models' that I encountered during my informal early exploration (which included models that were not a main focus of our course but tested to provide insight to potentially beneficial parameter adjustments, e.g., *K**, *NBTree*, *SimpleLogistic*). A sample of some exploratory analyses from the Experimenter supportive to my final discussion of the outcome is shown below.

4.2.2 Tuning Parameters

Of note in my initial exploratory analysis, most classifiers performed with slightly higher accuracy on the 3-dimension dataset vs. the full attribute dataset, with the exception of the best performing kNN classifiers (The results in the table below are from on non-discretized, non-binary attributes).

Most of my testing showed an overall trend across the full and subset datasets on algorithms of nearest neighbor, MLP, and Bayesian type: highest performance on the full dataset and 3 dimension dataset, followed by that with only correlated attributes. Many additional parameter configurations were tested in Experimenter than expressed in the output below. However, the dilemma around the inability to replicate performance scores across Explorer/Knowledge Flow and Experimenter inherent to the different methods for used in validation/evaluation and resulting conflicts when attempting to rank performance based on combined knowledge of the output from both led me to divert my focus from the Experimenter.

While our assignment specifically states to use the full data set for our final comparison, I interpret this to mean we cannot simply delete attributes (e.g., the attributes that differentiate the full data set and data set with 3 dimensions) when optimizing classifiers, but, appropriately applied filters that result in a subset of attributes selected for building the model is encouraged.

Tester: weka.experiment.PairedCorrectedTTester
Analysing: Percent_correct
Datasets: 6
Resultsets: 18
Confidence: 0.05 (two tailed)

Dataset	(1)	(2)		(3)		(7)		(8)		(9)		(11)		(12)		(14)		(15)		(17)		(18)	
3Dim	97.42	98.76		98.99		98.93		98.99		96.13		96.19		96.13		96.19		96.19		97.53		97.37	
AllData	97.35	98.71		98.02		98.08		98.02		95.12		95.23		95.51		95.23		95.23		97.70		97.41	
1stDim	86.37	84.44		89.87		89.30		90.20		80.00		81.31		82.26		81.31		81.31		82.30		82.48	
2ndDim	91.83	90.07		91.12		90.95		91.62		91.61		90.10		90.83		90.10		90.10		92.87		93.20	
NoCorr	92.29	92.92		90.60		90.71		90.82		86.30	*	86.20	*	86.92		86.20	*	86.20	*	85.95	*	87.98	
Corr	94.31	92.02		93.76		93.59		93.87		91.10		94.64		94.75		94.64		94.64		93.69		94.14	
(v/ /*)			(0/6/0)		(0/6/0)		(0/6/0)		(0/6/0)		(0/5/1)		(0/5/1)		(0/6/0)		(0/5/1)		(0/5/1)		(0/5/1)		(0/6/0)

Key	
(1)	bayes.NaiveBayesSimple '' -1478242251770381214
(2)	bayes.NaiveBayes -D 5995231201785697655
(3)	functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(7)	functions.MultilayerPerceptron '-L 0.3 -M 0.4 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(8)	functions.MultilayerPerceptron '-L 0.3 -M 0.1 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(9)	lazy.IB1 '' -6152184127304895851
(11)	lazy.IBk '-K 5 -W 0 -X -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(12)	lazy.IBk '-K 5 -W 0 -X -I -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(14)	lazy.IBk '-K 5 -W 0 -X -A \"weka.core.neighboursearch.BallTree -A \\\"weka.core.EuclideanDistance -R first-last\\\" -C \\\"weka.core.neighboursearch.balltrees.TopDownConstructor -N 40 -S weka.core.neighboursearch.balltrees.PointsClosestToFurthestChildren\\\"\"' -3080186098777067172
(15)	lazy.IBk '-K 5 -W 0 -X -A \"weka.core.neighboursearch.KDTree -A \\\"weka.core.EuclideanDistance -R first-last\\\" -S weka.core.neighboursearch.kdtrees.SlidingMidPointOfWidestSide -W 0.01 -L 40 -N\"' -3080186098777067172
(17)	lazy.IBk '-K 20 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(18)	lazy.IBk '-K 20 -W 0 -I -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172

4.3 On Classifier Comparison

While optimizing the learning schemes in Explorer and Knowledge Flow I was unable to produce better performance with a nearest neighbor algorithm beyond the naïve IB1 after improving both as a result of discretizing to binary attributes. Interestingly, the default parameters for ALL models discussed here provided the same or better outcome than any other configuration I could find (which includes the best possible outcome across all learning schemes covered this quarter that I found while employing the least bias method for supervised discretization and attribute subset selection).

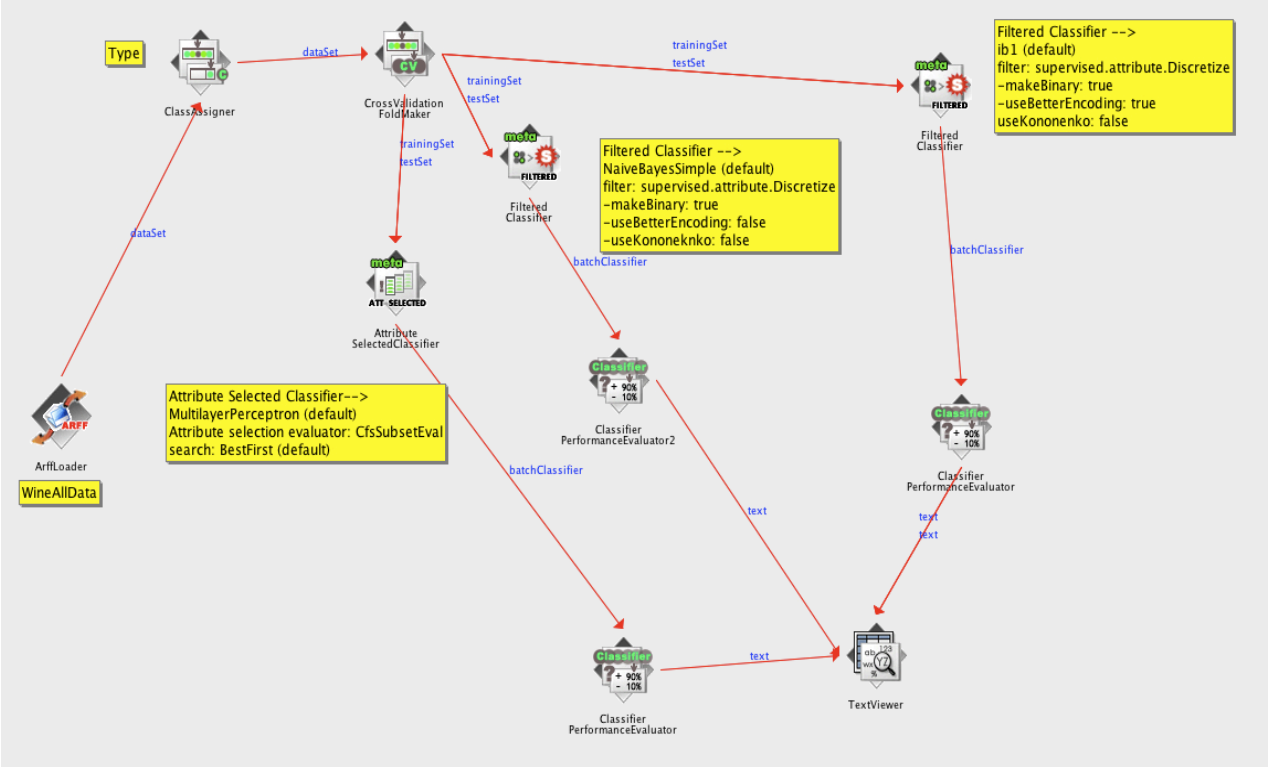
4.3.1 Metrics

As this is not domain where cost-sensitive learning or evaluation makes sense (i.e., none should be considered the positive class and misclassification holds the same level of cost for all 3 classes, I will base the majority of my evaluation on accuracy (% correct). Due to close comparisons between some of the classifiers and to provide further information, I will report additional statistics.

4.3.2 Procedure: Knowledge Flow

The Knowledge Flow and Explorer are similar in their functionality with the exception of some updatable incremental learners, though for my purposes, the results and functionality appear to be identical with the Knowledge Flow providing a visual representation of the Explorer process. Below you will see the Knowledge Flow canvas that diagrams my procedures in training and evaluating 3 classifiers: 1. *MultilayerPerceptron*; 2. *NaïveBayesSimple*; 3. *IB1*.

Note: The Knowledge Flow uses *IB1* and *NaiveBayesSimple* without filtered attribute selection, as I found the results to be identical to using only the filtered classifier (discretized) while tuning parameters in Explorer.



4.3.3 Classifier Performance: Results of Knowledge Flow & Explorer

After optimizing the classification models and running them through the Knowledge Flow, I present the results comparing the different schemes as well as comparing their performance with and without filtering for attribute subset selection (Tables A and B, below).

A. Performance of learning schemes on WineAllData prior to filtering for attribute selection; Validated with 1x 10-fold cross-validation

Classifier	TP Rate	FP Rate	ROC	Accuracy
MLP[1]	0.972	0.013	0.999	97.191%
NB[2]	**0.989**	**0.005**	**0.999**	**98.876%**
kNN[3]	0.983	0.007	0.988	98.315%

[1]Scheme:weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

[2]Scheme:weka.classifiers.meta.FilteredClassifier -F "weka.filters.supervised.attribute.Discretize -E -R first-last" -W weka.classifiers.bayes.NaiveBayesSimple

[3]Scheme:weka.classifiers.meta.FilteredClassifier -F "weka.filters.supervised.attribute.Discretize -D -E -R first-last" -W weka.classifiers.lazy.IB1

B. Performance of learning schemes on WineAllData after attribute selection filter; Validated with 1x 10-fold cross-validation

Classifier	TP Rate	FP Rate	ROC	Accuracy	Improve
MLP[1]	**0.994**	**0.002**	**1**	**99.438%**	**+2.247%**
NB[2]	0.989	0.005	0.999	98.876%	-
kNN[3]	0.983	0.007	0.988	98.315%	-

[1]Scheme:weka.classifiers.meta.AttributeSelectedClassifier -E "weka.attributeSelection.CfsSubsetEval " -S "weka.attributeSelection.BestFirst -D 1 -N 5" -W weka.classifiers.functions.MultilayerPerceptron -- -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

[2] Scheme: weka.classifiers.meta.FilteredClassifier -F "weka.filters.supervised.attribute.Discretize -E -R first-last" -W weka.classifiers.meta.AttributeSelectedClassifier -- -E "weka.attributeSelection.CfsSubsetEval " -S "weka.attributeSelection.BestFirst -D 1 -N 5" -W weka.classifiers.bayes.NaiveBayesSimple

[3]Scheme: weka.classifiers.meta.AttributeSelectedClassifier -E "weka.attributeSelection.CfsSubsetEval " -S "weka.attributeSelection.BestFirst -D 1 -N 5" -W weka.classifiers.meta.FilteredClassifier -- -F "weka.filters.supervised.attribute.Discretize -D -K -R first-last" -W weka.classifiers.lazy.IB1

5 Discussion

My experiment in training and fine-tuning an optimal classifier for the full wine data set results in one learned using the default settings for the *MultilayerPerceptron* by way of the metalearner, *AttributeSelectedClassifier*. This metalearner applies attribute selection methods while ensuring that attribute selection is based only on training data and not in testing the model (which would be the result of applying attribute selection prior to induction via the attribute selection panel in WEKA Explorer)[7].

With knowledge of the PCA and improvements in performance on the reduced 3 dimension attribute data set, I looked for an attribute selection evaluator that would take attribute correlation (with each other and with the class labels) into consideration. The default evaluator and search method, *CfsSubsetEval* and *BestFirst* respectively, are a good option resulting in the same subset of attributes comprising the wine 3 dimension

data set, which resulted in improved performance for the MLP. Note: This showed slight improvements among many of the learning schemes when evaluating 10x 10-fold cross validation in WEKA Experimenter, but these could not be replicated in single 10-fold cross-validation in Explorer/Knowledge Flow.

5.1 Performance by Classifier

5.1.1 Multilayer Perceptron

The complexity of the MLP Neural Network, which lends to the difficulty in the interpretation of classification models built with this algorithm, also plays an integral part in the high levels of performance in spaces with complex decision boundries (i.e., non-linearly seperable). I was surprised and disappointed that tuning the learning rate, which assesses and adjusts for the network's weights as they contribute to the error during training could not produce 100% correctly classified instances. It's possible that I didn't give an exhaustive search for the optimal parameters, or the persistently misclassified instance may be an outlier as a result of a small amount of noise in the dataset.

A concern with the MLP is its generalizability with unseen data and I would be interested to see how this performed if we had split the data into training and testing sets instead of using 10-fold cross validation (a potential future experiment).

5.1.2 Naïve Bayes

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 14 Type):
        Information Gain Ranking Filter

Ranked attributes:
 1.0151    7 Flavanoids
 0.8278   13 Proline
 0.7438   10 Color_Intensity
 0.7221   12 OD280_OD315
 0.6324   11 Hue
 0.6034    1 Alcohol
 0.5795    6 Total_Phenols
 0.4306    2 Malic_Acid
 0.3211    5 Magnesium
 0.2772    4 Ash_Alcalinity
 0.2653    9 Proanthocyanins
 0.2198    8 Nonflavanoid_Phenols
 0.1649    3 Ash



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 14 Type):
        Correlation Ranking Filter
Ranked attributes:
 0.57    13 Proline
 0.535    1 Alcohol
 0.49    10 Color_Intensity
 0.475   12 OD280_OD315
 0.445   11 Hue
 0.445    7 Flavanoids
 0.385    6 Total_Phenols
 0.339    4 Ash_Alcalinity
 0.333    2 Malic_Acid
 0.274    9 Proanthocyanins
 0.263    3 Ash
 0.253    8 Nonflavanoid_Phenols
 0.232    5 Magnesium
```

The Naïve Bayes model consistently performs well with this data set. Although the conditional independence assumption made by the NB classifier may degrade its performance in certain domains where feature dependencies are present, The NB classifier performs exceptionally well in other domains in spite of dependencies. One thing that is important to remember is that while *correct estimation usually leads to accurate prediction, accurate prediction does not require correct estimation*[8]. It has been posited that the performance of the NB classifier is not directly correlated with the degree of feature dependencies by class, but rather how much the assumption of conditional independence results in a loss of

information about the class[6]. Further, the distribution of dependencies across each class may be important to consider because the dependencies may effectively cancel each other out or result in improvements in classification.

A look at which attributes are correlated with each other (Alcohol with Proline .644; Flavanoid and Total Phenols: .85; OD280_OD315 with Flavanoids and Total Phenols: ~.7; Proanthocyanins with Flavanoid and Total Phenols: ~.6; according to Assignment 3) along with analysis of attributes providing high levels of information gain as well as those which are highly correlated with the class attribute (WEKA 7.7) via the Select Attributes panel lead me to believe that these factors are central to the high performance levels of NB as well as kNN and Logistic Regression.

5.1.3 k-Nearest Neighbor

Overall, I believe the data structure described above contributes to the success of the nearest neighbor algorithms in producing classifiers that perform well here as it relates to how the kNN classifies instances as similar using a distance measure. One thing to notice is that while increasing the amount of neighbors to consider results in improved performance prior to binarization, optimal performance is capped at 1 nearest neighbor once the data is transformed to binary. I suspect that this would vary as a function of how many bins are used when discretizing the data—here I used supervised discretization which only varied in the number of segments on two variables (magnesium and alcohol), depending on whether or not the options for better encoding and Kononenko's multi-interval discretization are selected.

References

[1] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," 2009.

[2] A. Frank and A. Asuncion, "UCI Machine Learning Repository." University of California, School of Information and Computer Science, Irvine, CA, 2010.

[3] A. G. K. Janecek, W. N. Gansterer, M. A. Demel, and G. F. Ecker, "On the Relationship Between Feature Selection and Classification Accuracy," in *JMLR: Workshop and Conference Proceedings*, 2008, pp. 90–105.

[4] P. Tan, M. Steinbach, V. Kumar Introduction to Data Mining Reading, MA: Addison-Wesley, 2005.

[5] T. M. Mitchell, "CHAPTER 1 GENERATIVE AND DISCRIMINATIVE CLASSIFIERS : NAIVE BAYES AND LOGISTIC REGRESSION Learning Classifiers based on Bayes Rule," in *Machine Learning*, 2010, pp. 1–17.

[6] I. Rish, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 2001, vol. 3, no. 22, pp. 41–46.

[7] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka : Practical Machine Learning Tools and Techniques with Java Implementations."

[8] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, vol. 1. Cambridge University Press Cambridge, 2008.