COMPUTER PROGRAM CATEGORIZATION WITH MACHINE LEARNING

MD MAHMUDUL HASAN RAFEE
Bachelor of Computer Science and Information Technology , Islamic University of Technology, 2011

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

COMPUTER PROGRAM CATEGORIZATION WITH MACHINE LEARNING

MD MAHMUDUL HASAN RAFEE

Date of Defence: August 2, 2017

Dedication

To my beloved parents and dearest wife

Abstract

Machine learning techniques have been applied to improve the learning process and to learn about the utilization of natural languages. Previous research has shown that similar techniques can be applied in the analysis of computer programming (artificial) languages. Several studies have demonstrated the influence of sociolinguistic characteristics such as age, gender, region, and social status in natural languages. This research focuses on determining the impact of sociolinguistic characteristics of the author, particularly gender and region on computer programs. We use machine learning and statistical techniques to find out the similarities and dissimilarities in the use of programming language based on the gender and region of the programmer. The results of various experiments are promising. We demonstrate that we can predict the gender of programmers with 83.1% accuracy and the region of the programmer with 92.5% accuracy.

Acknowledgments

All praise be to the Almighty for giving me the opportunity, strength and determination to complete this research. I would like to express my earnest gratitude to my supervisor and mentor Dr. Jacqueline E. Rice for her parent-like guidance, continuous support, and encouragement throughout my M.Sc. program. Without her incredible supervision, inspiring motivation, and insightful comments this research endeavor would hardly have been completed. Words are powerless to express my gratitude to her for what she has done for me. I consider myself very lucky indeed to have her as my supervisor.

I would also like to thank my committee members Dr. Kevin Grant and Dr. Fangfang Li for their time, constructive comments, and valuable suggestions for improving the quality of my thesis.

I am also thankful to Alberta Innovates -Technology Futures (AITF) for funding this research.

My appreciation also extends to all the members of my research group and my fellow graduate students whose cooperation has made this path easier for me. Last but not the least, I would like to thank my parents, sisters, and my wife for supporting me spiritually throughout this research and in my life in general. Especially, I would like to thank my wife, who was my inspiration and was by my side through all the difficult times of my life. I feel truly blessed to have her in my life. Thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's world, if we look at the types of language being used, we can classify them into two broad categories: the first is natural language and the second is artificial language. Natural language refers to an inter-human language such as English, French, Arabic, and Bengali. Humans use natural language to communicate with each other. In contrast, artificial language, which is also known as a programming language, is used by humans to communicate with computers. The use of natural language among humans differs depending on the region, society, religion, age, gender, and social or economic status [18]. These factors may also affect the users of the artificial language or programming language. Natural languages developed over thousands of years. However, artificial languages are synthesized by logicians and linguists to meet some specific design criteria. The most basic characteristic of the distinction is that an artificial language can be fully circumscribed and studied in its entirety [14].

A programming language is a formal language which is used to give instruction to computers to perform specific tasks [26]. Generally, a programming language is formed of two components: syntax and semantics. Syntax describes the format or structure of the programming elements, and semantics describes the meaning of the syntax of an element. Every programming language follows specific syntax and semantics quite strictly and that is why there are limited options left to the user of a programming language to express their thoughts while writing a program. Also the use of programming elements depends on the specific problem. However, programmers have some space to reflect their programming identity in the layout and structure of a program, such as the number of blank lines, number of comment lines, and number of mixed lines in a program.

One area where machine learning techniques have been applied is to improve the automated learning process and to learn about the utilization of natural languages. Previous research [23, 4, 17] has shown that similar techniques can be applied in the analysis of computer programming (artificial) languages. Several studies [23, 3, 4, 16] have demonstrated the influence of sociolinguistic characteristics such as age, gender, and social status in natural languages. The research we are proposing focuses on determining the impact of sociolinguistic characteristics of the author, particularly gender and region, on computer programs.

For decades linguistics researchers have sought to identify and interpret possible differences in linguistic styles between males and females [4]. As computer programs are also developed by male and female programmers, in this research we explore the possibility of identifying the gender of the programmer based on their programming style. Additionally, programming language is also used worldwide, and that is why we also explore the possibility of identifying the region of the programmer based on their programming style. Identification of the differences in programming language use in different groups of programmers could help us to understand possible shortcomings or advantages of their programming language knowledge. This will help to improve the teaching practice among the different groups of programmers. Additionally, if a common pattern is found among a group of programmers, this could help to design and develop specific Integrated Development Environments (IDEs) for specific groups of programmers.

In automated text categorization problems, machine learning techniques are widely used [3, 15]. In this study we also used machine learning techniques to investigate the gender and region difference in the use of programming among the programmers. We propose to use several popular machine learning techniques including Bayes net, Simple logistic, K star, bagging, classification via regression, DTNB, and random forest. To implement these machine learning techniques, we selected WEKA [13], which is an open-source machine learning tool and widely used in automatic text categorization problems [15, 6]. WEKA supports various functions of machine learning such as data processing, feature extraction, supervised and unsupervised learning, and model evaluation of different kinds of data. We also used Microsoft Excel to carry out statistical analysis on the variations of features among the different groups of programmers.

1.1 Motivation and Hypothesis

Misek-Falkoff [21] suggested that techniques from linguistics can be used to analyze a computer program. Naz [23] implemented machine learning techniques to classify the C++ programs based on the gender of the programmer. She used vocabularies of programming language (e.g., keywords, operators, loops, and comments) as features to categorize the programs based on the gender of the programmer. However, the use of these features or programming elements often depends on the problem to be solved rather than the choices of the programmers. That is why to find out the difference between male written and female written programs, we emphasize the layout and structure of the program rather than the use of different elements of the program. The layout or the structure of a program mainly depends on the programmer. For example, the use of comments in a program fully depends on the programmer. Also the use of commenting style usually varies from programmer to programmer. A programmer can decide where to write the comments, how to write the comments, and how many times the comments are needed in a program. That is why in this research we try to find out more significant features which will classify the programs based on the gender of the programmer more accurately. We also try to improve the performance of the learning models so that models can perform better to predict the gender of a programmer on future datasets.

William Labov [18] suggested that social variables such as age, gender, region, and ethnicity may influence an individual's linguistic expression [23]. Some researchers have sought to assess the impact of gender on natural languages [3, 16]. Naz [23] researched the classification of programs according to the gender of the programmer. She showed that gender does appear to affect the use of programming language. In accordance with this, our hypothesis is that an additional social variable, the region that the programmer originates from, may also correlate with variability in the use of programming language.

For this research, we investigate only whether we can identify the effect of gender and region in the use of computer language or programming language. In the future, we could include other social variables such as age and level of experience of the programmer.

1.2 Contributions

In this research we use machine learning to identify sociolinguistic characteristics of the authors of computer programs. Machine learning has been shown to be effective in the analysis of natural language, and our research focuses on extending this to the analysis of programming languages. Short-term outcomes include improving assessing whether we can identify variations in the use of programming languages associated with the programmers' gender and region. Thus, research affirms that, coding is a deliberate action across cultural and technological fields [8]. All languages, even programming languages, allow for choices that could reflect the programmer's social context. While the syntax of a computer program is quite strictly determined by the programming language, choices left up to the programmer include the use of different numbers and types of loops, data types, keywords, operators, and comments [30]. Based on this, the proposed research will use machine learning to analyze and identify individuals' coding styles and their associations with sociolinguistic variables. Three areas can potentially benefit from this work:

- It may be possible to enhance the design of compiler and to increase the efficiency of Integrated Development Environments (IDEs) so that particular styles or preferences of different sociolinguistic groups are supported; communication between different groups preferring different styles could also be facilitated.

- This work may aid software companies in the design of development teams by providing information about thinking and coding styles of different groups of programmers.
- Outcomes of our sociolinguistics analysis could help to enhance programming language teaching methods by providing knowledge about individuals' preferred use of programming languages.

1.3 Organization of Thesis

In Chapter 2 we provide an overview of sociolinguistics, machine learning, WEKA, and classification algorithms based on the related research on these fields.

Chapter 3 describes the data collection procedure and the methodology that we used to classify male and female written programs and also Canadian and Bangladeshi written programs. We explain all the steps that we implement in our methodology. We also describe all four experiment with their results. Threats to validity and programming environment are also discussed in this chapter.

Chapter 4 provides some analysis on features to identify gender and region based variations in the programs of our dataset.

In Chapter 5 we conclude this research with the discussion of possible research directions.

Chapter 2

Background and Literature Review

2.1 Related Work

Factors such as clothing, mannerisms, speech, and writing represent the social identity of a person in a society. Every member of a society tends to follow certain conventions in terms of socio-characteristics such as age, gender, and socio-economic status [23]. The study of social factors which influence or are influenced by the use of a language or linguistic variations within a society is known as sociolinguistics [18, 25]. An example of this is when an English-speaker's birthplace can be recognized simply by the individual's accent and use of various words. Similarly, an individual's accent and use of various words can determine a person's place in a society.

Many researchers have explored socio-linguistic factors of written documents in different natural languages [4,5,6]. For example, Argamon et al. [4] examined gender differences in English literature in the British National Corpus (BNC). The BNC consists of fiction and nonfiction writings from articles and books. The dataset contains 604 text documents, and each document on an average consists of more than 2,000 words. In [4], machine learning and statistical techniques were employed to find out the differences based on the author's writing style. The machine learning techniques were used to identify relevant features from the dataset. Lexical and syntactic features were analyzed according to the gender of authors and from that, more than 1000 features were selected to investigate gender difference in the textual documents. In order to select a small list of the most relevant features from those 1000 features, Argamon et al. [4] used a machine learning method named the EG algorithm

[20]. Using the EG algorithm, only 50 features were selected from the list of 1000 features. These 50 features had an impact on the differentiation between male written and female written textual documents. Statistical techniques were also used to find out the significance of the features. The student's t-test and Mann-Whitney U test were used to identify differences in the use of features in male written and female written documents. Machine learning and computational intelligence have also been used in the analysis of various data, ranging from text documents [15] to biological data such as genes or proteins [28].

Argamon et al. were able to then extend their work to an analysis of French literature [3]. They examined gender differences in 600 French literary and historic textual documents. In this case, Argamon et al. [3] used a machine learning algorithm named SVM (Support Vector Machine) to distinguish between male and female written documents. Using the support vectors from the SVM model, relevant features were identified based on the word distribution, usage, and their frequencies. Using those features, the SVM model was able to predict the gender of the authors accurately by 90%. The results from [4] and [3] were similar.

Authorship analysis or analysis of writing differences has also been carried out in the field of computer programs. Krusl and Sappford [17] explored classification of programmers style in order to find characteristics of coding style that might identify the author of a program. They used 88 C programs collected from various problem domains as their corpus or dataset. Machine learning methods implemented using LNKnet software were used for the authorship analysis. Nearest neighbor and neural network algorithms were applied for supervised learning and the k-means clustering algorithm was used for unsupervised learning. In order to evaluate the learning model, they used the n-fold cross-validation method. Indentation of C statements, use of conditional compilation, choice of while, for or do loops, and the number of lines in a function are some of the features they used in their study. The research also suggested that features such as the use of white spaces and more can be extracted by visual analysis for analyzing coding style.

Table 2.1: List of 50 Features

C++ Vocabulary	**Features**
Keywords	#include, #define, using, void, cout, cerr, cin, return, exit, int, float, char, const, double, bool, new, break, public, private
Operators	<, - >, >, &, &&, +, ++, !, !=, ==, =, -, –, *, /, |, ||, /=, +=, -=, *=, <=, >=
Comments	//*, //, /* */
Brackets	{ }, ( )
Block Execution	for, while, switch

Steven and Tahaghoghi also studied authorship attribution for computer programs [5]. In their experiment they used a corpus consisting of 1640 C programs written by 100 programmers. They used statistical analysis to determine the author of a program. Their research had a 67% success rate in terms of finding out the genuine author of a program.

University of Lethbridge M.Sc graduate Fariha Naz [23] investigated gender differences in programming language use. She used machine learning methods to categorize C++ programs according to the gender of the author/programmer. In her work she used open source implementations of machine learning algorithms: nearest neighbors (K*), decision tree (J48), and Bayes classifier (Nave Bayes).The dataset of her experiments consisted of 100 C++ programs. Among those 100 C++ programs, 50 programs were written by male programmers and 50 programs were written by female programmers. She treated each C++ program, as a text document. In order create the numerical representation of each C++ program, she used a list of features described in [6] and also showed in Table 2.1. The feature list included operators, keywords, loops, and comments. She used 50 features to convert the collected C++ programs into numeric form. She also applied term frequency and inverse document frequency (tf-idf) [31] technique to create numerical representations of original dataset. The tf-idf vectors represented the occurrences of frequencies of the features within each C++ program.

The machine learning tool WEKA [13] was used to construct three supervised learning models: K*(nearest neighbor), J48(decision tree), and Nave Bayes. These models were used to categorize the collected C++ programs based on the gender of the programmer (male/female). Supervised learning was used to train the three models. Cross validation and hold out techniques were used to evaluate the performance of classification models.

Among the three classification models the K*(nearest neighbor) model performed best with an accuracy of 72%. This model was able to correctly classify 72% of the programs according to the gender of the programmer. The Nave Bayes model achieved 66% accuracy, and the J48 (decision tree) model achieved 63% accuracy. The Kstar model also achieved the highest f-measure of 71.9%. Nave Bayes and Kstar achieved 66% and 63% f-measures respectively. The Kstar model was able to classify the highest number of female written programs accurately. Out of 50 male written programs, the Kstar model classified 39 female written programs accurately. The Nave Bayes model classified the highest number of male written programs correctly. 33 out of 50 male written programs were classified correctly by the Nave Bayes model.

Naz [23] also performed another experiment where she reduced the feature set. She used the same three classification model and evaluation techniques in this experiment. However, the number of features were reduced from fifty to seven. In this experiment, the Kstar (nearest neighbor) model again performed best with an accuracy of 71%. This model was able to correctly classify 71% of the programs according to the gender of the programmer. The J48 (decision tree) model performed better than the previous experiment with 50 features. J48 achieved 70% accuracy, whereas Naive Bayes achieved 61%. However, the Naive Bayes model was able to classify the highest number of male written programs accurately. Out of 50 male written programs, the Naive Bayes model classified 39 male written programs accurately. The Kstar (nearest neighbor) model classified the highest number of female written programs correctly. 36 out of 50 female written programs were classified correctly by the Kstar model.

2.2 Machine Learning

Machine learning is making its mark and is beginning to play a key role in a variety of critical and complex applications such as natural language processing, data mining, expert systems, pattern recognition, and image recognition. Machine learning has created many opportunities for researchers in these fields and this is just the beginning. According to [32], machine learning is likely to be the backbone of future development.

Machine learning is a field of artificial intelligence which has emerged from the fields of pattern recognition and computational learning theory. In 1959, Arthur Samuel defined machine learning as a *Field of study that gives computers the ability to learn without being explicitly programmed* [32]. Machine learning allows computers to learn and make predictions on data rather than strictly following the instructions of a static program. Tom Mitchell in 1997 gave a constructive definition of machine learning. He said, *A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E . The goal of machine learning is never to make perfect guesses, because machine learning deals in domains where there is no such thing. The goal is to make guesses that are good enough to be useful* [22].

2.2.1 Data

Data is the center point of machine learning. All learning and analysis in machine learning is based on data. Machine learning methods learn from training data, also known as examples. It is very important to have an appropriate amount and quality of data to train a machine learning model to predict accurately with unseen data. When we think about data, generally we think about databases consisting of rows and columns. In machine learning, the data is generally also organized like this. We use standard terms to refer to different aspects of the data:

i. Instance - A tuple or a row of a data table is called an instance. An instance contains features or attributes.

ii. Feature - A column of a data table is known as a feature. A feature is also known as an attribute. An attribute or feature is a property or characteristic of an instance (e.g., height, weight, blood type, body temperature)

iii. Data Type - Features or attributes can have various data types such as integers, strings, floats, dates, times, or several complex data types associated with learning methods.

iv. Datasets - Sets of multiple instances of a data table are called datasets. Datasets are important for training and testing purposes in machine learning. In the training period, the learning model will learn the characteristics of the input dataset. In the testing period, the learning model will use those characteristics to predict class labels on unseen datasets.

v. Training Datasets - A training dataset is a collection or set of instances that is used as an input to a machine learning algorithm to train a model.

vi. Testing Dataset - A collection or set of instances which is used to measure the accuracy of a machine learning model is known as a testing dataset. Since the testing dataset is already known to the learning model, the use of this testing dataset will increase the accuracy rate or performance of the learning model. In order to measure the performance of the learning model accurately, testing datasets are not used alongside the training datasets.

2.2.2 Data Preparation Process

In order to achieve reliable predictions from a machine learning model, it is very important to handle and organize the data precisely. There are three steps involved in preparing a dataset for use by a machine learning algorithm. First, we must select the data and then we must process and transform the data [33].

i. Selecting Data - Data which are relevant to the problem chosen should be selected. Sometimes it is good to collect all the relevant data that are available for the problem domain because it helps to train the model well. We can miss important features if we filter out some data without testing by the learning model.

ii. Processing Data - After data selection, the next step should focus on the utilization of the collected data by the learning algorithm. Data processing will help to create a framework for the collected data in order for the algorithm to work efficiently on the data. There are three steps for processing data:

(a) Formatting - The raw data which has been collected from different sources may not be in a suitable format for use by the machine learning algorithm. The data may need to be formatted according to the needs of the machine learning algorithm.

(b) Cleaning - There may be some incomplete or missing data in the raw data. A process may be needed to fix or remove the missing data in order to make the dataset consistent and useful for the problem being solved.

(c) Sampling - There may be more data available than is needed. A very large amount of data will increase the running time, computational complexity, and memory requirements of the learning algorithm. To solve this issue, it is better to take a small dataset that will represent the collected dataset. This is called sampling.

2.2.3 Data Transformation

The last step is transforming the data. The data must be transformed according to the needs of the learning algorithm and the knowledge of the problem domain. Scaling, feature aggregation, and feature decomposition are most common data transformation techniques. Scaling is done to bring all the features into the same scale for different quantities in the preprocessed data. For example: *The preprocessed data may contain attributes with mixtures of scales for various quantities such as dollars, kilograms and sales volume. Many machine learning methods like data attributes to have the same scale such as between 0 and 1 for the smallest and largest value for a given feature* [33]. There may be some features in the preprocessed data which will be more useful and productive for the learning method if they are aggregated into a single attribute or feature. This method is called feature aggregation. Feature decomposition consists of dividing the complex features in the preprocessed data into multiple features to make the learning process easier and more efficient.

2.2.4 Categories of Machine Learning

Depending on the nature of learning, machine learning tasks can be classified into two broad categories. They are [12]:

(1) Supervised Learning - In supervised learning a dataset is given to the learning algorithm where the dataset consists of sample inputs and their class labels. In supervised learning data instances are labeled with its class label. The goal is to learn the rule which will match the inputs to outputs. In supervised learning, the learning algorithm produces an inferred function by analyzing the training data. This inferred function is used to map the new data to its corresponding class labels. Supervised learning is generally used for generalizing a learning model. In supervised learning predictions are compared by the model with the known outputs and modification is done if there is any error. The most common example of supervised learning is classification. In this work we are using supervised learning.

(2) Unsupervised Learning - In unsupervised learning there are no labels on the input data. It is the job of the learning algorithm to determine any structure in its input data. The main goal of unsupervised learning is to discover either hidden patterns in the data, or a way to predict on data which is known as feature learning. Unsupervised learning is generally used for generalizing the structure of the learning model that does not have any predefined class labels along with the data. Unsupervised learning helps the model to naturally identify and explore the characteristics of data. The most common example of unsupervised learning is clustering.

2.2.5 Components of Text Categorization

In machine learning to solve text categorization problems there are four components to follow [5]:

- Document Representation - The first step is to choose a large set of text features which might be useful for categorizing a given text (typically words that are neither too common nor too rare) and represent each text as a vector consisting of values representing the frequency of each feature in the text.
- Dimension Reduction - The second step is to use various criteria for reducing the dimension of the vectors - typically by eliminating features which don't seem to be correlated with any category.
- Learning Method - Next, we need to use some machine learning method to construct one or more models of each category
- Testing Protocol - Lastly, we need to use some testing protocol to estimate the reliability of the system.

2.3 Model Evaluation Techniques

In order to build a best possible learning model for a specific problem we need to evaluate the learning algorithm. Various techniques to evaluate machine learning algorithms are discussed below [33]:

2.3.1 Test Harness

It is important to define a test harness before starting to use a machine learning algorithm for training a model. The data which will be used both for training and testing the learning algorithm is called a test harness. A test harness is used for analyzing the performance of the machine learning algorithm. A test harness allows the testing of the algorithm to perform quickly and regularly against a standard result of the problem domain. From the list of estimated algorithms for a problem, the test harness will assist the model to select the best algorithm for learning. The test harness also indicates how predictable the problem is. If the result from different learning algorithms is not satisfactory, then this would suggest that there is a lack of data to learn the structure of the problem, or the structure of the learning algorithm needs to be transformed in order to build the model.

2.3.2 Cross Validation

Cross validation is a widely used approach for testing learning algorithms. The first step of the cross validation technique is to create a number of folds. Folds are divisions of the entire data set into groups of instances of equal size. After that the learning model is trained using all folds except the one which will be used later on to test the model.

Step 1: Training by folds 1+2+3 testing by fold 4

Step 2: Training by folds 1+2+4 testing by fold 3

Step 3: Training by folds 1+3+4 testing by fold 2

Step 4: Training by folds 2+3+4 testing by fold 1

This is an iterative process and it continues until each fold gets a chance to test the model. Performance is measured by averaging the results of each fold to find out the efficiency of the algorithm. The number of folds generally depends on the problem and size of the dataset. For example, a 4-fold cross validation will consist of 4 iterations of training and 4 iterations of testing a learning model:

2.3.3 Evaluation Metrics

Performance measurement of a machine learning model is necessary in order to assess the solution. The performance measure should give an indication of the correctness or error in the prediction of the trained model. The classifier evaluation metrics include accuracy, precision, recall, and f-measure [13, 12].

A confusion matrix is a useful tool for analyzing the performance of a model. This indicates how well a classifier can recognize samples of different classes. To explain the working principle of a confusion matrix, a table of size "n x n" is shown in Table 2.2. n

Table 2.2: Confusion Matrix

Gender	Male	Female	Total
Male	TP	FN	P
Female	FP	TN	N
Total	P'	N'	P+N

represents the number of classes (male and female). In this confusion matrix, samples from male programmers are considered as positive samples and samples from female programmers are considered as negative samples. If we look at the Table 2.2 we find these terms: TP, FP, TN, FN, P, W, P' and N'. These terms are briefly described below:

- True Positives (TP) - TP refer to the positive samples that are predicted correctly by the model.
- True Negatives (TN) - TN refer to the negative samples that are correctly predicted by the model.
- False Positives (FP) - FP refer to the negative samples that are predicted incorrectly as positive samples.
- False Negatives (FN) - FN refer to the positive samples that were predicted incorrectly as negative samples.

P is the number of positive samples and N is the number of negative samples. P' is the number of samples that are predicted as positive samples (P' = TP+FP) and N' is the number of samples that are predicted as negative samples (N'=TN+FN). By using a confusion matrix, evaluation matrices such as accuracy, precision, recall, and f-measure can be calculated as follows:

- Accuracy - The accuracy rate of a classifier reflects how well the classifier recognizes samples of different classes. The accuracy of a classifier is the percentage of samples that are classified correctly by the classifier. The formula for calculating the accuracy of a model is as follows [13, 12]:

$$Accuracy = \frac{TP + TN}{P + N}$$

- Precision - Precision is a measure of exactness [12]. Precision measures the percentage of positive samples which are classified as a specific class and actually belong to that class. The formula for calculating the precision of a model is as follows [13, 12]:

$$Precision = \frac{TP}{TP + FP}$$

- Recall - Recall is the "measure of completeness" [13, 12]. Recall measures the percentage of positive data samples which are classified correctly as a specific class by the model. The formula for calculating the recall of a model is as follows [13, 12]:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

- F-measure - In classification problems occasionally there can be an inverse relationship between precision and recall [12]. So, it is possible that at times a model can achieve high precision but low recall. For this reason, f-measure is used as an alternative way to use precision and recall. F-measure is the harmonic mean of the precision and recall [12]. The formula for calculating the f-measure of a model is as follows [13, 12]:

$$F\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

2.4 WEKA

In this work we used open source machine learning software named WEKA (Waikato Environment for Knowledge Analysis) [13]. WEKA is widely used in the field of machine learning. The main reason behind using WEKA is that WEKA supports the core functions of machine learning data processing, feature selection, model building, and model testing. There are also many built-in machine learning algorithms available in WEKA.

2.4.1 WEKA Data Format

The data file with which WEKA works is known as an ARFF (Attribute-Relation-File-Format) file. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes [13]. An ARFF file consists of two distinct sections. The first section is the header and the second section is data. The header section of the ARFF file includes the name of the relation, a list of features/attributes, and the types of the attributes. In general, the first line of an ARFF file is defined as the relation name. The format to define a relation is:

@relation <relation-name>

Every section in an ARFF file starts with a @ symbol. It represents the start of a section in an ARFF file. @relation states that this is the start of the relation section. ¡relation-name¿ can be any string given by the user. The second part of the header section is the attribute/feature declaration. The format for attribute declaration is:

@attribute <attribute-name> <data-type>

Attribute declarations take the form of an ordered sequence of @attribute statements. Each attribute in the data set has its own @attribute statement which uniquely defines the name of that attribute and data type of that attribute. The order in which the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then WEKA expects that all of those attributes' values will be found in the third comma delimited column [23]. There are four data types available for the value of an attribute. These are:

1. Numeric (Integer or real numbers)
2. Nominal (List of possible values)
3. String (Textual values)

4. Date (Date format)

The last section of an ARFF file is the data section. This consists of the data declaration line and actual instance lines. The data section is denoted as: @data. This single line represents the starting of the data segment. After the data declaration line actual data instances are provided. On each single line, one data instance is represented with a carriage return denoting the end of the instance. Attribute/feature values of each instance are separated by commas. In each instance, attribute values must appear in the order in which they were declared in the @attribute section. A missing value in any instance is denoted by a single question mark ?. Values of nominal and string attributes are case sensitive. An example of an ARFF file is shown in Table 2.3.

There is another type of ARFF file which is known as Sparse ARFF. Sparse ARFF files are almost similar to ARFF files but data with zero values are not shown in the data section. A sparse ARFF file has similar relation and attribute sections but differs in the data section. The zero attributes are omitted and non-zero attributes are explicitly identified by the attribute number and their values. Suppose we have a data section as follows in an ARFF file:

```
@data
0, A, 0, B, yes
0, 0, A, 0, no
```

The representation of this data section in Sparse ARFF will be like this:

```
@data
{ 1  A, 3 B, 4 yes }
{ 2 A, 4 no }
```

In sparse ARFF, each data instance is surrounded by curly braces. The format for each data instance is : <index> <space> <value>. The attribute index starts from zero.

Table 2.3: Sample ARFF File

```
@relation weather

@attributeoutlook {sunny, overcast, rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,?,FALSE,yes
rainy,68,80,FALSE,yes
```

2.4.2 Classification Algorithms of WEKA

In WEKA there are six categories of classification algorithms available. From each category we selected at least one algorithm to build one of our classification models. Brief descriptions of the six categories of classification algorithms are given below:

- Bayesian Classifiers - Bayesian classifiers are generally statistical classifiers [12]. All the Bayesian classifiers are based on Bayes theorem. Bayes theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event [12]. In a Bayesian classifier, the role of a class is to predict the values of features for members of that class. Samples are grouped in to classes based on the common values of the features [29]. In a Bayesian classifier Bayes rule is used to predict the class given the feature values. The classifier builds a probabilistic model of the features and uses the model to predict the class of a new sample. Some examples of Bayesian classifiers in WEKA are NaiveBayes, AODE, BayesNet, and DMNBText. From the Bayesian classifiers, we used the BayesNet classifier to build one of our classification models.

- Tree Classifiers - All the classifiers in this category follow a flow chart tree like structure, where an internal node represents a test on an attribute, the outcome of the test is denoted by the branch and the leaf node denotes the class label. In tree classifiers, a “divide-and-conquer” approach is implied to the problem of learning from a set of independent instances which leads to a tree like representation [13]. Tree classifiers are most popular in machine learning because trees are expressive and easy to understand [10]. Some popular tree classifiers in WEKA are ADTree, DFTree, J48, RandomForest, RandomTree, and RepTree. From the tree classifiers, we used the RandomForest classifier to build one of our classification models.

- Rule Based Classifiers - Rule based classifiers generally make use of a set of IF-THEN rules for classification problem. A series of tests is a precondition of a rule. All the tests must succeed in order to execute the rule. The outcome of a rule execution provides the class that applies to instances covered by that rule [13]. Some examples of rule based classifiers in WEKA are DecisionTable, DTNB, JRip,OneR, Ridor, and ZEroR. From the rule based classifiers we used the DTNB classifier to build one of our classification models.

- Function Based Classifiers - Function category classifiers combine all the classifiers which can be written down as mathematical equations in a reasonably natural way [13]. Some examples of function based classifiers in WEKA are LinearRegression, SMO, LibSVM, SimpleLogistic, SMOreg, and Window. From the function based classifiers we used the SimpleLogistic classifier to build one of our classification models.

- Lazy Classifiers - All the previous classifiers, when given a set of training samples, will construct a classification model before testing new samples [12]. In contrast, lazy classifiers store the training instances and do no real work until testing new samples. When training sample is given, a lazy classifier simply stores it and waits until the model receives a test sample . When the model receives a test sample, it starts the classification process to classify the sample based on the similarity to the stored training sample. These kind of classifiers are also known as instance based learners [12]. Some examples of lazy classifiers in WEKA are IB1, IBK, Kstar, LBR, and LWL. From lazy classifiers, we used the Kstar classifier to build one of our classification models.

- Meta Classifiers - Classification algorithms under meta classifiers use or combine multiple classification algorithms into one classifier in order to make the classification decision more reliable. Some examples of meta classifiers in WEKA are: Bagging, LogitBoost, END, Dagging, Decorate, ClassificationViaRegression, and ClassifiactionViaClustering. From the tree meta we used Bagging and ClassificationViaRegression classifiers to build two of our classification models.

2.4.3 Feature Selection

In general a dataset contains a set of features or attributes. It may happen that some features are not relevant to the learning task. Leaving out the important features or keeping the irrelevant features may degrade the performance of a model. Redundant features can also slow down the learning process [13]. That is why it is important to choose a small set of features which is sufficient for learning and improvement of the quality of the concept description [9, 4]. WEKA provides some important algorithms to find out the useful features. To select useful features from our selected set of features, we used one of the attribute evaluators of WEKA named “InfoGainAttributeEval”. This is a statistical technique which evaluates features on the basis of Information Gain with respect to the class [13]. The difference between original information about the proportion of classes and new information which is obtained after the identification of the useful attribute is called information gain [23]. The following formula is used to calculate information gain [12, 33]:

Info Gain (Class, Attribute) = H (Classs) - H (Class | Attribute)

InfoGainAttributeEval extracts relevant features from the original feature set. This evaluator works with a search method in order to find out the best features. In WEKA, InfoGainAttributeEval uses the Ranker [33] algorithm as a searching algorithm. Based on the value of information gain, the ranker algorithm creates a list of features with the rank of individual features. The ranker method helps to identify the relevant and irrelevant features from the original set of features. A small set of features can be made by picking the relevant features which in turn may improve the performance of the learning model.

Chapter 3

Methodology

3.1 Data Collection

Data for this work were collected from two sources. The first source was the University of Lethbridge, Canada, and the second source was several universities in Bangladesh. Before starting the data collection procedure, it was necessary to receive the permission from the University of Lethbridge Human Subject Research Committee. We secured the approval (protocol 2012-012) from the committee after submitting all the necessary documents of the data collection procedure (described in Appendix A). We collected C++ programs as well as some information about the writer of each program. Before collecting the data from the students, we collected their permission to use their programs and information in this research. We collected 50 male written and 50 female written C++ programs from the students of Computer Science Department of the University of Lethbridge. These programs were collected as part of assignments and projects from the computer science classes. Other information about the writers of the programs was collected via a survey which was provided to each participant.

We also collected 40 male written and 20 female written programs from several uni-

Table 3.1: Data collection

Gender	**University of Lethbridge**	**Survey Monkey**	**Total Samples**
Male	50	40	90
Female	50	20	70
Total	100	60	160

versities in Bangladesh. We collected these 60 C++ programs plus authors' information through Survey Monkey which is an online survey development website [2]. In total, we collected 160 C++ programs written by both males and females. After collecting all the programs we cleaned all programs manually. This included removing all personal information such as students' names, id numbers, course information, and any other info which was not related to the programming or coding.

3.2 Document Representation

Our machine learning classification model requires inputs as a vector of numeric feature values and the model gives output as nominal values. In our classification model, feature values were numeric and class labels were nominal. There were two class labels: male and female. In our research work, we treated each collected C++ programs as a text document. In each document, we calculated the 16 selected feature values to transform the original text based dataset into a numerical dataset. Using the values of selected features, the programs were transformed into the appropriate data format for the classification model. We calculated the value of 16 features to generate the numeric feature vector of each C++ program. We represented each code by the programs feature values. Thus our dataset composed of the data instances represented by feature vectors (numeric values of 16 features of each code) and a class label (male or female). The dataset was then used to train and evaluate the learning model that we built using WEKA [13].

3.3 Feature Selection

To categorize the programs according to the gender of the author/ programmer, we selected 15 features of a code/program. These features were selected from "IEEE Standard for Software Productivity Metrics" [1]. These features are: total lines, total source code lines, source code lines percentage, total blank lines, blank lines percentage, total comments lines, comment lines percentage, total mixed lines, mixed lines percentage, total commentary words (cword), total physical executable lines of code (sloc-p), total logical executable lines of code (sloc-l), total functions, total code function lines, and average code lines in functions. We also selected region of the programmer as a feature. In total 16 features were selected for this research. Short descriptions of these features are given below.

1. Total lines - Count of lines that include number of source code lines, comment lines, and blank lines.

2. Total source code lines - All the code lines excluding total blank lines and total comment lines.

3. Source code lines percentage - Percentage of source code lines as compared to the total lines.

4. Total blank lines - Total number of empty lines.

5. Blank lines percentage - Percentage of blank lines as compared to the total lines.

6. Total comment lines - Total number of lines which are not executable by the compiler but useful for code description and other information about the code.

7. Comment lines percentage - Percentage of comment lines as compared to the total lines of code.

8. Total mixed lines - Count of lines that include both code and comments.

9. Mixed lines percentage - Percentage of mixed lines as compared to the total lines of code.

10. Total commentary words (CWORD) - Total words in all the comments of a single program.

11. Physical executable lines of code (SLOC-P) - Physical SLOC is a count of lines in the text of the program's source code excluding comment lines [24].

12. Logical executable lines of code - Logical SLOC attempts to measure the number of executable statements, but their specific definitions are tied to specific computer languages. For instance, one simple logical SLOC measure for C-like programming languages is the number of statement-terminating semicolons [27].

```
/* how many lines are here /*
for (i = 0; i < 200; i++)
{
 printf ("Thank You");
}
```

In the above code, we have 3 physical lines of code, 2 logical lines of code and 1 comment line.

13. Total functions - Total number of functions used in a single program.

14. Total function lines - Total number of lines in all the functions of a single program.

15. Average function lines - Total function lines divided by the total number of functions.

16. Region of the programmer - In our dataset, we have collected data from the students of two regions: Canada and Bangladesh. We used this region of the programmer/author as one of our features.

3.3.1 Region of the Programmer as a Feature

We selected the region of the programmer as one of the features while categorizing the programs according to the gender of the authors. The region of the programmer is not a feature of a program, but rather this is a feature of a programmer. Some initial experiments showed that inclusion of the region of the programmer in the feature list improved the performance of the machine learning models in predicting the gender of the author. In our dataset we had programs from two different regions: Canada and Bangladesh. Therefore, we performed two initial experiments that used the same 160 data samples, the same seven classification models, and the same cross validation technique to evaluate the performance of the models which we used in our main experiments (described in section 3.4). The only difference in the two experiments was the feature set. In one experiment we used all 16 features (described in section 3.3) where we included the region of the programmer in the

Table 3.2: Impact of Region of the Programmer

Classifaction Model	**Experiment 1 with 16 Features Including Region of the Programmer (Accuracy %)**	**Experiment 2 with 15 Features Excluding Region of the Programmer (Accuracy %)**
Simple Logistic	77.5	67.5
Kstar	75	71.9
Bagging	76.9	74.4
Classification Via Regression	78.1	70
DTNB	83.1	70.6
Random Forest	80.6	74.4
Bayes Net	70	70

feature set. In the other experiment we excluded the region of the programmer from the feature set and used the remaining 15 features to categorize the programs based on the gender of the authors. The results of the two experiments are shown in Table 3.2. From Table 3.2 we can see that except for the Bayes net model, all the other models performed significantly better when we used the region of the programmer as a feature. The inclusion of this feature helped the model to predict the gender of the programmer more accurately. The accuracy of the models increased from 2% to 12%, depending on the models, with the inclusion of the region of the programmer in the feature set. That is why we included this in the feature set while categorizing the programs based on the gender of the programmer. Further discussion of this inclusion of this feature is discussed in Section 4.5.

3.4 Experimental Work

In our research, the collected C++ programs were treated as text documents in order to apply several learning algorithms to categorize the programs according to the gender and the region of the programmer. We used WEKA to build classification models. These classification models used machine learning algorithms to classify the given input correctly. In our research, a vector of numeric feature values was provided as input to the classification model. Classification models provided outputs in the form of class labels. Class labels were male and female. In this research, we also tried to classify the collected C++ programs according to the region of origin of the programmer. We again used WEKA to build classification models. In this case, a vector of numeric feature values were also provided as input to the classification model. Classification models provide outputs in the form of class labels. Class labels were Canada and Bangladesh. For building the classification model in WEKA, we needed to build the appropriate data format. Using the feature values, C++ programs were transformed into the appropriate data format for WEKA.

We used supervised learning to build our classification models. In WEKA there are six categories of classification algorithm available (described in section 2.4.2). From each of the six categories we selected at least one algorithm to build the classification models.In total, we used seven machine learning algorithms:

- From the function category algorithms simple logistic algorithm was selected.
- From the lazy category nearest neighbor (kstar) algorithm was selected.
- From meta category bagging and classification via regression algorithms were selected.
- From rule category Decision table-Naive Bayes (DTNB) algorithm was selected.
- From the tree category algorithms random forest algorithm was selected.
- From the Bayes category Bayes net algorithm was selected.

These learning algorithms were used to train and test our dataset. To build each learning model for categorizing the programs according to the gender of the programmer, we used a total of 160 data instances and 16 features. All the data instances were labeled as male or female.To build each learning model for categorizing the programs according to the region of origin of the programmer, we used a total of 160 data instances and 15 features. All the data instances were labeled as Canada or Bangladesh. In this research, we performed 4 different experiments using our dataset of 160 programs. A brief description of these 4 experiments are given below:

- In the first experiment, we classified the collected C++ programs according to the gender of the author/programmer. The first experiment consisted of 16 features and used the cross validation technique to evaluate the results of the classification models. In the second experiment, we again classified the collected C++ programs according to the gender of the programmer/author. However, in this experiment, we reduced our feature set from 16 to 6 and used the cross validation technique to evaluate the results of the classification models.

- In the third experiment, the goal was to classify the collected C++ programs according to the region of the programmer/author. The third experiment consisted of 15 features and used cross validation technique to evaluate the result of the classification models. In the fourth and last experiment, we again classified the collected C++ programs according to the region of the programmer/author. However, in this experiment, we reduced our feature set from 15 to 7 and used the cross validation technique to evaluate the results of the classification models.

All four experiments used the same dataset. In experiment 1 and 2, data instances were associated with male and female class labels. In experiment 3 and 4, data instances were associated with Canada and Bangladesh class labels. To carry out the four experiments and to develop the classification models we used WEKA. Further details for each experiment are given below.

3.4.1 Parameter Settings

In this research, for all of the machine learning experiments we used WEKA. Using WEKA we tried to build some learning models and test those models with our dataset.

Besides using the seven classifiers (Bayes net, simple logistic, bagging, classification via regression, random forest, K star, and DTNB), we also used other features of WEKA. We used these features as a part of the data formatting, feature selection, and model evaluation process. Brief descriptions of the used features of WEKA are given below:

- Discretize Filter - In our experiments we used the Bayes net classifier to build one of the learning models. The Bayes classifier performs well when attribute/feature values are discretized rather than continuous [19]. So, to make the feature values discrete, we applied one of the supervised attribute filters of WEKA named as “Discretize”. The Discretize filter converts the numeric attributes to nominal [13].

- Randomized Filter - We applied one of the unsupervised instance filters of WEKA named “Randomize”. This filter was used to shuffle the order of the dataset randomly before implementing supervised learning methods. We set the seed value to the default value 42. The random number generator is reset with the seed value whenever a new set of instances is passed in [13].

- Cross validation - In order to evaluate the seven learning models, we used the cross validation technique of WEKA. In the cross validation process, we set the number of folds at 10. This means that our dataset was divided into 10 equal parts or folds. Then these 10 folds performed 10 iterations of training and 10 iterations of testing of each learning model.

- InfoGainAttributeEval - In order to reduce the feature set of our experiments, we used one of the attribute evaluators of WEKA named “InfoGainAttributeEval” (described in section 2.4.3). This attribute evaluator used the Ranker algorithm to rank all the features according to the information gain with respect to the class [13].

3.4.2 Programming Environment

In this research all experiments were run on an Acer Aspire laptop. The configuration of the laptop included Intel Core i5 processor, 6 GB of RAM, and 500 GB of hard disk. The computer was running by the Windows 7 operating system. We used WEKA 3.6.13 for the classification of the programs and Microsoft Excel 2016 for statistical analysis of the features of our research.

WEKA is widely used in the field of machine learning. WEKA is an open-source machine learning software. By using WEKA we implemented various filters to process the data, used seven classification algorithms to build the learning models and implemented an evaluation method to evaluate the performance of the models. We also used the feature selection function of WEKA to reduce the feature set. We used Microsoft Excel to perform some statistical analysis on the features of the programming language. We also tried to find out whether or not there exist significant differences between the use of features in male and female written programs and in the Canadian and Bangladeshi written programs.

3.4.3 Experiment 1

In experiment 1, our goal was to categorize the C++ programs according to the gender (male/female) of the programmer. In experiment 1, we used 160 data samples and 16 features. Here class labels were male and female. We first calculated the feature values for each of the 160 programs. Then we prepared the dataset according to the ARFF file format of WEKA using those feature values. Next, we used seven classification algorithms to build seven classification models using WEKA. The steps we followed are shown in Figure 3.1.

The algorithms used were: simple logistic, kstar, bagging, classification via regression, decision table - Naive Bayes (DTNB), random forest and Bayes net. After building the seven classification models, we applied the 10 fold cross validation technique to each classification model in order to evaluate the performance. The final step was to measure the performances of the seven models using our evaluation metrics. Results are given in

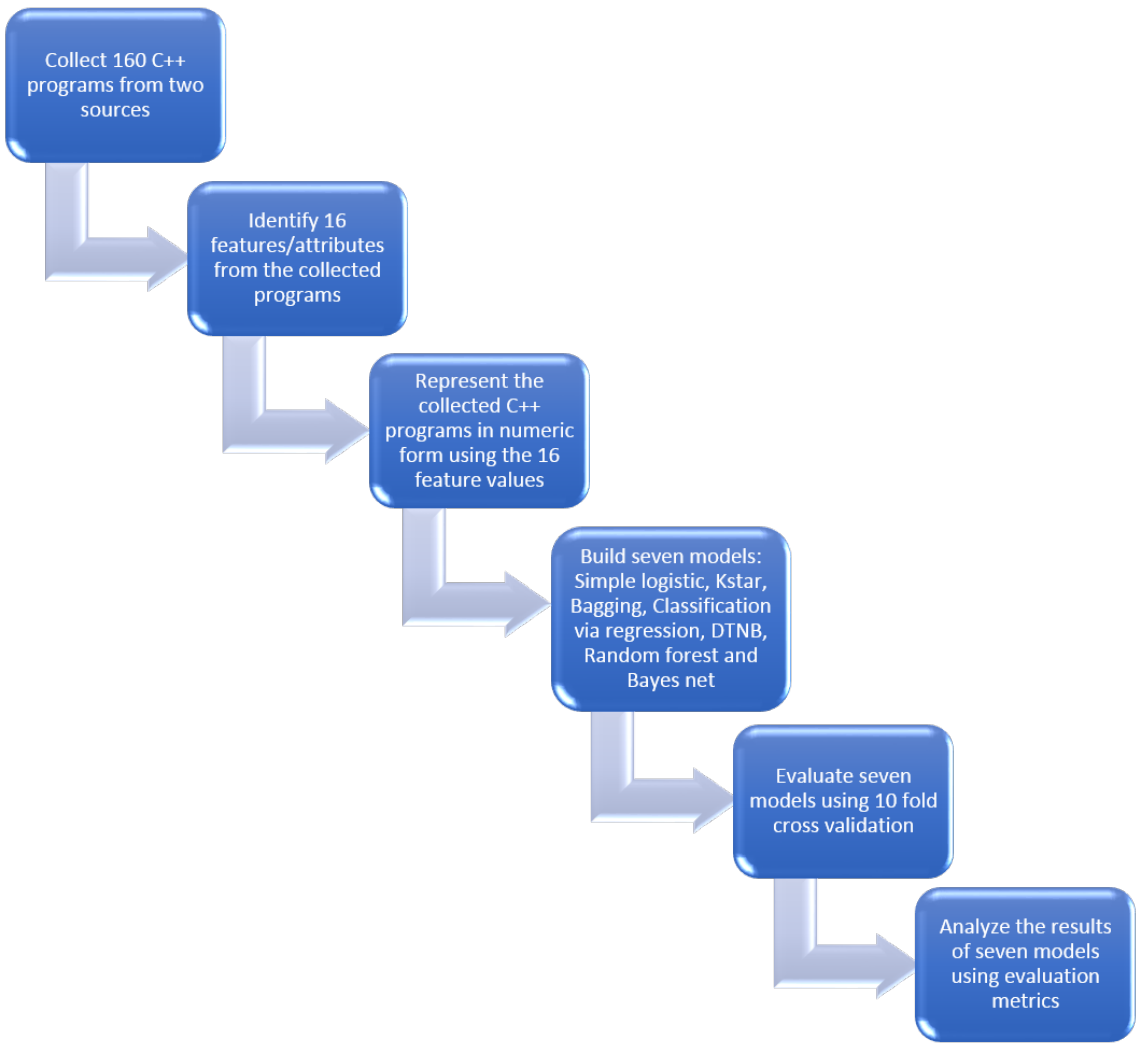

Collect 160 C++ programs from two sources
Identify 16 features/attributes from the collected programs
Represent the collected C++ programs in numeric form using the 16 feature values
Build seven models: Simple logistic, Kstar, Bagging, Classification via regression, DTNB, Random forest and Bayes net
Evaluate seven models using 10 fold cross validation
Analyze the results of seven models using evaluation metrics

Figure 3.1: Steps followed for experiment 1

section 3.5.

3.4.4 Experiment 2

In experiment 2, again our goal was to classify the 160 C++ programs according to the gender (male/female) of the programmer/author.We again used 160 data samples but here we used only 6 features to build the classification model and classify the programs. We reduced our feature set from 16 to 6 using the statistical measure Information Gain. WEKA has a built-in function named InfoGainAttributeEval for feature selection. This feature selection method used Ranker algorithm to rank all the features according to the information gain of each feature. From the ranking of each feature, we selected the top 6 ranked feature for this experiment. Those 6 features were: blank lines percentage, total blank lines, total functions, region of the programmer, source code lines percentage, and comment lines percentage. Steps of experiment 2 are shown in Figure 3.2.

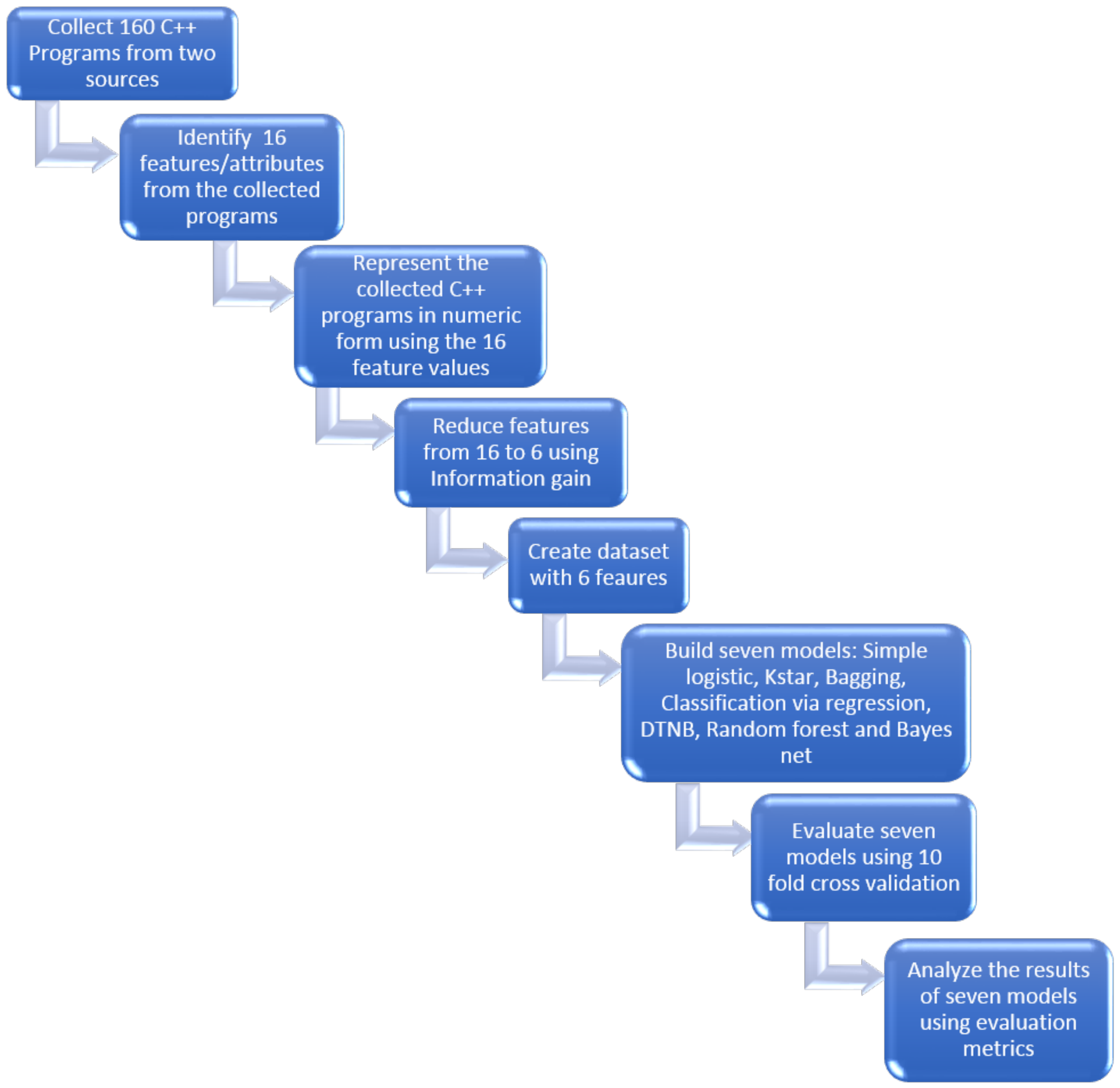

Collect 160 C++ Programs from two sources
Identify 16 features/attributes from the collected programs
Represent the collected C++ programs in numeric form using the 16 feature values
Reduce features from 16 to 6 using Information gain
Create dataset with 6 feaures
Build seven models: Simple logistic, Kstar, Bagging, Classification via regression, DTNB, Random forest and Bayes net
Evaluate seven models using 10 fold cross validation
Analyze the results of seven models using evaluation metrics

Figure 3.2: Steps followed for experiment 2

After selecting the 6 features we used the Remove filter of WEKA to remove the other features from our dataset. We then used same seven classification algorithms: simple logistic, kstar, bagging, classification via regression, decision table - Naive Bayes (DTNB), random forest and Bayes net to build our seven classification models. After building the seven classification model we again applied the 10 fold cross validation technique to each classification model in order to evaluate the performance. Lastly, we measured the performance of the models using our evaluation metrics. Results are given in section 3.5.

3.4.5 Experiment 3

We had C++ programs in our dataset written by programmers from two countries, Canada and Bangladesh. So, in experiment 3 we classified the collected C++ programs according to the region of the author (Canada/Bangladesh), another socio-linguistic characteristic. In experiment 3, we used 160 data samples and 15 features. Those 15 features were: total lines, total source code lines, source code lines percentage, total blank lines, blank lines percentage, total comment lines, comment lines percentage, total mixed lines, mixed lines percentage, total commentary words (CWORD), physical executable lines of code (SLOC-P), logical executable lines of code, total functions, total function lines, and average function line. We removed the feature region of the programmer from the feature list. Here class labels were Canada and Bangladesh. We first calculated the feature values of each of the 160 programs. Next we prepared the dataset according to the ARFF format of WEKA using the feature values. Next we used the seven classification algorithms to build classification models using WEKA. Steps of the experiment 2 are shown in Figure 3.3.

The seven classification models were built using simple logistic, kstar, bagging, classification via regression, decision table - Naive Bayes (DTNB), random forest and Bayes net classification algorithms. After building the models we again applied 10 fold cross validation to each classification model to evaluate the performance. Lastly, we measured the performance of the models using our evaluation metrics. Results are given in section 3.5.

3.4.6 Experiment 4

In experiment 4, again our goal was to classify the 160 C++ programs according to the country of the origin of the author (Canada/Bangladesh). Steps of this experiment are

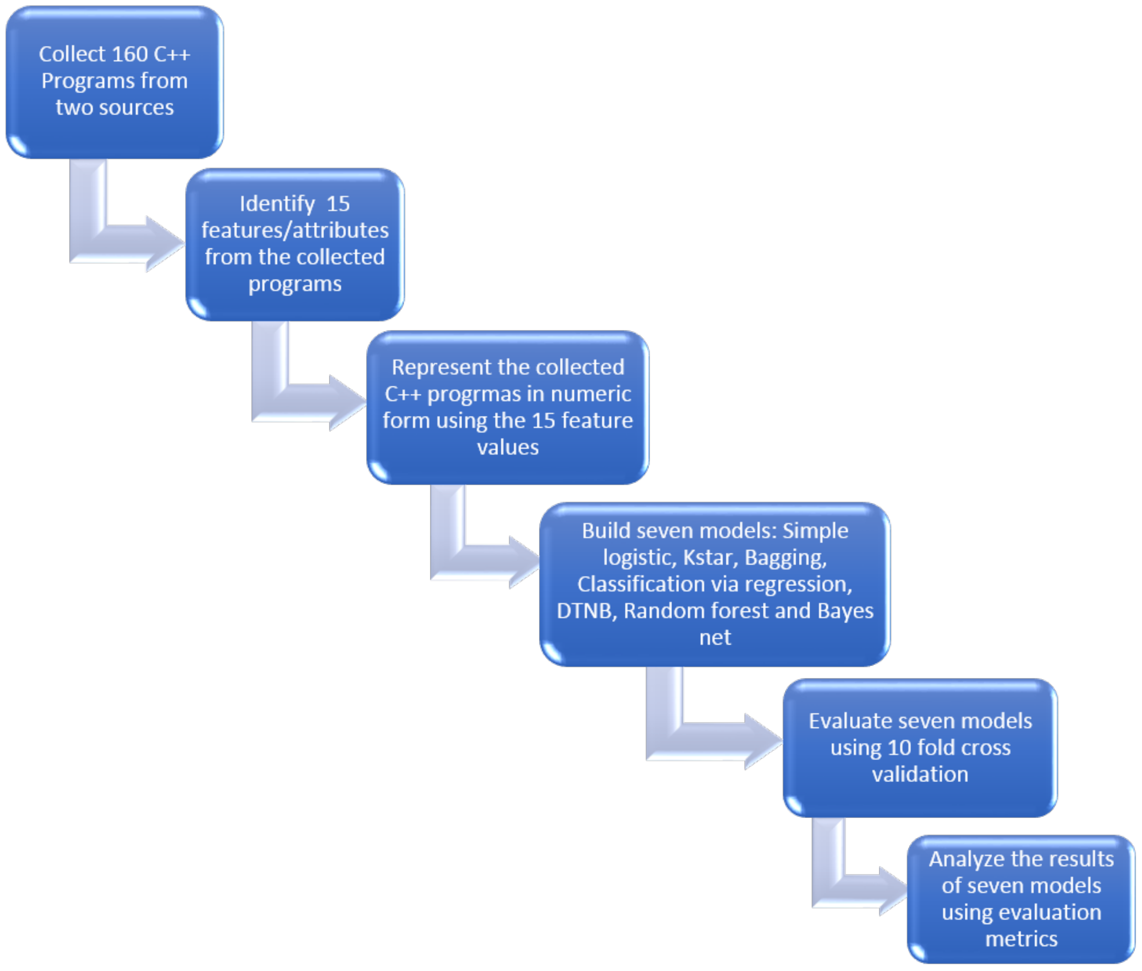

Collect 160 C++ Programs from two sources
Identify 15 features/attributes from the collected programs
Represent the collected C++ progrmas in numeric form using the 15 feature values
Build seven models: Simple logistic, Kstar, Bagging, Classification via regression, DTNB, Random forest and Bayes net
Evaluate seven models using 10 fold cross validation
Analyze the results of seven models using evaluation metrics

Figure 3.3: Steps followed for experiment 3

shown in Figure 3.4. In experiment 4, we again used 160 data samples but here we used only 7 features to build the classification model and classify the programs. We reduced our feature set from 15 to 7 using the statistical measure Information Gain. From the ranking of each feature, we selected top 7 ranked feature for this experiment. Those 7 features were: comment lines percentage, total comment lines, total blank lines, total commentary words, blank lines percentage, total functions, and average function line. After selecting the 7 features we used the Remove filter of WEKA to remove the other features from our dataset. We then used the same seven classification algorithms to build the classification models. After building those seven classification models we again applied the 10 fold cross validation technique to each classification model in order to evaluate the performance. Lastly, we measured the performance of the models using our evaluation metrics. Results are given in section 3.5.

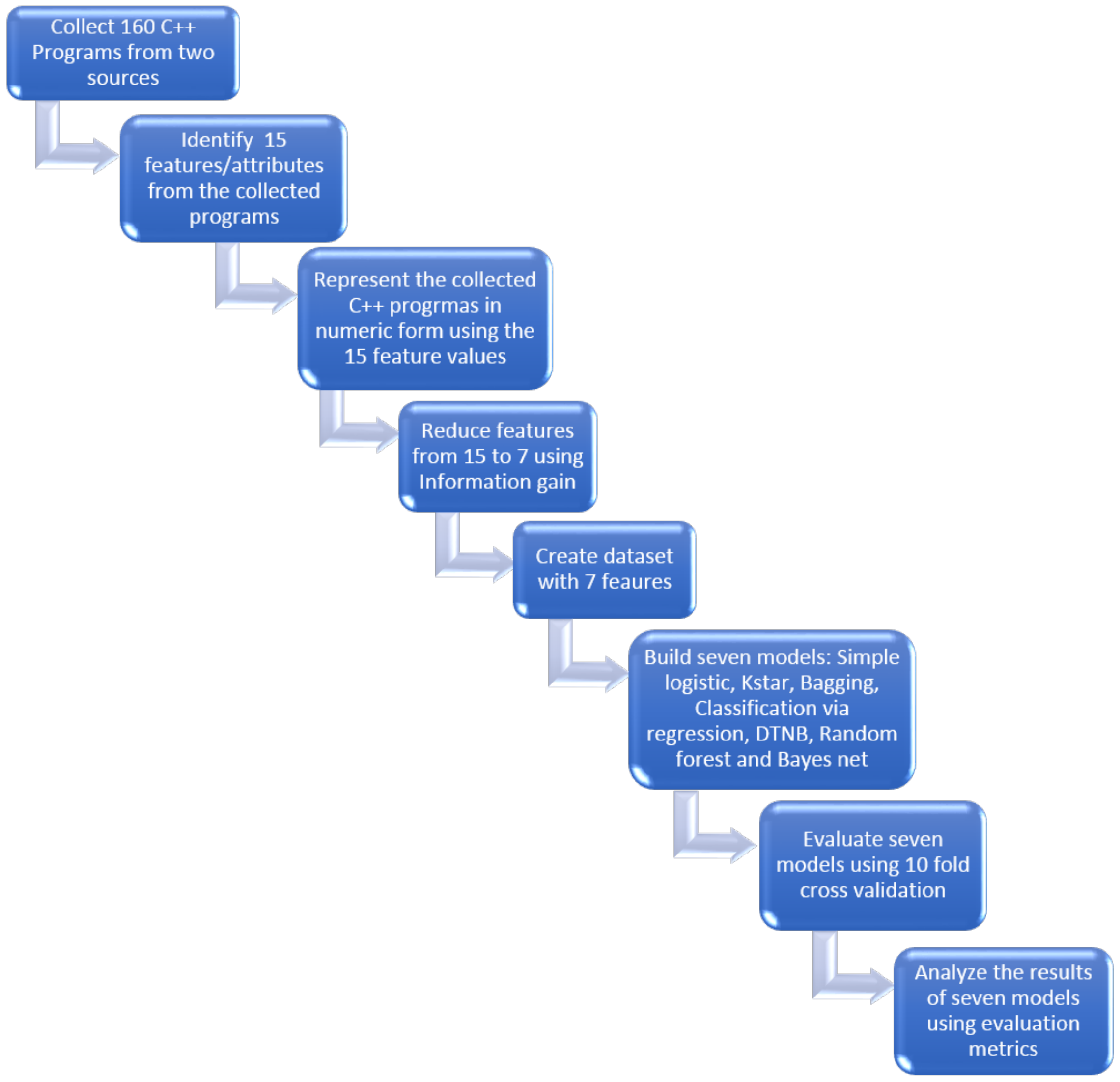

Collect 160 C++ Programs from two sources
Identify 15 features/attributes from the collected programs
Represent the collected C++ progrmas in numeric form using the 15 feature values
Reduce features from 15 to 7 using Information gain
Create dataset with 7 feaures
Build seven models: Simple logistic, Kstar, Bagging, Classification via regression, DTNB, Random forest and Bayes net
Evaluate seven models using 10 fold cross validation
Analyze the results of seven models using evaluation metrics

Figure 3.4: Steps followed for experiment 4

3.5 Results

In this section we discuss the results of four experiments. In each of the experiments, seven classification models were developed. In experiment 1 & 2, the main goal was to

Table 3.3: 16 Features and Cross Validation

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Simple Logistic	77.5	22.5	78.7	77.5	76.8
Kstar	75	25	74.9	75	74.9
Bagging	76.9	23.1	77.6	76.9	76.3
Classification Via Regression	78.1	21.9	80	78.1	77.2
DTNB	83.1	16.9	86.3	83.1	82.3
Random Forest	80.6	19.4	80.6	80.6	80.5
Bayes Net	70	30	72.6	70	69.9

categorize the C++ programs according to the gender (male/female) of the programmer. In these two experiments, class labels were male and female. In experiment 3 & 4, the main goal was to classify the programs according to the region of the author/programmer. In these two experiments, class labels were Canada and Bangladesh. In order to evaluate the seven classification models, the cross validation technique was used in all experiments.

3.5.1 Experiment 1

In experiment 1, we used 16 features and seven classification algorithms to build our seven models. As shown in Table 3.3, the DTNB model performed best with 83.1% accuracy. That means this model was able to classify 83.1% programs of our dataset correctly according to the gender of the programmer. The accuracy rate of the random forest model was also above 80%. The random forest model was able to classify 80.6% of the dataset correctly. Other models were also performed well. The classification via regression model classified 78.1% of the dataset accurately. The simple logistic and kstar models correctly classified 77.5% and 75% of the dataset respectively. The DTNB model also achieved the highest f-measure rate by scoring 82.3%. Next was the random forest model. The random forest model achieved 80.5% f-measure rate. Precision and recall of DTNB were 86.3% and 83.1% respectively.

Table 3.4: Performance of Seven Models with 16 Features

Model	**TP (/90)**	**TN (/70)**	**FP (/70)**	**FN (/90)**	**TP Rate (%)**	**FP Rate (%)**
Simple Logistic	82	42	28	8	77.5	26.4
Kstar	72	48	22	18	75	26.4
Bagging	80	43	27	10	76.9	26.6
Classification Via Regression	84	41	29	6	78.1	26.2
DTNB	89	44	26	1	83.1	21.4
Random Forest	78	51	19	12	80.6	21.1
Bayes Net	55	57	13	35	70	27.5

The DTNB model was able to classify the highest number of male written programs accurately compared to other models. In Table 3.4, we can see that among the 90 male written programs, the DTNB model was able to correctly classify 89 (TP=89) male written programs. Out of 90 male written programs the classification via regression model correctly classified 84 (TP=84) of them. The simple logistic and bagging model also correctly classified 82 (TP=82) and 80 (TP=80) male written programs respectively. The random forest model was able to correctly classify 78 (TP=78) male written programs out of 90 male written programs. The Bayes net model was in the last position among the seven classification models in terms of classifying the male written programs correctly. Bayes net model correctly classified 55 (TP=55) male written programs out of 90.

Although the Bayes net model scored lowest in terms of accuracy among the seven models, this model correctly classified the highest number of female written programs. The Bayes net model correctly classified 57 (TN=57) female written programs out of 70. The random forest model was second best in correctly classifying female written programs by classifying 51 (TN=51) out of 70 female written programs. The Kstar model correctly classified 48 (TN=48) out of 70 female written programs. The DTNB model correctly classified 44 (TN=44) out of 70 female written programs. The classification via regression model performed the worst among the seven classification models in terms of classifying the female written programs correctly. 41 (TN=41) out of 70 female written programs were

Table 3.5: 6 Features and Cross Validation

Model	**Correctly Classified (%)**	**Incorrectly Classified (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
Simple Logistic	76.9	23.1	77.8	76.9	76.2
Kstar	76.3	23.7	76.4	76.3	75.9
Bagging	79.4	20.6	80.6	79.4	78.7
Classification Via Regression	80	20	83.6	80	78.8
DTNB	83.1	16.9	86.3	83.1	82.3
Random Forest	76.9	23.1	77.3	76.9	76.4
Bayes Net	72.5	27.5	75.7	72.5	72.4

classified accurately by the classification via regression model.

3.5.2 Experiment 2

In experiment 2, we reduced our feature set from 16 to 6. As shown in Table 3.5, the DTNB model performed best with 83.1% accuracy. That means this model was able to classify 83.1% programs of our dataset correctly according to the gender of the programmer. The classification via regression model achieved 80% accuracy. The classification via regression model was able to classify 80% of the dataset correctly. Other models were also performed well. The Bagging model achieved almost 80% accuracy. 79.4% of the dataset was correctly classified by the bagging model. The random forest and simple logistic models achieved 76.9% accuracy. The Kstar model was close to the random forest and simple logistic model with an accuracy of 76.3%. The DTNB model also achieved the highest f-measure rate by scoring 82.3%. The classification via regression and bagging models achieved 76.8% and 76.7% f-measure respectively. Precision and recall of DTNB were 86.3% and 83.1% respectively.

The DTNB model was able to classify the highest number of male written programs accurately compared to other models. In Table 3.6, we can see that among the 90 male written programs the DTNB model was able to correctly classify 89 (TP=89). Out of 90

Table 3.6: Performance of Seven Models with 6 Features

Model	**TP (/90)**	**TN (/70)**	**FP (/70)**	**FN (/90)**	**TP Rate (%)**	**FP Rate (%)**
Simple Logistic	81	42	28	9	76.9	26.9
Kstar	77	45	25	13	76.3	26.4
Bagging	83	44	26	7	79.4	24.3
Classification Via Regression	80	40	30	2	80	25.1
DTNB	89	44	26	1	83.1	21.4
Random Forest	79	44	26	11	76.9	26.2
Bayes Net	56	60	10	34	72.5	24.6

male written programs, the bagging model correctly classified 83 (TP=83) of them. The simple logistic and classification via regression models also correctly classified 81 (TP=81) and 80 (TP=80) male written programs respectively. The random forest model was able to correctly classify 79 (TP=79) out of 90 male written programs. The Bayes net model performed the worst among the seven classification models in terms of classifying the male written programs correctly. The Bayes net model correctly classified only 56 (TP=56) male written programs out of 90.

Although the Bayes net model scored lowest in terms of the accuracy rate among the seven models, the model correctly classified the most female written programs. The Bayes net model correctly classified 60 (TN=60) female written programs out of 70 female written programs. The Kstar model was second in correctly classifying female written programs by classifying 45 (TN=45) out of 70 female written programs. All three bagging, DTNB and random forest models classified 44 (TN=44) out of 70 female written programs correctly. The classification via regression model performed the worst among the seven classification models in terms of classifying the female written programs correctly. Only 40 (TN=40) out of 70 female written programs were classified accurately by the classification via regression model.

Table 3.7: 15 Features and Cross Validation

Model	**Correctly Classified (%)**	**Incorrectly Classified (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
Simple Logistic	81.3	18.7	81.3	81.3	81.3
Kstar	85	15	85.5	85	85.1
Bagging	88.1	11.9	88.1	88.1	88
Classification Via Regression	88.8	11.2	88.8	88.8	88.8
DTNB	90.6	9.4	90.7	90.6	90.6
Random Forest	92.5	7.5	92.5	92.5	92.5
Bayes Net	82.5	17.5	83	82.5	82.6

3.5.3 Experiment 3

In experiment 3, our goal was to classify the programs according to the region of the programmer. We used 15 features and seven classification algorithms to build our seven models. As shown in Table 3.7, the random forest model performed best with 92.5% accuracy. That means this model was able to classify 92.5% programs of our dataset correctly according to the region of the programmer. The accuracy of the DTNB model was also above 90%. The random forest model was able to classify 90.6% of the dataset correctly. Other models were also performed well. The accuracy rates of the rest of the five models were all more than 80%. The classification via regression model achieved 88.8% accuracy. The classification via regression model was able to classify 88.8% of the dataset accurately. The bagging and kstar models correctly classified 88.3% and 81.3% of the dataset respectively. The random forest model also achieved the highest f-measure rate by scoring 92.5%. Next was the DTNB model. The DTNB model achieved 90.6% f-measure. Precision and recall of random forest model were both 92.5%.

The random forest model was able to accurately classify the most programs written by Canadian programmers compared to other models. In Table 3.8, we can see that among the 100 programs written by Canadian programmers random forest model was able to correctly classify 95 (TN=95) programs. The bagging and DTNB models also accurately classified

Table 3.8: Performance of Seven Models with 15 Features

Model	TP (/60)	TN (/100)	FP (/100)	FN (/60)	TP Rate (%)	FP Rate (%)
Simple Logistic	45	85	15	15	81.3	21.3
Kstar	51	85	15	9	85	15
Bagging	48	93	7	12	88.1	15.1
Classification Via Regression	52	90	10	8	88.8	12.1
DTNB	53	92	8	7	90.6	10.3
Random Forest	53	95	5	7	92.5	9.2
Bayes Net	49	83	17	11	82.5	17.8

more than 90 programs of Canadian programmers. Out of 100 programs written by Canadian programmers the bagging and DTNB model correctly classified 93 (TN=93) and 92 (TN=92) of them respectively. The classification via regression model accurately classified 90 (TN=90) programs which were written by Canadian programmers. Both the simple logistic and Kstar models correctly classified 85 (TN=85) programs. The Bayes net model was in the last position among the seven classification model in terms of classifying the Canadian written programs correctly. The Bayes net model correctly classified 83 (TN=83) Canadian written programs out of 100.

Although the Bayes net model scored lowest in terms of accurately classifying Canadian written programs among the seven models, this model correctly classified the highest number of Bangladeshi written programs. The DTNB and random forest models correctly classified the highest number of Bangladeshi written programs. Both the DTNB and random forest models correctly classified 53 (TP=53) Bangladeshi written programs out of 60. The classification via regression model was second in classifying correctly Bangladeshi written programs by classifying 52 (TP=52) out of 60. The Kstar model classified 51 (TP=51) Bangladeshi written programs correctly out of 60. From 60 Bangladeshi written programs, 49 (TP=49) programs were classified correctly by the Bayes net model. The simple logistic model performed the worst among the seven classification models in terms of classifying the Bangladeshi written programs correctly. 45 (TP=45) out of 60 Bangladeshi written

Table 3.9: 7 Features and Cross Validation

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Simple Logistic	81.3	18.7	81.3	81.3	81.3
Kstar	83.8	16.2	84.3	83.8	83.9
Bagging	85.6	14.4	85.7	85.6	85.4
Classification Via Regression	88.1	11.9	88.3	88.1	88.2
DTNB	83.8	16.2	84.3	83.8	83.9
Random Forest	89.4	10.6	89.3	89.4	89.3
Bayes Net	85	15	86	85	85.2

programs were classified accurately by the simple logistic model.

3.5.4 Experiment 4

In experiment 4, again our goal was to classify the programs according to the region of the programmer. We reduced our feature set from 15 to 7. We used seven classification algorithms to build seven classification models. As shown in Table 3.9, the random forest model performed best with 89.4% accuracy. This means that model was able to classify 89.4% of the programs of our dataset correctly according to the region of the programmer. The classification via regression model was able to classify 88.1% of the dataset correctly. Other models were also performed well. Bagging model achieved 85.6% accuracy. The Bayes net model also achieved 85% accuracy. The DTNB and kstar models both correctly classified 83.8% of the dataset. The random forest model also achieved the highest f-measure rate by scoring 89.3%. Next was the classification via regression model, which achieved 88.2% f-measure. Precision and recall of the random forest model were 89.3% and 89.4% respectively.

Both the random forest and bagging models were able to classify accurately most programs written by Canadian programmers compared to other models. In Table 3.10, we can see that among the 100 programs written by Canadian programmers random forest and

Table 3.10: Performance of Seven Models with 7 Features

Model	**TP (/60)**	**TN (/100)**	**FP (/100)**	**FN (/60)**	**TP Rate (%)**	**FP Rate (%)**
Simple Logistic	45	85	15	15	81.3	21.3
Kstar	50	84	16	10	83.8	16.4
Bagging	44	93	7	16	85.6	19.3
Classification Via Regression	52	89	11	8	88.1	12.5
DTNB	50	84	16	10	83.8	16.4
Random Forest	50	93	7	10	89.4	13
Bayes Net	53	83	17	7	85	13.7

bagging model were able to correctly classify 93 (TN=93) programs. The classification via regression model also accurately classified 89 (TN=89) programs out of 100 programs written by Canadian programmers.The simple logistic model correctly classified 85 (TN=85) programs which were written by Canadian programmers. Both the DTNB and Kstar models correctly classified 84 (TN=84) programs. The Bayes net model performed the worst among the seven classification model in terms of classifying the Canadian written programs correctly. The Bayes net model correctly classified 83 (TN=83) Canadian written programs out of 100.

The Bayes net model correctly classified the most Bangladeshi written programs. The Bayes net model correctly classified 53 (TP=53) Bangladeshi written programs out of 60. The classification via regression model was second in correctly classifying Bangladeshi written programs by classifying 52 (TP=52) out of 60 programs. All three DTNB, Kstar and random forest models accurately classified 50 (TP=50) Bangladeshi written programs out of 60. The bagging model performed the worst among the seven classification models in terms of classifying the Bangladeshi written programs correctly. Only 44 (TP=44) out of 60 Bangladeshi written programs were classified accurately by bagging model.

3.6 Discussion

In this section, we perform comparative analysis of the performance of learning models from the four experiments. We use accuracy, f-measure and confusion matrix to discuss the performance of the models.

3.6.1 Male and Female Written Programs

In experiment 1 and 2, the goal was to classify the programs according to the gender of the programmer. We used seven learning models for this purpose. In experiment 1, we used 16 features and in experiment 2, we used 6 features. Except for the Bayes net model, the other six models' accuracy was close to each other in both experiments. In both experiments, the DTNB model's performance was the best among the seven seven learning models. One reason for this better result is the underlying concept of the DTNB classification algorithm. This is a hybrid classifier. This classifier combines the power of two classifiers: decision table and Naive Bayes. The DTNB algorithm splits the set of attributes into two groups: one group assigns class probabilities based on Naive Bayes, the other group does this based on a decision table, and the resulting probability estimates are combined to take the classification decision [11].

The DTNB model achieved the highest accuracy of 83.13% and f-measure of 82.30%. If we look at the confusion matrix of the DTNB model in Table 3.11, we see that among the 90 male written programs, the DTNB model was able to correctly classify 89 of them. From the 70 female written programs this model correctly classified 44 programs. The DTNB model was able to classify the highest number of male written programs among the 7 models. One of the drawbacks of the DTNB model is the overfitting of data; we noticed that the model is more biased towards the male class label. As shown in Table 3.11, out of 160 programs, 133 programs were classified correctly. However, there were 26 programs mislabeled as male-written and 1 program mislabeled as female-written programs. The model was not able to categorize the female written programs accurately.

Table 3.11: DTNB Confusion Matrix of Experiment 1 & 2

Gender	Male	Female	Total
Male	**89**	*1*	90
Female	*26*	**44**	70
Total	115	45	160

Table 3.12: Bayes Net Confusion Matrix of Experiment 1

Gender	Male	Female	Total
Male	**55**	*35*	90
Female	*13*	**57**	70
Total	68	92	160

It is interesting to see that although we have reduced the number of features in experiment 2, this did not however, have any impact on the performance of the DTNB model. In both experiments, the performance of the DTNB model was same. The reason behind this is that in the DTNB model at each step, the algorithm removes an attribute entirely from the model [11]. For this reason in both experiments the DTNB model used only four features from the feature set to classify the programs. From the classifier output report of WEKA we found that these four features were the number of blank lines, number of comment lines, number of total functions, and region of the programmer. As both experiments used these four features, that is why the f-measure scores were the same.

In both experiments, the Bayes net model performed the worst among the seven learning models. In experiment 1, the Bayes net model achieved accuracy of 70% and in experiment 2, accuracy was 72.5%. F-measure was also lower than the other models. In experiment 1, the f-measure was almost 70%, and in experiment 2, the f-measure was 72.40%. If we look at the confusion matrices of the Bayes net model in Table 3.12 and Table 3.13, we observe that this model accurately classified 55 out of 90 male written programs and 57 out of 70 female written programs in experiment 1.

In experiment 2, 56 programs out of 90 male written programs and 60 out of 70 female written programs were correctly classified by the Bayes net model. The Bayes net model

Table 3.13: Bayes Net Confusion Matrix of Experiment 2

Gender	Male	Female	Total
Male	**56**	*34*	90
Female	*10*	**60**	70
Total	66	94	160

was able to classify the highest number of female written programs among the seven models. However, from the confusion matrix we found that because of the over fitting of data, the Bayes net model performed poorly among the seven models. It is clearly seen that this model is more biased towards the female class label, because in experiment 1, 35 programs and in experiment 2, 34 programs were misclassified as female written programs. Another reason is the underlying concept of the Bayes net algorithm. The Bayes net algorithm works well when the feature values are discretized [19]. In our dataset values were continuous, and for that reason we did not get the best performance from the Bayes net model. That is why when we applied the discretize filter in experiment 2, the accuracy of the model was increased by 2.5%.

3.6.2 Canadian and Bangladeshi Written Programs

In experiment 3 and 4, the goal was to classify the programs according to the region of the programmer. In experiment 3, we used 15 features and in experiment 4, we used 7 features. In both experiments, the random forest model achieved the highest accuracy (92.5% in experiment 3 and 89.39% in experiment 4) and f-measures (92.5% in experiment 3 and 89.30% in experiment 4). Table 3.14 and Table 3.15 shows that the random forest model accurately predicted the region of 95 out of 100 Canadian written programs and 53 out of 60 Bangladeshi written programs in experiment 3. In experiment 4 this model accurately predicted the region of 93 Canadian written programs and 50 Bangladeshi written programs. Therefore 148 programs in experiment 3 and 143 programs in experiment 4 were correctly classified by the random forest model. We noticed that the model is not biased towards one specific class label. The random forest model incorrectly predicted the region of the

Table 3.14: Random Forest Confusion Matrix of Experiment 3

Region	Bangladesh	Canada	Total
Bangladesh	**53**	*7*	60
Canada	*5*	**95**	100
Total	58	102	160

Table 3.15: Random Forest Confusion Matrix of Experiment 4

Region	Bangladesh	Canada	Total
Bangladesh	**50**	*10*	60
Canada	*7*	**93**	100
Total	57	103	160

programmer for 5 Canadian written and 7 Bangladeshi written programs in experiment 3, and 7 Canadian written and 10 Bangladeshi written programs in experiment 4.

In experiments 3 and 4, the simple logistic model performed the worst among the seven models. In both experiments, this model achieved the lowest accuracy (accuracy = 81.13%) and f-measures (f-measure = 81.30%). From Table 3.16, we observe that in both experiments, the simple logistic model accurately predicted the region of the programmer of 85 Canadian and 45 Bangladeshi written programs. In total 130 programs were classified correctly by the region of the programmer. The model incorrectly predicted the region of the programmer of 15 Canadian and 15 Bangladeshi written programs. Another interesting fact is that, the performance of the simple logistic model remained the same in experiment 4 although we reduced the number of features from 15 to 7. WEKA's classifiers output report showed that in the simple logistic model, the classification algorithm used four features to make the classification decision. These four features were number of blank lines, percentage of blank lines, number of comment lines, and number of total functions. As all of these four features were included in the reduced feature set, this helped to obtain the same f-measures in both experiments.

Table 3.16: Simple Logistic Confusion Matrix of Experiment 3 & 4

Region	Bangladesh	Canada	Total
Bangladesh	**45**	*15*	60
Canada	*15*	**85**	100
Total	60	100	160

3.7 Threats to Validity

According to Buse and Weimer [6], there exist some threats towards the validity of the results of this kind of research. In our research we also identified several potential threats to validity. These are:

- The dataset of this research was quite small. The dataset consisted of only 160 C++ programs. Because of this small dataset, there was a lack of data to train the machine learning model more effectively.
- The number of male written and female written programs were not evenly distributed. We had 90 male written and 70 female written programs in our dataset. The number of female participants in this research was lower than the number of male participants.
- The number of Canadian written and Bangladeshi written programs were not evenly distributed. The number of Canadian participants were higher than the number of Bangladeshi participants. We had 100 Canadian written programs and 60 Bangladeshi written programs in this study.
- In this research we tried to classify the programs according to the region of the programmer, which is Canada or Bangladesh. However, from Canada we only collected programs from University of Lethbridge, and from Bangladesh we collected programs from six universities. This is a limited selection (extremely limited in the case of Canada) which means results are not generalizable to the whole countries. This is a threat to the validity of the results of the machine learning models.

- In this research we only considered the C++ program, not other kinds of programming language. Other language may demonstrate different results.
- Another threat to validity is the type of participants in this research. All the participants of this research were students. They are not professional programmers. Their year of experience in programming knowledge varies from one to four. So it is quite obvious their use of programming language is quite different from expert and professional programmers.
- We had programs belonging to various problem domains which could result in variation in feature usage. This is also a threat to the validity of the results.
- In this research we selected a small set of features to classify collected programs based on the gender or the region of the programmer. We trained and evaluated our machine learning models for these features. There is the possibility that we missed some other important features for this research.
- In this study we selected region of the programmer as a feature while classifying the programs based on the gender of the author. Not having region of the programmer would impact any future results, if we wanted to expand this work to a large dataset. This is also a threat to validity of the results.

Chapter 4

Analysis of Features

4.1 Reducing the Set of Features

As our dataset was composed of only 160 programs, we reduced the feature set to avoid over fitting of data. In order to find out the most significant features for the learning models, we applied a feature evaluator. The selected feature evaluator was information gain, which in WEKA is named “InfoGainAttributeEval” (described in section 2.4.3). Using this feature evaluator, we selected the top six highest ranked features for the classification of programs according to the gender of the programmer. These six features were:

- Source code lines percentage
- Total number of blank lines
- Blank lines percentage
- Comment lines percentage
- Total number of functions
- Region of the programmer

Using these six features and the cross validation technique, we developed the same seven classification models. Our goal was to identify whether a reduced feature set is enough to categorize the collected programs according to the gender of the programmer.

To find out the significance of the reduced feature set, we performed comparative analysis between the models with all 16 features and the models with reduced six features.

Table 4.1: F-measures in Gender-wise Classification

Classification Algorithms	**F-measure Based On 16 Features (Experiment 1)**	**F-measure Based On 6 Features (Experiment 2)**
Simple Logistic	76.80%	76.20 %
Kstar	74.90 %	75.90 %
Bagging	76.30 %	78.70 %
Classification Via Regression	77.20 %	78.80 %
DTNB	**82.30 %**	**82.30 %**
Random Forest	80.50 %	76.40 %
Bayes Net	69.90 %	72.40 %

Table 4.1 shows the performance of the seven models with 16 features and the performance of the seven models with the reduced six features. Performances are shown in terms of f-measure of the models. From Table 4.1, we can see that classification models with reduced number of features performed almost similar to the models with all the 16 features. Among the seven classification models, four of them achieved the highest f-measure in comparison with the models with all 16 features. These four models were the K star, bagging, classification via regression and Bayes net model. As we removed some features in the second experiment, this helped to remove the noisy data from the dataset and helped the classification model to predict the class of the program more accurately.

Simple logistic and DTNB models achieved almost the same f-measure. Only the random forest model achieved a lower f-measure than the other six models in comparison with the models with all 16 features. After reducing the features, the DTNB model scored the same f-measure (f-measure = 82.30%) as the f-measure with 16 features. The main reason for this same score is that, in both experiments, the DTNB model used the same four features to classify the programs according to the gender of the programmer. From the classifier output report of WEKA, we found that these four features were the number of blank lines, number of comment lines, number of total functions, and region of the programmer. As in both experiments, these four features were present, which is why in both experiments the f-measure score was same.

Table 4.2: F-measures in Region-wise Classification

Classification Algorrithms	**F-measure Based On 15 Features (Experiment 3)**	**F-measure Based On 15 Features (Experiment 4)**
Simple Logistic	81.30 %	81.30 %
Kstar	85.10 %	83.90 %
Bagging	88.00 %	85.40 %
Classification Via Regression	88.80 %	88.20 %
DTNB	90.60 %	83.90 %
Random Forest	**92.50 %**	**89.30 %**
Bayes Net	82.60 %	85.20 %

We also tried to reduce the feature set for the classification of programs based on the region of the programmer. The same evaluator "InfoGainAttributeEval" was used to identify the most relevant features for the learning models. Using the feature evaluator, we selected the top seven highest ranked features. These seven features were:

- Total number of blank lines
- Blank lines percentage
- Total number of comment lines
- Comment lines percentage
- Total commentary words
- Total number of functions
- Average function lines

Using this subset of features and the cross validation technique, we again developed the same seven classification models. Here the goal was to identify whether a subset of features is enough to classify the collected programs according to the region of the programmer.

To find out the significance of the reduced feature set, we performed comparative analysis between the models with all 15 features and the models with reduced seven features.

Table 4.2 shows the performance of the seven models with 15 features and the performance of the seven models with the reduced seven features. Performances are shown in terms of the f-measure of the models. From Table 4.2 we can see that in the experiment of reduced features set, except for the simple logistic model all other models scored less in terms of the f-measure score in comparison with the models with all 15 features. The simple logistic model achieved the same f-measure of 81.30%. From WEKA's classifier output report, we found that in the simple logistic model four features were used to take the classification decision. These four features were the number of blank lines, percentage of blank lines, number of comment lines, and number of total functions. As all of these four features were included in the reduced feature set, this helped obtain the same f-measure in both experiments.

4.2 Statistical Approach

In this research we tried to find out the differences in programming styles and use of programming features among the male and female programmers. In order to do that, we built some machine learning models and ran a variety of experiments with our collected dataset and the selected list of features. We found some significant results from those models which indicates that there exists some differences among the male written and female written programs. To determine whether the differences are real or occur by chance, we carried out a statistical test. This statistical test will help to determine whether the features which we used to build the machine learning models are statistically significant or not.

As a form of statistical method, we used the "t-test" to carry out the statistical analysis. Results of this statistical analysis were measured in terms of ρ value that can be found in the output. The t-test is a statistical test which indicates whether or not there is a significant difference between the mean or average scores of two groups. This is a common approach to compare two samples or groups with small sample sizes. The t-test calculates the means of two groups and tells whether or not they are different from each other. The t-test also indicates how significant the differences are. In short, the t-test determines whether the differences of two groups are real or occur by chance.

In a t-test two hypotheses are tested. A null hypothesis is tested against the alternative hypothesis. The null hypothesis is denoted as H_0 and the alternative hypothesis is denoted as H_a. The null hypothesis, H_0 states that there is no difference in two groups ($\mu_1 = \mu_1$) because it might occur by chance. The alternative hypothesis, H_a states that there is a difference in two groups ($\mu_1 \neq \mu_1$). The results from the t-test either support or reject the null hypothesis, H_0. If the null hypothesis is rejected then the alternative hypothesis is supported but can not be proved. The ρ value provides evidence that the difference is either due to random chance or not [23]. A threshold value is used to indicate the difference of statistical significance between two groups. In general, the threshold value is 0.05. The difference between two groups is real if the ρ value of a t-test is less than 0.05. This rejects the null hypothesis, H_0. This means the differences between two groups are statistically significant. On the other hand, the likelihood of the result occurring by chance is high when the ρ value is greater than 0.05. This accepts the null hypothesis, H_0. This means the differences are not statistically significant. This can happen when the number of samples is not large enough. Studies with a small number of samples are bound to have “deficient power” [7], which means that it is not possible to know if results would be different with a large number of samples [23].

In order to find out the significance of our features in male written and female written programs we carried out the t-test. We performed a two-tailed t-test to find out if the mean of one group is greater than the other group. In this test we were interested to see whether the difference can be considered to be statistically significant or not. We performed the t-test for each of the features to determine which hypothesis is acceptable. In all the tests, features are treated as dependent variables and the group of programmers (male or female) are treated as independent variables. The ρ value of each feature from the t-test is shown in Table 4.3.

Table 4.3: T-Test (ρ-values) in Gender-wise Classification

Features	**ρvalue**
Total Lines	0.1612
Total Source Code Lines	0.3812
Source Code Lines Percentage	**0.0001**
Total Blank Lines	**0.0028**
Blank Lines Percentage	**0.0001**
Total Comment Lines	0.1886
Total Comment Lines Percentage	**0.0012**
Total Mixed Lines	0.2049
Mixed Lines Percentage	**0.0204**
Total Commentary Words	0.6285
Physical Executable Line of Code	0.4629
Logical Executable Line of Code	0.3917
Total Functions	0.0628
Total Function Lines	0.3128
Average Function Lines	0.0861

In Table 4.3 we observe that there are five features which have a ρ value less than 0.05. These features are:

- Source code line percentage
- Total number of blank lines
- Blank lines percentage
- Comment line percentage
- Mixed line percentage

The null hypothesis, H_0 is rejected and the alternative hypothesis, H_a is accepted for these five features. Therefore, we can consider these five features as statistically significant. For the remaining features, ρ values are greater than 0.05 indicating that null hypothesis, H_0 can not be rejected. So, these features are not considered statistically significant. This is not surprising as we have very small number of male written and female written programs

Table 4.4: Comparison of Feature Use in Male and Female Written Programs

Features	Male Written Programs	Female Written Programs
Source Code Lines Percentage	76%	24%
Total Number of Blank Lines	18%	82%
Blank Lines Percentage	18%	82%
Comment Lines Percentage	34%	66%
Mixed Lines Percentage	78%	22%

in our dataset. We can say that these remaining features have no relation to the gender of the programmer.

Based on the statistically significant features we performed a comparative analysis between male and female written programs. Table 4.4 shows the percentage of programs where the statistically significant features were used more in one group of programmers than other. From the table, we can see that source code line percentage was higher in 76% male written programs than the female written programs. Mixed line percentage was also higher in male written programs than the female written programs. In 78% of the programs mixed line percentage was higher in male written programs than female written programs. On the other hand total number of blank lines, blank line percentage and comment lines percentage were higher in female written programs than the male written programs. In 82% of the programs the total number of blank lines and blank line percentage were higher in female written programs. In 66% of the programs comment line percentage was higher in female written programs than male written programs.

From the comparison of statistically significant features in 160 programs of our dataset, it appears that female programmers tends to use more blank lines and comments in their programs compared to the male written programs and that is why we found that source code line percentage is higher in male written programs than the female written programs. Another interesting fact is that male programmers like to write comments in parallel with the code line, where as female programmers like to write comments in a separate new line.

In this research we also tried to find out the differences in programming styles and use

Table 4.5: T-Test (ρ-values) in Region-wise Classification

Features	**ρvalue**
Total Lines	0.88143
Total Source Code Lines	0.91161
Source Code Lines Percentage	0.08607
Total Blank Lines	0.25854
Blank Lines Percentage	0.44013
Total Comment Lines	**0.00279**
Total Comment Lines Percentage	**0.00016**
Total Mixed Lines	**0.00533**
Mixed Lines Percentage	**0.01799**
Total Commentary Words	**0.00001**
Physical Executable Line of Code	0.97261
Logical Executable Line of Code	0.74681
Total Functions	**0.00002**
Total Function Lines	0.62019
Average Function Lines	**0.00001**

of programming features among the Canadian and Bangladeshi programmers. In order to do that, we again carried out the t-test. We performed the t-test for each of the features to determine which hypothesis is acceptable. In all the tests, features are treated as dependent variables and the regions of the programmers (Canada or Bangladesh) are treated as independent variables. The ρ values of each feature from the t-test are shown in Table 4.5.

In Table 4.5 we observe that there are seven features which have a ρ value less than 0.05. These features are:

- Total number of comment lines
- Comment lines percentage
- Total number of mixed lines
- Mixed lines percentage
- Total number of commentary words

- Total number of functions
- Average function lines

The null hypothesis, H_0 is rejected and the alternative hypothesis, H_a is accepted for these seven features. Therefore, we can consider these seven features statistically significant. For the remaining features, ρ values are greater than 0.05, indicating that the null hypothesis, H_0 can not be rejected. So, these features are not considered statistically significant. This is not surprising as we have very small number of Canadian written and Bangladeshi written programs in our dataset. We can say that these remaining features have no relation to the region of the programmer.

Based on the statistically significant features we performed a comparative analysis between Canadian and Bangladeshi written programs. Table 4.6 shows the percentage of programs where the statistically significant features were used more in one group of programmers than other. From the table, we can see that except the average function lines all the other features were used more in Canadian written programs than the Bangladeshi written programs. In 66% programs of our dataset average function lines were higher in Bangladeshi written programs. On the other hand in total number of comment lines, comment lines percentage, total number of mixed lines, mixed lines percentage, total number of commentary words and total number of functions were higher in 86%, 90%, 78%, 76%, 86% and 74% Canadian written programs respectively.

From the comparison of statistically significant features in 160 programs of our dataset, it appears that Canadian programmers used more comments and functions than the Bangladeshi programmers. Use of comments and functions make a program more manageable and more understandable. So we might conclude that Canadian written programs are more manageable and more understandable than the Bangladeshi written programs.

Table 4.6: Comparison of Feature Use in Canadian and Bangladeshi Written Programs

Features	**Canadian Written Programs**	**Bangladeshi Written Programs**
Total number of comment lines	86%	14%
Comment lines percentage	90%	10%
Total number of mixed lines	78%	22%
Mixed lines percentage	76%	24%
Total number of commentary words	86%	14%
Total number of functions	74%	26%
Average function lines	34%	66%

4.3 Visual Analysis of Random Programs

In this study we also conducted visual inspection or visual analysis of collected programs alongside with machine learning and statistical analysis to find out the differences among the male written and female programs, as well as Canadian written and Bangladeshi written programs.

4.3.1 Male and Female Written Programs

We visually investigated the five features which were found to be statistically significant, as determined on the t-test, in classifying male and female written programs. These five features were source code line percentage, number of blank lines, blank lines percentage, comment line percentage, and mixed line percentage. We randomly chose three male written and three female written programs for visual inspection from our collected C++ programs. Though we have chosen the programs randomly, we selected the male and female written programs with same number of total lines for a better comparison of two groups of programmers. We did a one to one comparison of male and female written programs where both the programs contained almost the same number of lines.

The first male written program had a total of 222 lines and the first female written program had a total of 217 lines. In the male written program, source code line percentage was 91.89%, where in the female program source the code line percentage was 75.12%.

Table 4.7: First Male and Female written Program

Feature	Male Written Program	Female Written Program
Total Number of Lines	222	217
Source Code Line Percentage	91.89%	75.12%
Number of Blank Lines	0	32
Blank Lines Percentage	0%	14.75%
Comment Lines Percentage	5	16
Mixed Lines Percentage	5.86%	2.76%

Table 4.8: Second Male and Female written Program

Feature	Male Written Program	Female Written Program
Total Number of Lines	323	314
Source Code Line Percentage	95.36%	76.43%
Number of Blank Lines	0	23
Blank Lines Percentage	0%	7.32%
Comment Lines Percentage	2.79%	14.65%
Mixed Lines Percentage	1.86%	1.59%

In the male written program there were no blank lines. In the female written program there were 32 blank lines and the blank lines percentage was 14.75%. In the male written program, the programmer used 5 comments and the percentage was 2.25%. On the other hand, the female programmer used 16 comments and the percentage was 7.37%. Finally, in the male written program, 13 mixed lines were found with the percentage of 5.86%, while in the female written program, 6 mixed lines were found with the percentage of 2.76%.

The second male written program had a total of 323 lines and the second female written program had a total of 314 lines. In the male written program, source code line percentage was 95.36%, where in the female program the source code line percentage was 76.43%. In the male written program, there were no blank lines. In the female written program, there were 23 blank lines and the blank lines percentage was 7.32%. In the male written program, the programmer used 9 comments and the percentage was 2.79%. On the other hand, the female programmer used 46 comments and the percentage was 14.65%. Finally, in male written program, 6 mixed lines were found with the percentage of 1.86% , while in

Table 4.9: Third Male and Female written Program

Feature	Male Written Program	Female Written Program
Total Number of Lines	108	105
Source Code Line Percentage	63.89%	86.67%
Number of Blank Lines	0	13
Blank Lines Percentage	0%	12.38%
Comment Lines Percentage	17.59%	0.95%
Mixed Lines Percentage	18.52%	0%

the female written program, 5 mixed lines were found with the percentage of 1.59%.

The third male written program had a total of 108 lines and the third female written program had a total of 105 lines. In the male written program, source code line percentage was 63.89%, where in the female program the source code line percentage was 86.67%. In the male written program, there were no blank lines. In the female written program, there were 13 blank lines and the blank lines percentage were 12.38%. In the male written program, the programmer used 19 comments and the percentage was 17.59%. On the other hand, the female programmer used just 1 comment and the percentage was 0.95%. Finally, in male written program, 20 mixed lines were found with the percentage of 18.52%, while in the female written program, there were no blank lines.

From the visual comparison of three random programs of male and female programmers, we observed that there were significant differences in those five features. The visual inspection seems to match the findings of the statistical analysis. Source code line percentage and mixed line percentage were higher in male written programs than in the female written programs. Although in one male written program, the source code line percentage and mixed line percentage was lower than the female written program. On the other hand, the number of blank lines, blank line percentage, and comment line percentage were higher in all the female written programs than the three male written programs.

Table 4.10: First Canadian and Bangladeshi written Program

Feature	**Canadian Written Program**	**Bangladeshi Written Program**
Total Number of Lines	248	236
Total Comment Lines	27	0
Comment Lines Percentage	10.89%	0%
Total Mixed Lines	13	0
Mixed Lines Percentage	5.24%	0%
Total Commentary Words	155	0
Total Functions	8	5
Average Function Lines	26.25	43

4.3.2 Canadian and Bangladeshi Written Programs

In this section we visually investigate the seven features which were found to be statistically significant in Canadian and Bangladeshi written programs. These seven features were the number of comment lines, comment lines percentage, number of mixed lines, mixed lines percentage, total number of commentary words, and total number of functions. For visual inspection we randomly chose three Canadian written and three Bangladeshi written programs from our collected C++ programs. Although we have chosen the programs randomly, we selected the Canadian and Bangladeshi written programs to have the same number of total lines for a better comparison of the two group of programmers. We did a one to one comparison of Canadian and Bangladeshi written programs where both the programs contained almost the same number of lines.

The first Canadian written program had a total of 248 lines and the first Bangladeshi written program had a total of 236 lines. In the Canadian written program, 27 comment lines were found with the percentage of 10.89%. In the Bangladeshi written program, there were no comment lines. In the Canadian written program, there were 13 mixed lines with the percentage of 5.24%. In the Bangladeshi written program, there were no mixed lines. In the Canadian written program, we found total 155 commentary words, where in the Bangladeshi written program we did not find any commentary words. In the Canadian

Table 4.11: Second Canadian and Bangladeshi written Program

Feature	Canadian Written Program	Bangladeshi Written Program
Total Number of Lines	133	136
Total Comment Lines	21	5
Comment Lines Percentage	15.79%	3.68%
Total Mixed Lines	10	0
Mixed Lines Percentage	7.52%	0%
Total Commentary Words	254	11
Total Functions	3	9
Average Function Lines	32.67	11.33

written program, the programmer used a total of 8 functions and the average number of function lines was 26.25. In the Bangladeshi written program, the programmer used a total of 5 functions and the average number of function lines was 43.

The second Canadian written program consisted of a total of 133 lines and the second Bangladeshi written program consisted of a total of 136 lines. In the Canadian written program, 21 comment lines were found with the percentage of 15.79%. In the Bangladeshi written program, there were 5 comment lines with the percentage of 3.68%. In the Canadian written program, there were 10 mixed lines with the percentage of 7.52%. In the Bangladeshi written program there were no mixed lines. In the Canadian written program, we found a total of 254 commentary words, where in the Bangladeshi written program we found 11 commentary words. In the Canadian written program, the programmer used a total of 3 functions and the average number of function lines was 32.67. In the Bangladeshi written program the programmer used a total of 9 functions and the average number of function lines was 11.33.

The third Canadian written program consisted of a total of 152 lines and the first Bangladeshi written program consisted of a total of 156 lines. In the Canadian written program, 46 comment lines were found with the percentage of 30.26%. In the Bangladeshi written program, there was only one comment line and the percentage was 0.64%. In the Canadian written program, there were 4 mixed lines with the percentage of 2.63%. In the

Table 4.12: Third Canadian and Bangladeshi written Program

Feature	**Canadian Written Program**	**Bangladeshi Written Program**
Total Number of Lines	152	156
Total Comment Lines	46	1
Comment Lines Percentage	30.26%	0.64%
Total Mixed Lines	4	4
Mixed Lines Percentage	2.63%	2.56%
Total Commentary Words	228	10
Total Functions	8	6
Average Function Lines	8.88	18.67

Bangladeshi written program, there were also 4 mixed line and the percentage was 2.56%. In the Canadian written program, we found a total of 228 commentary words, where in the Bangladeshi written program we found a total of 10 commentary words. In the Canadian written program, the programmer used total 8 functions and the average number of function lines was 8.88. In the Bangladeshi written program. the programmer used total 6 functions and the average number of function line was 18.67.

From the visual comparison of three random programs of Canadian and Bangladeshi programmers, we noticed that there were significant differences in those seven features. We found that six features - number of comment lines, comment line percentage, number of mixed line, mixed line percentage, total commentary words, and number of total functions were all used more in Canadian written programs. In Bangladeshi programs, we found that the number of average function lines are higher than Canadian programmer written programs. Out of three programs, we found only one exception, where the number of total functions was higher and the number of average function lines was lower in a Bangladeshi written program.

4.4 Relationship between Features

In this section, we examined the relationship between the features. We focused on identifying the pairs of features which are linearly dependent on each other in male and female written programs and also in Canadian and Bangladeshi written programs. In order to find out the strength of the relationship between features, we measured the γ value between each pair of features. γ value is used to demonstrate the existence of a linear relationship between two features. Based on the value of γ there are three possibilities [33]:

(i) $\gamma = 0$ represents that features are independent and they are not correlated with each other;

(ii) $\gamma > 0$ represents that features are positively correlated. A higher value of γ shows that there is a strong correlation between the given features; and

(iii) $\gamma < 0$ represents that features are negatively correlated. This means that as the occurrence of one feature increases, the occurrence of the other feature decreases.

In Appendix C, we list γ values in a correlation matrix on the basis of the original feature frequencies in male written, female written, Canadian written and Bangladeshi written programs. A correlation (γ) of greater than 0.5 indicates a strong linear relationship [6]. In the following section we examine each pair of features with higher γ value in male written, female written, Canadian written and Bangladeshi written programs.

From Appendix C we can see that in male written programs there are total 21 pairs of feature which have γ value greater than 0.5. In female written programs there are total 31 pairs of feature which have γ value greater than 0.5. In Canadian written programs we found that there are total 22 pairs of feature which have γ value greater than 0.5. In Bangladeshi written programs there are 21 pairs of feature which have γ value greater than 0.5. Interesting fact is that in all the four categories (male, female, Canadian and Bangladeshi) of programs there are 12 pairs of feature which have γ value greater than 0.8. This indicates that there exists a strong relationship between those features. These pairs of feature are:

- Total number of lines of a program has a strong relationship with the total number of source code line, the total number of physical executable code, the total number of logical executable code and the total number of function lines of a program.

- Total number of source code line of a program has a strong relationship with the total number of physical executable code, the total number of logical executable code and the total number of function lines of a program.
- Total number of comment lines of a program has a strong relationship with the total number of commentary words.
- Total number of mixed lines of a program has a strong relationship with mixed lines percentage.
- Total number of physical executable code of a program has a strong relationship with the total number of logical executable code and the total number of function lines of a program.
- Total number of logical executable code of a program has a strong relationship with the total number of function lines of a program.

Besides these pairs of features in male written program we found that the total number of blank lines has a strong relationship with blank line percentage (γ=0.93). In comparison with male written programs, with female written programs we found that in female written programs there are more strongly related features than the male written programs. Some of the strongly connected features of female written programs are:

- Total number of lines and the total number of blank lines.
- Total number of lines and the total number of comment lines.
- Total number of lines and the total number of commentary words.
- Total number of functions and the total number of comment lines.
- Total number of function lines and the total number of comment lines.

4.5 Comparison with Previous Work

As this research is based on the work of Naz [23], which categorized computer programs according to the gender of the programmer, we compared the results of Naz's work with the results of ours. In order to find out the significance of our selected features, we performed the same experiments as Naz did. We used the same 100 C++ programs as Naz's work. We used the same three learning classifiers of WEKA: Naive Bayes, K star and J48. We kept the experimental environment unchanged except the feature set. Naz used 50 features (described in Table 2.1), whereas we used our 16 features (described in section 3.3) to classify the programs based on the gender of the programmer. The evaluation method for this experiment was also cross validation as in Naz's work. The result of this experiment is shown in Table 4.13. The result of Naz's work is shown in Table 4.14.

Comparing the results of the two tables indicates that the performances of our models were much better than Naz's models. The K star model achieved the highest accuracy (accuracy = 72%) and the highest f-measure (F-measure = 71.9%) among the three classification models in Naz's work. In our experiment, the Naive Bayes model achieved the highest accuracy (accuracy = 92%) and the highest f-measure (f-measure = 92%). In Naz's work, the J48 model achieved the lowest accuracy (accuracy = 63%), whereas the J48 model achieved 89% accuracy in our experiment. In our experiment, the lowest accuracy was 79% which was achieved by the K star model.

In Naz's work among the 50 male written programs, at most 33 programs were classified accurately by the K star and the Naive Bayes model. In our experiment, the Naive Bayes model also accurately classified the highest number of male written programs, correctly classifying 49 programs out of 50 male written programs. The K star model in Naz's work correctly classified 39 female written programs out of 50 female written programs, which was the highest rating among the three models. In our experiment, the Naive Bayes model once again performed the best, classifying 42 female written programs out of 50 programs among the three models.

Table 4.13: Result of this Research

Models	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**	**TP (/50)**	**FP (/50)**	**TN (/50)**
J48	89	89.4	89	89	42	3	47
Kstar	79	79.1	79	79	38	9	41
NB	92	92.6	92	92	43	1	49

Table 4.14: Result of Naz's Work

Models	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**	**TP (/50)**	**FP (/50)**	**TN (/50)**
J48	63	63	63	63	33	20	30
Kstar	72	72.3	72	71.9	39	17	33
NB	66	66	66	66	33	17	33

Although the methodology was the same in both Naz's and our experiment except for the feature set, we achieved a much better performance from our models. This is likely due to the significance of our feature set.

4.6 Impact of Region of the Programmer

In experiments 1 and 2, (described in section 3.4.3 and 3.4.5), while categorizing the programs based on the gender of the author, we used the region of the programmer as a feature in the feature set. In section 3.3.1 we showed that inclusion of this feature improved the performance of the model in predicting the gender of the author. In this section we further discuss the impact of region of the programmer in identifying the gender of the programmer. In our dataset we had programs from two different regions: Canada and Bangladesh; we performed two experiments with the programs of these two different regions. In both experiments, the goal was to categorize the programs based on the gender of the author. In both experiments, we used the same seven classification models and cross validation technique to build and evaluate the models. In both experiments we used the same 15 features (described in Section 3.3), and we excluded the region of the programmer from the feature list. In the first experiment, the dataset was composed of 100 Canadian written pro-

Table 4.15: 15 Features and Cross validation Technique

Classifaction Model	**Experiment 1 with 100 Canadian Written Progrmas (Accuracy %)**	**Experiment 2 with 60 Bangladeshi Written Programs (Accuracy %)**
Simple Logistic	90.8	65
Kstar	79	61.7
Bagging	91	63.3
Classification Via Regression	90	56.7
DTNB	91	65
Random Forest	88	58.3
Bayes Net	90.6	66.7

grams, and in the second experiment, the dataset was composed of 60 Bangladeshi written programs. The results of these two experiments are shown in Table 4.15.

From Table 4.15 we can see that machine learning models were more successful in Canadian written programs rather than Bangladeshi written programs while categorizing the programs based on the gender of the programmer. In Canadian written programs the highest accuracy was 91%; this means 91% of the Canadian written programs were classified accurately based on the gender of the author. In Bangladeshi written programs highest accuracy was 65%; this means 65% of the Bangladeshi written programs were classified accurately. The lowest accuracy was 79% in Canadian written programs and 56% in Bangladeshi written programs.

The reason behind this poor performance of the models in Bangladeshi written programs is the overfitting of data. We noticed that all the models were biased towards the male class label. Table 4.16 shows the performance of all the models in Bangladeshi male and female written programs. From Table 4.16, we can see that all the models are extremely biased towards the male class. For example, the DTNB model classified 39 (TP=39) male written programs accurately out of 40 male written programs. However, this model misclassified all the female written programs (FP=20) as male written programs. The Bagging model

Table 4.16: Confusion Matrices of Experiment 2

Model	**TP (/40)**	**TN (/20)**	**FP (/20)**	**FN (/40)**
Simple Logistic	38	1	19	2
Kstar	31	6	14	9
Bagging	36	2	18	4
Classification Via Regression	34	0	20	6
DTNB	39	0	20	1
Random Forest	30	5	15	10
Bayes Net	35	1	19	5

classified 36 (TP=36) male written programs accurately out of 40 male written programs. Again, this model misclassified 18 female written programs (FP=18) as male written programs. One of the reasons behind the bias towards the male written programs is the lack of female written programs in the dataset. We had only 20 female written Bangladeshi programs in our dataset. As the number of programs were very few, it was difficult to find the significant differences among the Bangladeshi male and female written programs. This is also reflected in the results.

Although everything was the same in both experiments, we experienced quite different performance level from the models. This indicates that differences exist in the male and female written programs based on the region of the programmers. For that reason we selected region of the programmer as a feature while categorizing the programs based on the gender of the programmer. In section 3.3.1 we found that inclusion of the region of the programmer in the feature set improved the model performances, which helped to identify the gender of the author more accurately.

Chapter 5

Conclusion

This research is based on the work of Naz [23], which categorized computer programs on the basis of the author's gender. Naz used machine learning techniques to categorize the male written and female written programs. Motivation for this work came from Argamon et al. [4, 3, 16] who also used machine learning methods to categorize French and English text documents according to the gender of the author. The goal of this research was to improve the performance of the learning models, so that they can classify computer programs according to the gender of the programmer more accurately than Naz's work. Another goal was to extend this approach to other sociological factors, in this case investigating the effects of the region of the programmer on the use of programming language.

In [21], Misek-Falkoff indicated that techniques from natural languages can be used to analyze computer programs. However, there are fewer linguistics variations in programming language than in natural language, because programming language imposes strict rules on the programmer while using the various elements of the programming language such as keywords, operators, variables, and comments. The use of the programming elements also depends on the problem to be solved more than the choices of the programmer. That is why, in order to find out the difference between male written and female written programs, we emphasized the layout and structure of the program rather than the use of different elements of the program. We selected the features for machine learning models based on the layout and structure of a program, such as number of blank lines, number of comment lines, number of mixed lines, and total number of functions used in a program.

In this research we used several techniques from the field of machine learning and statistics. Our dataset consisted of 160 C++ programs written by both male and female programmers. Among these 160 programs we had programmers from two regions: Canada and Bangladesh. We used a set of features from the programming language to create a numerical representation of computer programs. We used several machine learning techniques to categorize the computer programs according to the gender and region of the programmer. We used machine learning techniques in different stages of our experiment such as data processing, feature selection, model building, and model evaluation. We used seven machine learning algorithms to build seven learning models to categorize the programs. These seven models were the Bayes net, simple logistic, random forest, classification via regression, bagging, K star, and DTNB models. Using these seven models, we performed four different experiments with different numbers of features. In all the experiments we used the cross validation technique to evaluate the models' performance, because the cross validation technique reduces the risk of incorrect generalization of the dataset [6, 12].

In the first and second experiment, we categorized the programs based on the gender of the programmer. In the first experiment we used 16 features for the classification of programs. We were able to classify 83.1% of the programs accurately based on the gender of the authors. In the second experiment, we used an attribute evaluator to extract the best features from the feature set. We reduced the number of features from 16 to 6 in this experiment. Those six features were source code lines percentage, total blank lines, blank lines percentage, comment lines percentage, total functions, and region of the programmer. Using these features we were able to classify the same 83.1% of programs accurately based on the author's gender. The t-test showed that, from these six features four features (source code lines percentage, total blank lines, blank lines percentage, and comment lines percentage) were statistically significant.

In the third and fourth experiment, our goal was to classify the programs based on the region of the programmer. In the third experiment we used 15 features to classify the programs. We achieved the best result of 92.5% accuracy, which means 92.5% of the programs were accurately classified based on the region of the programmer. In the fourth experiment, we used an attribute evaluator to extract the best features and to reduce the number of features. We reduced the number of features from 15 to 7 for this experiment. Those 7 features were total comment lines, comment lines percentage, total blank lines, blank lines percentage, total commentary words, total functions, and average function line. The t-test showed that, from these seven features five features (total comment lines, comment lines percentage, total commentary words, total functions, and average function line) were statistically significant. We achieved the best accuracy of 89.4% from this experiment. This means 89.4% programs were accurately classified based on the region of the programmer.

We also performed statistical analysis of the features of our experiments. As the dataset of this study was small, therefore we tried to find out the significance of features by performing some statistical analysis. We also performed visual analyses on some programs to find out the difference of use of the statistically significant features among the different groups of programmers. The observations from the feature analysis are given below:

- Within our dataset of 160 programs we found that there were five features which were statistically significant while classifying the programs based on the gender of the programs. These five features were the source code line percentage, number of blank lines, blank lines percentage, comment line percentage, and mixed line percentage.

- From the visual analysis of the male written and female written programs, we found that the percentages of mixed line and source code line were higher in male written programs. On the other hand, the number of blank lines, percentage of blank lines, and percentage of comment lines were higher in female written programs.

- When we categorized the dataset of 160 programs according to the region of the programmer, we found that there were seven features which were statistically significant. These seven features were the number of comment lines, comment lines percentage, number of mixed lines, mixed lines percentage, total number of commentary words, and total number of functions.

- From the visual analysis of the Canadian written and Bangladeshi written programs, we observed that the number of comment lines, comment line percentage, number of mixed lines, mixed line percentage, number of total commentary words, and number of total functions were higher in Canadian written programs. In Bangladeshi written programs we noticed that the average number of function lines were higher than in Canadian written programs.

5.1 Future Research Directions

This research provides many possible directions for future research. Some of the potential research directions are given below:

- In this research we classified only programs written in C++ programming language. In the future we would like to carry out this research in other programming languages.
- Our dataset consisted of 160 programs. Our future plan is to enlarge our dataset by collecting programs from the open source program repository like Github. It would be interesting to implement the existing models using a large dataset.
- For building the machine learning model in this research, we used supervised learning. It would be interesting to implement the unsupervised learning [33] method for the classification of programs based on the gender and region of the programmer.
- In this study we used programs written by Canadian and Bangladeshi programmers to classify the programs based on the region of the programmer. In the future we would like to include programs written by programmers from several other countries of the world. Open-source online program repositories are our main attraction for this purpose.

- In our dataset, all the programs were written by university-level students. In the future we would like to carry out this research with the programs of experienced programmers. It would be intersecting to see how the level of experience of a programmer affects the use of programming language.
- For this research we selected a set of features without considering or doing any research on the connection of these features with the gender and region of a programmer. In the future, we would like to perform more detailed research on the features of a programming language as they connect to gender and region of the programmer.
- As a form of statistical analysis, we only performed the t-test on the features of this research. We would like to perform more statistical analysis such as principle component analysis to find out the significance of the features.

Bibliography

[1] IEEE standard for software productivity metrics. *Software Engineering Standards Subcommittee of the Technical Committee on Software Engineering of the IEEE Computer Society*, 1992.

[2] Survey monkey: Free online survey software & questionnaire tool, jun 2017.

[3] S. Argamon, J. Goulain, R. Horton, and M. Olsen. Vive la difference! text mining gender difference in french literature. *Digital Humanities Quarterly*, 3(2), 2009.

[4] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni. Gender, genre, and writing style in formal written texts. *Text - Interdisciplinary Journal for the Study of Discourse*, 23(3), 2003.

[5] S. Burrows and S. M. M. Tahaghoghi. Source code authorship attribution using n-grams. *School of Computer Science and Information Technology, RMIT University, Melbourne, Australia*, pages 32–39, 2007.

[6] R. P. L. Buse and W. Weimer. Learning a metric for code readability. *IEEE Transactions on Software Engineering*, 36(4):546–558, 2010.

[7] L. M. Connelly. t-tests. *Medsurg Nursing*, 20(6):341, 2011.

[8] G. Cox and A. McLean. *Speaking code: Coding as aesthetic and political expression*. The MIT Press, 2013.

[9] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78, 2012.

[10] P. Flach. *Machine learning: The art and science of algorithms that make sense of data*. Cambridge University Press, 2015.

[11] M. Hall and E. Frank. Combining naive bayes and decision tables. *In Proceedings of Twenty-First International Florida Artificial Intelligence Research Society Conference, AAAI Press, Coconut Grove, Florida, USA*, 2008.

[12] J. Han, M. Kamber, and J. Pei. *Data mining: Concepts and techniques*. Elsevier and Morgan Kaufmann, 2012.

[13] G. Holmes, A. Donkin, and I. Witten. Weka: A machine learning workbench. *Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*, 1994.

[14] P. Jaimai and H. S. Park. Representing unish grammars based on tree adjoining grammar formalisms. *Journal of Universal Language*, 4(1):1–16, 2003.

[15] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142, 1998.

[16] M. Koppel, S. Argamon, and A. Shimoni. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412, 2002.

[17] I. Krsul and H. E. Spafford. Authorship analysis: Identifying the author of a program. *Computers & Security*, 16(3):233–257, 1997.

[18] W. Labov. The linguistic variable as a structural unit. *Washington Linguistics Review*, 3:4–22, 1966.

[19] H. Langseth. Bayesian networks in reliability: The good, the bad and the ugly. *Advances in Mathematical Modeling for Reliability. IOS Press. Amsterdam, Netherland*, 2008.

[20] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[21] L. D. Misek-Falkoff. The new field of software linguistics. *ACM SIGMETRICS Performance Evaluation Review*, 11(2):35–51, 1982.

[22] T. M. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[23] Fariha Naz. Do sociolinguistic variations exist in programming? Master's thesis, University of Lethbridge, Canada, 2015.

[24] V. Nguyen, D. R. Sophia, T. Thomas, and B. Barry. A SLOC counting standard. *Center for Systems and Software Engineering, University of Southern California,*, 2007.

[25] W. O'Grady, J. Archibald, and F. Katamba. *Contemporary linguistics: An introduction*. 6th edition. Pearson Education Canada, 2011.

[26] J. C. Pane, A. Ratanamahatana, and B. A. Myers. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54(2):237–264, 2001.

[27] Park and Robert E. Software size measurement: A framework for counting source statements. *Software Engineering Institute, Carnegie Mellon University, Pennsylvania*, 1992.

[28] P. Pavlidis, I. Wapinski, and W. S. Noble. Support vector machine classification on the web. *Bioinformatics*, 20(4):586–587, 2004.

[29] D. A. Poole. *Artificial Intelligence: Foundations of computational agents*. Cambridge Univ Press, 2010.

[30] J. E. Rice, I. Genee, and F. Naz. Linking linguistics and programming: How to start? *in Proc. 25th Annual Psychology of Programming Interest Group Conference - PPIG*, 2014.

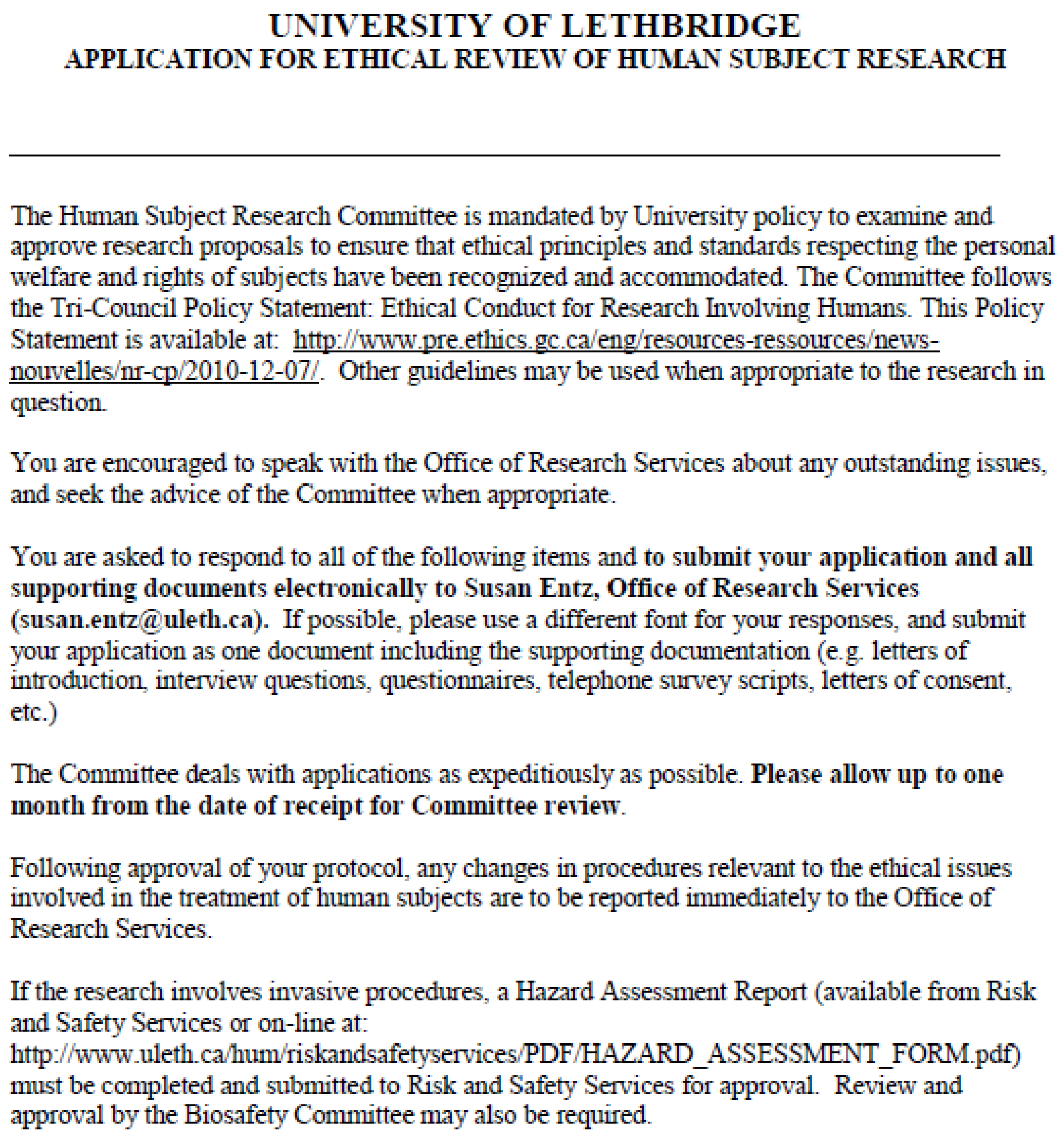
[31] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[32] P. Simon. *Too Big to Ignore: The Business Case for Big Data*. John Wiley & Sons, 2015.

[33] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining Practical Machine Learning Tools and Techniques*. 3rd edition. Elsevier and Morgan Kaufmann Publishers, 2011.

Appendix A

Ethical Review of Human Subject Research



UNIVERSITY OF LETHBRIDGE
APPLICATION FOR ETHICAL REVIEW OF HUMAN SUBJECT RESEARCH

The Human Subject Research Committee is mandated by University policy to examine and approve research proposals to ensure that ethical principles and standards respecting the personal welfare and rights of subjects have been recognized and accommodated. The Committee follows the Tri-Council Policy Statement: Ethical Conduct for Research Involving Humans. This Policy Statement is available at: http://www.pre.ethics.gc.ca/eng/resources-ressources/news-nouvelles/nr-cp/2010-12-07/. Other guidelines may be used when appropriate to the research in question.

You are encouraged to speak with the Office of Research Services about any outstanding issues, and seek the advice of the Committee when appropriate.

You are asked to respond to all of the following items and **to submit your application and all supporting documents electronically to Susan Entz, Office of Research Services (susan.entz@uleth.ca).** If possible, please use a different font for your responses, and submit your application as one document including the supporting documentation (e.g. letters of introduction, interview questions, questionnaires, telephone survey scripts, letters of consent, etc.)

The Committee deals with applications as expeditiously as possible. **Please allow up to one month from the date of receipt for Committee review.**

Following approval of your protocol, any changes in procedures relevant to the ethical issues involved in the treatment of human subjects are to be reported immediately to the Office of Research Services.

If the research involves invasive procedures, a Hazard Assessment Report (available from Risk and Safety Services or on-line at: http://www.uleth.ca/hum/riskandsafetyservices/PDF/HAZARD_ASSESSMENT_FORM.pdf) must be completed and submitted to Risk and Safety Services for approval. Review and approval by the Biosafety Committee may also be required.

SECTION A: GENERAL - *This information is collected under the authority of the* *Alberta Post-secondary Learning Act* *and will be used for administrative purposes associated with the ethical review of your human subject research protocol. It will be treated in accordance with the privacy protection provisions of Part 2 of the* *Alberta Freedom of Information and Protection of Privacy Act (*http://foip.alberta.ca/legislation/act/index.cfm). *Questions about the collection, use or disclosure of your personal information collected on this form can be directed to Susan Entz, Animal Welfare & Human Subject Research Coordinator, Office of Research Services, University of Lethbridge, Lethbridge, Alberta T1K 3M4, Phone: (403) 329-2747 and Email:* *susan.entz@uleth.ca*.

A1. Researcher/Applicant Information

Name: Jacqueline Rice
Department: Math and Computer Science
Telephone Number: 403 329-2783 (currently on sabbatical; email contact is preferable)
Email address: j.rice@uleth.ca

Are you: [✓ Faculty] ☐ Staff ☐ Graduate Student

☐ Undergraduate Student

A2. Co-Investigator's Information

Name:
Department:
Telephone Number
Email address:

Are you: ☐ Faculty ☐ Staff ☐ Graduate Student

☐ Undergraduate Student

A3. Student Thesis/Project Committee

a) Is this research for an undergraduate or graduate thesis/project? [✓ Yes] ☐ No

b) If yes, please provide the names, departments and phone numbers of your Committee members.

Name:	Department:	Telephone:
1.		
2.		

A4. Title of Project:

Indicate the title of your project. If this project is funded, the title should be the same as the title of your funded research.

Sociolinguistics in Computer Programming

A5. Location of Research

a) Indicate where the research will be conducted.

University of Lethbridge (Canada), Lancaster University (UK), and universities in Bangladesh (via online survey).

b) Does this project involve other centers, jurisdictions or countries? If so, please provide a list of the other groups who will be reviewing this protocol. (For example, the Lethbridge College Research Ethics Board must approve all posters to be posted on their campus.)

Yes. Lancaster University must also provide ethics approval for gathering data from their students.

A6. Start/End Dates of Research Involving Human Subjects

Please state the start and end dates of the research involving human subjects. **NOTE: Research involving human subjects cannot begin until Human Subject Research Committee approval has been received.**

Start date (dd/mm/yyyy): 01/03/2012 (or as soon as ethics approval is obtained)

End date (dd/mm/yyyy): 01/01/2018

A7. Funding

a) Is the project funded? ☐ Yes [✓ No]

☐ Funding approved – please specify source(s) :

1.
2.
3.

☐ Funding pending – please specify source(s):

1.
2.
3.

b) Is the project part of a course? ☐ Yes [✓ No]
Specify the course number and title:

SECTION B: DETAILS ABOUT THE PROJECT

B1. Purpose of Project

Provide a brief and clear statement of the context and objectives of the project, including the key questions and/or hypotheses of the project (in two pages or less).

The purpose of this study is to investigate the utilization of socio- and corpus-linguistics research approaches and practices in application to artificial (i.e. computer programming) languages. The goal of using these approaches is to identify information about the programmers or their sociological characteristics that could be used to improve current approaches to the development of computer programs. Futhermore, information may be extracted that could prove useful in computer science education, in much the way that applied linguistics techniques may be used in teaching English as a second language. Key questions include "Are gender differences apparent in how people use computer programming languages?" and "Do other sociological characteristics such as level of experience or first language spoken affect how people use computer programming languages".

B2. Description of Subjects

a) Indicate who you will recruit as potential participants in this study (e.g. undergraduates, school children, seniors) including any inclusion or exclusion criteria (e.g. over 65 years of age, self-identified as gay, speaks Blackfoot, speaks English), and the number of participants required.

The initial subjects will include students writing computer programs as part of their course work. I also intend to expand the study to include professionals who have submitted work to open source projects. I hope to have at least 30 participants from each group (30 students and 30 professionals). Although this study is examining how people use programming languages, part of the analysis may examine links to natural language, thus there is interest in whether the participants' first language is English or some other language. All participants will be of the age of majority (that is, able to legally give informed consent to participate in this study).

Participants will be recruited from the courses CPSC1620 and CPSC2620, currently being offered (Spring 2012). This should give me a large enough pool (I estimate that this is a total of approx. 80 students) that even if a significant portion of students choose not to participate, my goal of 30 should be reached.

2016 update: our data is significantly gender-skewed, and so we are still seeking additional participants to provide enough data for machine learning analysis. Since my new research assistant has connections to Bangladesh he intends to recruit participants from several universities of Bangladesh through an online survey. This should provide a large enough pool (we estimate 200 students) that even if a significant portion of students choose not to participate, our goal of 100 additional samples should be reached.

b) If the participants or facilities will be offered compensation or credit for participating in the research, provide details. Specify the amount, what the compensation is for, and how payment will be determined for subjects who do not complete the study.

No compensation will be offered.

B3. Recruitment of Subjects

a) Briefly describe how subjects will be recruited and who will do the recruiting. Researchers should avoid recruiting their own students. If this is unavoidable, researchers should provide the name of a research assistant, not associated with the course, who will do the recruiting and obtain consent when the researcher is not present.

If posters, newspaper advertisements, radio announcements or letters of invitation are being used, append these to this application.

Students will be recruited through their instructors or by myself. I will not be recruiting any students from my own courses; however the instructors of these courses will give a brief, oral introduction to my project, followed by which they will circulate to the course mailing lists a recruitment email from me. That is, I will write the email and send it to the instructors, so that I do not need access to the course mailing lists. The email will contain the information attached to this application (the "Project Information Sheet" and accompanying questionnaire), so that students are aware of exactly what is required of them before agreeing to participate. I have asked two of my colleagues (the instructors of CS1620 and CS2620) if they would be willing to do this for me, and they have agreed to do so pending approval of this ethics review.

Professionals will be recruited via email. Email addresses and contact information of the authors is often available as part of contributions to open source software (i.e. the information is made publically available as part of their contribution to the software project). I will look at a variety of well-known open-source projects such as Firefox and OpenOffice to determine whether the language used is appropriate for this study. For appropriate projects I will email individual contributors who have made their contact information public. The email will be similar to that sent to students, except of course any mention of grades and assignments will be removed and replaced with appropriate wording referring to their open source contributions.

My research assistant will contact the instructors of computer science departments in several Bangladeshi universities to request that they forward an email invitation their students that invites them to participate in the online survey. I will not be recruiting any students from my own courses; however the instructors will give a brief, oral introduction to my project, followed by which they will circulate to the course mailing lists a recruitment email (see attached). The email will contain the information attached to this application (the "Project Information Sheet" and accompanying online questionnaire link), so that students are aware of exactly what is required of them before agreeing to participate.

b) When and how will people be informed of the right to withdraw from the study? What procedures will be followed for people who wish to withdraw at any point during the study? What happens to the information contributed to the point of withdrawal?

All participants will be given a consent form and information letter explaining their right to decline participation or withdraw at any point. If a participant chooses to withdraw then any information related to that participant will be removed from the study and the data destroyed.

c) Indicate how subjects can obtain feedback on the research findings.

All publications based on this study will be made available to the subjects. My contact information will be provided in the letter of information, and participants can contact me to get an executive summary before publication if they so wish.

B4. Description of Research Procedures

Provide a summary of the design and procedures of the research. Provide details of data collection, and time commitment for the subjects, etc. NOTE: all study measures (e.g. questionnaires, interview guides, surveys, rating scales, etc.) must be appended to this application. If the procedures include a blind, indicate under what conditions the code will be broken, what provisions have been made for this occurrence, and who will have the code.

Data collection will consist of two parts: computer programs generated as part of their course work will constitute one part of the data, and a questionnaire will constitute the other part. The questionnaire should take less than 10 minutes to complete. The instructor of the course(s) will provide the programs for all students who agree to participate in the study. Participation will not have any impact on the students' grades. I am not connected in any way to the courses which I am suggesting provide the data, and there is no potential of conflict of interest between myself and the students, or between myself and the instructors.

The instructors of the courses will provide an oral overview of the study to their students, followed by which the information letter, consent form, and questionnaire will be circulated to all students via email. Students will have the opportunity to respond or not in a confidential setting. That is, because they are receiving the invitation via email, there will be no way that another individual (other than myself) will know if they have responded to the invitation. Only students who sign and submit the consent form will have their data used in the study.

Participants who agree to be involved in this study will respond with their agreement and answers to the questionnaire via email to me. When their email is received I will a) respond to the student with an acknowledgement of their participation in the study (this allows students to reconsider their agreement, should they wish to do so), b) process their answers (see following paragraph), c) delete the email, and then d) inform the instructors that I have received permission from the student to view their work. The instructors in question have agreed to then release the work to me. The acknowledgement email will remind the student that they can withdraw from the study at any time should they wish to do so.

Because it is essential to link the programs with the questionnaire data, when the questionnaires are received I will record the data in a database with a computer-generated unique identifier replacing the participant's name. This unique identifier will then be used as the file-name for the program submitted by that participant, thus removing any need for the names to be stored.

At this point in the research various options for analysis of the data have been proposed, although final decisions have not yet been made on how the data will be analysed. I anticipate the development of computer programs to count various metrics in the code, such as average line lengths, average identifier (word) lengths, numbers of semi-colons, and other computer-language related metrics. I am also planning to use programs such as WMatrix (http://ucrel.lancs.ac.uk/wmatrix/) and CQP (http://cqpweb.lancs.ac.uk/) for analysis of the natural language portions of the programs such as comments. Concordance of various tokens is of interest, as is examining whether identifiers are from the dictionary, or close to words from the dictionary.

B5. Privacy Protection

The next set of questions deals with anonymity and confidentiality. Refer to the brief descriptions below to assist you in answering these questions.

a) Anonymity** refers to the protection of the identity of participants.* ***Anonymity protection can be provided along a continuum, from "complete" to "no" protection, where complete protection means that no identifying information will be collected. *We remind applicants that university researchers should treat any personal information in accordance with the privacy protection provisions of Part 2 of the Alberta Freedom of Information and Protection of Privacy Act (*http://foip.alberta.ca/legislation/act/index.cfm*). If you have any questions about the collection, use, or disclosure of personal information under the Act, please contact Karen Mahar, FOIP Coordinator, The University of Lethbridge, 4401 University Drive, Lethbridge, Alberta T1K 3M4, Phone: (403) 380-1811, Email: karen.mahar@uleth.ca.*

1. Will the anonymity of the participants be protected?

☐ Yes (completely) ☒ Yes (partially) ☐ No

2. If "yes", explain how anonymity will be protected, and describe how this will be explained in the consent process.

All names will be removed from the data collected; however since I am collecting information such as gender and years of experience, this has the possibility to identify an individual if, for instance, there are very few women in a class. However individual information will not be identified in any publications in order to maintain the individual's anonymity.
Names must be initially given, and indeed identity is linked to responses that are given via email. However this is strictly for linking the questionnaire information to each computer program that is submitted, and as described in Section B4 these names will then be removed and the original data with the name attached will be destroyed.

3. If "no", justify why loss of anonymity is required, and describe how this will be explained in the consent process.

***b) Confidentiality** refers to the protection, access, control and security of the data and personal information.*

Confidentiality or non-disclosure agreements are recommended for all the individuals involved with the project (e.g. transcriptionists, research assistants, co-investigators, etc.).

1. How will confidentiality be protected and how will this be explained in the consent process? Specify which personnel will have access to the listing of names and study ID numbers as well as other study information collected (use job titles rather than individual names.) Provide details on the location, manner of storage, and the proposed retention period of the information collected.

The only personnel with full access to this information will be myself and a graduate student. Data will be stored on university computing resources, with the appropriate firewalls and other security measures in place. Data from this study will only be accessible through the use of a password, available only to myself, the graduate student and administrators of the computing resources. The graduate student and myself will also sign a non-disclosure agreement.

The primary role of the graduate student will be to design the software for processing the computer programs. Identification and computation of the metrics described in section B4 will be quite complex, and this will be a task well-suited to a graduate student. In addition, the graduate student must examine the other tools (WMatrix and CQPWeb) to see how the text (computer programs) will need to be processed in order to use these tools to process the data; this too will involve the development of fairly complex software. Finally, the graduate student will be aiding in analysis of the results, and refinement of the analysis approach. My goal is to link the style shown in the use of programming language to the sociological characteristics of the individuals, and the graduate student will be involved (with me) in tabulating the results of the metric measurements and determining whether there are any correlations with the data given in the questionnaires.

B6. Potential Risks and Benefits

To facilitate Human Subject Research Committee review and to determine whether the study involves more than minimal risk, please respond to the following questions. Does this project involve…	**Check those that apply**
1. Collection of data through invasive clinical procedures that are not required for normal patient care.	
2. Collection of data through noninvasive clinical procedures involving imaging or microwaves that are not required for normal patient care.	
3. Collection, use, or disclosure of health information or biological samples where the researcher is requesting that the requirement for informed consent be waived.	
4. Any procedures involving deception or incomplete disclosure of the nature of the research for purposes of informed consent.	
5. Any possibility that a breach of confidentiality could place subjects at risk of Criminal or civil liability or be damaging to subjects' financial standing, Employability or reputation.	
6. Research questions or procedures that might be expected to cause subject psychological distress, discomfort or anxiety beyond what a reasonable person might expect in day to day social interactions (e.g., questions that raise painful memories or unresolved emotional issues).	
7. Research questions that involve sensitive issues (e.g. sexual orientation or practices, etc.).	
8. Investigations in which there is a previous or existing relationship between the investigator and subjects (e.g., manager/employee, therapist/client, teacher/student).	
9. Investigations in which there is a conflict of interest between an investigator and the sponsor of the investigation.	
10. Any other non-therapeutic risks that arise from procedures not directly related to patient care.	

a) Outline any risks of potential physical or emotional harm or discomfort to the subjects, and describe how the anticipated benefits outweigh the potential risks.

No risks to the subjects are anticipated.

b) Indicate the steps taken to inform subjects of the possible consequences of releasing information in the public domain, and describe how subjects will be given an opportunity to review material before it is released.

Publication of this material will maintain anonymity of the individuals, thus no consequences to the subjects are anticipated. My contact information as well as an executive summary will be made available to participants who wish to review the results before publication.

c) Outline the exit strategy for termination of the study. Some types of research involve intense or lengthy contact between a researcher and the study participant(s), which may result in a close personal relationship, especially if the research itself involves matters close to the heart of participants. For this section, applicants should consider the possibility that a strategy may be required for participants who have difficulty in disengaging from the project after their role is completed or the project has terminated. If this does not apply to your research, please indicate n/a. If the research involves vulnerable populations, carefully clarify the boundaries between the researcher and participants.

No exit strategy is needed for this study.

B7. Obtaining Consent

Advise the Committee how informed consent will be obtained. The Tri-Council Policy Statement ensures that informed consent be obtained in writing from all subjects or, when appropriate from parents or legal guardians, unless there is a good reason for not doing so. If a consent form will be used, attach copies for the Committee. The Human Subject Research - Sample Letter of Consent is available from the Office of Research Services or our web site under Certification at: http://www.uleth.ca/rch/funding/online_forms.cfm. Please ensure that the reading level of the consent form is appropriate to the population involved.

a) Clearly detail who will be obtaining consent and the procedures for doing so. If appropriate, specify whether subjects will be randomly assigned to groups before or after consent has been attained.

The instructor of the course will provide an overview of the study, followed by which an information letter, consent form, and questionnaire (so that the students can see what is involved before agreeing to participate) will be circulated via email (to maintain confidentiality). Circulating the information via email is a more confidential approach as it allows the potential participant to respond (or not) while at home, or in some other private setting, rather than having them submit a paper response in a public setting. The researcher will be the collection point for the consent forms. Email responses will obviously not be anonymous, but as described in Section 4B, the email responses will be destroyed once the data has been anonymised and recorded. The consent form and accompanying letter are attached to this application.

b) If the subjects are not able/competent to give fully informed consent (cognitive impairment, age, etc.), or if there are significant power differences in operation (professor/student, employer/employee, political or economic minorities, etc.), please specify, and describe steps you will take to obtain free and informed consent. If participants are not competent to consent, specify who will consent on their behalf.

All subjects are anticipated as being able to give informed consent for this study.

c) Do any of the procedures include the use of deception or partial disclosure of information to subjects? If yes, provide a rationale for the deception or partial disclosure. Describe the procedures for debriefing the subjects.

No deception/partial disclosure is being used in this study.

d) **For the letter of consent/consent form:**

1. Extend an invitation to participate in the research project.

2. Provide a brief description of the project, including the purpose of the research, and a description of what is expected of the participant (e.g, the time commitment and the frequency of contact).

3. Describe the risks and discomforts (e.g. distress, inconvenience, psychological or social discomforts, fatigue, or physical safety issues). If the research project has the potential to identify upset, distressed or disturbed individuals, describe what arrangements will be made to assist these individuals, if need be.

4. Describe the benefits, including an explicit statement if there are no potential benefits to the participants (e.g. "You will not benefit directly from participation in this research").

5. Provide assurance of anonymity and confidentiality – this statement should describe the steps taken to ensure anonymity and confidentiality, and should include information regarding who will have access to the data collected. **NOTE: Participants should be advised that their privacy cannot be guaranteed when electronic surveys are used.**

6. Outline compensation for participation in the research project, if applicable.

7. Provide a non-coercive disclaimer – this statement should indicate that participation is voluntary, and that refusal to participate will not initiate prejudice, penalty or loss of benefits to which the subject is otherwise entitled.

8. Provide an option to withdraw – this statement should indicate that participants may discontinue participation at any time without prejudice, penalty or loss of benefits, and if they choose to withdraw, that they will be consulted regarding what should be done with their data.

9. Indicate the instances when the researcher may be obligated by law to report, to law enforcement or another agency, information revealed as a result of the research. **NOTE: Questions likely to result in reportable activities must be flagged for the respondent, and the respondent must be given the option to skip these questions.**

10. Provide a brief description of the anticipated use of the data.

11. Provide information on how participants will be informed of the results of the research.

12. Provide the name of the researcher, along with their institutional affiliation, and contact information for questions/clarification about the research project. Also include the following statement: "Questions regarding your rights as a participant in this research may be addressed to the Office of Research Services, University of Lethbridge (Phone: 403-329-2747 or Email: research.services@uleth.ca)."

e) **For telephone surveys**, informed consent should take place in the form of a verbal explanation of the above points. Append the "script" for this explanation to this application.

f) **For anonymous questionnaires**, include a cover letter that includes all the information normally provided in a consent form. Append a copy of this cover letter to this application.

B8. Continuing Review

Propose a process for continuing review if the research is ongoing. Continuing review should consist of, at least, the submission of a succinct annual status report. Notify the Committee when the research concludes.

The protection of human subjects will be assured in accordance with the Tri-Council Policy Statement or with other guidelines if these have been agreed upon as more appropriate.

[illegible]

Signature of Researcher/Applicant Date: April 22 2016

When the Researcher/Applicant is a student, the supervisor must sign the following statement:

"I have reviewed this application and I deem it ready to submit to the Human Subject Research Committee for review."

________________________	________________________
Signature of Supervisor	Date

(Revised January 20, 2012)

EMAIL INVITATION:

Dear student,

you are invited to participate in an online survey about the use of programming language. This survey is not required, and your submission will be anonymous. More information about the project is in the attached document, but if you have any questions please contact the primary investigator on this research, Dr. Jacqueline Rice (j.rice@uleth.ca).

In order to participate in the study please go to https://www.surveymonkey.com/r/68LKRN5.

Your time is very much appreciated.

Sincerely,

Dr. Jacqueline E. Rice
Associate Professor of Computer Science
University of Lethbridge

Project Information Sheet:
Sociolinguistics in Computer Programming

I would like to invite you to be a participant in a study combining the fields of linguistics and computer science.

The purpose of the study is to investigate how analytic tools from the field of linguistics might be applied to samples of computer programs. We hope this will allow us to identify how information about the programmers or their sociological characteristics might be used to improve current approaches to the development of computer programs. In addition, information from this work may also be used in improving computer science education practices, in much the way applied linguistics has been used in teaching English as a second language.

Should you agree to participate your participation in this study will consist of submitting a program of your choice, as well as completing a short follow-up questionnaire. The questionnaire should take less than 10 minutes to complete and is attached to this letter.

Your name will be removed from the data gathered during this study, and the data gathered will be used only for research and teaching publication purposes. There are no anticipated risks to this study. Anonymised data will be stored on a secure password-protected system to maintain participants' confidentiality. A student may be aiding me in this study, and she/he will also have access to the data that you submit. Both the student and I will sign a confidentiality agreement. Although there is no direct benefit to the participants, information from this study will be used to advance the field of computer science.

Participation is entirely voluntary, and you may withdraw at any time. Should you wish to withdraw the data you have submitted to us will be destroyed. To withdraw from the study, ask any further questions about the study, or to request an executive summary before the results are published please contact me:

Dr. Jackie Rice

j.rice@uleth.ca

Associate Professor, Dept. of Math and Computer Science, University of Lethbridge, Lethbridge, AB, Canada

Questions regarding your rights as a participant in this research may be addressed to the Office of Research Services, University of Lethbridge (Phone: 403-329-2747 or Email: research.services@uleth.ca).

The following pages, along with the project information sheet, will be presented to the participants who choose to complete the online survey:

Consent

YES/NO	I have read and understood the project information sheet.
YES/NO	I understand the nature of this study and agree to participate.
YES/NO	I understand that I can withdraw at any time.

1. What is your gender?

2. What is the name of your university?

3. Provide your valid email address.

4. What is your gender?

5. What was the first computer programming language that you learned?

   - C
   - C++
   - Java
   - Python
   - Other (please specify): ________________________

6. How many years of experience would you say you have as a programmer?

   - Less than 1
   - 1
   - More than 1 (please specify): ____________________

7. Provide your code here (a box for pasting in your code will be provided in the online survey).

End of survey.

Appendix B

Frequency of Features

Frequency of Features in Female Canadian Written Programs

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
1.	461	98.48	0	0.00	1.30	25
2.	153	62.09	45	29.41	8.50	8
3.	389	76.86	48	12.34	10.80	11
4.	217	75.12	32	14.75	7.37	7
5.	75	80.00	11	14.67	4.00	2
6.	186	74.73	12	6.45	13.44	7
7.	328	81.71	42	12.80	3.66	14
8.	157	48.41	38	24.20	21.02	6
9.	174	53.45	38	21.84	24.71	8
10.	82	57.32	23	28.05	14.63	5

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
11.	302	68.21	51	16.89	14.90	11
12.	172	97.09	0	0.00	2.33	9
13.	134	66.42	12	8.96	24.63	6
14.	314	76.43	23	7.32	14.65	11
15.	152	55.26	18	11.84	30.26	8
16.	739	56.02	114	15.43	23.27	17
17.	289	87.54	22	7.61	4.15	11
18.	83	49.40	15	18.07	32.53	5
19.	139	56.12	34	24.46	19.42	6
20.	262	80.53	51	19.47	0.00	3
21.	221	71.04	34	15.38	13.57	3
22.	121	74.38	31	25.62	0.00	8
23.	167	95.81	0	0.00	2.99	8
24.	96	71.88	27	28.12	0.00	6
25.	216	87.96	11	5.09	6.94	11
26.	587	62.52	108	18.40	16.87	15
27.	183	71.04	22	12.02	16.94	5
28.	89	67.42	23	25.84	6.74	6

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
29.	163	74.85	33	20.25	4.91	7
30.	74	64.86	17	22.97	12.16	5
31.	119	72.27	27	22.69	5.04	9
32.	273	87.91	24	8.79	3.30	12
33.	474	81.65	48	10.13	4.01	10
34.	205	87.80	0	0.00	11.71	9
35.	318	71.07	24	7.55	10.38	10
36.	88	72.73	15	17.05	10.23	1
37.	64	51.56	24	37.50	10.94	1
38.	320	61.88	38	11.88	26.25	18
39.	380	52.63	72	18.95	28.16	17
40.	82	70.73	16	19.51	9.76	1
41.	56	71.43	10	17.86	10.71	1
42.	109	62.39	34	31.19	6.42	6
43.	148	62.16	22	14.86	22.97	8
44.	172	48.26	46	26.74	25.00	9
45.	255	87.06	0	0.00	6.67	9
46.	105	86.67	13	12.38	0.95	8

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
47.	461	77.22	0	0.00	22.78	27
48.	166	91.57	0	0.00	8.43	3
49.	180	90.00	0	0.00	1.11	6
50.	260	100.00	0	0.00	0.00	3
TOTAL COUNT FOR ALL 50 FILES	**10960**	**73.61**	**1348**	**12.30**	**12.36**	**422**

Frequency of Features in Male Canadian Written Programs

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
1.	40	100.00	0	0.00	0.00	2
2.	281	88.61	0	0.00	8.54	12
3.	248	83.87	0	0.00	10.89	8
4.	222	91.89	0	0.00	2.25	14
5.	294	72.79	1	0.34	7.48	7
6.	893	93.84	0	0.00	4.48	8
7.	67	94.03	0	0.00	2.99	1
8.	108	63.89	0	0.00	17.59	4
9.	118	75.42	0	0.00	24.58	2
10.	133	76.69	0	0.00	15.79	3
11.	100	94.00	0	0.00	5.00	2
12.	181	95.58	0	0.00	3.87	5
13.	238	97.90	0	0.00	1.26	12
14.	131	70.99	1	0.76	12.98	5
15.	225	79.11	0	0.00	19.56	11
16.	244	97.95	0	0.00	2.05	11
17.	99	100.00	0	0.00	0.00	9

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
18.	328	64.02	0	0.00	18.90	11
19.	325	93.23	0	0.00	6.77	12
20.	256	97.27	0	0.00	2.73	11
21.	268	66.79	1	0.37	32.84	11
22.	173	72.83	0	0.00	27.17	6
23.	181	100.00	0	0.00	0.00	3
24.	103	75.73	0	0.00	24.27	4
25.	125	71.20	0	0.00	28.80	5
26.	94	77.66	0	0.00	22.34	4
27.	128	79.69	0	0.00	20.31	4
28.	193	73.58	0	0.00	21.76	5
29.	143	84.62	0	0.00	15.38	5
30.	122	71.31	0	0.00	26.23	4
31.	65	100.00	0	0.00	0.00	2
32.	41	73.17	0	0.00	26.83	1
33.	248	96.37	0	0.00	3.63	12
34.	181	85.64	0	0.00	14.36	11
35.	277	70.76	0	0.00	24.55	11

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
36.	302	76.82	0	0.00	23.18	11
37.	76	100.00	0	0.00	0.00	7
38.	181	80.66	0	0.00	17.13	5
39.	182	96.70	0	0.00	1.65	11
40.	148	81.76	0	0.00	14.19	9
41.	290	87.93	0	0.00	8.62	11
42.	141	93.62	0	0.00	6.38	11
43.	266	73.68	0	0.00	22.93	11
44.	116	100.00	0	0.00	0.00	7
45.	323	95.36	0	0.00	2.79	7
46.	151	90.73	0	0.00	3.31	11
47.	191	78.53	0	0.00	14.14	10
48.	144	86.11	0	0.00	9.72	5
49.	380	74.47	0	0.00	25.53	12
50.	205	86.34	0	0.00	6.34	10
TOTAL COUNT FOR ALL 50 FILES	**9,969**	**84.65**	**3**	**0.03**	**12.03**	**376**

Frequency of Features in Female Canadian Written Programs

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
1.	287	80.84	55	19.16	0.00	5
2.	206	90.78	19	9.22	0.00	8
3.	106	92.45	5	4.72	2.83	4
4.	101	100.00	0	0.00	0.00	1
5.	290	80.69	56	19.31	0.00	5
6.	97	87.63	11	11.34	1.03	1
7.	649	86.29	89	13.71	0.00	5
8.	129	86.82	17	13.18	0.00	1
9.	107	86.92	14	13.08	0.00	2
10.	1,677	84.68	237	14.13	1.19	20
11.	102	76.47	24	23.53	0.00	1
12.	297	95.62	11	3.70	0.67	6
13.	105	86.67	14	13.33	0.00	1
14.	101	88.12	12	11.88	0.00	7
15.	91	81.32	7	7.69	7.69	1
16.	295	75.93	41	13.9	10.17	13
17.	90	72.22	14	15.56	12.22	2

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
18.	178	82.58	31	17.42	0	18
19.	95	90.53	9	9.47	0	6
20.	124	83.06	21	16.94	0	1
TOTAL COUNT FOR ALL 20 FILES	**5,127**	**85.10**	**687**	**13.40**	**1.44**	**108**

Frequency of Features in Male Bangladeshi Written Programs

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
1.	196	45.41	106	54.08	0.00	1
2.	115	91.30	10	8.70	0.00	1
3.	116	87.93	13	11.21	0.86	7
4.	152	69.74	33	21.71	0.00	6
5.	312	59.94	125	40.06	0.00	1
6.	236	98.31	4	1.69	0.00	5
7.	220	86.36	21	9.55	0.45	11
8.	137	86.13	4	2.92	10.95	1
9.	140	80.71	24	17.14	0.00	1
10.	104	85.58	5	4.81	9.62	4
11.	311	93.57	20	6.43	0.00	1
12.	56	69.64	13	23.21	1.79	3
13.	156	77.56	30	19.23	0.64	6
14.	100	79.00	11	11.00	10.00	6
15.	118	70.34	30	25.42	4.24	6
16.	143	91.61	12	8.39	0.00	5
17.	536	79.48	53	9.89	10.26	10

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
18.	101	89.11	3	2.97	7.92	1
19.	109	76.15	17	15.60	2.75	3
20.	110	78.18	24	21.82	0.00	6
21.	98	78.57	11	11.22	10.20	1
22.	135	100.00	0	0.00	0.00	1
23.	119	93.28	7	5.88	0.84	1
24.	136	80.88	22	16.18	2.94	5
25.	107	85.05	16	14.95	0.00	7
26.	120	65.00	16	13.33	10.00	6
27.	124	83.06	21	16.94	0.00	1
28.	647	55.33	269	41.58	0.93	7
29.	101	88.12	12	11.88	0.00	7
30.	125	84.00	18	14.40	0.00	7
31.	227	91.19	20	8.81	0.00	2
32.	153	100.00	0	0.00	0.00	1
33.	119	73.95	29	24.37	0.84	8
34.	101	88.12	12	11.88	0.00	7
35.	198	82.83	34	17.17	0.00	13

Programs	Total Number of Lines	Source Code Lines Percentage	Total Number of Blank Lines	Blank Lines Percentage	Comment Lines Percentage	Total Number of Functions
36.	214	95.33	1	0.47	2.34	7
37.	227	91.19	20	8.81	0.00	2
38.	244	95.08	12	4.92	0.00	1
39.	136	86.03	14	10.29	3.68	9
40.	373	88.47	42	11.26	0.27	1
TOTAL COUNT FOR ALL 40 FILES	**7172**	**80.98**	**1134**	**15.81**	**2.16**	**179**

Appendix C

Relationship of Features

In the following Tables, we present four correlation matrix computed for male written, female written, Canadian written and Bangladeshi written programs. The matrix is symmetric, so the diagonal values in the matrix are always 1. These values represent that there is a strong relationship between the same features which is of no interest. The values above and below the diagonal are the same. Thus, we report the values only below the diagonal.

Correlation Based on Raw Frequency of Features in Male-Authored Programs

Features	Total Lines	Total Source Code Lines	Source Code Lines Percent -age	Total Blank Lines	Blank Lines Percent -age	Total Comment Lines	Total Comment Lines Percent -age	Total Mixed Lines	Mixed Lines Percent -age	Total Comme -ntary Words	Physical Executa -ble Line of Code	Logical Executa -ble Line of Code	Total Function	Total Function Lines	Average Function Lines
Total Lines	1														
Total Source Code Lines	**0.98**	1													
Source Code Lines Percentage	0.00	0.18	1												
Total Blank Lines	0.06	-0.01	-0.33	1											
Blank Lines Percentage	0.01	-0.05	-0.30	**0.93**	1										
Total Comment Lines	0.42	0.25	-0.71	0.20	0.12	1									
Total Comment Lines Percentage	-0.03	-0.19	-0.90	0.14	0.11	**0.77**	1								
Total Mixed Lines	0.29	0.19	-0.40	0.41	0.34	0.17	0.01	1							
Mixed Lines Percentage	0.07	-0.03	-0.47	0.44	0.45	0.07	0.03	**0.89**	1						
Total Commentary Words	0.47	0.30	-0.68	0.26	0.18	**0.81**	**0.58**	**0.53**	0.39	1					
Physical Executable Line of Code	**0.99**	**1.00**	0.13	0.03	-0.02	0.26	-0.18	0.28	0.06	0.35	1				
Logical Executable Line of Code	**0.98**	**0.98**	0.11	0.02	-0.03	0.28	-0.16	0.29	0.06	0.38	**0.99**	1			
Total Functions	0.48	0.46	0.10	0.01	-0.03	0.29	-0.12	0.14	0.02	0.27	0.46	0.47	1		
Total Function Lines	**0.96**	**0.97**	0.09	-0.07	-0.07	0.32	-0.10	0.17	-0.01	0.35	**0.97**	**0.96**	0.42	1	
Average Function Lines	**0.69**	**0.72**	0.09	-0.09	-0.06	0.10	-0.08	0.07	-0.04	0.15	**0.71**	**0.70**	-0.21	**0.77**	1

Correlation Based on Raw Frequency of Features in Female-Authored Programs

Features	Total Lines	Total Source Code Lines	Source Code Lines Percent -age	Total Blank Lines	Blank Lines Percent -age	Total Comment Lines	Total Comment Lines Percent -age	Total Mixed Lines	Mixed Lines Percent -age	Total Comme -ntary Words	Physical Executa -ble Line of Code	Logical Executa -ble Line of Code	Total Function	Total Function Lines	Average Function Lines
Total Lines	1														
Total Source Code Lines	**0.94**	1													
Source Code Lines Percentage	0.11	0.41													
Total Blank Lines	**0.58**	0.32	-0.52	1											
Blank Lines Percentage	-0.36	-0.55	-0.75	0.45	1										
Total Comment Lines	**0.72**	0.47	-0.43	**0.65**	-0.05	1									
Total Comment Lines Percentage	0.13	-0.11	-0.75	0.30	0.16	**0.67**	1								
Total Mixed Lines	**0.58**	0.47	-0.04	0.39	-0.21	0.44	0.04	1							
Mixed Lines Percentage	0.29	0.26	0.04	0.07	-0.29	0.16	-0.03	**0.86**							
Total Commentary Words	**0.69**	0.44	-0.36	**0.66**	-0.02	**0.89**	0.49	0.67	0.33	1					
Physical Executable Line of Code	**0.94**	**1.00**	0.39	0.34	-0.54	0.48	-0.11	**0.53**	0.31	0.48	1				
Logical Executable Line of Code	**0.94**	**0.97**	0.36	0.40	-0.50	0.48	-0.11	**0.58**	0.35	**0.52**	**0.98**	1			
Total Functions	**0.75**	**0.74**	0.10	0.26	-0.34	**0.60**	0.20	0.22	0.08	0.42	**0.72**	**0.67**	1		
Total Function Lines	**0.95**	**0.97**	0.31	0.46	-0.45	**0.51**	-0.08	**0.55**	0.30	**0.52**	**0.98**	**0.98**	**0.65**	1	
Average Function Lines	0.07	0.15	0.28	-0.01	-0.14	-0.15	-0.28	0.04	-0.01	-0.08	0.15	0.13	-0.40	0.23	1

Correlation Based on Raw Frequency of Features in Canadian-Authored Programs

Features	Total Lines	Total Source Code Lines	Source Code Lines Percent-age	Total Blank Lines	Blank Lines Percent-age	Total Comment Lines	Total Comment Lines Percent-age	Total Mixed Lines	Mixed Lines Percent-age	Total Comme-ntary Words	Physical Executa-ble Line of Code	Logical Executa-ble Line of Code	Total Function	Total Function Lines	Average Function Lines
Total Lines	1														
Total Source Code Lines	**0.95**	1													
Source Code Lines Percentage	0.03	0.28	1												
Total Blank Lines	0.38	0.15	-0.56	1											
Blank Lines Percentage	-0.13	-0.27	-0.68	**0.69**	1										
Total Comment Lines	**0.60**	0.36	-0.50	0.45	0.01	1									
Total Comment Lines Percentage	0.04	-0.15	-0.72	0.14	0.05	**0.69**	1								
Total Mixed Lines	0.39	0.30	-0.14	0.09	-0.18	0.28	0.03	1							
Mixed Lines Percentage	0.12	0.07	-0.12	-0.11	-0.25	0.08	0.02	**0.87**	1						
Total Commentary Words	**0.60**	0.37	-0.41	0.42	-0.04	**0.86**	**0.51**	**0.56**	0.31	1					
Physical Executable Line of Code	**0.95**	**1.00**	0.26	0.15	-0.28	0.37	-0.15	0.38	0.14	0.41	1				
Logical Executable Line of Code	**0.95**	**0.98**	0.23	0.19	-0.26	0.38	-0.14	0.40	0.15	0.44	**0.98**	1			
Total Functions	0.64	**0.59**	0.04	0.23	-0.12	0.50	0.06	0.15	0.02	0.37	**0.58**	**0.56**	1		
Total Function Lines	**0.95**	**0.97**	0.17	0.24	-0.18	0.41	-0.09	0.30	0.07	0.43	**0.97**	**0.97**	**0.53**	1	
Average Function Lines	0.33	0.41	0.16	0.03	-0.04	-0.06	-0.19	0.05	-0.04	0.00	0.40	0.38	-0.33	0.48	1

Correlation Based on Raw Frequency of Features in Bangladeshi-Authored Programs

Features	Total Lines	Total Source Code Lines	Source Code Lines Percent-age	Total Blank Lines	Blank Lines Percent-age	Total Comment Lines	Total Comment Lines Percent-age	Total Mixed Lines	Mixed Lines Percent-age	Total Comme-ntary Words	Physical Executa-ble Line of Code	Logical Executa-ble Line of Code	Total Function	Total Function Lines	Average Function Lines
Total Lines	1														
Total Source Code Lines	**0.98**	1													
Source Code Lines Percentage	-0.05	-0.10	1												
Total Blank Lines	**0.79**	**0.69**	-0.12	1											
Blank Lines Percentage	0.12	0.00	-0.23	**0.60**	1										
Total Comment Lines	0.37	0.33	0.15	0.20	-0.09	1									
Total Comment Lines Percentage	-0.06	-0.10	0.29	-0.12	-0.17	**0.70**	1								
Total Mixed Lines	0.07	-0.01	-0.10	0.31	0.27	0.08	0.10	1							
Mixed Lines Percentage	-0.08	-0.12	-0.09	0.01	0.15	0.04	0.16	**0.86**	1						
Total Commentary Words	0.06	0.00	0.16	0.07	0.01	**0.66**	**0.74**	0.44	0.47	1					
Physical Executable Line of Code	**0.99**	**1.00**	-0.03	**0.69**	0.00	0.34	-0.08	0.00	-0.11	0.03	1				
Logical Executable Line of Code	**0.99**	**0.99**	-0.03	**0.71**	0.02	0.32	-0.09	0.01	-0.11	0.01	**1.00**	1			
Total Functions	0.50	0.50	-0.13	0.36	0.07	0.33	0.02	0.13	0.05	0.22	0.49	0.48	1		
Total Function Lines	**0.90**	**0.89**	-0.02	**0.72**	0.08	0.25	-0.07	0.09	-0.04	0.06	**0.90**	**0.89**	**0.53**	1	
Average Function Lines	0.12	0.11	0.21	0.11	-0.02	-0.09	-0.09	-0.10	-0.15	-0.12	0.12	0.11	-0.42	0.22	1