DESIGN AND DEVELOPMENT OF NAÏVE BAYES CLASSIFIER

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Bandana Garg

In Partial Fulfillment
for the Degree of
MASTER OF SCIENCE

Major Program:
Computer Science

June 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

DESIGN AND DEVELOPMENT OF NAÏVE BAYES CLASSIFIER

By

Bandana Garg

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Kendall Nygard
Chair

Brian Slator

Limin Zhang

Approved:

07/01/2013
Date

Brian Slator
Department Chair

ABSTRACT

The naïve Bayes classifier is a simple form of Bayesian classifiers which assumes all the features are independent of each other. Despite this assumption, the naïve Bayes classifier's accuracy is comparable to other sophisticated classifiers.

In this paper we designed and developed a naïve Bayes classifier for a better understanding of the algorithm. The classifier is tested on two different data sets from the University of California at Irvine machine learning repository. Different cross validation methods are used to calculate the accuracy of the developed classifier. The different accuracies obtained are compared to get the best accuracy of the classifier. This value is also compared with accuracies obtained for the same data sets using different algorithms reported in other papers. It was observed from the comparisons that the naïve Bayes classifier's results are very comparable to other algorithms.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Kendall Nygard for his continuous guidance, support, patience, encouragement, and motivation. I would also like to thank all my committee members for their support.

I also thank my family for their continuous support. Finally, I would like to thank all my friends, especially Basudha Pradhan, Erin Mullen, and Sydney Addy, for their motivation and for being there for me.

DEDICATION

I dedicate this paper to my father Dr. Govind Ram Agrawal and my mother Mrs. Rama Agrawal for being the inspiration in my life.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

DEDICATION v

LIST OF TABLES ix

LIST OF FIGURES x

1. INTRODUCTION 1

1.1. Objectives 2

1.2. Document Outline 2

2. LITERATURE REVIEW 4

2.1. Machine Learning 4

2.1.1. Types of Machine Learning Algorithms 4

2.1.2. Supervised Machine Learning 5

2.1.2.1. Process of Supervised Learning 6

2.2. Cross Validation 7

2.2.1. Holdout Method 7

2.2.2. K-Fold Cross Validation 7

2.2.3. Leave-One-Out Cross Validation 8

2.3. Bayes' Theorem 8

2.4. Naïve Bayes Classifier 9

2.4.1. Algorithm 10
2.4.2. Zero-Frequency Problem 11
2.4.3. Numerical Predictors 11
2.4.4. Avoiding Floating Point Underflow 12
3. NAÏVE BAYES CLASSIFIER 13
3.1. Design 13
3.1.1. Implementation Choices 13
3.1.1.1. Microsoft Visual Studio, C# and .NET Framework 13
3.1.1.2. EPPlus 14
3.1.2. System Design 14
3.2. Development 18
3.2.1. Read Input File 18
3.2.2. Select Cross Validation Method 19
3.2.3. Calculations 24
3.2.3.1. Calculate Prior Probabilities of Each Class 24
3.2.3.2. Calculate Conditional Probabilities 24
3.2.3.3. Calculate Posterior Probabilities 25
3.2.3.4. Classification 26
3.2.3.5. Calculate Accuracy 26
3.3. Experimentation 28

3.3.1. Data Sets 28
3.3.1.1. Mushroom Data Set 28
3.3.1.2. Iris Data Set 31
3.3.2. Data Pre-Processing 32
3.3.2.1. Mushroom Data Set 32
3.3.2.2. Iris Data Set 33
3.3.3. Experiments 33
3.3.3.1. Experiment 1: Accuracy Using K = 50 in K-Fold Cross Validation 33
3.3.3.1.1. Results 35
3.3.3.2. Experiment 2: Accuracy for All Cross Validation Methods 35
3.3.3.2.1. Results 37
3.4. Comparison with Other Algorithms 37
4. CONCLUSION 40
5. REFERENCES 41

LIST OF TABLES

Table Page

1. Instances with Known Classes in Supervised Machine Learning 5
2. Summary Statistics of Iris Data Set 32
3. Results of Experiment 1 for Both Data Sets 33
4. Results of Experiment 2 for Both Data Sets 35
5. Description of Algorithms Used in Comparison 38
6. Comparison of Accuracies from Different Algorithms 39

LIST OF FIGURES

Figure Page

1. The Structure of the Naive Bayesian Network (4) 10
2. Flowchart for Naive Bayes Classifier 16
3. Pseudo Code: Read the Input File using EPPlus Library 18
4. Pseudo Code: Creating Training Set and Test Set in Holdout Cross Validation..... 19
5. Pseudo Code: Classifier Using Holdout Cross Validation 20
6. Pseudo Code: Creating Training Set and Test Set in K-Fold Cross Validation 21
7. Pseudo Code: Classifier Using K-Fold Cross Validation 22
8. Pseudo Code: Classifier Using Leave-One-Out Cross Validation 23
9. Pseudo Code: Calculating Prior Probabilities 24
10. Pseudo Code: Calculation Posterior Probabilities 25
11. Screenshot: Naive Bayes Classifier Running with K-fold Cross Validation 27
12. Accuracy of Classifier Using K-Fold Cross Validation (Mushroom Data Set) 34
13. Accuracy of Classifier Using K-Fold Cross Validation (Iris Data Set)..... 34
14. Accuracy of Classifier Using Different Cross Validation Methods 36

1. INTRODUCTION

Classification in data analysis is the task of assigning a class to instances of data described by a set of attributes. Classification or supervised classification includes the construction of a classifier which is trained on a set of training data that already has the correct class assigned to each data point. This builds a concise model of the distribution of class labels. It is then used to classify new data where the values of features are known but the class is unknown.

Many algorithms have been developed for supervised classification based on artificial intelligence (logical algorithms such as decision trees), perceptron-based techniques (single layered perceptron, multilayered perceptron), and statistical learning techniques (Bayesian networks, instance based techniques) [1] .

Bayesian classification is based on Bayes theorem. Bayesian classifiers assign the most likely class to a given instance described by a set of attributes. The Naïve Bayes Classifier assumes that the effect of each attribute on a class is statistically independent of all other attributes [2]. This assumption, called class conditional independence, is made to simplify computation, and in this sense, is considered 'naive' [3]. Despite this assumption, the naïve Bayes classifier's performance is competitive with other sophisticated techniques and has proven effective in many practical applications [4] [2] [5].

Naïve Bayes classifier works best in two cases: when the features are completely independent and secondly when the features are functionally dependent. The worst performance is seen in between these two extremes [5]. Many extensions and optimizations have been put forth to improve the naïve Bayesian classifier's performance. In [4], the authors suggest a Tree Augmented Naïve Bayes (TAN) classifier, which has significant

performance improvements over the naïve Bayesian classifier. The authors in [6] show that choosing structures by maximizing conditional likelihood while setting parameters by maximum likelihood also yields better results.

The popularity of naïve Bayes classifier has increased and is being adopted by many because of its simplicity, computational efficiency, and it's good performances for real-world problems.

1.1. Objectives

The objectives of this paper are:

1. To design and develop a naïve Bayes classifier for better understanding of the algorithm.
2. To test the performance of the classifier on the selected datasets.
3. To use different cross validation methods to evaluate the performance of the algorithm.

1.2. Document Outline

Section 1 of this document gives a general introduction of the Naive Bayes Classifier, its advantages over other algorithms, and enhancements. It also includes the objectives of this paper.

Section 2 gives a summary of where and how the Naïve Bayes algorithm is used for classification. It describes the process of supervised machine learning and the different cross validation methods. This section includes an explanation of the Bayes' theorem on which the naïve Bayes classifier is based on, and it describes the algorithm for naïve Bayes classifier.

Section 3 includes the design, development, and the testing of the classifier that was developed. It describes in detail the design of the system and its development. It includes a

description of the dataset that was used to test the algorithm and the results of those test. This section ends with the comparison of the performance using different cross validation methods

Section 4 concludes this paper. It presents conclusions drawn from the experiments done using the algorithm on the selected datasets.

2. LITERATURE REVIEW

2.1. Machine Learning

"Machine learning is programming computers to optimize a performance criterion using example data or past experience" [7]. Learning is done by the execution of a computer program to optimize the parameters of a defined model, which can be predictive to make predictions in future or descriptive to gain knowledge from data or both. There are two things that need to be achieved in the learning process. First, the training needs to be done with an efficient algorithm to solve the optimized problem, and to store and process the data. Second, the trained model needs an efficient representation and algorithm for inference [7].

2.1.1. Types of Machine Learning Algorithms

As listed in [7], machine learning algorithms are organized in a taxonomy based on their desired outcomes. The common algorithm types are:

- Supervised learning – These algorithms generate a function based on the training data set that maps inputs to predetermined labels or classes.
- Unsupervised learning – These algorithms are not provided with classification. They seek out similarities between data points to form clusters.
- Semi-supervised learning – The algorithms combine both labeled and unlabeled data to generate a classifier.
- Reinforcement learning – These algorithms learn by interacting with an environment. Instead of being trained on an existing data, these algorithms learn from their actions which are selected based on past actions (exploitation) or by new choices (exploration). The algorithm receives a numeric reward from the environment for a

successful outcome of an action and it selects actions that maximize the accumulated reward over time [8].

- Transduction – These algorithms are similar to supervised algorithms but instead of explicitly specifying a function, they try to predict new outputs based on training inputs, training outputs, and testing inputs.
- Learning to learn – These algorithms learn their own inductive bias based on previous experience.

2.1.2. Supervised Machine Learning

Supervised learning is common in classification problems where the goal is to have the computer learn a predefined classification. Supervised learning algorithms learn from externally supplied instances of data, represented by a set of features and a class, to create a function that makes predictions about the classification of future instances or new data. Table 1 shows the structure of data used in supervised learning. Each instance of data is defined by a set of features and a class.

Table 1: Instances with Known Classes in Supervised Machine Learning

Data in standard format					
Case	Feature 1	Feature 2	…	Feature n	Class
1	XX	X		XXX	Yes
2	XX	X		XXX	No
3	XX	X		XXX	Yes
…					

2.1.2.1. Process of Supervised Learning

The process of supervised learning is described in the steps below.

Step 1: The first step in supervised learning is the collection of dataset. If the expertise is available then data on the informative features can be collected. If not, then data is collected on all features in hopes that the relevant ones can be isolated. Data collected in such a way contains noise and missing values and needs intensive pre-processing [1].

Step 2: The data-preprocessing step involves reducing noise by instance selection. There are several methods available to handle missing data. Instance selection is also used to handle the infeasibility of learning from extremely large datasets.

Step 3: The training set is defined by feature subset selection, in which the irrelevant and redundant features are removed. In cases where some features are dependent on each other, new features are constructed from the basic feature set.

Step 4: Algorithm selection is a very important step. There are numerous algorithms available for machine learning.

Step 5: The algorithm is trained using a training data set.

Step 6: The algorithm is tested with a test data set. Once the evaluation from the preliminary testing is satisfactory, then the classifier can be used for predictions. The classifier is evaluated based on prediction accuracy (the percentage of correct predictions divided by the total number of predictions) [1]. The evaluation technique used is called cross validation which is described in section 2.2.

Step 7: If the accuracy is not satisfactory then the process needs to return to a previous step to re-examine certain choices made. As listed in [1], some of the factors that need to be reexamined can be:

- Relevant features for the problem may be missing
- Larger dataset may be needed
- The dimensionality of the problem may be too high
- The selected algorithm may be inappropriate
- Parameter tuning may be needed

2.2. Cross Validation

Using the same set of data for the training and validation of an algorithm yields an overoptimistic result [9]. Cross Validation is based on the principle that testing the algorithm on a new set of data yields a better estimate of its performance [10].

Most real applications have a limited amount of data. Because of this the dataset is split into the training sample and the validation sample. The training sample is used to train the algorithm and the validation sample is used as "new data" to evaluate the performance of the algorithm [11].

2.2.1. Holdout Method

The holdout method is the simplest kind of cross validation in which the dataset is separated into two sets: the training set and the validation set. The algorithm is trained using the training set only. The algorithm is then used to evaluate the data in the validation set [11]. The errors it makes are accumulated to give the mean absolute test set error, which is used to evaluate the algorithm. However, since the evaluation may depend on which data is in the training set and in the validation set, its evaluation can have a high variance.

2.2.2. K-Fold Cross Validation

K-fold cross validation is an improvement of the holdout method. In the k-fold cross validation method the dataset is divided into k subsets and the holdout method is repeated k

times. In each run, one of the k sets is used as the validation set and the remaining k-1 sets are put together to form the training set. Then the average error across all *k* trials is computed. This reduces the dependency of the evaluation on how the dataset is separated. Every data point is in the training set k times and in the validation set k-1 times. The variance in the result is reduced as k is increased. The disadvantage of this method is that it takes k times as much computation for an evaluation since the training algorithm has to be run k times [12].

2.2.3. Leave-One-Out Cross Validation

Leave-one-out cross validation is a logical extreme of k-fold cross validation where k is equal to the number of data points. The training on the algorithm is done on all data points except for one. The evaluation given by the leave-one-out cross validation is good but computationally expensive [12].

2.3. Bayes' Theorem

Bayes' Theorem is a statement from probability theory that allows for the calculation of certain conditional probabilities. Conditional probabilities are those probabilities that reflect the influence of one event on the probability of another event. The term generally used in Bayes' theorem are prior probability and posterior probability. The prior probability of a hypothesis or event is the original probability obtained before any additional information is obtained. The posterior probability is the revised probability of the hypothesis using some additional information or evidence obtained.

Bayes' Theorem can be written as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \text{(Eq. 1)}$$

Where,

P(A) is the prior probability of A

P(B) is the prior probability of B

P(A|B) is the posterior probability of A given B

P(B|A) is the posterior probability of B given A

Since the denominator P(B) in Eq. 1 is the probability of the evidence without any knowledge of the event A, and since the hypothesis A can be true or false, Bayes' theorem can also be written as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A) \times P(A) + P(B|\neg A) \times P(\neg A)} \quad \text{(Eq. 2)}$$

Where,

P(¬A) is the probability of A being false

P(B|¬A) is the probability of B given A is false

2.4. Naïve Bayes Classifier

A naïve Bayes classifier is a probabilistic classifier based on applying Bayes' theorem with strong independence assumptions. The naïve Bayesian classifier was first described in [13] in 1973 and then in [2] in 1992. When represented as a Bayesian network, a naïve Bayes classifier has the structure depicted in Figure 1. It shows the independence assumption among all features in a data instance.

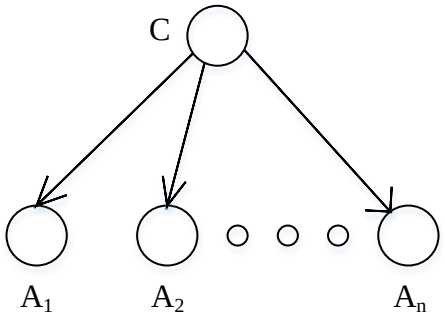

Figure 1: The Structure of the Naive Bayesian Network (4)

2.4.1. Algorithm

As written in [5], a naïve Bayes classifier can be defined as below. Variables are denoted using capital letters such as X_i, and their values will be denoted by lower-case letters such as x_i, and sets of variables are denoted by boldface letters such as **X**.

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a finite set of observed random variables, called features, where each feature takes values from its domain D_i. The set of all feature sets is denoted by $\Omega = D_1 \times \ldots \times D_n$. Let C, such that $c \in \{0, \ldots, u-1\}$, be an unobserved random variable denoting the class of a set of features.

A hypothesis $h : \Omega \rightarrow \{0, \ldots, u-1\}$, that assigns a class to any given set of variables is defined as a classifier. Each class c is assigned a discriminant function $f_c(\boldsymbol{x}), c = 0, \ldots, u-1$. The classifier selects the class with the maximum discriminant function on a given set of variables, written as $h(\boldsymbol{x}) = \arg max_{c \in \{0,\ldots,u-1\}} f_c(x)$.

The Bayes classifier $h^*(\boldsymbol{x})$ uses the posterior probabilities given a set of variables as the discriminant function, i.e. $f^*(\boldsymbol{x}) = P(C = c \mid X = x)$. Applying Bayes' theorem from Eq. 1 to this function gives $P(C = c \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{P(\boldsymbol{X}=\boldsymbol{x} \mid C=c)\, P(C=c)}{P(\boldsymbol{X}=\boldsymbol{x})}$. Since P(**X** = **x**) is the same for all classes it can be ignored. Hence, the Bayes' discriminant function can be written as $f^*(\boldsymbol{x}) = P(\boldsymbol{X} = \boldsymbol{x} \mid C = c)\, P(C = c)$, where $P(\boldsymbol{X} = \boldsymbol{x} \mid C = c)\, P(C = c)$ is called the

class-conditional probability distribution (CPD) [5]. Thus the Bayes' classifier written as in Eq. 3 finds the maximum posterior probability hypothesis given **x**.

$$h^*(\boldsymbol{x}) = \arg max_c \, P(\boldsymbol{X} = \boldsymbol{x} \,|C = c) \, P(C = c) \qquad \text{(Eq. 3)}$$

Applying the assumption that features are independent given the class on Eq. 3, we can get the naïve Bayes classifier.

$$f_c^{NB}(\boldsymbol{x}) = \prod\nolimits_{j=1}^{n} P\big(X_j = x_j \,\big|C = c\big) \, P(C = c) \qquad \text{(Eq. 4)}$$

2.4.2. Zero-Frequency Problem

When a feature value does not occur with every class value, a value 1 is added to the count of every feature value-class combination.

2.4.3. Numerical Predictors

When features have continuous data or numerical data, there are two ways to handle the data:

1. The first option is to transform the numerical values to their categorical counterparts, called binning, before constructing their frequency tables.
2. The second option is to use the distribution of the numerical values to guess the frequency. One of the common practices is to assume normal distribution for the numerical values, which is defined by two parameters: mean and standard deviation.

$$P(X = x \,|C = c) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad \text{(Eq. 5)}$$

Where,

Mean = $\mu = \frac{1}{n}\sum_{j=1}^{n} x_j$

Standard Deviation = $\sigma = [\frac{1}{n-1}\sum_{j=1}^{n}(x_j - n)^2]^{0.5}$

n = number of times the feature occurs with the given class

2.4.4. Avoiding Floating Point Underflow

Because of the floating point limitations during computation, when working with large data the Naïve Bayes Classifier needs to be optimized as below. Instead of storing the probabilities, the natural logarithm of those probabilities are stored. And instead of multiplying them, the numbers are added.

$$f_c(x) = log_e\big(P(C = c)\big) + \sum_{j=1}^{n} log_e(P\big(X_j = x_j \,\big|C = c)) \qquad \text{(Eq. 6)}$$

3. NAÏVE BAYES CLASSIFIER

3.1. Design

This section describes the implementation choices and the design and development process of the Naïve Bayes classifier built.

3.1.1. Implementation Choices

The classifier was implemented in C# using Microsoft Visual Studio. The implementation choices have been described in this section.

3.1.1.1. Microsoft Visual Studio, C# and .NET Framework

Microsoft Visual Studio (VS) is an Integrated Development Environment (IDE) that provides a set of development tools for developing .NET web applications, XML Web Services, desktop applications, and mobile applications. VS makes development easier through time-saving and convenient features such as intelliSense, designers, and debugging.

The .NET framework is a software framework that provides a large library and language interoperability between several programming languages. The Base Class Library (BCL) includes a large number of common functions. Some of the namespaces from the BCL being used in the development of the naïve Bayes classifier include System.Collections, System.IO, System.Threading, System.Linq, and System.Windows.Forms.

C# is an object oriented programming language designed to be fully compatible with the Microsoft .NET framework.

The naïve Bayes classifier has been developed using Visual Studio 2012, .Net Framework 4.5, and C#.

3.1.1.2. EPPlus

EPPlus is a .net library that reads and writes Excel 2007/2010 files using the Open Office Xml format (.xlsx) [14]. The project is licensed under the GNU Library General Public License (LGPL). EEPlus version 3.1 is used in the development of the naïve Bayes classifier because of its complete and simple integration with .NET and its speed in loading excel files (50,000 cells in seconds [14]).

3.1.2. System Design

The naïve Bayes classifier is a desktop application developed using windows forms. The classifier is designed for categorical data for features and is generalized to read any dataset with categorical data.

The following structure is required in the input file for the file to be read correctly:

- The input file is a Microsoft Excel file. The file should be in .xlsx format.
- The feature values are categorical and not numerical (continuous data)
- The first column in the data is the class followed by the feature values
- The first row in the data is the feature names

The system is designed to run the algorithm using three cross validation methods: holdout method, k-fold cross validation, and leave-one-out cross validation. These cross validation methods are described in section 2.2.

The system is designed to read an input file (.xlsx file) containing the data set selected by the user. Depending on the cross validation method selected by the user, the input data set is split into training set and test set.

Using the training set, the prior probabilities of each class is calculated. Using a single instance from the test set, the conditional probabilities for each feature value is calculated.

These values are then used to calculate the posterior probabilities for each class. The class with the highest posterior probability is assigned as the class for that test instance. This process is done on each instance in the test set.

By comparing these assigned class values to the actual class values of the test data, the number of correct classification is obtained which is then used to calculate the accuracy of the classifier.

For k-fold classification, the dataset is split into the number k entered by the user. The entire process from creating training set and test set to calculating the accuracy is performed k times using each set as the test set in each iteration. The training set is formed by merging the remaining k-1 sets. The accuracies obtained from all iterations are averaged to get the accuracy of the classifier.

The workflow of the system is shown in the flowchart below (Figure 2).

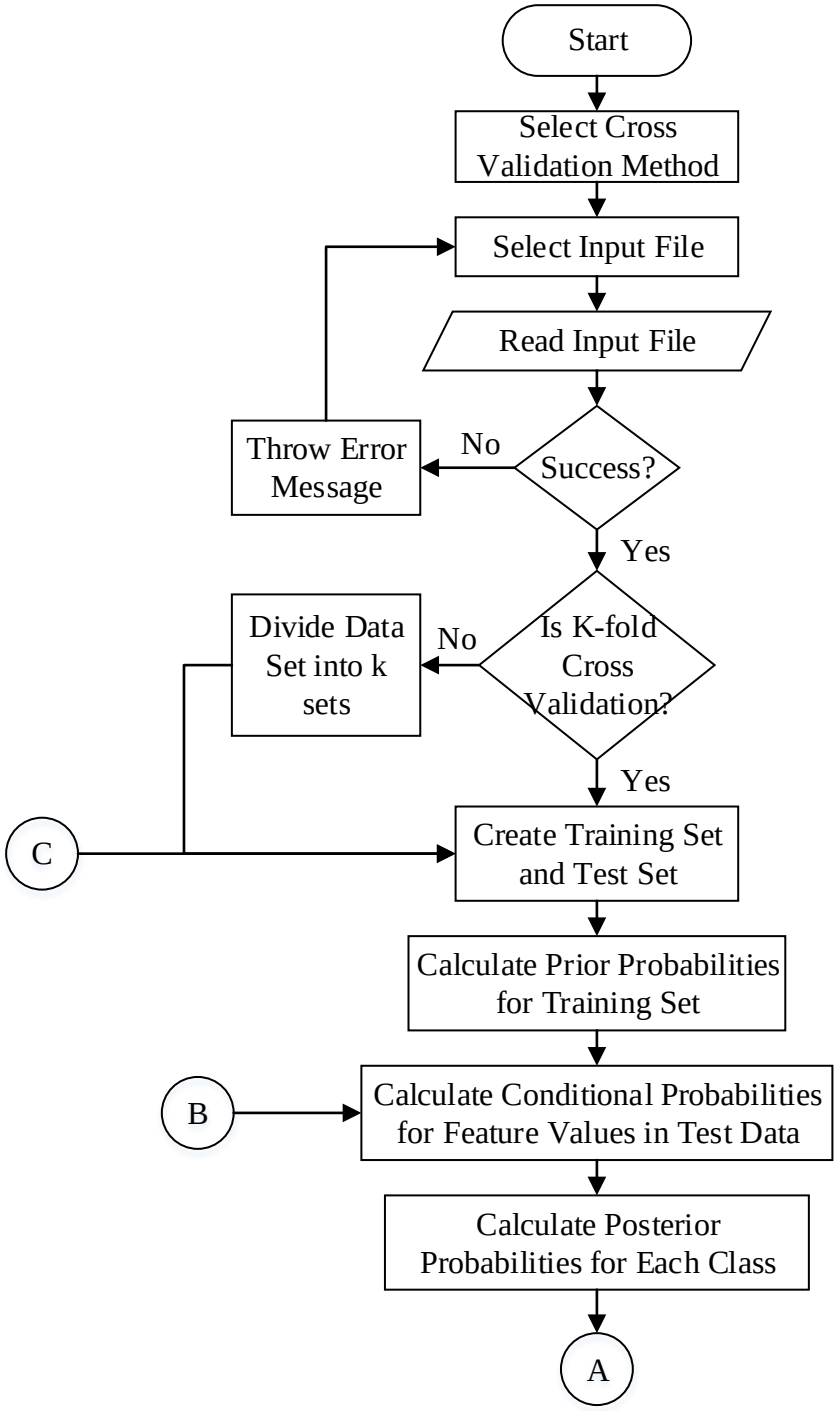

Figure 2: Flowchart for Naive Bayes Classifier

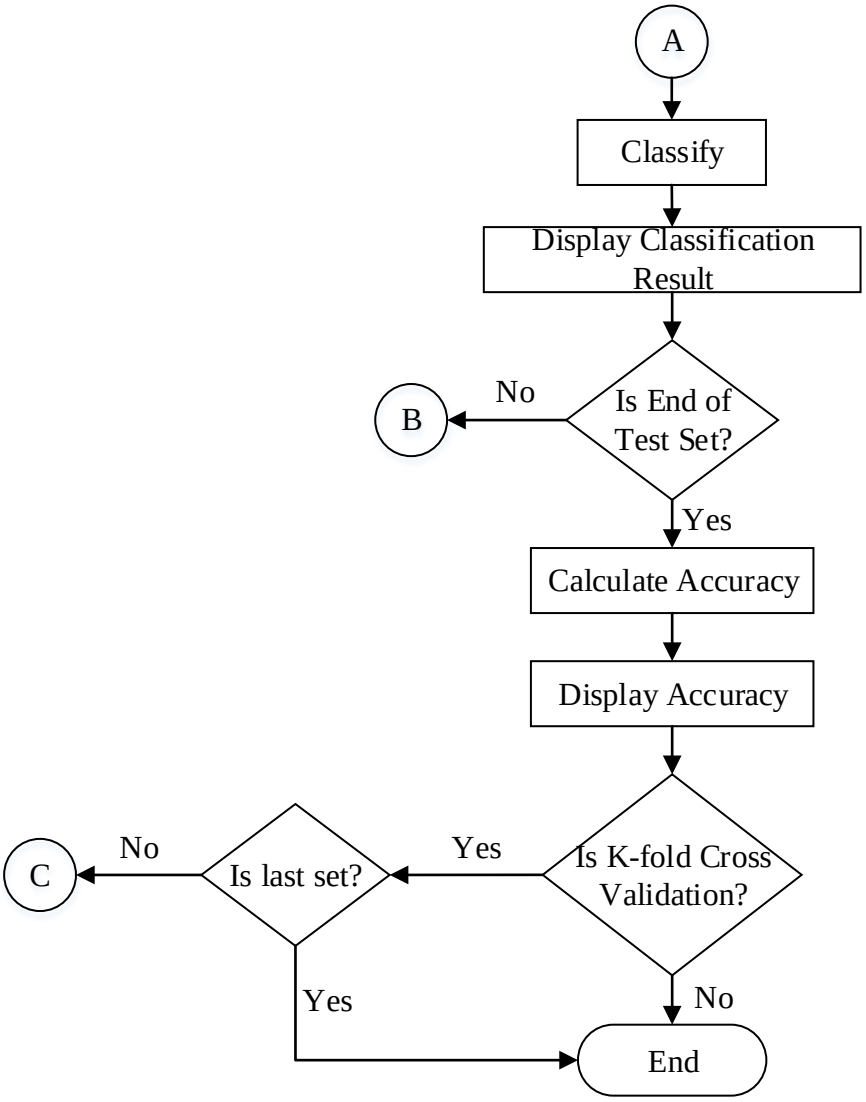

Figure 2: Flowchart for Naive Bayes Classifier (continued)

3.2. Development

This section describes each step of the general workflow of the classifier using pseudo code.

3.2.1. Read Input File

The user is required to select the data file in .xlsx format using the UI. The input file selected by the user is read using EPPlus. The EPPlus library can be accessed by adding a reference to its dll and adding "OfficeOpenXml" to the list of included libraries.

```
for (int rowNumber = 2; rowNumber <= currentWorksheet.Dimension.End.Row;
rowNumber++)
{
        var instanceValues = new List<string>();
        for (int colNumber = 1; colNumber <=
        currentWorksheet.Dimension.End.Column; colNumber++)
        {
                string featureValue = currentWorksheet.Cells[rowNumber,
                colNumber].Value.ToString();
                instanceValues.Add(featureValue);
        }

        var dataInstance = new DataInstance();
        dataInstance.Features = instanceValues;
        dataSet.DataInstances.Add(dataInstance);
}
```

Figure 3: Pseudo Code: Read the Input File using EPPlus Library

Explanation: The rowNumber is started at 2 because the first row contains the feature names. Each row is added as an instance of the class DataInstance which is a list of feature values for one instance of the data. All the DataInstances are then added as an instance of the class DataSet which is a list of the object DataInstance, and represents our data set.

3.2.2. Select Cross Validation Method

Depending on the cross validation method selected by the user through the UI, the training set and the test set are created for training and validation of the classifier.

Holdout Method: For the holdout method the dataset is split into half creating the training set and the test set (Figure 4).

```
INPUT: DataSet

OUTPUT: TrainingSet, TestSet

1 BEGIN
2 {

3     SET trainingSetSize = DataSet.Count / 2

4     SET testSetSize = DataSet.Count - trainingSetSize

5     SET TrainingSet = DataSet.Take(trainingSetSize)

6     SET TestSet = DataSet.Skip(trainingSetSize).Take(testSetSize)

7 }

8 END
```

Figure 4: Pseudo Code: Creating Training Set and Test Set in Holdout Cross Validation

The algorithm is then applied to each instance in the test set to get the accuracy of the classifier (Figure 5).

```
INPUT: TrainingSet, TestSet

OUTPUT: Accuracy of Classifier

1 BEGIN
2 {

3   Set correctClassificationCount to 0

4   FOREACH TestInstance Ti in TestSet

5    {

6       CALL TrainClassifier with TrainingSet

7       CALL ClassifyTestData with Ti

8       IF classification = Ti.Class THEN

9         INCREMENT  correctClassificationCount

10      ENDIF

11   }

12  CALL CalculateAccuracy with correctClassificationCount and TestSet.Count
    RETURNING Accuracy

13 }

13 END
```

Figure 5: Pseudo Code: Classifier Using Holdout Cross Validation

K-Fold Method: In the k-fold method the dataset is split into k different sets. In the developed algorithm the value of k is entered by the user using the UI. For each of the k sets, that set is used as the test set and the remaining k-1 sets are combined to form the training set (Figure 6).

```
INPUT: SplitLists
        List of k sets created by splitting the DataSet
OUTPUT: TrainingSet, TestSet
1 BEGIN
2 {
3    FOREACH SplitSet Si in SplitLists
4     {
5        SET TestSet = Si
6        SET TrainingSet = Remaining sets merged
7     }
8 }
9 END
```

Figure 6: Pseudo Code: Creating Training Set and Test Set in K-Fold Cross Validation

The algorithm for the classifier is then applied to each instance of the test set for each of the k iterations. The accuracy for each iteration is calculated which is then averaged out to get the classifiers accuracy (Figure 7).

```
INPUT: SplitLists
OUTPUT: Accuracy of Classifier
1 BEGIN
2 {
3   SET noOfSplits = k
4   FOR noOfSplits = 1 to k
5    {
6      Set correctClassificationCount to 0SplitLists
7      Call SetTrainingSetAndTestSet with
7      FOREACH TestInstance Ti in TestSet
8       {
9          CALL TrainClassifier with TrainingSet
10           CALL ClassifyTestData with Ti
11           IF classification = Ti.Class THEN
12              INCREMENT  correctClassificationCount
13          ENDIF
14       }
15     CALL CalculateAccuracy with correctClassificationCount and TestSet.Count
        RETURNING Accuracy
16   }
17   Calculate average accuracy
18 }
19 END
```

Figure 7: Pseudo Code: Classifier Using K-Fold Cross Validation

Leave-One-Out Method: In this cross validation method, the algorithm is carried out n times where n is the number of instances in the data set. In each run, one of the instances is used as the test data and the remaining instances are used as the training set (Figure 8)

```
INPUT: DataSet

OUTPUT: Accuracy of Classifier

1 BEGIN
2 {

3    Set correctClassificationCount to 0

4    FOREACH DataInstance Di in DataSet

5     {

6.        SET TestData = Di

7.        SET TrainingSet = DataSet.RemoveAt(i)

6         CALL TrainClassifier with TrainingSet

7         CALL ClassifyTestData with Ti

8         IF classification = Ti.Class THEN

9           INCREMENT  correctClassificationCount

10        ENDIF

11     }

12   CALL CalculateAccuracy with correctClassificationCount and TestSet.Count
      RETURNING Accuracy

13 }

13 END
```

Figure 8: Pseudo Code: Classifier Using Leave-One-Out Cross Validation

3.2.3. Calculations

The calculations done for classification are described in this section.

3.2.3.1. Calculate Prior Probabilities of Each Class

The prior probability of each class is calculated by dividng the number of data instances with that class in the training set by the total number of instances in the training set (Figure 9).

```
INPUT: ClassCount, TotalCount
        ClassCount is the count of instances with that class value in the training set
        TotalCount is the count of all instances in the training set
OUTPUT: Prior Probability of the class
1 BEGIN
2 {
3     Compute PriorProbability as ClassCount / TotalCount
4  }
5 END
```

Figure 9: Pseudo Code: Calculating Prior Probabilities

3.2.3.2. Calculate Conditional Probabilities

The conditional probabilities for each feature value in the test data are calculated by getting the count of instances with that feature value in a particular class and dividing it by the count of instances with the same class in the training set. This is done for each class in the data set.

```
INPUT: PriorProbabilityList, ConditionalProbabilityList
        PriorProbabilityList contains the prior probabilities of all classes
        ConditionalProbabilityList contains the list of conditional probabilities for all
        features in the test data for that class
OUTPUT: PosteriorProbabilityList
            List of posterior probabilities of each class
1 BEGIN
2 {
3    FOREACH ClassProbability Pc in PriorProbabilityList
4     {
5        SET posteriorProbability = Pc
6        FOREACH ConditionalProbability CPi in ConditionalProbabilityList
7          {
8            Compute posteriorProbability as posteriorProbability * CPi
9          }
10       Add posteriorProbability to PosteriorProbabilityList
11      }
12 }
13 END
```

Figure 10: Pseudo Code: Calculation Posterior Probabilities

3.2.3.3. Calculate Posterior Probabilities

The posterior probability for each class given the feature values in the test data are calculated by using the naïve Bayes classifier formula in Eq. 4 on the prior probability and conditional probability values.

3.2.3.4. Classification

The class with the highest posterior probability is assigned as the class for the test data.

3.2.3.5. Calculate Accuracy

The class assigned to the test data is compared with the actual class of the test data to get the count of correct classifications. The accuracy of the classifier is calculated by dividing the number of correct classifications by the total number of classifications.

In case of k-fold cross validation, the accuracies obtained from all runs are averaged to get the average accuracy of the classifier. The standard deviation is also calculated to get the range of acceptable accuracy, and to see if any of the accuracies are too high or too low.

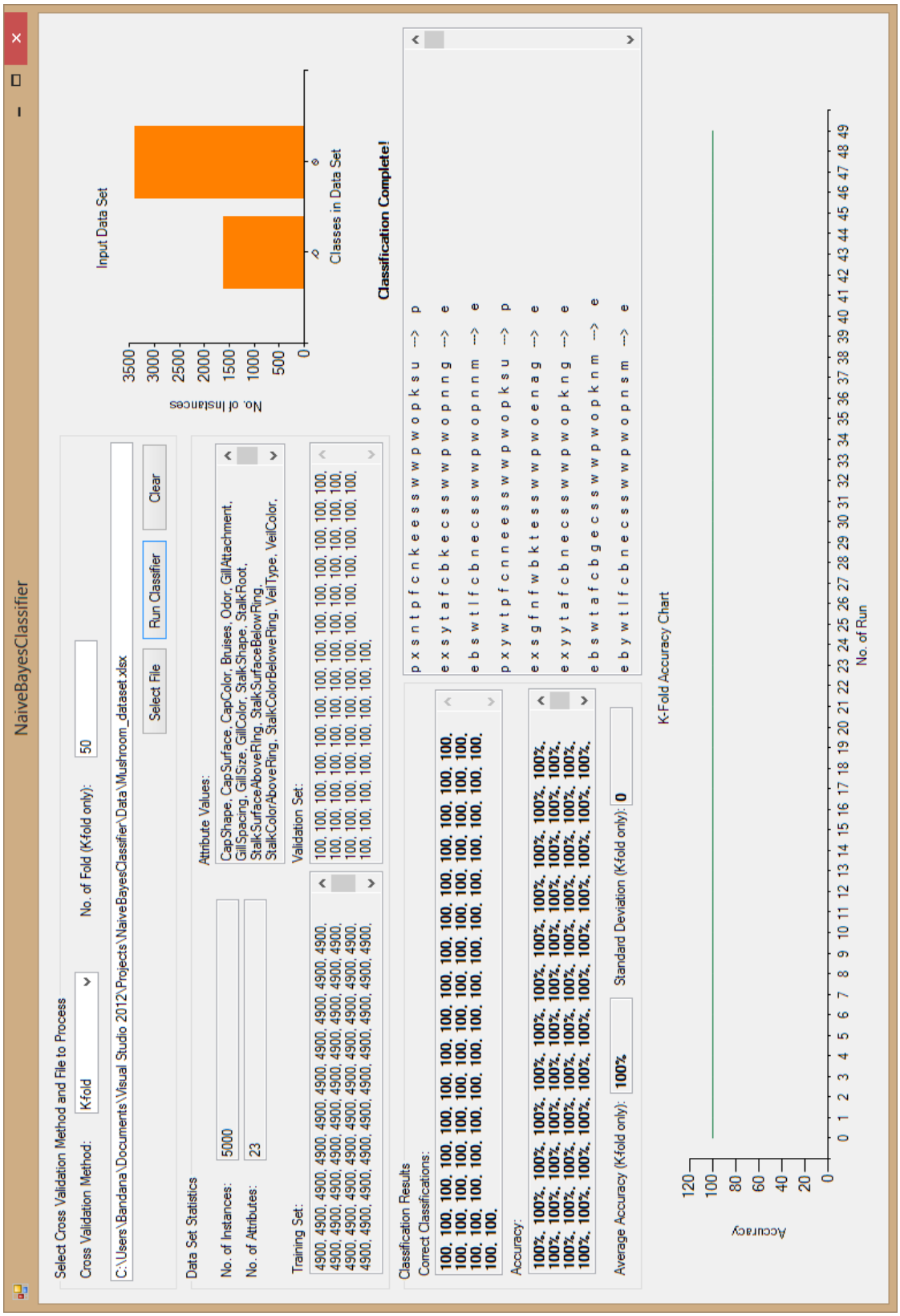

Figure 11: Screenshot: Naive Bayes Classifier Running with K-fold Cross Validation

3.3. Experimentation

The Naïve Bayes classifier developed was tested using two of the datasets from the machine learning repository at the Center for Machine Learning and Intelligent Systems at the University of California, Irvine (UCI) [15] [16].

3.3.1. Data Sets

Two different data sets from the UCI machine learning repository were used to test the developed classifier: Mushroom data set, and Iris data set. The data sets are described below.

3.3.1.1. Mushroom Data Set

All information on the data set has been taken from the UCI website [15].

1. Title: Mushroom Data Set
2. Source:
   - Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf
   - Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)
   - Donated Date: 27 April 1987
3. Data Set Information:

   This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like ``leaflets three, let it be'' for Poisonous Oak and Ivy.

4. Number of Instances: 8124
5. Number of Attributes: 22
6. Attribute Characteristics: Categorical
7. Attribute Information:
    1) cap-shape: bell = b, conical = c, convex = x, flat = f, knobbed = k, sunken = s
    2) cap-surface: fibrous = f, grooves = g, scaly = y, smooth = s
    3) cap-color: brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, yellow = y
    4) bruises?: bruises = t, no = f
    5) odor: almond = a, anise = l, creosote = c, fishy = y, foul = f, musty = m, none = n, pungent = p, spicy = s
    6) gill-attachment: attached = a, descending = d, free = f, notched = n
    7) gill-spacing: close = c, crowded = w, distant = d
    8) gill-size: broad = b, narrow = n
    9) gill-color: black = k, brown = n, buff = b, chocolate = h, gray = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y
    10) stalk-shape: enlarging = e, tapering = t
    11) stalk-root: bulbous = b, club = c, cup = u, equal = e, rhizomorphs = z, rooted = r, missing = ?
    12) stalk-surface-above-ring: fibrous = f, scaly = y, silky = k, smooth = s
    13) stalk-surface-below-ring: fibrous = f, scaly = y, silky = k, smooth = s
    14) stalk-color-above-ring: brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y

15) stalk-color-below-ring: brown = n, buff = b, cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, yellow = y

16) veil-type: partial = p, universal = u

17) veil-color: brown = n, orange = o, white = w, yellow = y

18) ring-number: none = n, one = o, two = t

19) ring-type: cobwebby = c, evanescent = e, flaring = f, large = l, none = n, pendant = p, sheathing = s, zone = z

20) spore-print-color: black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, purple = u, white = w, yellow = y

21) population: abundant = a, clustered = c, numerous = n, scattered = s, several = v, solitary = y

22) habitat: grasses = g, leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d

8. Missing Attribute Values:

   2480 of them (denoted by "?"), all for attribute number 11.

9. Classes:

   - Edible = e
   - Poisonous = p

10. Class Distribution

    - Edible: 4208 (51.8%)
    - Poisonous: 3916 (48.2%)

3.3.1.2. Iris Data Set

All information on the data set has been taken from the UCI website [16].

1. Title: Iris Plants Database
2. Source:
    - Creator: R. A. Fisher
    - Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    - Donated Date: July 1988
3. Data Set Information:

    This data set is one of the post popular in pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are not linearly separable from each other. The predicted attribute is the class of the iris plant.
4. Number of Instances: 150
5. Number of Attributes: 4
6. Attribute Characteristics: Real
7. Attribute Information:
    1) Sepal length in cm
    2) Sepal width in cm
    3) Petal length in cm
    4) Petal width in cm
8. Missing Attribute Values: None
9. Classes:
    - Iris Setosa

- Iris Versicolour
- Iris Virginica

10. Class Distribution:
    - Iris Setosa: 33.3%
    - Iris Versicolour 33.3%
    - Iris Virginica: 33.3%
11. Summary Statistics:

Table 2: Summary Statistics of Iris Data Set

	Min	Max	Mean	SD	Class Correlation
Sepal Length	4.3	7.9	5.84	0.83	0.7826
Sepal Width	2.0	4.4	3.05	0.43	-0.4194
Petal Length	1.0	6.9	3.76	1.76	0.9490 (high)
Petal Width	0.1	2.5	1.20	0.76	0.9565 (high)

3.3.2. Data Pre-Processing

Some data pre-processing was required in both the data sets used.

3.3.2.1. Mushroom Data Set

For the experiment, the data instances with missing values were removed. A subset of the remaining dataset containing 5000 data instances was used for testing the classifier.

1. Number of Instances: 5000
2. Class Distribution:
    - Edible: 3397 (67.94%)
    - Poisonous: 1603 (32.06%)

3.3.2.2. Iris Data Set

Since the classifier is built for categorical data, the real data in this data set has been integerized to fit the classifier design.

3.3.3. Experiments

Two different experiments were done using both the data sets to test the accuracy of the naïve Bayes classifier.

3.3.3.1. Experiment 1: Accuracy Using K = 50 in K-Fold Cross Validation

In the first experiment, the classifier was tested using the k-fold cross validation method. The value of k was input as 50. For each of the 50 runs, the accuracy was calculated. Using all the accuracies obtained, the average accuracy and the standard deviation was calculated.

For the mushroom data set, each run had 4900 data instances in the training set and 100 instances in the test set.

The iris data set had 147 instances in the training set and 3 instances in the training set for each of the runs.

Table 3: Results of Experiment 1 for Both Data Sets

	Accuracy	Standard Deviation
Mushroom Data Set	100%	0
Iris Data Set	89.33%	22

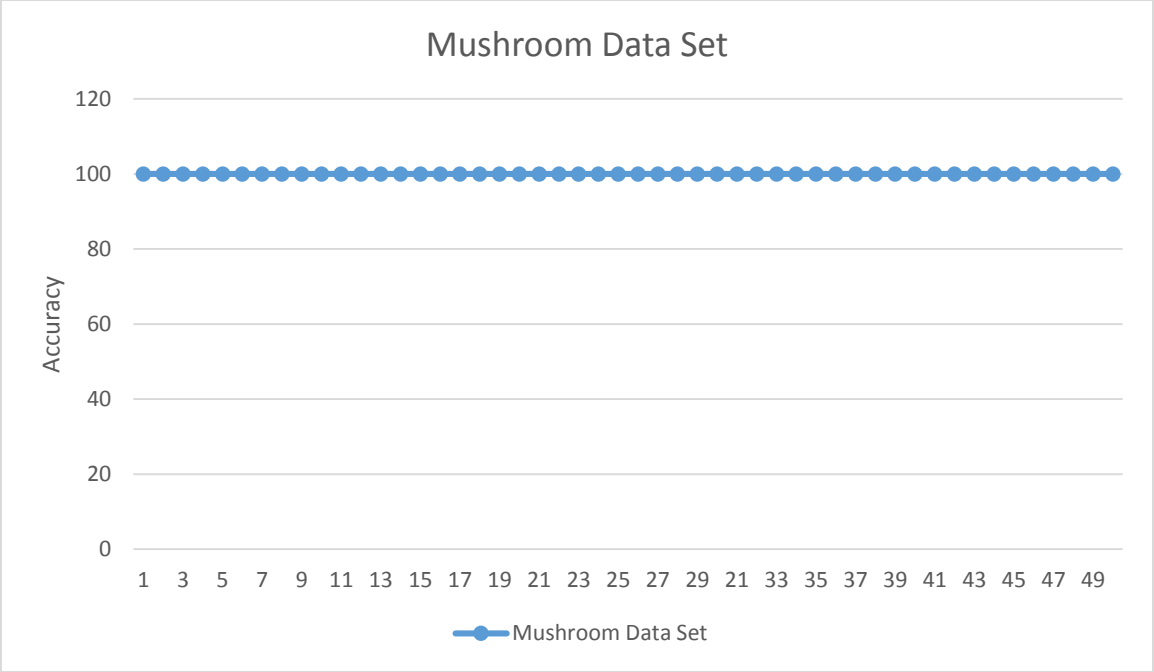

Figure 12: Accuracy of Classifier Using K-Fold Cross Validation (Mushroom Data Set)

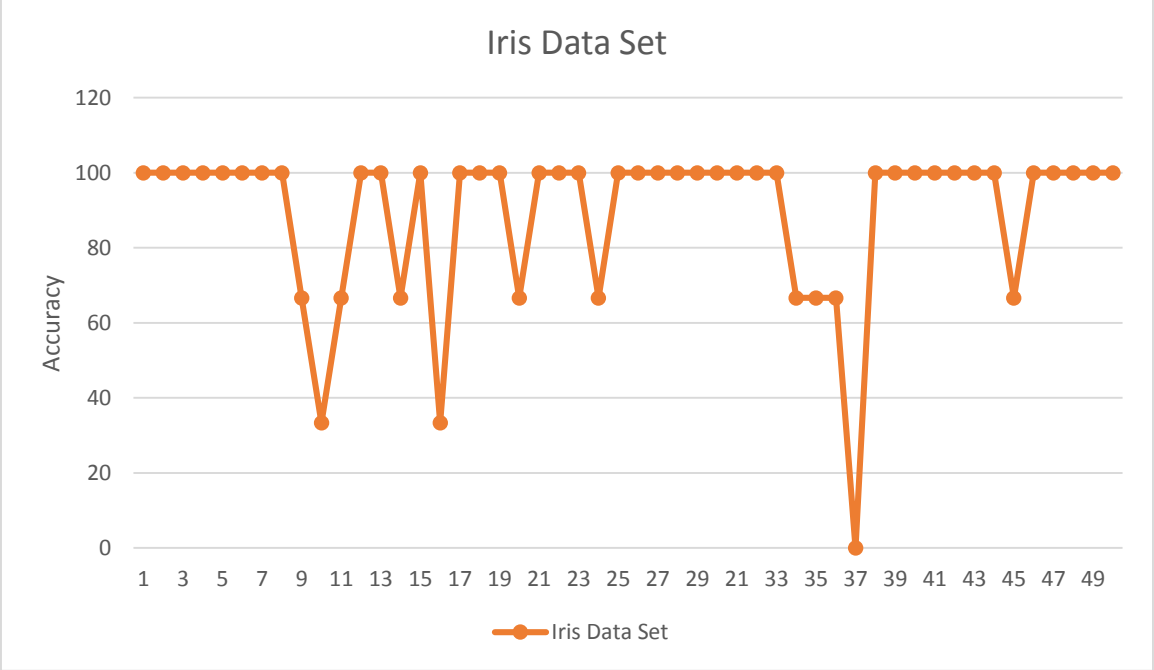

Figure 13: Accuracy of Classifier Using K-Fold Cross Validation (Iris Data Set)

3.3.3.1.1. Results

From the accuracies obtained from the classifier, listed in Table 3, it is seen that the accuracy for the mushroom data set is consistent at 100% for all 50 runs. On the other hand, for the iris data set the average accuracy is calculated at 89.33% with a standard deviation of 22. The accuracies for this data set were varying from 0% to 100% with more than half of them above the 60% mark. It can be seen that the results of the mushroom data set are much better than that of the iris set. The better accuracy for the mushroom data set can be attributed to the following:

1. A large data set: The mushroom data set had 5000 instances compared to 150 in the iris data set.
2. Larger number of attributes: The mushroom data set had 22 attributes whereas the iris data set has 4.

3.3.3.2. Experiment 2: Accuracy for All Cross Validation Methods

Both the data sets were run through the classifier using all three cross validation methods and the accuracies were calculated. The same number of instances as in Experiment 1 were used for this experiment as well.

Table 4: Results of Experiment 2 for Both Data Sets

	Holdout	**K-Fold**	**Leave-One-Out**
Mushroom Data Set	99.8%	100%	100%
Iris Data Set	92%	89.33%	90%

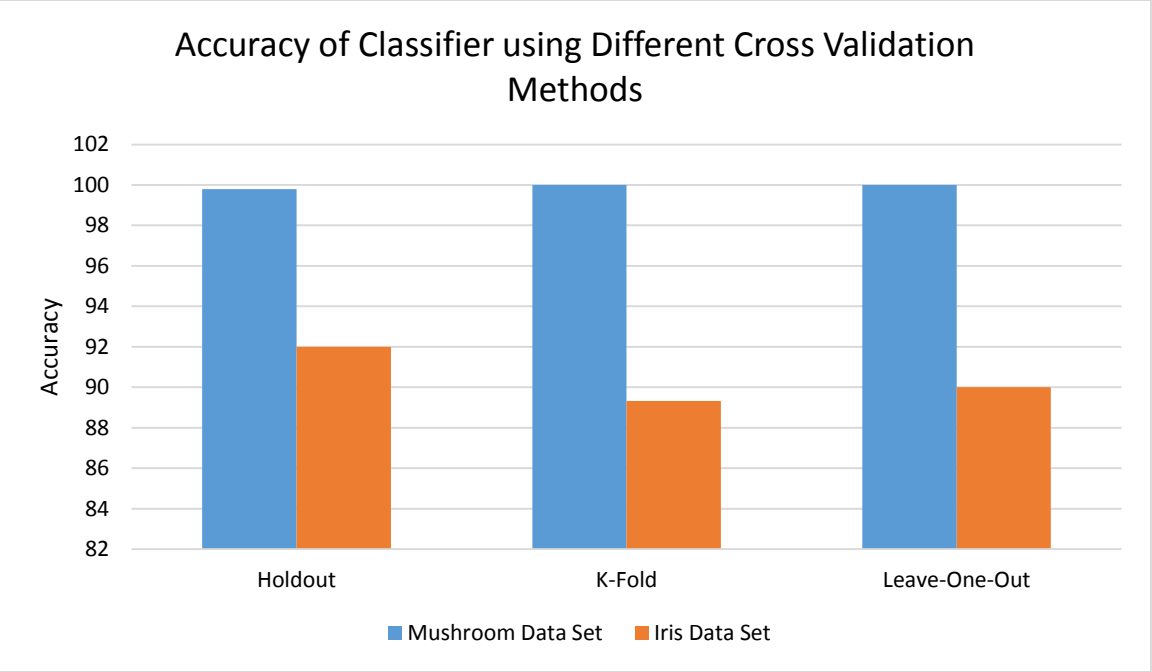

Figure 14: Accuracy of Classifier Using Different Cross Validation Methods

Holdout Cross Validation: The mushroom data set had 2500 instances in the training set and 2500 instances in the test set. The accuracy obtained was 99.8%. The iris data set had 75 instances in the training set and 75 instances in the test set. The accuracy obtained was 92%.

K-Fold Cross Validation: This is the same as Experiment 1 where the accuracy of the mushroom data set was 100% and that of the iris data set was 89.33%. For the mushroom data set, each run had 4900 data instances in the training set and 100 instances in the test set. The iris data set had 147 instances in the training set and 3 instances in the training set for each of the runs.

Leave-One-Out Cross Validation: In each run, the mushroom data set had 4900 instances in the training set and 1 instance in the test set. The accuracy obtained was 100%.

The iris data set had 149 instances in the training set and 1 instance in the test set in each run. The accuracy of the classifier was 90%.

3.3.3.2.1. Results

1. Mushroom data set: The accuracies obtained for this data set using all three cross validation methods were all very good. The lowest was 99.8% using holdout method. This shows that a larger training set can give higher accuracy.
2. Iris data set: The accuracies obtained using the three cross validation methods were around the 90% mark. The highest accuracy of 92% was obtained from the holdout method. This is contrary to other results where larger training sets generated higher accuracies. This behavior could have been caused by the distribution of classes in the training set and the test set.
3. From the results obtained from the two data sets it can be observed that this naïve Bayes classifier has better accuracy with larger data sets and larger number of attributes. The accuracies on the mushroom data set averages close to 100% and that of the iris data set averages around 90%.

3.4. Comparison with Other Algorithms

The two data sets, mushroom data set and iris data set, used to test the classifier in this paper have been used in several other papers to test different classification algorithms. In this section we will compare our results with results from some of the other papers to compare the accuracy obtained.

The accuracy of the developed naïve Bayes classifier was compared with the accuracies from the following algorithms: J48 unpruned tree algorithm [17], CBA [18], C4.5 [19], DeEP [20], and multi layer feed forward neural network [21].

Table 5: Description of Algorithms Used in Comparison

Algorithm	Description
C4.5	An algorithm based on the ID3 algorithm that tries to find simple decision trees [19]. The algorithm is based on certain premises described below: • The tree is a leaf labelled with a class if all cases are that same class • For each attribute, the potential information and the gain are calculated from a test on the attribute. • Find the best attribute to branch on depending on the current selection.
J48 unpruned tree algorithm	A version of the C4.5 algorithm employing two pruning methods. • Subtree replacement – nodes in a decision tree may be replaced by a leaf reducing the number of tests along a certain path • Subtree raising – a node may be moved upwards towards the root of a decision tree replacing other nodes along the way
Classification Based on Associations (CBA)	An integration of classification rule mining and association rule mining. It consists of two parts: rule generator for finding association rules and the classifier builder.
Decisions through Emerging Patterns (DeEP)	An instance based classifier which makes decisions through emerging patters defined as itemsets whose frequencies change significantly from one class to another. [20]
Multi layer feed forward neural network	Feed forward neural network consisting of intermediary layers called hidden layers that perform intermediary computations before directing the input to the output layer. [21]

The results of the comparisons are summarized in Table 6. For each of the algorithms the tables lists the maximum accuracies obtained from experiments.

Table 6: Comparison of Accuracies from Different Algorithms

	C4.5	J48	CBA	DeEP	NN	NB
Mushroom Data set	-	100%	-	100%	-	100%
Iris Data Set	95.3%	-	92.9%	96%	96.66%	92%

From Table 6 it can be seen that for the mushroom data set the accuracies from different algorithms are all 100%. The iris data set showed satisfactory accuracy with all algorithms. It can be seen that the accuracies from C4.5, DeEP, and the multi layer feed forward neural network are higher than the accuracy from the naïve Bayes classifier developed in this paper.

Table 6 also shows that the accuracy of the developed naïve Bayes classifier is very comparable to other sophisticated algorithms described in Table 5.

4. CONCLUSION

In this paper, we designed and developed a naïve Bayes classifier that was generalized to read any data set with categorical data and a prescribed structure in the input excel file. The classifier was tested using two different data sets from the UCI machine learning repository. Two experiments were carried out using different cross validation methods to calculate the accuracy of the classifier. The results were used to make some observations about the classifier.

For the mushroom data set, with 5000 data instances and 22 attributes, the accuracy obtained was close to 100% for all experiments performed. The iris data set, which has 150 data instances and 4 attributes, had accuracies from all experiments closer to 90%. From the results obtained it was observed that the classifier performed better on the mushroom data set, which is a larger data set and has a larger number of attributes. The attributes in both the data sets were independent of each other. Since the accuracies for both data sets were above 90%, we can say that the naïve Bayes classifier has satisfactory results even with the independence assumption.

From the comparisons in section 3.4, it can be seen that the accuracy of the mushroom data set is good for most algorithms. On the other hand, the iris data set had better accuracy from almost all other algorithms than the naïve Bayes classifier. The lower accuracy could be because of the fact that the values of the features in the iris data set were converted to integers to work with the built classifier.

It can be observed from this paper, that the results of the naïve Bayes classifier closely competes with that of other sophisticated classifiers.

5. REFERENCES

[1] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification," 2007.

[2] P. Langley, W. Iba and K. Thompson, "An analysis of Bayesian Classifiers.," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.

[3] S. M. Kamruzzaman, "Text Classification using Artificial Intelligence," *Journal of Electrical Engineering*, Vols. EE 33, No. I & II, December 2006.

[4] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers.," *Machine Learning*, vol. 29, pp. 131-163, 1997.

[5] I. Rish, "An empirical study of the naive Bayes classifier," *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 22, pp. 41-46, 2001.

[6] D. Grossman and P. Domingos, "Learning Bayesian network classifiers by maximizing conditional likelihood," in *Proceedings of the twenty-first international conference on Machine learning*, Banff, Cananda, 2004.

[7] T. O. Ayodele, "Types of Machine Learning Algorithms," in *New Advances in Machine Learning*, Y. Zhang, Ed., InTech, 2010.

[8] F. Woergoetter and B. Porr, "Reinforcement learning," *Scholarpedia*, vol. 3, p. 1448, 2008.

[9] S. C. Larson, "The shrinkage of the coefficient of multiple correlation.," *J. Edic. Psychol.*, pp. 45-55, 1931.

[10] F. Mosteller and J. W. Tukey, "Data analysis, including statistics.," in *Handbook of Social Psychology, Vol.*, Addison-Wesley, 1968.

[11] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," in *Statistics Surveys*, vol. 4, 2010, pp. 40-79.

[12] P. Rafaeilzadeh, L. Tang and H. Liu, "Cross Validation," *Encyclopedia of Database Systems*, 2009.

[13] R. O. Duda and P. E. Hart, Pattern classification and scene analysis, John Wiley and Sons, 1973.

[14] "EPPlus - Create Advanced Excel 2007 Spreadsheets on the Server," [Online]. Available: http://epplus.codeplex.com/.

[15] "UCI Machine Learning Repository: Mushroom dataset," [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Mushroom. [Accessed 13 March 2013].

[16] "UCI Machine Learning Repository: Iris Data Set," [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris. [Accessed 16 April 2013].

[17] C. Eusebi, C. Gliga, D. John and A. Maisonave, "Data Mining on a Mushroom Database," in *Proceedings of Student-Faculty Research Day, CSIS, Pace University*, 2008.

[18] B. Liu, W. Hsu and Y. Ma, "Integrating classification and association rule mining," in *Proceedings of the Fourth International Conference on Knowledge Discovery in Databases and Data Mining*, New York, USA, 1998.

[19] J. R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann Publishers Inc., 1993.

[20] J. Li, G. Dong and K. Ramamohanarao, "Instance-Based Classification by Emerging Patterns," in *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2000.

[21] M. Swain, S. K. Dash, S. Dash and A. Mohapatra, "An Approach for Iris Plant Classification using Neural Network," *International Journal on Soft Computing*, vol. 3, no. 1, February 2012.