CLASSIFICATION OF COMPUTER PROGRAMMING CONTEST PROGRAMS BASED ON GENDER, REGION AND SOFTWARE METRICS

SARA BINTE ZINNAT
Bachelor of Science, Military Institute of Science and Technology, 2013

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

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

CLASSIFICATION OF COMPUTER PROGRAMMING CONTEST PROGRAMS BASED ON GENDER, REGION AND SOFTWARE METRICS

SARA BINTE ZINNAT

Date of Defence: September 09, 2021

Dr. J. Rice Thesis Supervisor	Professor	Ph.D.
Dr. J. Morris Thesis Examination Committee Member	Professor	Ph.D.
Dr. J. Zhang Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. W. Osborn Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

To the almighty

Abstract

This research focuses on determining the effect of sociolinguistics characteristics (particularly, gender and region) on computer programs. Previous studies have demonstrated the use of machine learning techniques to analyze the relationship between sociolinguistics features and programming language. We collected C++ programs from an open source programming contest website. The features were calculated based on three software metrics: lines of code, cyclomatic complexity and Halstead metrics. Using five machine learning algorithms we trained several models and performed experiments to compare their performance. To investigate the significance of the features, we also carried out statistical and correlation analysis. As indicated by the experimental results, our models successfully predicted the gender of the programmers with 91.7% accuracy when programmers solved the same problems. When the programmers solved different problems, the model achieved an accuracy of 86.4%. Our models also efficiently classified the region of the programmer with 75.2% accuracy.

Acknowledgments

All praise be to the Almighty.

First of all, I would like to thank Dr. Rice for giving me the opportunity to work under her supervision. I feel really blessed to learn from such a nice and humble person. Thank you, Dr. Rice for your continuous support, perceptive comments and guidelines that helped me to determine my goals in every part of my M. Sc. program.

I would like to thank my committee members Dr. Joy Morris and Dr. John Zhang for their support and valuable suggestions.

I am thankful to Alberta Innovates- Data Enabled Innovation (AI-DEI) program for funding this research. My appreciation also goes to Dr. Rice and the School of Graduate Studies for their financial support.

From the core of my heart I would like to thank my parents Md. Azharul Hoque Chowdhury and Zinnat Mahal, my siblings Ali and Afrin, Tamanna, my two lovely nieces Ohi and Namirah for their prayers and support.

My heartfelt appreciation goes to all my faculty members, my fellow Sociolinguistics research group members and every person who encouraged me throughout my M.Sc. program.

Last but not the least I would like to thank my husband Deen Md. Abdullah and my daughter Shifa Abdia Deen for powering and cheering me up in hard times. I can never thank you with a few words.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Sociolinguistics is the study of language and other social factors including gender, region, and class [43]. The use of a language may vary depending on social variables such as gender, region, age, social or economic status of the human users. Linguistic variables are correlated with social variables and can be analyzed to study sociolinguistics variations [32]. There exist a wide variety of languages, including natural languages and programming languages. According to Wardhaugh, "Natural language is a rich system that allows its users to communicate with each other by following some grammar rules". Examples include English, Bengali, Hindi, and French [43]. In contrast, a programming language is defined as a formal language that is employed to provide instructions to computers to perform specific tasks [22]. Several studies have demonstrated gender-based linguistic variations in textual documents [3, 5, 16]. Machine learning techniques are widely used to categorize text documents [2, 3, 19]. Recently, gender-based analysis of computer programs has also become an area of interest for researchers [16, 17, 32, 35, 36].

1.1 Motivation and Hypothesis

To examine the effect of gender on computer programs, Fisher et al. [9] analyzed computer programming contest problems and determined several factors that influenced male and female contestants' participation while solving a contest problem. The authors claimed that female participants solved fewer problems than male participants as the female participants tended to develop efficient solutions rather than rapidly obtaining a working solution.

Rafee [35] examined C++ computer programs to determine gender and region-based sociolinguistics variations. The author generated several features (a feature is a measurable unique characteristic that can be observed to predict labels) based on the software metric lines of code and suggested that male programmers use fewer comments and blank lines than female programmers. All of these studies have helped us to generate the following research questions: Do sociolinguistic variations exist in the computer programs of computer programming contests? Do male and female contestants write programs differently to solve a particular programming contest problem?

To answer our research questions, we analyze computer programming contest data to investigate the effect of sociolinguistics variables, gender and region on the computer programs. To solve any programming contest problem, a programmer has to follow a series of steps. The first step is to identify the problem. Next, some brainstorming should be done to figure out the possible ways to start. The next step is to determine the best possible solution and possibly write some algorithms or pseudo code. The final step is to write the program and debug if needed [21]. The entire process of solving a programming contest problem differs from person to person. For example, one programmer may solve a programming contest problem by writing 100 lines of code. In contrast, another programmer may solve the same problem by writing 80 lines of code. In the software industry, the development of a piece of software may require the involvement of many people: analysts, programmers, developers, testers and users. Thus, the software development process generally follows a Software Development Life Cycle (SDLC) model, for example the Waterfall model or an Iterative model [39]. While solving any programming contest problem a contest programmer would not follow any SDLC model because of time constraints. As well, they may be the only contributor. Thus, we hypothesize that computer programs from programming contests can be analyzed to find differences in programming styles of individual programmers, particularly male and female programmers. Since programmers from different parts of the world participate in programming contests, programming contest data can also be a

potential data source to determine the effect of region on the programmer.

In [35], Rafee considered only one software metric, lines of code. Based on this he calculated several features and executed his experiments. However, the author did not analyze information such as the cyclomatic complexity and effort of the programs.

Several studies, e.g. [26], have demonstrated a strong correlation between lines of code and effort analysis, and a stronger correlation between lines of code and complexity analysis. Therefore, in our research, we aim to analyze contest programs and generate features based on three software metrics: lines of code [7], McCabe's cyclomatic complexity [25], and Halstead metrics [12]. We hypothesize that this additional information may offer better results or tell us more about the effort, the time required to complete the program, and the number of instances of decision logic that may vary depending on the gender of the programmer. These metrics are used in analysis to investigate the gender-based and region-based sociolinguistics differences of the programming contest programmers through various machine learning approaches.

Although Rafee carried out a similar work, that work was performed with a smaller data set using only one software metric. We aim to repeat the work with larger data sets and use three different software metrics. In this research our aim is to identify the underlying information whether males and females or people from different regions have different approaches to programming rather than just the accuracy. Therefore, to identify the different programming approaches we plan to use additional features that may help us to get the underlying information about the approaches of programming.

1.2 Contributions

In this research, we investigate computer programming contest programs to find differences in the programming styles of the computer programmers. We attempt to classify the computer programs based on the two sociolinguistics variables: gender, and region. The classification is based on analysis of features within the programs based on three soft-

ware metrics: lines of code, McCabe's cyclomatic complexity, and Halstead metrics. The machine learning tool *WEKA* [45] is used to classify the computer programs based on the gender and region of the programmers. The contributions of our research are as follows:

- In this research, we analyze the programming styles of male and female programmers. We also determine which programming features vary according to the region of the programmers. Therefore, the results of our analysis can potentially benefit researchers who aim to investigate the thought process of computer programmers.
- We conduct our research based on professional software metrics which are used to measure the readability and standard of programs. Thus, our sociolinguistics research could be used to develop a tool/system to guide contest programmers in developing professional programming styles.
- We determine which features are essential for gender-based and region-based classification. Our work may assist project managers in software companies to group programmers who have analogous programming styles such as preference for fewer lines of code, more comments in the programs, or similar logic implementation patterns. Alternatively, our research may help managers to group programmers preferring different programming styles, which may be beneficial to achieve a successful product.

1.3 Organization of Thesis

In Chapter 1, the motivation and hypothesis of this research are discussed with a brief introduction.

Chapter 2 provides the definitions and details on sociolinguistics, three software metrics, machine learning, WEKA, classification algorithms or classifiers, validation technique, and evaluation methods. The related work is also discussed in this chapter.

Chapter 3 describes our methodology for data collection, data preparation, and feature selection tasks.

Chapter 4 discusses the programming environment, all the experiments with numerical results, and the threats to the validity of this research.

Chapter 5 presents an analysis of the features that are highly relevant to identify different programming styles of the programming contest programmers.

Chapter 6 concludes this thesis with a summary of the research and suggestions for future research.

Chapter 2

Background and Literature Review

2.1 Sociolinguistics

The study of sociolinguistics examines social variables such as language, gender, age, region, education, and ethnicity to find connections between language and society [43]. Holmes [13] described how the usage of words varies according to the gender of the speakers. The author also provided examples that demonstrate the regional-based linguistic and pronunciation variations among English-speaking nations. According to Labov [20], social variables such as gender, region, age, socio-economic class, and ethnicity may influence an individual's linguistic expression. Several studies have explored the effect of sociolinguistic variables on written documents including [2, 3, 5, 14, 16]. For example, Ishikawa investigated university students writing under controlled conditions to determine text-based linguistic variations [14]. In her study the author provided two specified topics and word limitations to the participants who were asked to write an argumentative essay. The results showed that male participants used more nouns to describe facts, whereas female participants used more pronouns, intensifiers, and modifiers to express their opinions.

2.2 Software Metrics

Software metrics are "Measures of evaluating some characteristics or attributes of a software entity that are computable or countable" [15]. To determine the quality of a piece of software, various standard metrics are used. Each metric measures some properties of the software which are known as features. In machine learning, the performance of a model

highly depends on the selection of appropriate features. Irrelevant features may negatively affect the performance of the machine learning model. In our research, we consider three software metrics: Lines of Code [7], McCabe's cyclomatic complexity [25] and Halstead metrics [12] to calculate the relevant features. Details of these metrics are given in the sections below.

2.2.1 Lines of code

Lines of code is one of the simplest and most widely used software metrics. Fenton et al. [7] defined lines of code as a total of non commented source statements and commented source statements. Non commented source statements could be any statement in the program, including program headers, declarations, executable and non executable statements, whereas commented source statements could be comments and blank lines.

The features that we use for our research based on lines of code are defined as follows [35]:

- **Total Lines of code (LOC) -** The total number of lines or statements including source code lines, comments and blank lines is referred to as total lines of code.

- **Comments -** The lines in source code that are useful to provide information about the source code are termed as comments. All lines provide information, but comments are not executable.

- **Physical executable source lines of code (SLOCP) -** Physical executable source lines of code is defined as the total number of lines in the program excluding the comment lines. Blank lines are included in SLOCP. The equation for calculating SLOCP is as follows:

$$SLOCP = LOC - Comments \tag{2.1}$$

- **Logical executable lines of code (LLOC) -** Logical executable lines of code is defined as the total number of lines in the program excluding comments and blank lines.

The value of LLOC can be calculated, using Equation 2.2:

$$LLOC = LOC - Comments - blank\ lines \tag{2.2}$$

- **Blank lines -** The total number of lines that are empty and do not contain any characters are referred to as blank lines.

Let us consider the sample program shown in Listing 2.1:

- Total number of lines of code (LOC) = 5,
- total number of comments = 1, and
- total number of blank lines = 0.

From Equation 2.1 SLOCP is calculated as

$$\begin{aligned} SLOCP &= 5 - 1 \\ &= 4. \end{aligned}$$

Using Equation 2.2, the value of LLOC is calculated as

$$\begin{aligned} LLOC &= 5 - 1 - 0 \\ &= 4. \end{aligned}$$

```
1 // This is a comment
2 for (i = 0; i < 100; i++)
3 {
4     cout<<"This is a sample program";
5 }
```

Listing 2.1: Sample program for LOC metrics example.

2.2.2 McCabe's cyclomatic complexity

According to McCabe [25], cyclomatic complexity is a software complexity metric that measures the amount of decision logic in a single software module. Cyclomatic complexity is based on the structure of the software's control flow graph, and control flow graphs express the logic structure of software modules. Watson et al. [44] defined a module as "A single function that could be used as a design component via a call or return mechanism, and each module has a single entry as well as an exit point". The authors [44] provided an example of a control flow graph based on the C programming language. In C, a function is considered to be a module. Each module has a corresponding flow graph which consists of nodes and edges to express the logic structure of that module or function. The nodes correspond to the computational statements, and the edges represent the transfer of control between nodes. Each possible execution path of a software module has a corresponding path from the entry node to the exit node in the module's control flow graph [44].

In order to determine the cyclomatic complexity of each module, the flow graph G of the module is drawn. For each connected component there is only one single entry node and one exit node for which 2 is added in the equation of the cyclomatic complexity. Then the cyclomatic number is found using Equation 2.3

$$C(G) = e - n + 2p \tag{2.3}$$

where

- $C(G)$ is the cyclomatic number of graph G,
- e is the number of edges in the graph,
- n is the number of nodes, and
- p is the number of the strongly connected components [44].

The value of $C(G)$ for each module should be less than 10, otherwise the module is considered to be complex. Since complex modules are generally more difficult to maintain and modify, they tend to have more errors and bugs [15].

For example, we can consider the sample program shown in Listing 2.2. The program compares two integer values and returns the greater value.

```
int compare(int x, int y)
{
   if (x > y)
      int z = x;
   else
      z = y;
   return z;
}
```

Listing 2.2: Sample program for cyclomatic complexity example.

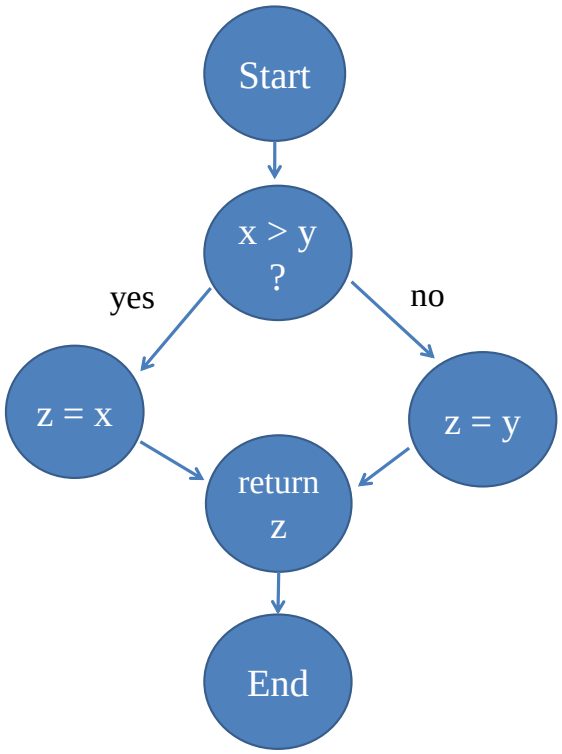

Figure 2.1: Control flow graph for the sample program in Listing 2.2.

If we consider the control flow graph in Figure 2.1, excluding the start and end nodes,

the number of nodes n is 4, the number of edges e is 4 and the number of connected components p is 1 since we are considering one control flow graph of a single program.

The cyclomatic complexity for the sample program shown in Listing 2.2 can be calculated using Equation 2.3.

$$\begin{aligned} C(G) &= 4-4+2\times 1 \\ &= 2. \end{aligned}$$

The sample program shows a 'compare' function where two executable paths for each instances of decision logic exist. Hence, the cyclomatic complexity number for the 'compare' function is 2.

2.2.3 Halstead metrics

Using Halstead metrics a program is considered to be a collection of tokens, and the tokens are classified as either operators or operands [15]. The goal of the Halstead metrics is to measure various attributes of a program; for example, program length, vocabulary, volume, level, difficulty, effort and required programming time. Halstead provided formulas to measure these attributes [12, 15].

In order to calculate Halstead complexity metrics, we use the following definitions:

- n_1= the number of distinct operators,
- n_2= the number of distinct operands,
- N_1 = the total number of operators,
- N_2 = the total number of operands,

Using these definitions, the Halstead metrics can be calculated as follows:

Length of the program is defined as

$$N = N_1 + N_2. \tag{2.4}$$

Calculated estimated program length is defined as

$$\hat{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2. \tag{2.5}$$

Vocabulary of the program is defined as

$$n = n_1 + n_2. \tag{2.6}$$

The volume or size measure of the program is defined as

$$V = N \times \log_2 n. \tag{2.7}$$

The difficulty of the program is defined as

$$D = \frac{n_1}{2} \times \frac{N_2}{n_2}. \tag{2.8}$$

Finally, the effort to implement defined as

$$E = D \times V. \tag{2.9}$$

During Halstead metrics calculation the function names are ignored, but elements inside the functions are considered for the calculations.

Listing 2.3 shows a program that averages three integer values. As shown in Tables 2.1 and 2.2, the number of operators and operands are calculated for the sample program shown in Listing 2.3.

```
int main()
{
    int a, b, c, avg ;
    cout  <<  " a = " ;
    cin >> a ;
    cout << " b = " ;
    cin >> b ;
    cout << " c = " ;
    cin >> c ;
    avg = ( a + b + c ) / 3 ;
    cout <<  " avg = " << avg ;
    return 0 ;
}
```

Listing 2.3: Sample program for Halstead metrics example.

Table 2.1: Number of operators for the sample program in Listing 2.3.

operators	**count**	**operators**	**count**
int	2	“”	4
cout	4	;	10
cin	3	,	3
()	2	=	5
{}	1	+	2
<<	5	/	1
>>	3	return	1

In this example,

- the number of distinct operators $n_1 = 14$,
- the number of distinct operands $n_2 = 6$,
- the total number of operators $N_1 = 46$, and
- the total number of operands $N_2 = 18$.

Replacing the values of n_1 and n_2 in Equation 2.4 gives us

$$N = 46 + 18 = 64.$$

Table 2.2: Number of operands for the sample program in Listing 2.3.

operands	**count**
a	4
b	4
c	4
avg	4
3	1
0	1

From Equation 2.5 the calculated estimated program length is

$$\begin{aligned}\hat{N} &= 14\log_2 14 + 6\log_2 6 \\ &= 53.30 + 15.51 \\ &= 68.81.\end{aligned}$$

From Equation 2.6 the vocabulary of the program is

$$\begin{aligned}n &= 14 + 6 \\ &= 20.\end{aligned}$$

From Equation 2.7 the volume of the program is

$$\begin{aligned}V &= 64 \times \log_2 20 \\ &= 276.60.\end{aligned}$$

Replacing the values of n_1, n_2 and N_2 in Equation 2.8 gives the difficulty of the program:

$$\begin{aligned}D &= \frac{14}{2} \times \frac{18}{6} \\ &= 21.\end{aligned}$$

Lastly, from Equation 2.9 the effort required to maintain the program is

$$\begin{aligned} E &= D \times V \\ &= 21 \times 276.60 \\ &= 5808.6. \end{aligned}$$

2.3 Machine Learning

Machine learning (ML) is an application of artificial intelligence (AI) in which a mathematical model has the ability to automatically learn and improve from experience without being explicitly programmed [37]. In the 1990s, Mitchell offered another detailed definition of machine learning: "A computer program is able to learn from experience E with respect to some task T and some performance measure P, if its performance as measured by P, improves with the experience E" [29].

2.3.1 Machine learning approaches

Recently, machine learning approaches such as supervised learning, unsupervised learning and reinforcement learning have been widely used for analyzing text-based documents [2, 19, 32, 35]. In our research, we use a supervised machine learning approach. Details of various machine learning approaches are given below.

Supervised Learning

Classifying data instances based on predefined class labels is called supervised learning. In supervised learning, a data set consisting of sample inputs and their class labels is provided to the learning algorithm. The goal is to learn the rule(s) which will match the inputs to outputs (labels). 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. For example, if the class labels are male or female, then

the data instances in the data sets should be classified either in the male or female class [32]. In our research, we use supervised machine learning approaches implemented in the *WEKA* machine learning tool [46].

Unsupervised Learning

In unsupervised learning, a training data set is provided to the learning algorithms where the data set consists of sample inputs without any class labels. The goal of unsupervised learning is to determine any hidden structure or patterns in the input data [41]. For example, clustering is a form of unsupervised learning.

Reinforcement Learning

In reinforcement learning, a system learns from its own actions and experiences in an interactive environment to achieve a certain goal or reward. In this machine learning approach no training data set is given to the system. The system has an expected goal and to achieve this goal the system takes actions based on experience. According to Sutton [41], "Reinforcement learning problems involve learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. Moreover, the learner is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out". For example, learning to play chess is a form of reinforcement learning.

2.4 Data

Data are an essential part of any machine learning task. In machine learning, data are usually constructed as a table with values [45]. The terms used to describe our data are defined below.

- **Instance:** Each instance is an individual, independent row of data which consists of a set of column values.

- **Features:** Each column in an instance is a feature or attribute. Multiple columns or sets of features in a row form a single instance.
- **Data set:** A table that holds multiple instances and attributes is a data set.
- **Training data set:** The data set (or portion of the data set) which is used as input to train the model is termed the training data set.
- **Testing data set:** The data set which is applied to measure the performance of the model is defined as the testing data set.

Table 2.3 shows a sample data set representing the conditions of ten cars. In this table, the features are engine condition, mileage and transmission. There are ten instances. Here, the first instance is Good, Low, Automatic.

For the example shown in Table 2.3 we might use 70% of the data to train the system and the other 30% data to test the system.

Table 2.3: Sample data set representing the conditions of ten cars.

Engine condition	**Mileage**	**Transmission**	**Buy?**
Good	Low	Automatic	Yes
Moderate	Low	Manual	Yes
Bad	High	Manual	No
Good	Medium	Automatic	Yes
Moderate	High	Manual	No
Moderate	Medium	Automatic	Yes
Bad	Medium	Automatic	No
Good	High	Manual	No
Bad	Low	Automatic	No
Moderate	Medium	Manual	Yes

2.4.1 Data preparation steps

The learning and performance of any machine learning system is dependent on a well-organized and processed data set. Before applying any machine learning algorithms, the

training and testing data sets must be created by following three significant steps: selecting data, pre-processing the data, and transforming the data.

(i) **Data Selection:** The selection of appropriate data related to the defined task is the first step of data preparation. The selected data set must hold the relevant features that are necessary to apply the learning algorithms.

(ii) **Data Preprocessing:** After gathering appropriate data from the original source the next step is to prepare the data set. This preprocessing step involves formatting, cleaning and sampling of the selected data.

(iii) **Formatting:** The data which has been collected from different sources may not be suitable for use by the machine learning models. The data should be formatted according to the requirements of the machine learning algorithms. For example, an input file may contain information as text format; however the machine learning algorithms are not capable of working with text input files. Therefore, the text data must be formatted into an array or vector of integer, string or other compatible formats suitable for input to the machine learning algorithms.

(iv) **Cleaning:** The removal of irrelevant, duplicate, missing, or incorrect data is called cleaning the data. Data which has errors or redundant features are removed from the selected data set in this cleaning process. For example, in our region-based data set multiple records or instances have the country name listed as 'USA', 'United States of America', or 'United States America'. To fix the different representation of the same message, we updated all the strings to be 'USA'.

(v) **Sampling:** In order to reduce memory requirements and computational time and complexity, subsets or sample prototypes of a large data set may be used as sample representatives. The task of selecting a portion of a large data set as input is called sampling.

(vi) **Data Transformation:** The process of converting data or information from one format to another is called data transformation. The data is converted from the source system format into the target system format. For instance, if the data is in *CSV*-Comma-Separated Values format, the process of converting it into an *ARFF*-Attribute Relation File Format [45] file for the machine learning task can be termed as data transformation. Other information may include scaling, or digitizing non-numeric data.

2.5 *WEKA*: A Machine Learning Tool

WEKA is a machine learning package that provides a collection of data preprocessing tools and implemented state-of-the-art machine learning algorithms [45]. Several studies have carried out classification experiments using different machine learning tools such as *WEKA* [31, 35, 45].

2.5.1 Data Format

(a) **ARFF Data Format:** In *WEKA ARFF* (Attribute-Relation File Format) files, the data sets are represented in such a way that the data sets hold independent and unordered instances [45].

ARFF files have two main sections: the header and the data.

The header section is composed of two parts: the relation and the attribute. The first line of an ARFF file is defined as the relation name [45]. The format to write a relation is $@relation < relation-name >$. Similarly, the format to write an attribute statement is $@attribute < attribute-name >< datatype >$.

The $@data$ statement denotes a single line which represents an instance. Each attribute or feature value in an instance is separated by commas.

A sample *ARFF* file is given in Listing 2.4. In the header section the relation is "@relation balanceGender" and one of the attributes is "@attribute program vocabulary numeric". In the data section, an instance or record is denoted by "62, 263, 307.346455,

```
@relation balanceGender

@attribute program vocabulary numeric
@attribute program length numeric
@attribute calculated estimated program length numeric
@attribute volume numeric
@attribute difficulty numeric
@attribute gender {male,female}

@data
62,263,307.346455,1565.95363,male
68,258,345.989912,1570.565413,female
```

Listing 2.4: Sample file for ARFF File example.

1565.95363, male".

(b) **CSV data format:** A *CSV* (Comma Separated Value) file is a comma-separated values file, which allows data to be saved in a tabular format. In this format the data is laid out in a table of rows and columns, and a comma is used to separate the values in a row.

2.5.2 Classification Algorithms

In *WEKA*, each machine learning algorithm is implemented using a classifier. A classifier is a hypothesis or discrete-valued function that assigns class labels to particular data points. In our research work, five *WEKA* classifiers are used to perform the classification tasks: Bayes, Functions, Meta, Rules and Trees classifier. The details of these five *WEKA* classifiers are described in the sections below [45].

Bayes Classifiers

Bayes classifiers are based on Bayes theorem. The Bayes classifier builds a model where each data instance belongs to a class with some features and Bayes theorem is applied to predict the class of a new instance. In *WEKA*, some Bayesian classifiers are Bayes Net, NaiveBayes, NaiveBayesSimple, ComplementNaiveBayes, and BayesianLogisticRegression. In our research, we use the Bayes Net or Bayesian Network algorithm.

A simple Bayesian Network is a directed acyclic graph that consists of nodes and edges.

Each attribute in the Bayesian Network is represented by a node. In the Bayesian Network the parent-child relationship can be shown as an edge from the parent node to child node [45]. Each node in the network holds a probability distribution table that is used to determine the class of an instance. For each child node the probability table holds two parts and the parts are partitioned by a vertical line. The left side in the table holds values for the parent node and the right side of the table holds values for that particular child node. Table 2.4 shows data representing the conditions of five cars. Figure 2.2 represents a simple Bayesian Network based on the data from Table 2.4. The node 'Buy' is a parent node that points to the two child nodes: 'Engine condition' and 'Mileage'. Each node in the Bayesian Network holds a probability distribution table for the two attribute values. One of the child nodes 'Engine condition' has a left column for the 'Buy' parent node values (Yes, No) and a right column for the Engine condition feature values (Good, Moderate, Bad). Similarly, the other child node: 'Mileage' has a left column for the 'Buy' parent node values (Yes, No) and a right column for the 'Mileage' feature values (Low, Medium, High).

Table 2.4: Sample data on cars engine conditions and mileages.

Engine condition	**Mileage**	**Buy**
Good	Low	Yes
Moderate	Low	Yes
Bad	High	No
Good	Medium	Yes
Moderate	High	No

The Bayes Net algorithm uses Bayes theorem to calculate the conditional probability for each attribute value. The attribute values with maximum probabilities are selected for the classification of an instance. The conditional probability is the probability that an event will occur based on the occurrence of a previous event.

Let the conditional probability that event A will occur given that another event B occurs be denoted by $P(A|B)$. Then according to Bayes theorem,

Table 2.5: The car condition data with counts and probabilities.

Engine condition			Mileage			Buy	
	Yes	No		Yes	No	Yes	no
Good	2	0	Low	2	0	3	2
Moderate	1	1	Medium	1	0		
Bad	0	1	High	0	2		
Good	2/3	0/2	Low	2/3	0/2	3/5	2/5
Moderate	1/3	1/2	Medium	1/3	0/2		
Bad	0/3	1/2	High	0/3	2/2		

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.10}$$

where

- the probability that an event A will occur is denoted by $P(A)$,
- the probability that an event B will occur is denoted by $P(B)$, and
- $P(B|A)$ is the probability that an event B will occur given that event A occurs.

Based on the Table 2.4 we create Table 2.5 with counts and probabilities on five cars engine condition and mileage.

For example, given Engine Condition = Good and Mileage = Medium, we can calculate the probability of Buy = Yes.

To handle the zero probabilities, Laplace smoothing is used [45]. In Laplace smoothing, 1 is added to the numerator and for each attribute value 1 is added to the denominator. Applying Laplace smoothing in Table 2.5, we get Table 2.7 and 2.8.

Table 2.6: A new instance.

Engine condition	Mileage	Buy
Good	Medium	?

Table 2.7: Computing the car condition data probabilities using Laplace smoothing.

Engine condition			Mileage			Buy	
	Yes	No		Yes	No	Yes	no
Good	2+1/3+3	0+1/2+3	Low	2+1/3+3	0+1/2+3	3+1/5+2	2+1/5+2
Moderate	1+1/3+3	1+1/2+3	Medium	1+1/3+3	0+1/2+3		
Bad	0+1/3+3	1+1/2+3	High	0+1/3+3	2+1/2+3		

Table 2.8: Car condition data probabilities values after Laplace smoothing.

Engine condition			Mileage			Buy	
	Yes	No		Yes	No	Yes	no
Good	0.50	0.2	Low	0.50	0.2	0.57	0.43
Moderate	0.33	0.4	Medium	0.33	0.2		
Bad	0.17	0.4	High	0.17	0.6		

From Table 2.8, the likelihood probability for 'Yes' is calculated as

$$\begin{aligned} P(Yes|Good,Medium) &= P(Good|Yes)P(Medium|Yes)P(Yes) \\ &= 0.5 \times 0.33 \times 0.57 \\ &= 0.094. \end{aligned}$$

From Table 2.8, the likelihood probability for 'no' that the person will not buy the car is calculated as

$$\begin{aligned} P(No|Good,Medium) &= P(Good|No)P(Medium|No)P(No) \\ &= 0.2 \times 0.2 \times 0.43 \\ &= 0.017. \end{aligned}$$

We apply normalizing to the probabilities (P(Yes), P(No)) so that the sum of P(Yes) and P(No) equals to 1.

The probability of Yes after normalizing is

$$\begin{aligned} P(Yes|Good,Medium) &= \frac{0.941}{0.941+0.0172} \\ &= \frac{0.941}{0.9582} \\ &= 0.98 \\ &= 98\%. \end{aligned}$$

The probability of No after normalizing is

$$\begin{aligned} P(No|Good,Medium) &= \frac{0.0172}{0.941+0.0172} \\ &= \frac{0.0172}{0.9582} \\ &= 0.02 \\ &= 2\% \end{aligned}$$

So, for the situation where a car's engine condition is good and the car has medium mileage, then in this example the decision of a buyer will be 'Yes' with a probability value 0.98.

Function Classifiers

Function classifiers combine several machine learning classifiers and develop mathematical functions for data classification [45]. In *WEKA*, examples of some Function classifiers are Linear Regression, Logistic, SMO, Multilayer perceptron and Linear Support Vector Machine (Linear SVM). In our research, we use Sequential Minimal Optimization (SMO). In *WEKA*, the Sequential Minimal Optimization (SMO) algorithm is used to train the Support Vector Machine (SVM) classifier. The goal of the SVM classifier is to solve any classification problem by creating a hyperplane that uniquely classifies the data instances of the given classes. A 'hyperplane' is a decision boundary that divides the input space into

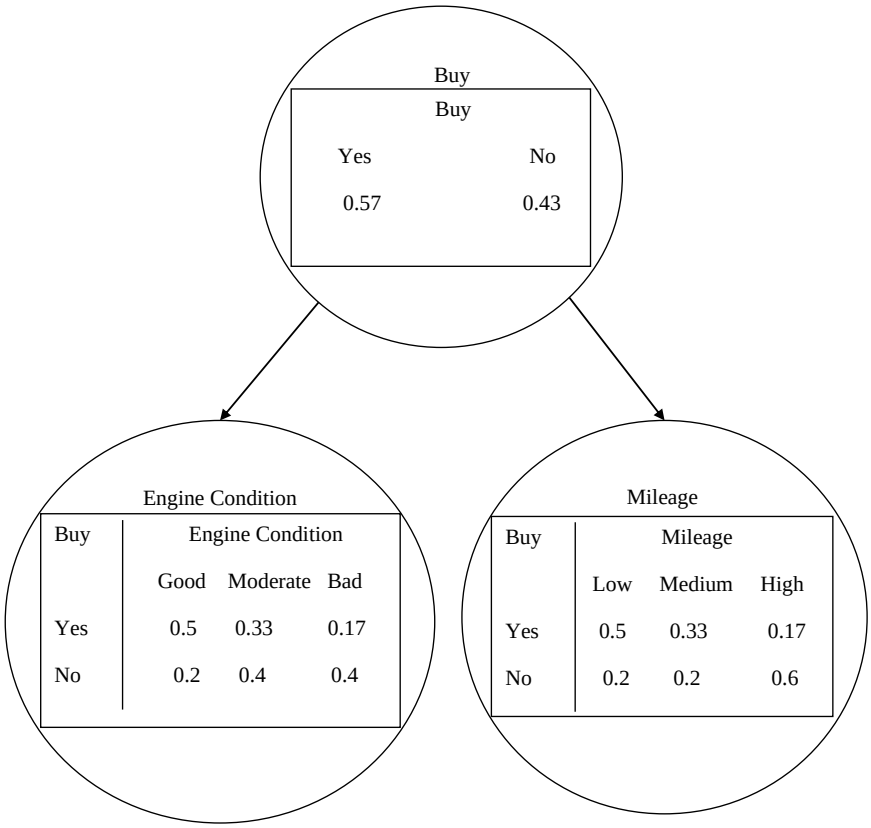

Figure 2.2: Example of a Bayes classifier, values are calculated based on Table 2.8.

the given number of classes and classifies each data point to a class.

Figure 2.3 shows an SVM classifier for a two-class classification task. Let us assume that a maximum margin hyperplane classifies all the training instances accurately for a two class classification task. The two different classes are denoted by blue circles and green squares. The black line in the center is the maximum margin hyperplane. The dotted line perpendicular to the maximum margin hyperplane is the 'maximum margin' which shows the maximum distance of the hyperplane from each class. The points that have the minimum distance to the maximum margin hyperplane are termed the 'support vectors'. The orange lines on which the support vectors are located are defined as the 'decision boundaries'. The decision boundaries are parallel to the maximum margin hyperplane.

Equation 2.11 defines the maximum margin hyperplane [45]:

$$x = b + \sum \alpha_i y_i a(i) \cdot a \tag{2.11}$$

In Equation 2.11, $a(i)$ and a are vectors. $a(i)$ is the set of attributes for the ith support vector and y_i is the class of the training instance. The a vector is the test instance. There are two numeric parameters: b and α_i. The learning algorithm determines the value of the two parameters. The Sequential Minimal Optimization (SMO) algorithm uses the polynomial kernel to train the support vectors efficiently. A polynomial kernel is a function that calculates the dot product of two vectors such as $a(i) \cdot a$ and turns the result to the power $(a(i) \cdot a)^n$ until the value of errors decreases.

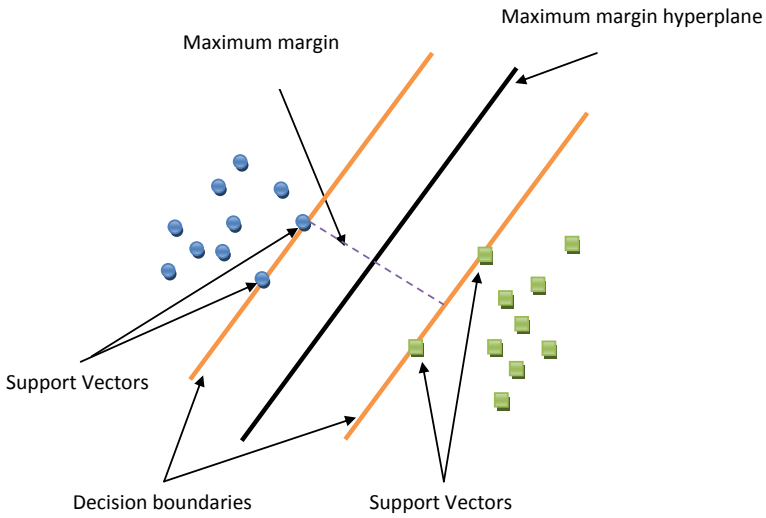

Figure 2.3: Example of a Functions classifier (SVM).

Meta Classifiers

Meta-learning algorithms take classifiers and turn them into more powerful learners. A meta classifier incorporates multiple algorithms to form one classifier and generate bet-

ter results based on performance measures. Some meta-classifier algorithms in *WEKA* include Bagging, Boosting, AdaBoostM1, ClassificationViaClustering and ClassificationViaRegression. For our research, we use the ClassificationViaRegression classifier.

ClassificationViaRegression classifier uses the $M5'$ algorithm for the classification task. $M5'$ algorithm is a combination of Decision Tree and Linear Regression algorithms. During the training process in the $M5'$ algorithm, the original data set is used to generate several new data sets for each class label. The total number of instances in each new data set is equal to the total number of instances in the original data set. In each new data set the class label value is set to 1 for a specific class label and 0 for the remaining class labels. For each data set, several model trees are created. A model tree is a binary decision tree that can be used for any regression task. The leaf nodes in the decision tree use linear regression functions to generate probabilities about the class of each instance. Based on the attribute values of each instance, the class probabilities are calculated. The formula for linear regression function is calculated using Equation 2.12

$$x = \sum_{i=1}^{n} \sum_{j=1}^{k} w_j a_j^i \tag{2.12}$$

where

- n is the total number of attributes,
- k is the total number of instances,
- x is the class,
- a is the value of each attribute, and
- w is the weight of each attribute which is calculated during the training process.

During the testing process of a new instance, the attribute values of the given instance are passed through each model tree. For each class label, the probabilities for the new

instance are calculated. The class label for which the new instance has the maximum probability is selected as the final class.

Figure 2.4 shows an example of a Meta classifier. In Figure 2.4, the original data set has two class labels, Male and Female. From the original data set, two data sets are derived based on each class label. In data set A: Male, the class value for Male is set to 1 and for Female the class value is set to 0. In contrast, in data set B: Female, the class value for Female is set to 1 and for Male the class value is set to 0. Based on the attribute values of each instance, multiple model trees are generated during the training process. The leaf nodes in the model trees hold linear regression functions which calculate probabilities of a class based on the attribute values.

For example, given attribute values of (2.2 3.2 4.3 ?), the new instance is passed through the model trees and the probabilities for class Male = 0.93 and class Female = 0.07 are calculated according to the attribute values. Based on the higher probability value, the class of the new instance is predicted as Male.

Rules Classifiers

Rules classifiers generate rules for classification. For example, the Decision Table is one of the algorithms that builds a table based on the features of the data. Some other examples of Rules classifiers are DTNB, OneR and ZeroR [45]. In our research, we use the Decision Table classifier to build one of the classification models.

Table 2.4 shows data on the engine conditions and mileages of five cars. Based on the data, the decision-makers will decide which cars they should buy and which cars they should not.

Let us assume that Table 2.4 is used to train the Decision Table classifier. Based on the data, the Decision Table classifier generates if-then rules where the class is 'buy' and class labels are: yes, no. The generated rules are shown as follows:

if (Engine condition = Good and Mileage = Low) then Buy = Yes;

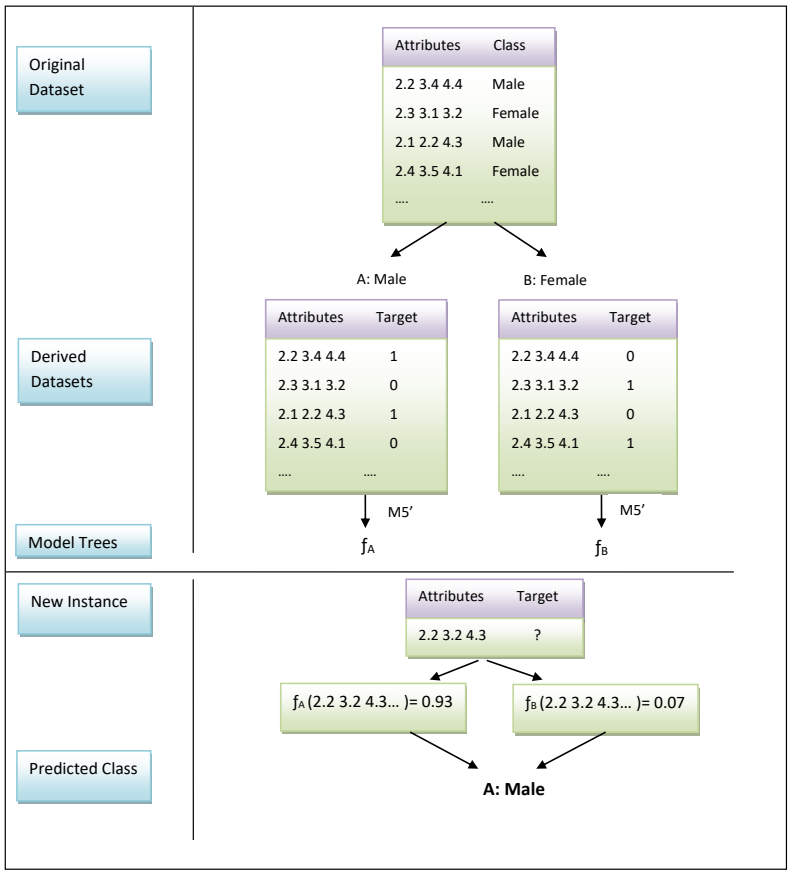

Figure 2.4: Example of a Meta classifier.

if (Engine condition = Moderate and Mileage = Low) then Buy = Yes;

if (Engine condition = Bad and Mileage = High) then Buy = No;

if (Engine condition = Good and Mileage = Medium) then Buy = Yes;

if (Engine condition = Moderate and Mileage = High) then Buy = No;

For a new instance, the Decision Table algorithm will use these if-then rules to make a decision. For example, if a given instance is

Engine condition = Good, Mileage = Medium, Buy = ?

the Decision Table classifier will classify the new instance as Buy = Yes.

During the training process the Decision Table algorithm generates decision rules based on the given training data. Given a test instance the algorithm will search the generated decision rules to find a decision for the test instance. If the algorithm is unable to find a rule

that matches the instance then the instance is labeled an invalid condition. For example, if a given instance is

Engine condition = Good, Mileage = High, Buy = ?

the Decision Table classifier will classify the new instance as Buy = Invalid condition. The analyst may use this data to further train the model.

Trees Classifiers

A tree is a simple representation of classification. Tree classifiers offer a supervised machine learning approach where the data is continuously split according to a certain parameter. Tree classifiers in *WEKA* are BFTree, J48, Random Forest and NBTree. In our research, we use the Random Forest classifier for the classification tasks.

A decision tree is a tree-like structure that consists of a 'root', 'nodes', 'edges or branches' and 'leaf nodes'. An attribute is selected as the root node. Then, the 'edges' are created which correspond to the outcome of a test and connect to the next node or leaf. The 'leaf nodes' denote class labels. Random Forest is one of the most popular and powerful supervised machine learning classification algorithms. The algorithm creates a forest of multiple decision trees. Each decision tree in the forest makes a decision to predict the class of an instance. Based on the majority decisions of the decision trees, the algorithm performs the classification tasks with more accurate results [4]. The algorithm uses Information Gain to decide which attribute to split at each step of the decision tree. Figure 2.5 illustrates the approach of the Random Forest classifier. The training data are used to develop a Random Forest classifier with n decision trees. For a new instance, each decision tree predicts a class label.

2.5.3 *WEKA* Attribute selection

WEKA has multiple algorithms for attribute or feature selection from the data set. Each data set will contain some features that are more or less relevant for the chosen tasks. The attribute selection option in *WEKA* selects the features that are likely to be more reliable

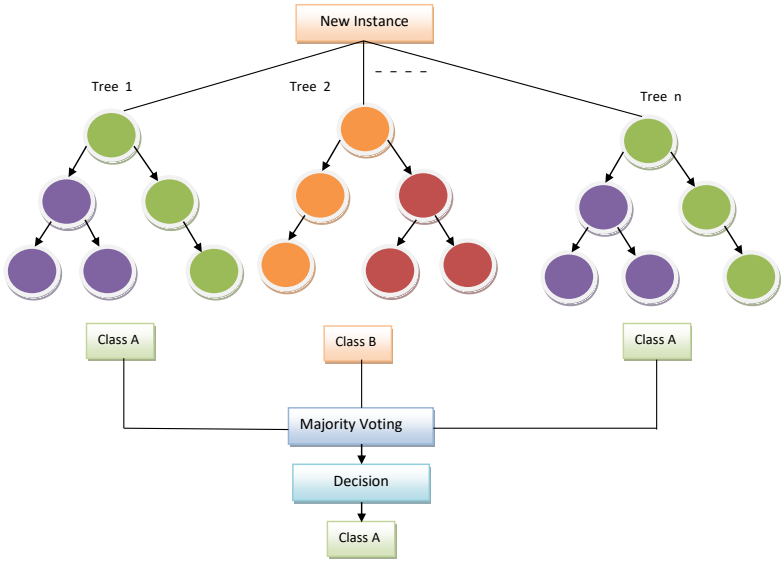

Figure 2.5: Example of a Random Forest classifier.

in making predictions and less redundant from the experimental data. One of the popular attribute evaluators in *WEKA* is 'InfoGainAttributeEval' which uses the 'Ranker' method for ranking features that are more relevant to the experiment. Some other feature selection algorithms are CfsSubsetEval, CostSensitiveSubsetEval and WrapperSubsetEval.

To select the best features for our research, we use 'InfoGainAttributeEval' approach. This approach measures the significance of an attribute by evaluating the information gain according to the class value. "Information Gain" measures the changes of entropy of the given attribute values. "Entropy" estimates the impurity or amount of information available in the values of an attribute. The lower the value of entropy, the higher the value of information gain. The attribute which has the highest information gain value is selected as the "best" attribute. In Section A.1 (Appendix A), an example on the calculation of information gain is shown in details.

2.6 Validation Technique

Cross validation is a widely used technique for testing models created by machine learning algorithms. If most of the data belong to one specific class during the training process, the learning model may become biased towards the majority class which has more data. To overcome a learning algorithm's biasing problem, cross validation is used [42]. Cross validation uses a fixed number of folds for partitioning a data set [45]. With limited data a model can make inappropriate predictions during training. This can be the cause of a situation termed overfitting, when a trained model has a zero error rate for a small training data set but makes inaccurate predictions with new data [46]. Cross validation can help to attenuate the risk of overfitting. In *WEKA*, the k-fold cross validation technique is used for model validation [32, 45]. In our research work, we use 10-fold (where k=10) cross validation. In 10-fold cross validation, the entire data set is divided into 10 groups of instances (each instance is a row of a data table) of equal size which are called folds. The model is then trained using all folds except one which will be used later on to test the model. Cross validation is an iterative process and it continues until each fold is used to test the model. Performance is measured by averaging the results of each fold in order to determine the efficiency of the model or classifier.

2.7 Evaluation Metrics

To demonstrate the performance of a supervised learning model various evaluation metrics are used, including a confusion matrix, precision, recall, and F-measure. A confusion matrix is a table that represents a model's performance when assigning data instances to corresponding class labels [10].

(i) **Confusion Matrix:** A confusion matrix is a table that represents the performance of a classifier on a set of test data for which the true or actual values are known. Table 2.9 shows the terms used in the confusion matrix to describe the models's performance.

Table 2.9: Structure of a confusion matrix showing the results of machine learning classification.

	Predicted Class		
Actual Class		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

True Positives (TP): The predicted samples that are classified correctly as positives are referred to as True Positives.

False Positives (FP): The predicted samples that are actually negatives but classified incorrectly as positives are referred to as False Positives.

True Negatives (TN): The predicted samples that are classified correctly as negatives are referred to as True Negatives.

False Negatives (FN): The predicted samples that are actually positives but classified incorrectly as negatives are referred to as False Negative.

Based on the confusion matrix, we determine the values of precision, recall, and F-measure of each machine learning model or classifier using the WEKA machine learning tool.

An example confusion matrix is given in Table 2.10. In this confusion matrix, programs from male programmers are labeled as positive samples and programs from female programmers are labeled as negative samples. In this example,

- TP (True male written programs): The model correctly classifies 1211 programs as male written programs.
- TN (True female written programs): The model correctly classifies 1168 programs as female written programs.
- FN (False female written programs): The model incorrectly classifies 222 pro-

grams as female written programs.

- FP (False male written programs): The model incorrectly classifies 265 programs as male written programs.

Table 2.10: Example confusion matrix.

	Predicted Class		
Actual Class		Male	Female
	Male	1211 (TP)	222 (FN)
	Female	265 (FP)	1168 (TN)

(ii) **Accuracy:** Accuracy is the ratio of the number of correct predictions over the total number of predictions calculated as a percentage as shown in Equation 2.13.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.13}$$

If we compute the accuracy using Table 2.10 we get

$$\begin{aligned} Accuracy &= \frac{1211+1168}{1211+1168+265+222} \\ &= 0.8300 \\ &= 83.00\%. \end{aligned}$$

(iii) **Precision:** The ratio of correctly classified positive (TP) values over predicted values (TP+FP) is referred to as precision, as calculated by Equation 2.14. Precision can be thought of as the percentage of the results that are relevant.

$$Precision = \frac{TP}{TP+FP} \tag{2.14}$$

The precision for the confusion matrix shown in Table 2.10 can be calculated as

$$\begin{aligned} Precision &= \frac{1211}{1211+265} \\ &= 0.8204 \\ &= 82.05\%. \end{aligned}$$

(iv) **Recall:** The ratio of correctly classified positive (TP) values over actual values (TP+FN) is referred to as recall, as calculated by Equation 2.15. Recall is the percentage of the results that are relevant and correctly classified.

$$Recall = \frac{TP}{TP+FN} \tag{2.15}$$

The recall for the confusion matrix shown in Table 2.10 can be calculated as

$$\begin{aligned} Recall &= \frac{1211}{1211+222} \\ &= 0.8450 \\ &= 84.50\%. \end{aligned}$$

(v) **F-measure:** F-measure is a weighted average of precision and recall, as calculated by Equation 2.16.

$$F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.16}$$

The F-measure for the confusion matrix shown in Table 2.10 can be calculated as

$$F - measure = \frac{2 \times 0.8205 \times 0.8450}{0.8205 + 0.8450}$$
$$= 0.8325$$
$$= 83.25\%.$$

2.8 Programming Language: Python

Python is an open source programming language with a large standard library including various packages [33]. In our research, we use the programming language Python to extract data and relevant features from the primary data set. We also used Python to calculate new features from the source code in our data set. We wrote Python programs to calculate features based on the three metrics: lines of code, cyclomatic complexity, and Halstead metrics. We used the Python library 'lizard' 'lizard' [1] to calculate features based on the lines of code, cyclomatic complexity. Due to the lack of free tools and libraries for calculating the Halstead metrics, we wrote Python programs using the algorithm of Halstead metrics to calculate the features.

2.9 Related Work

Ronald Wardhaugh described how sociolinguistics is related to language, society and gender in [43]. He described how men and women have used language in their own way. For example, he theorized that women use more polite and surprised language patterns than men to express their opinion on a particular topic. Regional effect on language is also described by Wardhaugh [43].

Various research has investigated written text documents to demonstrate the effect of social variables in natural language [2, 3, 5, 14]. For example, Argamon et al. examined the British National Corpus (BNC) to explore gender-based differences using machine learning

[1] https://pypi.org/project/lizard/

approaches [3]. In other work, Argamon et al. performed analysis on French literature [2] using the SVM (Support Vector Machine) algorithm. Both sets of work reported 90% accuracy in predicting the gender of the authors. In recent years, other research have also analyzed gender-based writing variations in computer programs [17, 32, 35].

In [31], Naz analyzed computer programs to investigate the relationship between sociolinguistics features and programming languages. The authors conducted their research on 100 C++ programs in which 50 computer programs were from male and 50 from female. The authors used 50 features and conducted five experiments. In a second set of experiments the authors reduced the number of features from 50 to 7 and acquired an accuracy of 71%. The authors analyzed the features and determined that the keywords *bool*, /, ==, >= were used with higher frequency in male written programs, whereas the keywords *char*, *double*, + were used more often in female written programs.

Rafee [35] examined C++ computer programs based on gender and region of the programmers. In his experiments, Rafee considered only one software metric, lines of code, to calculate features for the classification task. Rafee also investigated the effect of region in computer programs. The author demonstrated that Canadian programmers wrote more comments than Bangladeshi programmers. However, in other work, Meulen et al. [26] claimed that the software metric lines of code has a strong correlation with other software metrics, such as McCabe's cyclomatic complexity and Halstead metrics.

In [42], Tasnim analyzed computer programs to classify computer programs based on the experience of the programmer. Programmers were categorized as being at a beginner level or an expert level. Their work suggested that beginner level programmers used fewer functions, fewer executable statements and more blank lines in their programs. On the other hand, the number of user-defined functions and the executable lines of code (SLOCP) were higher in expert written programs.

To further investigate the relationship between sociolingiustics features and different software metrics, Alam [1] conducted four machine learning experiments on GitHub and

Codeforces data. The author classified the data based on the gender and region of the programmers. The author analyzed 103 features and used Bagging classifier, Decision Table, Random Forest, Bayes Net and Logistic Regression algorithms for the classification tasks. The research showed that the number of average lines of code was higher in North American programs than the South Asian programs, and suggested that the difficulty measurement of female written programs tended to be higher than the male written programs. In contrast, male programmers used exception handling functions more than the female programmers.

McCabe identified that an individual programmer's style was related to the complexity measure [25]. The author determined that several programmers who never had any training in structured programming consistently managed to write programs in the 3 to 7 complexity range, and the programs with complexity value less than 10 are termed as well-structured programs. Watson et al. [44] applied the cyclomatic complexity metric to several FORTRAN programs and tested the complexity of the programs and found a high correlation between cyclomatic complexity and non-commented lines of code.

In [12] the Halstead Metric was analyzed on different Python and C++ programs where the authors calculated the bugs, effort and time required to run the code. Their work suggested that coding in Python requires less effort and time than C++ language.

Chapter 3

Methodology

3.1 Data Source: codeforces.com

Our research goal is to analyze the programming styles of male and female contest programmers from varying regions. We focus on collecting contest problems in which both male and female programmers solve the same problems. We also aim to analyze the programming styles of the programmers based on their region. Online programming websites such as codeforces, codechef[2], leetcode[3], and hackerrank[4] individually host different programming contests and allow the contest programmers to submit their solutions to a database. Codeforces is a competitive programming website as well as a social networking site that offers the users a platform to conduct and organize programming contests [27].

3.1.1 Advantages

We selected codeforces to collect data for our research based on the following advantages.

- The user source code is accessible. Anyone can access or view the source code and user information such as name and country. Other online programming websites usually have minimal access to such information. The availability of this information makes codeforces a useful repository for competitive programming research.

- Participants from all over the world participate in the programming contests hosted by

[2] https://www.codechef.com/
[3] https://leetcode.com/
[4] https://www.hackerrank.com/

codeforces. The source code may have region-based variations and these variations can be analyzed to investigate the differences of the programming styles.

- Codeforces stores the total number of submissions for each contest problem, and so we used this facility for collecting our required data.
- Codeforces also has an API that provides access to some of the stored data in machine-readable JSON format [28].

3.1.2 Disadvantages

Codeforces also has some disadvantages which are detailed below:

- In codeforces, data are retrieved from HTML pages. If there is any update in the HTML pages, we may also have to make updates to our data collection program to match the changes.
- The source code is stored in pop-up links. Codeforces allows the web driver to retrieve 50 programs per page. By clicking on the pop-up links, the web driver can fetch up to 50 programs from each page. If the number of submissions is higher, the automatic collection of the source code for the solution programs may become a time-consuming task.
- The participation of female programmers in any programming contest is comparatively lower than male programmers [9]. In codeforces, the number of the male user is considerably higher than the female user. However, this is likely to be true of most coding websites.

3.2 Data Selection Criterion

We selected nine contest problems based on the number of participants and the difficulty rating of the problems. While selecting the problems, we made the following observations,

1. High difficulty rated problems (difficulty level: $2300 - 4000$) were attempted by a small number of participants, approximately 100 participants. Therefore, the codeforces organizers tend to receive a small number of submissions for these problems.

2. Low difficulty rated problems (difficulty level: $800 - 1600$) were solved by a huge number of programmers, nearly 6000 to 100000 participants. However, after analyzing some low difficulty rated problems, we determined that the programs were too short. For example, the programs were solved by writing one or two lines of code. As a result, the scope for calculating features from the programs was limited although the number of participants is very large. In our work we only considered programs that were at least 20 lines long, including the comments and blanklines.

3. The medium difficulty rated problems, (difficulty level: $1700 - 2200$) were solved by 1500 to 4500 participants. Moreover, the programs had the scope (length) for calculating features that are required for our research. For example, each program should have at least 2 logic statements for calculating the cyclomatic complexity.

3.3 Data Collection Process

We divided our data collection process into three major parts:

1. user information collection,

2. source code collection, and

3. adding gender labels.

The details are described in the following subsections.

3.3.1 User information collection

In codeforces, a user profile holds several items of user information including: handle (which uniquely identifies a user), first name, last name, region, the user's blog, teams,

submissions, and a list of contests in which the user participated. Codeforces has several APIs which allow access to some information from the website. We used the 'user.info' API to collect the information on each user who participated in the different contests hosted and organized by codeforces operators.

3.3.2 Source code collection

Codeforces has a 'Problemset' page which contains information on a problem. The information on the 'Problemset' page is stored in different tabs such as the problem id, problem name, problem difficulty rating, and solved problems. If the 'solved problems' tab is clicked, a pop-up link directs to the corresponding 'status' page. The status page shows information regarding a submission such as submission id, submission time, handle, problem name, language, verdict, time, and memory. Each submission id contains the link to the corresponding source code. By clicking on the submission id, the source code of a submission can be retrieved. As the codeforces API does not provide the ability to retrieve source code, we used the Selenium[5] web driver to fetch the source code or programs authored by the users or participants. The submission id is linked to each user's handle, which directs to the user profile. Thus, we used the handle and submission id to connect the user information to the collected source code.

We created a Python program that used the Selenium web driver to collect the solution programs. The link to the solution programs can be found via a uniform resource locator or url (which is the address of a web page). In our program, we set the url of the targeted problem to scrape our required data. The program opens the url page and then opens the pop up links of the submission id to collect the corresponding source code. From each page, the web driver is allowed to collect programs from 50 submissions, and we set the web driver to continue scraping the next pages until no records were found. If the submission id did not contain any pop-up links, the web driver ignored that submission and moved forward to scrape the next submission.

[5] `https://selenium-python.readthedocs.io/index.html/`

3.3.3 Adding gender labels

Codeforces only provides the first name and last name of a user. Hence, gender information is not available for users of codeforces. To add gender labels to the user names, we used an API genderize.io[6]. Genderize.io uses the information from user profiles across major social networks and provides data on user names through the API. Genderize.io permits 100 free name requests per day. The API adds a gender label (male, female or null) to the given first names and gives a probability value between 0.0 and 1.0. To reduce the number of requests, we stored all the responses from the API in the local database. For a given first name, the name is searched at first in the local database to check if the gender label is already added to the name. If a gender label is added to the given first name, the request to the genderize.io is skipped. Otherwise, a request is made to the API and the response from genderize.io is added to the local database. To verify the added gender labels, we used a human annotation approach and manually searched the names in the google search engine.

3.4 Data Information

In machine learning, the performance of a model is highly dependent on the preparation of the data set. Therefore, inconsistent, incomplete, or redundant data are cleaned or removed in the data preparation step [11].

3.4.1 User information data

We retrieved 11 attribute fields that contained user information. Table 3.1 shows that only rating and max rating are integer type data, and the rest of the attributes are string type data. The attribute handle is a unique string identifier for each user profile. A user name is indicated by first name and last name of the user. The region of a user is denoted by the country and city. The attribute organization provides information about the user's institution such as high school, university. To determine the expertise of a user rank, max rank, rating

[6] https://www.genderize.io/

and max rating attributes are used. For our research we selected only 4 attributes from Table 3.1 and did not use any of the other attributes. The 4 attributes that we selected for our research are handle, first name, last name and country.

Table 3.1: Attributes in codeforces related to user information.

Attribute	Type	required
Handle	String	yes
First Name	String	yes
Last Name	String	yes
Country	String	yes
City	String	no
Organization	String	no
Rank	String	no
Rating	Integer	no
Max Rank	String	no
Max Rating	Integer	no
Registered	String	no

3.4.2 Source code data

We collected 38430 computer programs for our research. We observed that a user could submit multiple solutions. The reason for multiple submissions might be that a user wanted to write an optimized solution. Thus, we selected only one program per user. With this restriction we retrieved 37451 programs for nine contest problems. Table 3.2 shows the problem id, title, difficulty,total submissions and the number of programs written in C++ and other languages for the selected contest problems. The problems were solved using different languages such as 'C++', 'Python', and 'JAVA'. As shown in Table 3.2 the number of programs written in C++ programs is much higher than the number of programs written in other languages. For our research we selected only programs which were written in the C++ language and have valid first names, of which there were 18986.

Table 3.3 shows the list of attributes that we collected in this research.

Table 3.3 illustrates that the submission id, memory are integer type data and the source

Table 3.2: Information about the nine selected contest problems.

Problem id	Title	Difficulty	Submissions	Programs in C++ language	Programs in other languages
1288D	Minimax Problem binary search	2000	3802	3612	190
1301D	Time to run	2000	2926	2793	133
1304E	1-Trees and queries	2000	2956	2757	199
1324F	Maximum White Subtree	1800	4196	3978	218
1326D2	Prefix-suffix palindrome	1800	5990	5694	296
1328E	Tree queries	1900	4783	4544	239
1332E	Height all the same	2100	2604	2432	172
1340B	Nastya and Scoreboard	1700	5768	5467	301
1344B	Monopole Magnets	2000	4426	4158	268

code, handle, programming language, problem title, submission date are string type data. When a submission is made a unique submission id is given to each submission and the submission is saved in the source code field. Each user who made a submission is identified by the user's handle. The attribute problem title holds the name of each problem. The programmers may solve the problem according to his/her choice of programming language. The programming language attribute shows which programming language is used to solve the problem. The submission date stores information about the submission day, month and year. The time and memory required to compile the program are stored in the time and memory fields of the codeforces website. We did not have to calculate how much time and memory a programmer used to solve a problem, we just used the data collected from the website. From Table 3.3, we selected 6 attributes for our research and did not use the other attributes. The 6 selected attributes were submission id, source code, handle, programming language, time and memory.

Table 3.3: Attributes in codeforces related to source code collection.

Attribute Field	Type	required
Submission id	Integer	yes
Source code	String	yes
Handle	String	yes
Programming language	String	yes
Problem title	String	no
Time	String	yes
Submission date	String	no
Memory	Integer	yes

3.5 Data Preparation

To prepare to analyze the nine problems that were solved using the C++ language, we first used data cleaning techniques and removed the records which had empty values for the first name and country attributes. Using a Python program, we merged the valid user information (in JSON format) and source code (in CSV format) and stored the data set in a CSV file. After combining the user information and source code, we had a total of 18986 records. In the next step, we extracted all the first names from the CSV file. We used the 'genderize.io' API to add the gender label based on the probability of a user's first name. The 'genderize.io' added gender label to 17415 names and was unable to add the gender label to 1571 names, and so we removed the records which had an unknown gender label. This left us with a data set with 17415 records. We then applied the human annotation approach and used the Google search engine to verify the gender label. Figure 3.1 represents the data collection and preparation process of our research.

Table 3.4 summarizes the 17415 records remaining after the data preparation steps. 15982 records contained the source code and information on male participants and 1433 contained code and information on female participants. We then categorized the regions of the programmers into two large regions: the Eastern zone and the Western zone. The Eastern zone contains the users from Asia, whereas the Western zone holds data from North

America and Europe. Of the 17415 records, 11792 records were from the Eastern zone and 2944 records were from the Western zone.

Table 3.4: Information on number of records after data preparation steps.

Total records: 17415	Gender-based records	Male : 15982
		Female : 1433
	Region-based records	Eastern zone : 11792
		Western zone : 2921

3.6 Balanced Data Set Creation

The performance of a machine learning model is highly dependent on the prepared data set. An imbalanced data set may affect the performance of a machine learning model, and the resulting model may have a substandard performance [8, 18]. If a data set has an unequal number of records for different class labels for a classification problem, the data set can be referred to as imbalanced [8, 18]. For example, in our gender-based data set the number of records for male programmers is 15982 while the number of records for female programmers is 1433. For our research, we prepared eleven smaller, but balanced data sets. Ten of these were used for gender-based analysis (data sets $1-10$) and one data set (data set 11) was used for region-based analysis. Details of these are given below.

3.6.1 Gender-based data sets

We prepared ten data sets to explore the effect of gender in programming styles. Table A.3 shows the information on our gender-based data sets.

- **Individual problem based data sets**

  One of our research goals is to determine whether male and female contestants write programs differently to solve a particular programming contest problem. For this we wish to study the solutions from the male and female participants who solved the same problem. Because our initial data sets had 15982 records from male participants

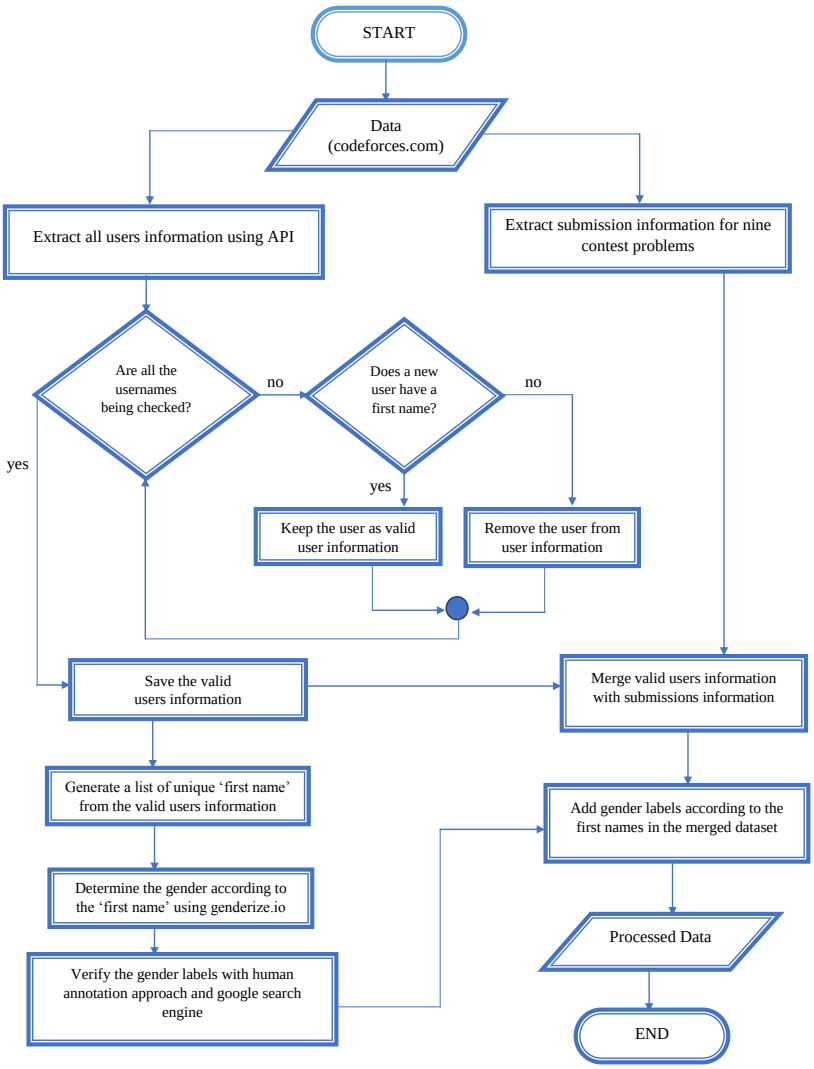

Figure 3.1: Data collection and preparation process.

and 1433 records for female participants, we balanced our data sets by using the under-sampling method. In this approach, which is also known as random under-sampling [8], we randomly removed data from the majority class in order to balance the data set. For each smaller data set we randomly removed some male records since the number of records for male programmers was significantly higher in each smaller data set. Following this approach we prepared nine data sets (data sets $1-9$) for nine different programming problems, with one data set per problem. Each data set contains an equal number of records for male and female participants. In Appendix A,

Table A.3 shows the number of records for each of these data sets.

- **Combined data set**

  In machine learning, the size of the data set has an impact on the learning process of the machine. For smaller data sets, the over-fitting problem may affect the learning process [8]. Therefore, we combined the different solutions for the nine different contest problems and prepared one larger data set (data set 10). In this data set different solutions for the nine different problems that we collected in data set $1-9$ were analyzed collectively to identify some general programming styles of male and female programmers. Thus our gender-based combined balanced data set 10 has a total of 2866 records, which holds the source code and information on two genders, male: 1433 records, and female: 1433 records.

3.6.2 Region-based data set

Our second research goal is to determine the effect of region on the programmers. Table 3.5 represents the information on the region-based data set. We grouped the number of countries into two regions: Eastern region and Western region. Countries such as Bangladesh, China, Japan, India, Pakistan, Sri Lanka, Iran, Philippines, Singapore, and Malaysia are categorized into the Eastern region. We grouped the countries United States of America, Canada, Italy, France, Germany, Romania, and other European countries into the Western region. Thus data set 11 contains a total of 5842 records with 2921 records from the Eastern region and 2921 records from the Western region. We used the under sampling method since the number of programs from the Eastern region seemed to be higher than the programs from the Western region.

3.7 Document Representation

Machine learning models require a vector of numeric feature values as input and provide nominal values as output. In our gender-based classification model, the input feature values

Table 3.5: Information on the region-based data set.

	Region		
Data set	No. of records from Eastern region	No. of records from Western region	Total Records
11	2921	2921	5842

are numeric and the output class value (male or female) is nominal. Similarly, our region-based classification model has numeric input feature values and a nominal output class value (Eastern region or Western region). In this research, we treated each collected C++ program as a text document. Using a Python program, we calculated numeric feature values based on lines of code, McCabe's cyclomatic complexity, and Halstead metrics from each text document. For each C++ program, we counted the numeric values of 13 features. Next, we represented our data file in the *WEKA* data format, known as an ARFF (Attribute-Relation-File-Format) file. To ensure that the feature values are within a similar scale and represent word occurrence information, we used a Python program and converted the input feature values into tf-idf (term frequency–inverse document frequency) vectors [45]. Tf-idf computes the relevance of a feature/term to a document and the occurrences of that feature or term in all the documents. In total, the ARFF data file will hold several data instances or records. Each data instance or record in the ARFF data file is represented by a set of numeric feature values and a class label. For our region-based classification model, the numeric input feature values are the same as used in our gender-based classification model.

3.8 Selected Features

In our work we use three types of features: sociolinguistics features, software metrics features and program execution features. We selected seventeen features for our experiments, as shown in Figure 3.2. We consider two sociolinguistics features, thirteen software metrics features, and two program execution features. We analyzed the gender and continent region of the programmer as sociolinguistics features. We calculated three types of software metrics features from the programs. We calculated loc, slocp, lloc, comments and

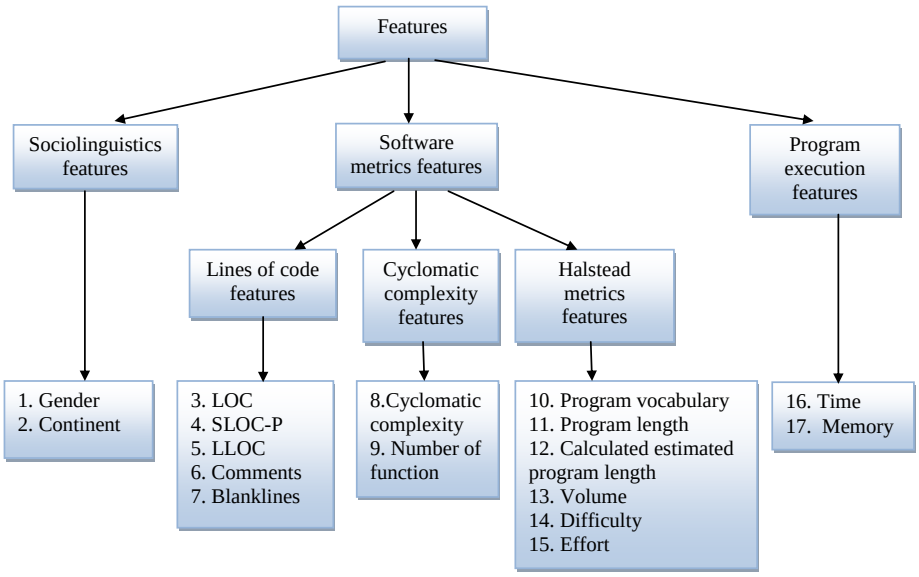

Figure 3.2: Types of features used in our work.

blanklines as lines of code features; cyclomatic complexity number and number of function as cyclomatic complexity features; and program vocabulary, program length, calculated estimated program length, volume, difficulty and effort as Halstead metrics features. Details of these features are given in Sections 2.2.1, 2.2.2 and 2.2.3. We also consider two execution features, required time and memory to execute the program. Program execution time is measured from the start of the program when inputs are initiated to the end of the program when the outputs are delivered [47]. The amount of memory required to store the instructions of a program is defined as required memory [47]. We extracted the program execution features from the codeforces submissions.

We use the features listed in Figure 3.2 to conduct the experiments which are described in Chapter 4.

Chapter 4

Experiments and Results

In this chapter we discuss the set up of the experiments and the results achieved from the experiments. The chapter concludes with a discussion on the results and the threats to the validity of this research.

4.1 Experimental Work

In this work we collected C++ programs from the programming contest website codeforces.com. We created a total of 11 data sets for our research which were used to train the classification models, and then tested them using 10-fold cross validation. The goal was to classify the C++ programs based on the gender and region of the programmer. In the gender-based classification task, the models predict the class labels male or female while in the region-based classification task, the models predict the class labels Eastern or Western.

We used five supervised machine learning algorithms from the *WEKA* tool. The algorithms that we used in our research are as follows:

- From the Bayes classifier, we used the Bayesian Network or Bayes Net algorithm.
- From the Function classifier, we used the Sequential Minimal Optimization (SMO) algorithm.
- From the Meta classifier, we used the Classification Via Regression algorithm.
- From the Rules classifier, we used the Decision Table algorithm.

- From the Trees classifier, we used the Random Forest algorithm.

Using these five algorithms we performed a total of 13 experiments. An overview of these experiments is given below.

- In experiments $1-10$ we classified the gender of the programmers. We used 16 features and five machine learning algorithms to build the classification models. In experiments 1-9 we used 9 different but smaller data sets (data sets $1-9$), and in experiment 10 we used a larger data set (data set 10). Finally, we applied 10-fold cross-validation and measured the performance of the five classification models.

- In experiment 11 we classified the region of the programmers. We used 16 features and five machine learning algorithms to build the models. In experiment 11 we used one region-based data set (data set 11) to classify the region of the programmers. Again, we applied 10-fold cross-validation and measured the performance of the five models.

- In experiment 12 we reduced the number of features based on the 'Information Gain' values and selected 6 features to classify the gender of the programmers. Using these 6 features and five machine learning algorithms, we built the models. In experiment 12 we used data set 10. As before, we applied 10-fold cross-validation and measured the performance of the five models. We used the top 6 features to determine the effect of the reduced features on the performance of the five models.

- In experiment 13 we again reduced the number of features based on the 'Information Gain' values and selected 6 features, but this time to classify the region of the programmers. This experiment (13) used data set 11. As before, we applied 10-fold cross-validation and measured the performance of the five classification models.

4.2 WEKA Tools

In this work we used *WEKA* to build the machine learning models. We also used additional *WEKA* tools during the data formatting, feature selection, and model evaluation processes. These tools included the discretize filter, remove filter, info gain attribute selector, and cross-validation tool. A brief description of these tools are given below:

- **Filters**

  Filters in *WEKA* are used for preparing the data set before applying the classifiers. They may be supervised, which means the class values are taken into consideration, or unsupervised when the class values are not considered. We used the supervised discretize filter to discretize the attribute or feature values for the Bayesian Network algorithm. The discretize filter converts the numeric input attributes or features into nominal input features or attributes [45]. We used a bin size of 10 for discretization. For example, we applied the discretize filter on the lines of code (loc) feature and discretized the values into 10 bins.

  We also used the unsupervised remove filter to remove features for experiments 12 and 13, where we removed all but the top 6 features. This is done by selecting the 10 features that we wish to remove and then applying the remove filter.

- **InfoGainAttributeEval**

  To determine the most suitable features and reduce the number of features, we used one of the attribute evaluators of *WEKA* called 'InfoGainAttributeEval'. Using the attribute evaluator, each attribute in the dataset is evaluated in the context of the output variable or the class. Applying the Ranker algorithm on the training data set, this evaluator measures the information gain (described in Section 2.5.3) for each feature and rank the features. In the Ranker algorithm each attribute is evaluated and the results are listed in a rank order.

- **Cross-validation**

  To evaluate the performance of the learning models, we used the cross-validation tool (described in Section 2.6) built into *WEKA*. In our experiments we set the number of folds at 10 for the cross-validation process.

4.3 Programming Environment

We used the Python programming language and the machine learning tool *WEKA*. The experiments were performed on a *Dell* laptop with an Intel Core i5 processor, 4 GB RAM, and a 1 TB Hard Disk. We used the *Linux* operating system, which is an open-source command-based operating system, and *Windows 8*© which is a proprietary graphical operating system. *Linux* is used for writing and executing Python code using version *Python 3.6*. We used *Windows 8* for running the machine learning tool *WEKA* and Microsoft Excel© for the statistical analysis of the features.

4.4 Experiment Details

4.4.1 Experiments 1-9

In experiments $1-9$ we aimed to classify the programs based on the gender of the programmers, however in these experiments each data set contained programs for a single problem. Table 4.1 shows the number of records used for the gender (individual-problem) based classification.

We first calculated the numeric feature values of 16 features. Using a Python program, we transformed the features into TF-IDF form and converted the feature values into the ARFF file format. Table 4.2 shows the total 16 features that we used in our gender-based classification.

We used five machine learning algorithms: Bayesian Network, Sequential Minimal Optimization (SMO), Classification Via Regression, Decision Table, and Random Forest to build the classification models. Of these five algorithms, the Bayes Net algorithm is the

Table 4.1: Total records used for experiments $1-9$.

Experiment	Total records
1	296
2	228
3	248
4	212
5	564
6	314
7	168
8	480
9	356

only one that required discretized values. Thus we applied the discretize filter prior to training using the Bayes Net algorithm. Then we applied the 10-fold cross-validation technique and evaluated the models' performance. The steps followed for the first 11 experiments are shown in Figure 4.1. The same process was used for experiments $1-9$, using data sets $1-9$. Results are given in Section 4.5.

4.4.2 Experiment 10

In experiment 10 our goal was to classify the programs based on the gender of the programmers, however in this experiment our data set was a mix of different problems. We used data set 10 (described in Section 3.6.1) and calculated the feature values for 2866 data instances. We followed the same approach as in experiments $1-9$.

4.4.3 Experiment 11

In experiment 11 we classified the C++ programs based on the programmer's region (Eastern region or Western region). We used data set 11 for this experiment. We calculated the numeric feature values representing the data set and prepared the data instances in ARFF file format. In experiment 11 we used 5842 data instances, 16 features and five algorithms to develop models to predict the classes. The 16 features consisted of the same features as shown in Table 4.2 except instead of continent, we used gender. As before, we applied

Table 4.2: Features used in gender-based classification.

Number	Features
1	Lines of code (LOC)
2	Physical executable lines of code (SLOCP)
3	Logical executable lines of code (LLOC)
4	Comments
5	Blanklines
6	Cyclomatic complexity
7	Number of function
8	Program vocabulary
9	Program length
10	Calculated estimated program length
11	Volume
12	Difficulty
13	Effort
14	Time
15	Memory
16	Continent

10-fold cross-validation to evaluate the performance of the classification models. As in experiments $1-10$ Figure 4.1 shows the steps that we followed to conduct experiment 11. The results are given in Section 4.5.11.

4.4.4 Experiment 12

In experiment 12 our goal was to see whether we could reduce the features and still classify the programs with comparable accuracy. Therefore, we reduced the number of features from 16 to 6 and classified the programs based on the gender of the programmers. We used data set 10 (described in Section 3.6.1) and calculated the feature values for 2866 data instances using all 16 features. Then we transformed the data set into TF-IDF form and prepared the data set in the ARFF file format. We next used the 'InfoGainAttributeEval' filter of *WEKA* and selected the top 6 features based on the Information Gain values (described in Section 2.5.3) of the attributes. Applying the remove filter, we removed 10 features and reduced the number of features to the 6 which are shown in Table 4.3. Results are shown in

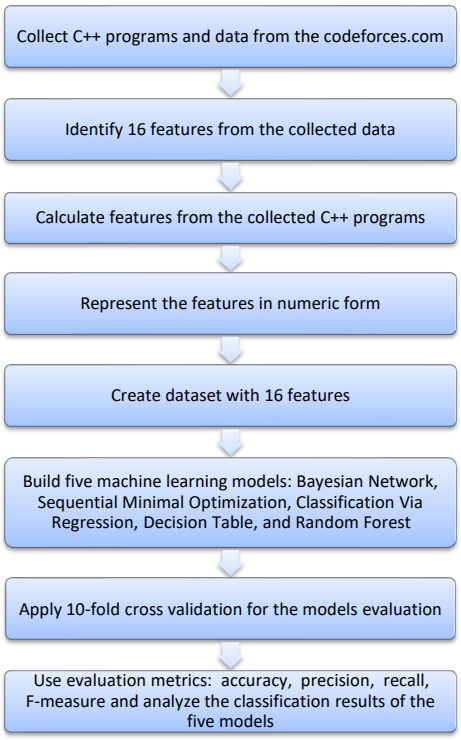

Figure 4.1: Steps followed for experiments 1-11.

Section 4.5.12.

Table 4.3: Reduced six features for gender-based classification.

Number	Features
1	Memory
2	Cyclomatic complexity
3	Time
4	Program Length
5	Physical executable lines of code (SLOCP)
6	Volume

4.4.5 Experiment 13

In experiment 13 we used 5842 data instances and 6 features to predict the class labels Eastern and Western. We again reduced the number of features from 16 to 6, but for this experiment we classified the C++ programs based on the region (Eastern, Western) of the

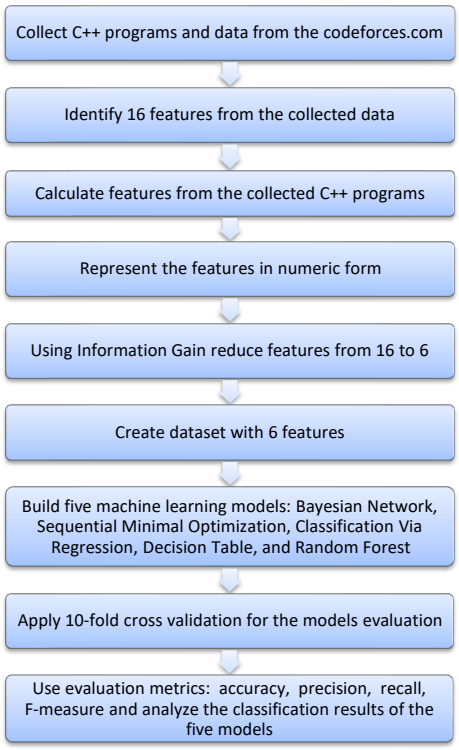

Figure 4.2: Steps followed for experiments 12-13.

programmer. We used data set 11 to perform the experiment. We selected the top 6 features based on the Information Gain values (described in Section 2.5.3) of the attributes. Table 4.4 shows the top 6 features selected for this experiment. Figure 4.2 shows the steps that we followed for experiment 13. The results of experiment 13 are discussed in Section 4.5.13.

Table 4.4: Reduced six features for region-based classification.

Number	Features
1	Memory
2	Time
3	Cyclomatic complexity
4	Program vocabulary
5	Calculated estimated program length
6	Difficulty

4.5 Results

This section describes the classification results of the 13 experiments that were performed using 11 different data sets. We tried to investigate the reasons why different models achieved the best results. Therefore, we visually analyzed 10 randomly chosen programs (5 female-written programs and 5 male-written programs) in each data set and found some interesting relationships among the classification models and the problem tags. These relationships are shown in Table 4.5. The concepts or problem tags required to solve the problems in data sets $1-9$ are shown in Table 4.6.

Table 4.5: Relationships between the problem tags and the classification models.

Serial Number	Problem Tag	Model
1	Dynamic Programming/ Depth-first search	Bayes Net
2	Tree/ Graph/ Search	Decision Table
3	Structure/ Matrix	Random Forest
4	Greedy	Classification Via Regression
5	Hashing	Sequential Minimal Optimization

Table 4.6: Concepts required to solve the problems in data sets $1-9$.

Problem Id	Data set number	Problem tags
1288D	1	Binary search, dynamic programming
1301D	2	Constructive algorithm (Greedy, Dynamic Programming)
1304E	3	Tree, graph
1324F	4	Graph, dynamic programming, depth-first search
1326D2	5	Search, greedy
1328F	6	Tree, graph, depth-first search
1332E	7	Matrix, constructive algorithm (greedy), hashing
1340B	8	Graph, dynamic programming
1344B	9	Depth-first search, graph, constructive algorithm (matrix)

4.5.1 Experiment 1

The goal of experiment 1 was to classify the programs when an equal number of male and female programmers solved a given programming problem. Table 4.7 shows the overall performance measures of the five models. The table shows that the Bayes Net model achieves the highest accuracy of 80.4%. As shown in Table 4.8, the Bayes Net model classified the highest number of male-written programs accurately at 133 (TP). Out of the 148 female-written programs, both the Bayes Net and Decision Table models accurately classified the highest number of female-written programs at 105 (TN).

As shown in Table 4.6 the problem tags for data set 1 were binary search and dynamic programming. From our visual analysis of the 10 random programs we found that the male programmers used the dynamic programming concepts more than the female programmers, and the Bayes Net model classified the highest number of male-written programs accurately. In contrast, the female programmers used the search (binary search) concept in their programs, and the Decision Table identified the highest number of female-written programs. However, the female-written programs also showed the use of dynamic programming concepts, and the Bayes Net model also classified the highest number of female-written programs. Therefore, we hypothesize that there is a relationship between the problem tags and the models (Bayes Net and Decision Table), as shown in Table 4.5. If the programmers used dynamic programming concepts, the Bayes Net model showed a better performance, whereas the Decision table model performed better when the programmers used search concepts. Further discussions are provided in Section 4.6.1.

4.5.2 Experiment 2

In experiment 2 the Classification Via Regression model showed the highest accuracy of 82.5% as shown in Table 4.9. The precision, recall, and F-measure values of the Classification Via Regression model are 83.0%, 82.5%, and 82.4%. Table 4.10 shows that the Bayes Net model correctly classified 103 (TP) male-written programs out of a total of 114 male-

Table 4.7: Cross-validation results from experiment 1 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	80.4	19.6	81.5	80.4	80.2
Sequential Minimal Optimization	75.7	24.3	76.4	75.7	75.5
Classification Via Regression	69.6	30.4	69.7	69.6	69.5
Decision Table	79.0	21.0	79.8	79.1	78.9
Random Forest	75.7	24.3	75.8	75.7	75.6

Table 4.8: Confusion matrix for experiment 1 (16 features).

Model	TP (/148)	FP (/148)	TN (/148)	FN (/148)	TP Rate (%)	FP Rate (%)
Bayes Net	133	15	105	43	80.4	19.6
Sequential Minimal Optimization	124	24	100	48	75.7	24.3
Classification Via Regression	109	39	51	97	69.6	30.4
Decision Table	129	19	105	43	79.1	20.9
Random Forest	118	30	42	106	75.7	24.3

written programs. However, the Classification Via Regression model correctly classified the highest number of female-written programs at 87 (TN).

In Table 4.6, the problem tag for data set 2 is the constructive algorithm. If a program holds the problem tag constructive algorithm, there can be many possible solutions for the problem. To obtain a suitable solution, programmers can use a variety of appropriate algorithms. In data set 2 the male programmers seemed to use dynamic programming concepts, and we see that the Bayes Net model correctly classified the highest number of male-written programs. In contrast, the female programmers used greedy concepts, and the Classification Via Regression model correctly classified the highest number of female-written programs. This suggests the existence of the relationships between the problem tags and the models (Bayes Net and Classification Via Regression) as shown in Table 4.5. If the

programmers used dynamic programming concepts, the Bayes Net model showed a better performance, whereas the Classification Via Regression model performed better when the programmers used greedy concepts. Further discussions are provided in Section 4.6.1.

Table 4.9: Cross-validation results from experiment 2 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	81.1	18.9	82.2	81.1	81.0
Sequential Minimal Optimization	64.0	36.0	64.0	64.0	64.0
Classification Via Regression	82.5	17.5	83.0	82.5	82.4
Decision Table	75.9	24.1	77.0	75.9	75.6
Random Forest	78.5	21.5	78.9	78.5	78.4

Table 4.10: Confusion matrix for experiment 2 (16 features).

Model	TP (/114)	FP (/114)	TN (/114)	FN (/114)	TP Rate (%)	FP Rate (%)
Bayes Net	103	11	82	32	81.1	18.9
Sequential Minimal Optimization	72	42	74	40	64.0	36.0
Classification Via Regression	101	13	87	27	82.5	17.5
Decision Table	98	16	75	39	75.9	24.1
Random Forest	96	18	83	31	78.5	21.5

4.5.3 Experiment 3

In experiment 3 the Decision Table model shows the highest accuracy of 81.5% as shown in Table 4.11. The precision, recall, and F-measure values of the Decision Table model are 82.3%, 81.5%, and 81.3%, respectively. Table 4.12 shows that the Decision Table model accurately classified the highest number of male-written programs at 111 (TP). The Decision Table model also correctly classified the highest number of female-written programs at 91 (TN).

As shown in Table 4.6 the problem tags for data set 3 are tree and graph. The male and female programmers both used the concepts of tree and graph. Again, the Decision table model correctly classified the highest number of male-written and female-written programs. These suggested relationships between the problem tags and the models are shown in Table 4.5. If the programmers used the tree or graph concept, the Decision Table model showed a better performance. Further discussions are provided in Section 4.6.1.

Table 4.11: Cross-validation results from experiment 3 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	77.8	22.2	78.6	77.8	77.7
Sequential Minimal Optimization	74.6	25.4	74.9	74.6	74.5
Classification Via Regression	68.1	31.9	68.2	68.1	68.1
Decision Table	81.5	18.5	82.3	81.5	81.3
Random Forest	75.8	24.2	76.2	75.8	75.7

Table 4.12: Confusion matrix for experiment 3 (16 features).

Model	TP (/124)	FP (/124)	TN (/124)	FN (/124)	TP Rate (%)	FP Rate (%)
Bayes Net	107	17	86	38	77.8	22.2
Sequential Minimal Optimization	99	25	86	38	74.6	25.4
Classification Via Regression	89	35	80	44	68.1	31.9
Decision Table	111	13	91	33	81.5	18.5
Random Forest	102	22	86	38	75.8	24.2

4.5.4 Experiment 4

In experiment 4 the Bayes Net and Decision Table demonstrate the same highest accuracy of 81.6% as shown in Table 4.13. Again, Table 4.14 shows that the Bayes Net model correctly classified the highest number of male-written programs at 98 (TP). However, the

Decision Table model correctly classified the highest number of female-written programs at 81 (TN).

As shown in Table 4.6 the problem tags for data set 4 were graph, dynamic programming and depth-first search. From the visual analysis we saw that the male programmers used the concept of dynamic programming or depth-first search, and the Bayes Net model correctly classified the highest number of male-written programs. In contrast, the female programmers used primarily graph concepts and we see that the Decision Table model correctly classified the highest number of female-written programs. Again, these proposed relationships between the problem tags and the models (Bayes Net and Decision Table) are shown in Table 4.5. If the programmers used dynamic programming or depth-first search concepts in their programs, the Bayes Net model performs well, whereas the Decision Table model showed a better performance when the programmers used graph concepts. Further discussions are provided in Section 4.6.1.

Table 4.13: Cross-validation results from experiment 4 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	81.6	18.4	83.2	81.6	81.4
Sequential Minimal Optimization	70.3	29.7	70.7	70.3	70.1
Classification Via Regression	76.4	23.6	76.8	76.4	76.3
Decision Table	81.6	18.4	81.9	81.6	81.6
Random Forest	78.3	21.7	78.6	78.3	78.3

4.5.5 Experiment 5

In experiment 5 the Random Forest model provides the highest accuracy of 86.2% as shown in Table 4.15. Again, Table 4.16 shows that the Decision Table model correctly classified the highest number of male-written programs at 265 (TP). In contrast, the Classification Via Regression model accurately classified the highest number of of female-written

Table 4.14: Confusion matrix for experiment 4 (16 features).

Model	TP (/106)	FP (/106)	TN (/106)	FN (/106)	TP Rate (%)	FP Rate (%)
Bayes Net	98	8	75	31	81.6	18.4
Sequential Minimal Optimization	82	24	67	39	70.3	29.7
Classification Via Regression	87	19	75	31	76.4	23.6
Decision Table	92	14	81	25	81.6	18.4
Random Forest	88	18	78	28	78.3	21.7

programs at 230 (TN).

As shown in Table 4.6 the problem tags for data set 5 were search and greedy. From the visual analysis we see that the male programmers used search concepts in their programs, and the Decision Table model correctly classified the highest number of male-written programs. In contrast, the female programmers used greedy concepts to write their programs and the Classification Via Regression model accurately classified the highest number of female-written programs. Although the problem tags were search and greedy, the programmers (male and female) also used arrays as the data structure. This could be why the Random Forest model showed the highest performance. If the programmers used greedy concepts the Classification Via Regression model performs better, whereas the Decision Table performed well when search concepts were used. In addition, the Random Forest model performed well when the programmers used arrays as the data structure. Further discussions are provided in Section 4.6.1.

4.5.6 Experiment 6

In experiment 6 the Decision Table model showed the highest accuracy of 86.6% as shown in Table 4.17. Again, Table 4.18 shows that the Bayes Net model correctly classified the highest number of male-written programs 147 (TP). However, the Decision table model accurately classified the highest number of female-written programs at 132 (TN).

Table 4.15: Cross-validation results from experiment 5 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	83.5	16.5	84.2	83.5	83.4
Sequential Minimal Optimization	82.6	17.4	82.9	82.6	82.6
Classification Via Regression	83.7	16.3	83.7	83.7	83.7
Decision Table	85.6	14.4	86.7	85.6	85.5
Random Forest	86.2	13.8	86.5	86.2	86.1

Table 4.16: Confusion matrix for experiment 5 (16 features).

Model	TP (/282)	FP (/282)	TN (/282)	FN (/282)	TP Rate (%)	FP Rate (%)
Bayes Net	256	26	215	67	83.5	16.5
Sequential Minimal Optimization	245	37	221	61	82.6	17.4
Classification Via Regression	242	40	230	52	83.7	16.3
Decision Table	265	17	218	64	85.6	14.4
Random Forest	257	25	229	53	86.2	13.8

In Table 4.6 the problem tags to solve the problem in data set 6 were tree, graph and depth-first search. From the visual analysis of 10 programs it appeared that the male programmers used the concept of depth-first search more in their programs, and we saw that the Bayes Net model correctly classified the highest number of male-written programs. In contrast, the female-programmers used tree or graph concepts to write the programs, and the Decision table model accurately classified the highest number of female-written programs. These relationships between the problem tags and the models (Bayes Net and Decision Table) are shown in Table 4.5. In this table we can see that the Bayes Net performed well when the problem tag was depth-first search and the Decision Table showed better performance when the problem required the concept of tree or graph.

Table 4.17: Cross-validation results from experiment 6 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	85.0	15.0	86.1	85.0	84.9
Sequential Minimal Optimization	80.6	19.4	80.9	80.6	80.5
Classification Via Regression	76.1	23.9	76.4	76.1	76.0
Decision Table	86.6	13.4	86.7	86.6	86.6
Random Forest	81.8	18.2	82.3	81.8	81.8

Table 4.18: Confusion matrix for experiment 6 (16 features).

Model	TP (/157)	FP (/157)	TN (/157)	FN (/157)	TP Rate (%)	FP Rate (%)
Bayes Net	147	10	120	37	85.0	15.0
Sequential Minimal Optimization	134	23	119	38	80.6	19.4
Classification Via Regression	128	29	111	46	76.1	23.9
Decision Table	140	17	132	25	86.6	13.4
Random Forest	138	19	119	38	81.8	18.2

4.5.7 Experiment 7

In experiment 7 the Random Forest model demonstrates the highest accuracy of 91.7% as shown in Table 4.19. Again, Table 4.20 shows that the Sequential Minimal Optimization (SMO) model correctly classified the highest number of male-written programs at 82 (TP), whereas the Random Forest model accurately classified the highest number of female-written programs at 74 (TN).

In Table 4.6 the problem tags for data set 7 were matrix, constructive algorithm (greedy) and hashing. From the visual analysis it appeared that the female programmers used matrix approaches in their programs, while the Random Forest model accurately classified the highest number of female-written programs. In contrast, the male programmers used the concept of hashing, and the Sequential Minimal Optimization (SMO) model correctly classified the highest number of male-written programs. As before in experiments 2 and 5 we

saw a relationship between the problem tag greedy and the Classification Via Regression model. In this experiment the male and female programmers also used the concept of constructive algorithm (greedy). This may be why the Classification Via Regression model correctly classified the second highest number of male-written and female-written programs. The relationships between these problem tags and the models (Random Forest, Sequential Minimal Optimization and Classification Via Regression) are shown in Table 4.5. If the programmers used the concept of matrix the Random Forest model showed a better performance and the Sequential Minimal Optimization (SMO) model performed well when the concept of hashing was used. Again, the Classification Via Regression model performed better when the concept of greedy algorithm was used. Further discussions are provided in Section 4.6.1.

Table 4.19: Cross-validation results from experiment 7 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	88.7	11.3	89.1	88.7	88.7
Sequential Minimal Optimization	83.9	16.1	86.7	83.9	83.6
Classification Via Regression	90.5	9.5	91.1	90.5	90.4
Decision Table	88.1	11.9	88.4	88.1	88.1
Random Forest	91.7	8.3	91.9	91.7	91.7

Table 4.20: Confusion matrix for experiment 7 (16 features).

Model	TP (/84)	FP (/84)	TN (/84)	FN (/84)	TP Rate (%)	FP Rate (%)
Bayes Net	79	5	70	14	88.7	11.3
Sequential Minimal Optimization	82	2	59	25	83.9	16.1
Classification Via Regression	81	3	71	13	90.5	9.5
Decision Table	78	6	70	14	88.1	11.9
Random Forest	80	4	74	10	91.7	8.3

4.5.8 Experiment 8

In experiment 8 the Bayes Net model demonstrates the highest accuracy of 83.1% as shown in Table 4.21. Again, Table 4.22 shows that the Decision Table model correctly classified 222 (TP) male-written programs, and the Bayes Net model correctly classified the highest number of female-written programs at 178 (TN).

As shown in Table 4.6, the problem tags for data set 8 were graph and dynamic programming. From the visual analysis we observed that the female programmers seemed to use the dynamic programming concepts, while the Bayes Net model correctly classified the highest number of female-written programs. Again, the male programmers used the concept of graph in their programs and the Decision Table model correctly classified the highest number of male-written programs. The male programmers also used the concept of dynamic programming, and the Bayes Net model correctly classified the second highest number of male-written programs at 221 (TP). Again the relationships between the problem tags and the models (Bayes Net and Decision Table) are shown in Table 4.5. If the programmers used the dynamic programming concepts, the Bayes Net showed a better performance, whereas the Decision Table performed better when the programmers used the graph concepts. Further discussions are provided in Section 4.6.1.

Table 4.21: Cross-validation results from experiment 8 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	83.1	16.9	84.2	83.1	83.0
Sequential Minimal Optimization	72.3	27.7	73.2	72.3	72.0
Classification Via Regression	77.3	22.7	77.5	77.3	77.2
Decision Table	81.7	18.3	83.2	81.7	81.4
Random Forest	79.6	20.4	80.1	79.6	79.5

Table 4.22: Confusion matrix for experiment 8 (16 features).

Model	TP (/240)	FP (/240)	TN (/240)	FN (/240)	TP Rate (%)	FP Rate (%)
Bayes Net	221	19	178	62	83.1	16.9
Sequential Minimal Optimization	197	43	150	90	72.3	27.7
Classification Via Regression	197	43	174	66	77.3	22.7
Decision Table	222	18	170	70	81.7	18.3
Random Forest	206	34	176	64	79.6	20.4

4.5.9 Experiment 9

In experiment 9 the Bayes Net model shows the highest accuracy of 79.5% as shown in Table 4.23. However, Table 4.24 shows that the Bayes Net model also correctly classified the highest number of male-written programs at 159 (TP). Out of 178 female-written programs, the Decision Table and Random Forest models each accurately classified the same highest number of female-written programs at 128 (TN).

From Table 4.6 we can see that the problem tags for data set 9 were depth-first search, graph and constructive algorithm (matrix). From the visual analysis of 10 programs it seemed that the male programmers used the depth-first search concept, and the Bayes Net model correctly classified the highest number of male-written programs. In contrast, the female programmers used the concepts of graph and matrix in their programs. Again, the Decision Table and Random Forest model each accurately classified the same highest number of female-written programs. The resulting (hypothesized) relationships between the problem tags and the models (Bayes Net, Decision Table and Random Forest) are shown in Table 4.5. If the programmers used the depth-first search concept, the Bayes Net model showed a better performance, whereas the Decision Table and Random Forest models performed better when the programmers used the concepts of graph and matrix in their programs, respectively.

Table 4.23: Cross-validation results from experiment 9 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	79.5	20.5	80.7	79.5	79.3
Sequential Minimal Optimization	74.2	25.8	74.4	74.2	74.1
Classification Via Regression	73.3	26.7	73.6	73.3	73.2
Decision Table	78.9	21.1	79.5	78.9	78.8
Random Forest	77.5	22.5	77.9	77.5	77.5

Table 4.24: Confusion matrix for experiment 9 (16 features).

Model	TP (/178)	FP (/178)	TN (/178)	FN (/178)	TP Rate (%)	FP Rate (%)
Bayes Net	159	19	124	54	79.5	20.5
Sequential Minimal Optimization	141	37	123	55	74.2	25.8
Classification Via Regression	140	38	121	57	73.3	26.7
Decision Table	153	25	128	50	78.9	21.1
Random Forest	148	30	128	50	77.5	22.5

4.5.10 Experiment 10

In experiment 10 all the programs of from data sets 1-9 were combined for the gender-based classification using 16 features. Data set 10 was a combination of data sets $1-9$ and contained all the problems tags. All of the five models show an accuracy over 81.0%. As shown in Table 4.25 the Random Forest model showed the highest accuracy of 86.4%. Table 4.26 shows that the Sequential Minimal Optimization (SMO) model correctly classified the highest number of male-written programs at 1295 (TP). In contrast, the Random Forest model accurately classified the highest number of female-written programs at 1201 (TN). Further discussion is given in Section 4.6.1.

Table 4.25: Cross-validation results from experiment 10 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	83.5	16.5	83.9	83.5	83.4
Sequential Minimal Optimization	85.9	14.1	86.2	85.9	85.9
Classification Via Regression	84.5	15.5	84.5	84.5	84.5
Decision Table	81.0	19.0	81.4	81.0	81.0
Random Forest	86.4	13.6	86.5	86.4	86.4

Table 4.26: Confusion matrix for experiment 10 (16 features).

Model	TP (/1433)	FP (/1433)	TN (/1433)	FN (/1433)	TP Rate (%)	FP Rate (%)
Bayes Net	1277	156	1116	317	83.5	16.5
Sequential Minimal Optimization	1295	138	1167	266	85.9	14.1
Classification Via Regression	1222	211	1199	234	84.5	15.5
Decision Table	1237	196	1085	348	81.0	19.0
Random Forest	1276	157	1201	232	86.4	13.6

4.5.11 Experiment 11

In experiment 11 we built five classification models to classify the region of the programmer. The Random Forest model has the highest accuracy of 75.2% as shown in Table 4.27. Again, Table 4.28 shows that the Classification Via Regression model classified the highest number of programs from the Eastern region accurately at 2309 (TP), and the Bayes Net model correctly classified the highest number of programs from the Western region at 2311 (TN). Further discussions are given in Section 4.6.2.

4.5.12 Experiment 12 (Gender-based classification with reduced features)

In experiment 12 our goal was to see whether the reduced features could classify the programs with comparable accuracy. In this experiment the Bayes Net model demonstrates the highest accuracy of 84.6% as shown in Table 4.29. Table 4.30 shows that the Bayes Net

Table 4.27: Cross-validation results from experiment 11 (16 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	69.6	30.4	70.3	69.6	69.3
Sequential Minimal Optimization	72.9	27.1	73.2	72.9	72.8
Classification Via Regression	74.4	25.6	74.6	74.4	74.3
Decision Table	70.0	30.0	70.3	70.0	69.9
Random Forest	75.2	24.8	75.3	75.2	75.2

Table 4.28: Confusion matrix for experiment 11 (16 features).

Model	TP (/2921)	FP (/2921)	TN (/2921)	FN (/2921)	TP Rate (%)	FP Rate (%)
Bayes Net	1753	1168	2311	610	69.6	30.4
Sequential Minimal Optimization	2280	641	1980	941	72.9	27.1
Classification Via Regression	2309	612	2036	885	74.4	25.6
Decision Table	2221	700	1870	1051	70.0	30.0
Random Forest	2245	676	2151	770	75.2	24.8

model correctly classified the highest number of male-written programs at 1305 (TP). However, the Classification Via Regression and the Random Forest models accurately classified the same highest number of female-written programs at 1177 (TN). We observed that the reduced features were successful to achieve comparable results to experiment 10. Further discussions are provided in Chapter 5, Section 5.1.1.

4.5.13 Experiment 13 (Region-based classification with reduced features)

In experiment 13 we reduced the number of features from 16 to 6 to see whether the reduced features could achieve comparable results to experiment 11. In this experiment the Classification Via Regression model has the highest accuracy of 73.6% as shown in Table 4.31. Again, Table 4.32 shows that the Classification Via Regression model also accurately classified the highest number of programs from the Eastern region at 2301 (TP).

Table 4.29: Cross-validation results from experiment 12 (6 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	84.6	15.4	85.2	84.6	84.5
Sequential Minimal Optimization	83.6	16.4	83.8	83.6	83.5
Classification Via Regression	84.0	16.0	84.0	84.0	84.0
Decision Table	79.2	20.8	79.8	79.2	79.1
Random Forest	84.0	16.0	83.9	83.9	83.9

Table 4.30: Confusion matrix for experiment 12 (6 features).

Model	TP (/1433)	FP (/1433)	TN (/1433)	FN (/1433)	TP Rate (%)	FP Rate (%)
Bayes Net	1305	128	1120	313	84.6	15.4
Sequential Minimal Optimization	1261	172	1134	299	83.6	16.4
Classification Via Regression	1230	203	1177	256	84.0	16.0
Decision Table	1235	198	1035	398	79.2	20.8
Random Forest	1227	206	1177	256	83.9	16.1

In contrast, the Bayes Net model accurately classified the highest number of programs from the Western region at 2173 (TN). Further discussions are provided in Chapter 5, Section 5.1.2.

4.6 Discussion of experimental results

In this section, we discuss the summary of the experimental results. We tried to investigate the best models based on the performance measures of the 13 experimental results. A description of the findings is given below.

Table 4.31: Cross-validation results from experiment 13 (6 features).

Model	Correctly Classified (%)	Incorrectly Classified (%)	Precision (%)	Recall (%)	F-measure (%)
Bayes Net	72.0	28.0	72.0	72.0	72.0
Sequential Minimal Optimization	72.8	27.2	73.1	72.8	72.7
Classification Via Regression	73.6	26.4	73.8	73.6	73.5
Decision Table	67.5	32.5	67.8	67.5	67.4
Random Forest	72.7	27.3	72.7	72.7	72.7

Table 4.32: Confusion matrix for experiment 13 (6 features).

Model	TP (/2921)	FP (/2921)	TN (/2921)	FN (/2921)	TP Rate (%)	FP Rate (%)
Bayes Net	2030	891	2173	748	72.0	28.0
Sequential Minimal Optimization	2283	638	1970	951	72.8	27.2
Classification Via Regression	2301	620	1997	924	73.6	26.4
Decision Table	2147	774	1799	1122	67.5	32.5
Random Forest	2125	796	2122	799	72.7	27.3

4.6.1 Gender-based classification discussion

Individual problem based discussion

- We analyzed the relationships between the problem tags and the models to find a connection between them. The Bayes Net model showed better performance when the problem tags were the dynamic programming and depth-first search. To create a Bayesian Network the concept of depth-first search and dynamic programming are used [38]. This is likely why the Bayes Net model showed a better performance when the programmers used depth-first search and dynamic programming concepts in their programs.

- Again, the concept of graph and tree are used for creating the decision tree of a Decision Table model [40]. This could explain why the Decision Table model performed

better when the problem tags were tree and graph.

- A structure creates a data type that can be used to categorize different items into a single type. However, a Random Forest model creates multiple decision trees and based on the majority decisions of the trees, the model classifies an instance to a specific class. To identify the structure in the data, a Random Forest model generates a matrix [24]. Since Random Forest uses the concept of matrix and structure, this could explain why the model was successful in identifying the male-written and female-written programs that used the concept of matrix and structure.

- In contrast, the Classification Via Regression model uses the original data set to generate several new data sets for each class label. Using a model tree and a linear regression function, the model generates probabilities about the class of each instance. A greedy algorithm makes one greedy choice continuously and reduces each given problem into a smaller one. Researchers have used the concept of greedy algorithm to address the linear regression problems [34]. This may be a key factor in why the Classification Via Regression model showed a better performance when the programmers used the greedy algorithm concept.

- In [23] the authors demonstrated that hashing can be used with the Support Vector Machines learning algorithm. Hence, this could be why the Sequential Minimal Optimization model performed better when the programmers used the concept of hashing in their programs.

- One of the possible reasons for the differences in the performance of the five models is the problem tags or concepts used to solve the problems. We can choose a particular model for better if we know what the underlying problem tags are. The models were successful in identifying the classes potentially because of the underlying effect of the problem tags and how the programs were designed based on these tags.

- In gender-based (individual problem) classification tasks (experiments $1-9$), the Bayes Net model generally performed better than the other models. As shown in Table 4.33 the Bayes Net model achieved the highest accuracy in four experiments (experiments 1, 4, 8, and 9). In Table 4.6 the problem tags for the data sets 1, 4, 8, and 9 show the requirement of either dynamic programming or depth-first search concept. From 4.5 we can see that the Bayes Net model performed well when the problem tags were the dynamic programming and depth-first search. This could be why the Bayes Net model showed efficient performance in these four experiments.

- In contrast, the Sequential Minimal Optimization model showed the lowest accuracy, in particular for experiments 2, 4, 5, 7, and 8. From Table 4.5 we can see that the Sequential Minimal Optimization model performed well when the problem tag was hashing. Although data sets 5 and 7 required the concept of hashing, the overall uses of the other tags were higher in the submitted programs. This could be the possible reason why the other models performed well in these five experiments and the Sequential Minimal Optimization model showed the lowest performance.

- The Bayes Net model classified the highest number of male-written programs for five experiments out of the nine experiments. Using 16 features, the Bayes Net model correctly classified the highest number of male-written programs in experiments 1, 2, 4, 6, and 9. To solve the problems in data sets 1, 2, 4, 6, and 9, the male programmers used more dynamic programming or depth-first search concept in their programs than the other concepts. This could be why the Bayes Net model classified the highest number of male-written programs.

- In five experiments, the Decision Table model performed better than the other models and correctly classified the highest number of female-written programs. The Decision Table model using 16 features accurately classified the highest number of female-written programs in experiments 1, 3, 4, 6, and 9. To solve the problems in data

sets 1, 3, 4, 6, and 9, the female programmers used more tree or graph concepts than the other concepts. This could be a possible reason why the Decision Table model classified the highest number of female-written programs.

- Analyzing the results of experiments $1-9$, we observed that in all experiments the male-written and female-written programs were classified by different models, except experiment 3. In experiment 3 there was limited scope to apply different concepts, therefore the male and female programmers used the same concepts. In other eight experiments the male and female programmers seemed to use different approaches or problem tags while solving the same problems. That could be why different models classified male-written and female-written programs based on the use of different problem tags. We can also hypothesize that male and female programmers use different approaches when they solve a particular problem. Thus we can see a gender-based variations in the computer programs of the computer programming contests. Therefore, these observations give the possible answers to our two research questions: Do sociolinguistic variations exist in the computer programs of computer programming contests? Do male and female contestants write programs differently to solve a particular programming contest problem?

Combined problem based discussion (with 16 features)

- Table 4.34 demonstrates that the Random Forest has the highest accuracy in the gender (combined problem) based classification. The Random Forest model using 16 features showed an accuracy of 86.4%, which is the highest accuracy of the five models. From a total of 1433 records (in data set 10), 732 records were from the data sets 5 and 7. From Table 4.1 we can see that the experiments 5 and 7 used the data sets 5 (564 records) and 7 (168 records), respectively. In these two experiments the Random Forest model showed the highest performance. In experiment 10 we used the combined data set 10 where the same records of the data sets 5 and 7 were

Table 4.33: Comparison of the best and worst models for gender-based (individual problem) classification.

Experiment	Best Model	Accuracy of the Best model	Worst Model	Accuracy of the Worst model
1	**Bayes Net**	**80.4%**	Classification Via Regression	69.6%
2	Classification Via Regression	82.5%	**Sequential Minimal Optimization**	**64.0%**
3	Decision Table	81.5%	Classification Via Regression	68.1%
4	**Bayes Net, Decision Table**	**81.6%**	**Sequential Minimal Optimization**	**70.3%**
5	Random Forest	86.2%	**Sequential Minimal Optimization**	**82.6%**
6	Decision Table	86.6%	Classification Via Regression	76.1%
7	Random Forest	91.7%	**Sequential Minimal Optimization**	**83.9%**
8	**Bayes Net**	**83.1%**	**Sequential Minimal Optimization**	**72.3%**
9	**Bayes Net**	**79.5%**	Classification Via Regression	73.3%

present. This could be why the Random Forest model showed the highest performance in experiment 10. Again, data set 10 was comparatively large and the Random Forest model performs well for the larger data sets [4]. These are likely the reasons the Random Forest model showed the highest accuracy.

- The Sequential Minimal Optimization model using 16 features (experiment 10) accurately classified the highest number of male-written programs. The Sequential Minimal Optimization (SMO) model performed well when the programmers used the concept of hashing. Hashing is a data structure that uses the array and mapping

concepts. From the visual analysis we found that the male-written programs used the concept of array or mapping in their programs, this could be a possible reason that the SMO model correctly classified the highest number of male-written programs.

- In contrast, the female-written programs seemed to use the matrix and structures in their programs, this could be why the Random Forest model correctly classified the highest number of the female-written programs.

Table 4.34: Comparison of the best and worst models for gender-based (combined problem) classification.

Experiment	Best Model	Accuracy of the Best model	Worst Model	Accuracy of the Worst model
10	Random Forest	86.4%	Decision Table	81.0%
12	Bayes Net	84.6%	Decision Table	79.2%

4.6.2 Region-based classification discussion (with 16 features)

- As shown in Table 4.35, the Random Forest shows the highest accuracy in the region-based classification tasks (experiment 11). The Random Forest model using 16 features performed better with an accuracy of 75.2% than the other models. We used the submissions of all of the nine problems and prepared the data set 11. Therefore, the data set 11 contained all the problem tags that are shown in Table 4.6. From the visual analysis we found that the programmers from the Eastern and Western regions both used the concepts of structure and matrix. From Table 4.5 the Random Forest model showed better performance when the problem tags were structure and matrix. Again, data set 11 was comparatively large and the Random Forest model performed well for the larger data sets [4]. These are likely the reasons the Random Forest model showed the highest accuracy.

- We observed that the Classification Via Regression model correctly identified the highest number of programs from the Eastern region. From the visual analysis of

the 10 random programs we saw that the programmers from the Eastern region used the greedy concept in their programs, and the Classification Via Regression model classified the highest number of programs from the Eastern region. As shown in Table 4.5 the Classification Via Regression model showed a better performance when the programmers used the concept of greedy algorithm.

- In contrast, the programmers from the Western region seemed to use the concept of dynamic programming and depth-first search, and the Bayes Net model correctly classified the programs from the Western region. As shown in Table 4.5 the Bayes Net model performed better when the programmers used the concept of dynamic algorithm or depth-first search.

Table 4.35: Comparison of the best and worst models for region-based classification.

Experiment	Best Model	Accuracy of the Best model	Worst Model	Accuracy of the Worst model
11	Random Forest	75.2%	Bayes Net	69.6%
13	Classification Via Regression	73.6%	Decision Table	67.5%

4.7 Limitations

There may be some potential threats to the validity of this research.

- In this research, the gender of a programmer is predicted based on the first name of that person. A programmer may use a name to hide his/her identity or other errors may occur in the gender prediction process. Therefore, there may be some inaccurate information regarding the gender of the programmer.

- Country and city information is provided by programmers, and we use this to determine the origin of the programmer. If the country information is inaccurate, this could impact the model development.

- We used only three types of features based on software metrics for this research. There may be other features that may give us better results. Some suggestions on this are given in Section 6.1 Future Work.

- In codeforces the number of female-written programs is significantly lower than the male-written programs. If we could explore more female-written programs, the outcome might differ from our current analysis. For a small number of data the risk of over-fitting may occur which may lead to an inaccurate representation of all female programmers. This is a potential threat to the validity of this research.

Chapter 5

Discussion

This chapter discusses the statistical analysis of the features that are used to determine the gender-based and region-based programming styles of the contest programmers.

5.1 Reduction of Features

In experiments 10 and 11 (as described in Chapter 4), we used 16 features for the classification tasks. Using the 'InfoGainAttributeEval' attribute evaluator of *WEKA*, we reduced the number of features from 16 to 6 to conduct experiments 12 and 13. The selected 6 features for gender-based classification were

- memory,
- cyclomatic complexity,
- time,
- program length,
- slocp, and
- volume.

Again, for region-based classification the selected 6 features were

- memory,
- cyclomatic complexity,

- time,
- program vocabulary,
- calculated estimated program length, and
- difficulty.

Using these features, we built new classification models for experiments 12 and 13. Then, we compared the results to determine whether the feature reduction approach helped to improve the models' performance.

5.1.1 Gender-based classification

We hypothesized that analysis of the computer programs from programming contests can help to determine the differences in programming styles of groups of programmers, particularly male and female programmers. Table 5.1 shows the comparative analysis of the performance of the five classification models for gender-based classification. Overall we found that the Bayes Net model's accuracy improved slightly with the reduced features among the five classification models. The accuracy of the other classification models decreased by 0.5% to 2.6%. Although the Random Forest model achieved the highest accuracy (86.4%) using 16 features, the Bayes Net model's performance increased with the reduction to 6 features. However, the removal of the irrelevant features had minor effect on the performance of all the models and the changes in the performance might be evaluated as insignificant. Overall, the models using 6 features were successful in achieving results comparable to the models using 16 features.

The Bayes Net model creates a directed acyclic graph that consists of nodes and edges, where each attribute or feature is represented by a node. As described in Section 2.5.2, the Bayes Net model calculates the conditional probability for each attribute value and the attribute values with maximum probabilities are selected for the classification of an instance. When the number of attributes is reduced, consequently the number of nodes in

the Bayes Net graph will also decrease. Using the reduced number of nodes, the Bayes Net model can create a less complex probabilistic model and predict the most important relationships among the features (represented as nodes). This could be why the performance of the Bayes Net model increased with reduced features. However, the Sequential Minimal Optimization model uses a function, the Classification Via Regression model uses a model tree and a linear regression algorithm, the Decision Table creates a tree and the Random Forest model uses multiple decision trees which all may perform better for larger data sets using more features. This is one possible reason that these models showed a decrease in performance.

Table 5.1: Accuracy in gender-based classification.

Model	**Accuracy (%)**	
	16 features	6 features
Bayes Net	83.5	84.6
Sequential Minimal Optimization	85.9	83.6
Classification Via Regression	84.5	84.0
Decision Table	81.0	79.2
Random Forest	86.4	84.0

5.1.2 Region-based classification

Programmers from many different parts of the world participate in programming contests. Therefore, we hypothesized that programming contest data can also be analyzed to determine the effect of region on computer programs. Again, we tried to investigate whether the reduction of the features had any significant effect on the performance of the five models. In Table 5.2, the performance of the models with 16 and 6 features is shown. After reducing the features, again only the Bayes Net model saw an increase in accuracy (from 69.9% to 72.0%, or about 2.1%). A possible reason for the Bayes Net model's slightly improved performance is the probabilistic approach underlying this model (as described in Section 4.6.2 and in Section 2.5.2). Of the five models using 6 features, the Classification Via Regression model achieved the highest accuracy of 73.6%. Although the Random

Forest model with 16 features shows the highest accuracy (75.2%), the performance of the Random Forest model slightly decreased by 2% with the reduced 6 features.

The Classification Via Regression model is a meta learning algorithm that incorporates a decision tree and a linear regression algorithm to calculate probabilities for a specific class. The leaf nodes in the decision tree use linear regression functions to generate probabilities about the class of each instance. Based on the attribute values of each instance, the class probabilities are calculated. After the feature reduction, the leaf nodes in the decision tree use the reduced features and linear regression functions to generate probabilities about the class of each instance. If the reduced features set contains less highly correlated features, the model can generate more accurate probabilities about the class of an instance [30]. For example, in region-based classification the features lines of code and calculated estimated program length were highly correlated (as described in Section 5.3). Based on the information gain we selected the calculated estimated program length and removed the lines of code. The selection of less highly correlated features could be a reason that the Classification Via Regression model achieved the highest performance with the reduction to 6 features. However, the changes in the performance of the models were insignificant and the models using 6 features achieved very close results to the models using 16 features.

Table 5.2: Accuracy in region-based classification.

Model	**Accuracy (%)**	
	16 features	6 features
Bayes Net	69.6	72.0
Sequential Minimal Optimization	72.9	72.8
Classification Via Regression	74.4	73.6
Decision Table	70.0	67.5
Random Forest	75.2	72.7

In gender-based and region based classifications three features seemed to be present in both of the reduced features set. Time, memory and cyclomatic complexity feature values have variance for which these three features showed higher information gain values. Out of six features other three features were different in gender-based and region-based classi-

fication. In gender-based classification we can see that the feature program length is one of the top features which is measured by counting total number of operators and operands. Again, in gender-based classification the other two features are volume and slocp which measure the size of the programs. Therefore, in gender-based classification we can see that the length of the programs, total operator and operands based features provided higher information gain.

In contrast, in region based classification program vocabulary and calculated estimated program length are two of the features which are calculated using unique operator and operands. The other feature difficulty is used to measure the effort that is required to maintain a program. Therefore, in region-based classification we can see that the difficulty, unique operator and operands based features provided higher information gain.

5.2 Statistical Analysis

In this research, one goal was to determine any differences in the programming styles of male and female programmers. Our other goal was to investigate the programming styles of Eastern and Western programmers. We hypothesized that the analysis of computer programs from programming contests may help us to identify any variations in the programming styles. To answer our questions, we conducted a total of 13 experiments with five classification models. The experimental results demonstrated that the models performed with an average 84.5% accuracy for gender-based (combined problem) classification and 72.4% accuracy for region-based classification. To determine the significance of the results we performed a statistical test termed a 'T-test'. This statistical test investigates the difference between the two groups and compares the mean or average values between them [35]. For an example, in gender-based classification data sets we have two classes, male and female with 15 features. For each feature we conducted a T-test and compared the average values of the male and female classes to determine any difference.

In a T-test, two hypotheses are tested in contrast to the other; that is, a null hypothesis

H_0 is tested against an alternative hypothesis H_A. The null hypothesis states that there is no remarkable difference between the mean of the two groups. In contrast, the alternative hypothesis states that there are effective differences between the two groups. To demonstrate that the difference is authentic or coincidental, ρ value is calculated during the T-test. We used the T-test formula $= T.TEST(array1, array2, tails, type)$ in Microsoft Excel©. We used the male data in $array1$, the female data in $array2$, $tail = 2$ for two tailed distribution, and $type = 2$ for two-sample equal variance T-test.

In general, a threshold ρ value of 0.05 is used for the hypothesis testing. If $\rho < 0.05$, H_0 is rejected and H_A is accepted; this also indicates that the statistical difference between the two groups is authentic. Alternatively, if $\rho > 0.05$, H_0 is accepted and H_A is rejected; this indicates that the statistical difference between the two groups is not significant.

5.2.1 Gender-based statistical analysis

We performed a T-test for individual features to see if there was a statistically significant difference in each feature values for each group. We conducted a two-tailed T-test and determined whether the mean of one group is similar to the other group [6]. In these T-tests the gender (male or female) of the programmer is used as an independent variable and the features are used as dependent variables. We performed T-tests for the individual problem based data sets (where the programmers solved the same problems) as well as the combined problem data set (where the programmers solved different problems).

Individual problem based analysis

We first performed T-tests for the 15 features on data sets $1-9$ to investigate any statically significant difference in each feature for male-written and female-written programs when they solved the same problems. In Appendix B, Table B.1 shows that 13 features have ρ values less than 0.05. For these 13 features, the null hypothesis H_0 (there is no difference between the two groups) is rejected, and the alternative hypothesis H_A (there is difference between the two groups) is accepted. Therefore, these features show statistically significant

differences between the two groups of programmers (male, female) when they solved the same problem. These 13 features are loc, slocp, lloc, comments, blanklines, cyclomatic complexity, number of function, program vocabulary, program length, calculated estimated program length, volume, difficulty, and effort.

As shown in Table B.1 in Appendix B, the ρ values for time (for seven data sets out of 9 data sets) and memory (for four data sets out of 9 data sets) are greater than 0.05. For these two features, the null hypothesis H_0 is accepted and alternative hypothesis H_A is rejected. Thus we observed that time and memory seem to have no impact on the results when male and female programmers solved the same programming problem. The usage of time and memory were almost equal in male-written and female-written programs. This may be the reason that the differences for these two features (time and memory) do not appear to be statistically significant.

Next, we performed a comparative analysis of the statistically significant features between the male-written and female-written programs using the nine single-problem data sets. In Appendix C, Table C.1 shows that female-written programs have significant usage of 6 features: comments, effort, loc, volume, blanklines, and number of functions. The use of these 6 features are significantly higher (from 62% to 75%) in female-written programs than in male-written programs, when the programmers solved the same problem. The usage of the other 6 features are moderately higher (from 52% to 61%) in female-written programs.

The female programmers used more lines of code, logical lines of code, physical lines of code, comments, and blanklines in their programs than the male programmers. While solving the same programming problems, female programmers used more operators, operands and wrote difficult programs that require more effort. For example, the use of more variables (as operands) and arithmetic operators will increase the volume of a program. This increase in volume will make the program more complex, and the programmers have to give more effort to maintain a difficult programs. Since female-written programs used more op-

erators and operands, this usage of operators and operands increased the volume of their programs.

Combined problems based analysis

We next performed T-tests on the results from data set 10 to investigate the statistically significant difference in each feature for male-written and female-written programs when they solved different problems. In Appendix B, Table B.2 demonstrates that time is the only feature that has ρ value greater than 0.05. For this feature, H_0 (there is no difference between the two groups) is accepted, and H_A (there is difference between the two groups) is rejected. Therefore, time does not show statistically significant differences between the two groups when the programmers solved different problems. The use of time was almost equal in male-written and female-written programs. This could be the reason that time do not appear to be statistically significant. Table B.2 also shows that 14 features have ρ value less than 0.05. For these 14 features, H_0 is rejected and H_A is accepted. These 14 features are loc, slocp, lloc, comments, blanklines, cyclomatic complexity, number of function, program vocabulary, program length, calculated estimated program length, volume, difficulty, memory, and effort.

Next we conducted a comparative analysis of the statistically significant features between the male and female-written programs when they solved different problems. 13 of the statistically significant features are the same as when we used the 9 data sets (individual problem).

In Appendix C, Table C.2 shows the relative usage of the features. From this table we can see that the male-written and female-written programs show similar usage of the features when they solved a particular problem. As before, the female programmers seemed to use more logic decisions, and write more longer and difficult programs than the male programmers when they solved different problems. Therefore, these features may help to identify the programming styles of the male and female programmers when they solve a par-

ticular or different problems. Additionally, they may tell us about the different approaches used by male and female programmers. We can hypothesize that male and female programmers write programs differently regardless of whether they solve the same problems or different problems.

5.2.2 Region-based statistical analysis

We also performed T-tests on the region-based data set to determine the difference between the programs from Eastern and Western regions. In this two-tailed T-test, we treated the region of the programmer (Eastern or Western) as the two groups and performed T-tests on 15 features. In Appendix B, Table B.3 demonstrates that 12 features have a ρ value less than 0.05. For these 12 features, H_0 is rejected and H_A is accepted and we can say that these 12 features are statistically significant features for region-based data set. These 12 features are loc, lloc, comments, blanklines, number of function, program vocabulary, program length, calculated estimated program length, volume, difficulty, memory, and time.

We observed that the cyclomatic complexity, slocp and effort have ρ value greater than 0.05. For these three features, H_0 (there is no difference between the two groups) is accepted, and H_A (there is difference between the two groups) is rejected. We found that except these 3 features the other 11 features are statistically significant in both gender-based (combined problems) and region-based data sets. As the size of the gender and region-based data sets are large, these statistically significant 11 features are the same for both of the data sets.

In Appendix C, Table C.3 shows a comparative analysis of features between the programs from Eastern and Western regions. From the table, we observed that the use of time (70%) appears to be significantly higher in the programs from the Western region than from the Eastern region. The features volume, calculated estimated program length, and difficulty are also higher (56%) in the programs from the Western region. In contrast, the programs from the Eastern region used more memory (55%) and blanklines (52%) than

the Western region. We can thus hypothesize that the 11 statistically significant features as shown in Table 5.3 can help to identify the programming styles based on the gender and region of the programmers.

Table 5.3: Statistically important features in gender-based (combined problem) and region-based data sets.

Feature number	Features
1.	Loc
2.	Lloc
3.	Comments
4.	Number of function
5.	Program vocabulary
6.	Program length
7.	Calculated estimated program length
8.	Volume
9.	Difficulty
10.	Blanklines
11.	Memory

5.3 Relationships between features

We next performed correlation analysis to investigate the relationships between pairs of features. We were interested to know which pairs of features are linearly dependent on each other for each data set. A linear relationship means that the increase or decrease in one feature value will cause an equivalent increase or decrease in the other feature value. We tried to determine the relationship between the features by calculating the correlation coefficient (γ) value between each pair of features. For gender-based experiments, we measured the γ value for male-written programs and female-written programs. From the statistical T-tests we found that the analysis for the smaller data sets (individual problems) gave the same results for the larger data set (combined problems). Therefore, we did not perform correlation analysis for the smaller gender-based data sets (individual problems), rather we performed correlation analysis for the larger gender-based data set (combined problems).

Similarly, we also measured the γ value for programs from the Eastern and Western regions for our region-based research. The value of γ ranges from -1 to $+1$ [46]. Based on the value of γ, there can be three types of relationship between a pair of features:

- For $\gamma = 0$, the pair of features is said to be independent of each other, and there is no relationship between the features.
- For $\gamma > 0$, the pair of features is positively correlated and if the frequency of one feature increases, the frequency of the other feature will also increase. For higher γ values ($\gamma > 0.5$), the features are said to be strongly correlated.
- For $\gamma < 0$, the pair of features is negatively correlated. If the frequency for one feature increases, the frequency of the other feature will decrease.

Relationship between features for gender-based data set

In Appendix D, the correlation matrix of features for the gender-based data set is shown in Table D.1. From Table D.1, there are 3 pairs of features that have $\gamma = 0$. Thus, these features are independent of each other. These 3 pairs of features are

- time with memory,
- time with program vocabulary, and
- time with calculated estimated program length.

The program execution time depends on factors such as the algorithm time complexity, input data size, instructions and CPU speed [47] while the required memory depends on other factors such as the processor, operating system, and registers. Again, an increase in program vocabulary means more unique operators and operands are used. Hence, time does not depends on how many unique operators or operands are used. This may be a possible reason that time is independent of the memory and the number of the unique operators and operands.

As shown in Table D.1, there exist 12 pairs of features for which $\gamma < 0$. The pairs of features which are negatively correlated are listed in Table 5.4. As we can see time has negative correlation with 10 features which are used as instructions to the computer. This is logical, as the use of these features will certainly influence the performance of the execution time. For example, if a programmer gives more effort and writes an efficient program, then the execution time may be reduced.

Table 5.4: Negatively related pairs of features for gender-based data set.

Pair number	Pairs of features
1.	Time, Program length
2.	Time, Volume
3.	Time, Difficulty
4.	Time, Effort
5.	Time, Loc
6.	Time, Slocp
7.	Time, Lloc
8.	Time, Blanklines
9.	Time, Cyclomatic complexity
10.	Time, Number of function
11.	Memory, Program vocabulary
12.	Memory, Calculated estimated program length

There are 90 pairs of features that have $\gamma > 0$ which means these pairs of features are positively correlated. Out of 90 pairs of positively correlated features, 33 pairs of features have $\gamma > 0.5$. Thus these 33 pairs of features are strongly correlated. In Appendix D, Table D.2 shows the pairs of features that are strongly correlated. For example, for the pair volume and difficulty $\gamma = 0.6$ thus these features are strongly correlated. Again there is a clear reason for this: if the size of the program (or volume) increases, the programs becomes more difficult to maintain. Similarly, for the pair cyclomatic complexity and number of function the value of $\gamma = 0.65$. The use of more functions with if-else statements will increase the cyclomatic complexity of a program. There are other clearly evident connections for the pairs lines of code with comments ($\gamma = 0.72$), lines of code with physical executable

lines of code ($\gamma = 0.88$), lines of code with logical executable lines of code ($\gamma = 0.78$), and lines of code with blanklines ($\gamma = 0.54$). If more comments, blanklines, physical executable lines of code and logical executable lines of code are added to a program, the value of the total lines of code will definitely increase.

Relationship between features for region-based data set

In Appendix D, Table D.3 demonstrates the correlation matrix of features for the region-based data set. In this table there are 3 pairs of features that have $\gamma = 0$, indicating that these pairs of features have no relationship between them. These pairs of features are

- time with number of functions,
- time with blanklines, and
- time with volume.

The program execution time depends on factors including the algorithm time complexity, input data size, instructions and CPU speed, and time has no linear relationship with the number of the functions used, the blanklines and the volume or the size of the programs. For example, a program using more functions to implement an inefficient algorithm may not always help to reduce the execution time.

Table 5.5: Negatively related pairs of features for region-based data set.

Pair number	Pairs of features
1.	Time, Program length
2.	Time, Comments
3.	Time, Effort
4.	Time, Loc
5.	Time, Slocp
6.	Time, Lloc
7.	Time, Cyclomatic complexity

Table D.3 shows that 7 pairs of features are negatively correlated to each other. These 7 pairs of features are listed in Table 5.5. As we can see time is negatively correlated

with 7 features. These features may negatively influence a program's execution time. For example, a program using more logic decisions and an efficient algorithm may require less execution time. In contrast, a different program using fewer logic decisions and an inefficient algorithm may require more execution time.

We noticed that 95 pairs of features have a positive correlation with each other. Out of these 95 pairs of features, 39 pairs of features are strongly correlated. In Appendix D, Table D.4 shows the pairs of strongly correlated features for the region-based data set. For example, for the pair volume with difficulty the value of $\gamma = 0.72$, thus these features are strongly correlated. If the size of the program or volume increases, the program becomes more difficult to maintain. Again, for the pair number of function with logical executable lines of code (lloc) the value of $\gamma = 0.52$. If more functions are added, this will increase the logical executable lines of code. Similarly, for the pair effort with difficulty the value of $\gamma = 0.85$, this is clearly evident that more effort will be required to maintain a difficult program.

In this research, we hypothesized that the analysis of features based on three software metrics (lines of code, cyclomatic complexity and Halstead metrics) may offer better results or tell us more about the use of these features depending on the gender and region of the programmer. We observed that 31 pairs of features are strongly correlated in both gender-based and region-based data sets. These pairs of features are shown in Table 5.6. For example, in Table 5.6 we can see that the feature physical executable lines of code (slocp) has a positive relationship with the program length which measures total number of operator and operands. If the total number of operators and operands increase, the number of physical executable lines of code (slocp) will also increase. Since a strong correlation may indicate redundancy, possibly only one feature would be needed from a pair of strongly correlated features for better accuracy.

Table 5.6: Strongly related pairs of features in gender-based (combined problem) and region-based data sets.

Pair number	Pairs of features	
1.	Number of function	Cyclomatic complexity
2.	Number of function	Volume
3.	Number of function	Lloc
4.	Slocp	program length
5.	Slocp	Loc
6.	Slocp	Cyclomatic complexity
7.	Slocp	Volume
8.	Slocp	Lloc
9.	Program length	Loc
10.	Program length	Program vocabulary
11.	Program length	Cyclomatic complexity
12.	Program length	Volume
13.	Program length	Effort
14.	Program length	Calculated estimated program length
15.	Program length	Lloc
16.	Program length	Difficulty
17.	Loc	Comments
18.	Loc	Cyclomatic complexity
19.	Loc	Volume
20.	Loc	Lloc
21.	Cyclomatic complexity	Lloc
22.	Volume	Effort
23.	Volume	Calculated estimated program length
24.	Volume	Difficulty
25.	Volume	Lloc
26.	Effort	Difficulty
27.	Calculated estimated program length	Difficulty
28.	Program Vocabulary	Volume
29.	Program Vocabulary	Calculated estimated program length
30.	Program Vocabulary	Difficulty
31.	Program Vocabulary	Volume

5.4 Comparison with previous work

In [32], Naz analyzed 100 C++ programs that were collected from students'assignments. The author conducted six experiments and built four classification models using 50 features. From the Tree and Bayes classifiers, the author used the J48 algorithm and Naive Bayes al-

gorithms, respectively. In [35], Rafee collected 160 C++ programs from students and tried to classify the programs based on the gender and region of the programmers. The author used 16 features and built seven classification models. In [35], Rafee used the Bayes Net algorithm from the Bayes classifier and the Random Forest algorithm from the Tree classifier to build two models out of the seven classification models. Similarly, in [1], Alam classified 12034 programs according to gender and 12680 programs according to the region of the programmers. Alam used 103 features and five classification models for the research, including the Bayes Net and Random Forest algorithms.

Table 5.7: Comparison with previous models for gender-based classification.

Classifier	**Naz's result**	**Rafee's result**	**Alam's result**	**Our result**
Trees	63%	89%	62.6%	86.4%
Bayes	66%	92%	54.3%	83.5%

Table 5.8: Comparison with previous models for region-based classification.

Classifier	**Rafee's result**	**Alam's result**	**Our result**
Trees	92.5%	78.36%	75.2%
Bayes	82.5%	71.28%	69.6%

Tables 5.7 and 5.8 show the comparison of models for gender and region-based classification. Comparing all the results, we observed that the Tree-based classifier performed best for the classification tasks. For gender-based classification, our result outperformed Alam's and Naz's work. Our Tree classifier successfully achieved an accuracy of 86.4%, whereas Naz's and Alam's Tree classifier achieved an accuracy of 63% and 62.3%. Rafee's Tree classifier has the highest accuracy of 92%. For our gender-based (combined problem) classification, our Tree classifier's accuracy was close to Rafee's Tree classifier's accuracy. However, we observed that one of our gender-based (individual problem) experiments demonstrated an accuracy of 91.7%, which was similar to Rafee's Tree classifier's accuracy. For the region-based classification, our Tree classifier's accuracy was 75.2% which was close to Alam's Tree classifier's accuracy of 78.3%.

The accuracy of Rafee's Tree classifier is the highest among all the classification results. However, the reason for Rafee's Tree classifier's high performance may be that the size of the data set is comparatively small, and the programmers themselves directly provided the gender information. Moreover, the author collected data only from two specific regions. In comparison the size of our data set is comparatively larger than Rafee's data set, and we predicted the gender of the programmers using an API. As well our region-based data set consisted of programs that were submitted by programmers from a wide variety of regions. Rafee calculated features based on only lines of code of the programs, whereas we considered cyclomatic complexity, Halstead metrics, and program execution based features for our research. Since our data collection approach, prepared data sets and classification models are completely different than Rafee's work, a comparison may not be particularly useful. It is worth noting that our classification models were successful in achieving better accuracy results for larger data sets considering different features.

Chapter 6

Conclusion

In this work, our research goal was to investigate the effect of the sociolinguistics features such as gender and region on computer programs. Our other goal was to determine whether male and female contestants write programs differently to solve a particular programming contest problem. Therefore, we collected the user information and C++ programs from an open-source programming contest website codeforces.com. We added gender labels (male and female) to the user names by using an API. Additionally, we categorized the programs into two regions (Eastern and Western). From the collected C++ programs, we calculated 13 features based on three software metrics: lines of code, cyclomatic complexity, and Halstead metrics. Applying different data preprocessing techniques we prepared 11 balanced data sets. The preparation of 11 different data sets was one of our research challenges and we successfully completed this challenge.

Next, we performed 13 experiments using different data sets ($1-11$) and features (16 and 6). To conduct the experiments five algorithms were used from the *WEKA* machine learning tool: Bayesian Network, Sequential Minimal Optimization (SMO), Classification Via Regression, Decision Table, and Random Forest algorithms. We used an attribute selector to find out the best features. In each experiment, we applied 10-fold cross-validation technique. To measure the models' performance, we used the accuracy values. Table 6.1 shows information on experiments $1-13$. In experiments $1-9$ we used data sets $1-9$ with 16 features and classified the gender of the programmers when they solved the same problems. Again, in experiment 10 we used 16 features and classified the gender of the pro-

grammers when they solved different problems. To classify the region of the programmers using 16 features, we used data set 11 and performed experiment 11. We also reduced the number of features from 16 to 6, used data sets 10 and 11, and performed experiments 12 and 13, respectively. We saw a variety in the performances of the classification models and found some interesting relationships among the classification models and the problem tags based on the results of these 13 experiments.

Table 6.1: Information on experiments $1-13$.

Experiment	Classification	Number of features	Data set
1-9	Gender-based (Individual problems)	16	1-9
10	Gender-based (Combined problems)	16	10
11	Region-based	16	11
12	Gender-based (Combined problems)	6	10
13	Region-based	6	11

In experiments $1-9$ we used data sets consisting of the same problems addressed by all the programmers. In these nine experiments, we were able to achieve the best accuracy of 91.7% with a Random Forest model using 16 features in experiment 7. Other models also showed better performances. Of the five classification models, the Bayes Net model performed best, achieving the highest accuracy in four experiments. The Bayes Net model also classified the highest number of male-written programs. Out of nine experiments the Bayes Net model accurately classified the highest number of male-written programs in five experiments. In contrast, the Decision Table model successfully predicted the highest number of female-written programs.

We also classified the gender of the programmers when they solved different problems in a programming contest. In experiment 10 the Random Forest model demonstrated the best accuracy of 86.4%. In this classification, the Sequential Minimal Optimization model successfully predicted the highest number of male-written programs, whereas the Random Forest model predicted the highest number of female-written programs.

After reducing features in experiment 12, the Bayes Net model achieved the highest ac-

curacy of 84.6% among five models and classified the most male-written programs. However, the Random Forest model and Classification Via Regression model both classified the highest number of female-written programs. In addition, the models using 6 features showed comparable accuracy results to the models using 16 features.

In region-based classification the Random Forest model demonstrated the best performance with an accuracy of 75.2%. After reducing the features, the Classification Via Regression model showed the highest accuracy of 73.6% and achieved comparable results. The Classification Via Regression model classified the highest number of programs from the Eastern region. In contrast, the Bayes Net model accurately predicted the highest number of programs from the Western region.

We also executed statistical T-tests to determine the statistically significant features and analyzed the use of the features to demonstrate how the programmers of a particular region or gender used these features in their programs. Additionally, we performed correlation analyses to investigate the relationship between features.

The findings from the experiments and feature analysis are given below:

- Analyzing the results of experiments $1 - 10$ and 12 we observed that in all experiments different models classified male-written and female-written programs based on the use of different problem tags. Based on this relationship between the problem tags and the classification models we hypothesize that male and female programmers use different approaches to solve a particular or different problems. Thus we can see a gender-based variations in the computer programs of the computer programming contests.

- Again, analyzing the results of experiments 11 and 13 we observed that different models classified programs from the Eastern and Western region based on the use of different problem tags. Thus we can see a region-based variation in the computer programs of the computer programming contests. We hypothesize that the programmers from the Eastern and Western regions use different approaches when they solve

the contest problems.

- From the statistical analysis we noticed that the female programmers use more comments, effort, lines of code, volume, blanklines, and number of function than the male programmers. Moreover, the female programmers seemed to use more logic decisions and write longer programs than the male programmers when they solved the same or different problems.

- We also observed that the use of effort was higher in the programs from the Western region than from the Eastern region. The features volume, calculated estimated program length, and difficulty are also higher in the programs from the Western region. In contrast, the programs from the Eastern region used more memory and blanklines than the Western region.

6.1 Future work

Some of the suggestions for the future research are given below:

- We analyzed the computer programs based on the gender and region of the programmers. In our future work, we would like to investigate other sociolinguistics factors such as age, ethnicity and experience level of the programmers.

- In this research, we only collected C++ programs from an online programming contest website. However, there are other programming contest websites and software repositories such as codechef and Github. It would be interesting to analyze the programs of these repositories.

- We only collected the programs written in the C++ language. Our future work may include the analysis of other programming languages such as Java and Python.

- In this research, we tried to determine the programming styles of the programmers when they solved programming contest problems. Our future work may include the

comparative analysis of stress during programming. We would like to analyze how a programmer solves competitive programming contest problems as opposed to regular programming problems.

- We calculated three software metrics based features such as lines of code, Halstead Metrics, and cyclomatic complexity. We would like to incorporate other software metrics for example, maintainability index with the sociolinguistics features in our future work.

- While collecting data, we observed that programmers in a programming contest website submit multiple solutions as they intended to achieve the optimized solutions. Therefore, our future work may include developing a tool that will help a novice programmer to develop his/her programming skills based on professional software metrics and code optimization techniques.

- Some programs were used to analyze the programs that performed better on different types of programs/concepts. To choose a machine learning tool based on the program concept we need to test more programs. If we could test more than 10 programs, this could provide some quantitative support to our work. Our future work may include analysis of more programs to support the possible connection between problem tags and the algorithms.

In this research we observed that the gender and region of the programmers influence the programming styles of the contest programmers. Using different software metrics as features we were successful to identify the programming styling differences of the programmers. Therefore, based on the hypotheses we can say that the analysis of sociolinguistics features with different software metrics can be used to identify the variations of programming styles of the programmers.

Bibliography

[1] S. Alam. Computer program complexity and its correlation with program features and sociolinguistics. Master's thesis, University of Lethbridge, Canada, 2021.

[2] 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.

[3] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni. Gender, genre, and writing style in formal written texts. *Text (The Hague)*, 2003.

[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] N. Cheng, R. Chandramouli, and K.P. Subbalakshmi. Author gender identification from text. *Digital Investigation*, 8(1):78 – 88, 2011.

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

[7] N. Fenton and J. Bieman. *Software metrics: a rigorous and practical approach*. CRC press, 2014.

[8] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera. *Learning from imbalanced data sets*, volume 11. Springer, 2018.

[9] M. Fisher and A. Cox. Gender and programming contests: Mitigating exclusionary practices. *Informatics in Education*, 5(1):47–62, January 2006.

[10] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[11] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[12] T. Hariprasad, G. Vidhyagaran, K. Seenu, and C. Thirumalai. Software complexity analysis using Halstead metrics. In *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pages 1109–1113. IEEE, 2017.

[13] J. Holmes. *An introduction to sociolinguistics*. Routledge, 2013.

[14] Y. Ishikawa. Gender differences in vocabulary use in essay writing by university students. *Procedia - Social and Behavioral Sciences*, 192:593 – 600, 2015.

[15] A. A. Khan, A. Mahmood, S. M. Amralla, and T. H. Mirza. Comparison of software complexity metrics. *International Journal of Computing and Network Technology*, 4(01), 2016.

[16] M. Koppel, S. Argamon, and A. R. Shimoni. Automatically categorizing written texts by author gender. *Computing Reviews*, 45(1):43, 2004.

[17] I. Krsul and E. H. Spafford. Authorship analysis: Identifying the author of a program. *Computers Security*, 16(3):33 – 257, 1997.

[18] M. Kuhn, K. Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[19] J.T. Kwok. Automated text categorization using Support Vector Machine. In *In Proceedings of the International Conference on Neural Information Processing (ICONIP*, pages 347–351, 1998.

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

[21] A. Lai and S. Yang. The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. In *2011 International Conference on Electrical and Control Engineering*, pages 6940–6944, 2011.

[22] C. Lewis and G. Olson. *Can Principles of Cognition Lower the Barriers to Programming?*, page 248–263. Ablex Publishing Corp., USA, 1987.

[23] P. Li, A. Shrivastava, J. Moore, and A. C. Konig. Hashing algorithms for large-scale learning. *arXiv preprint arXiv:1106.0967*, 2011.

[24] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[25] T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 4:308–320, 1976.

[26] M. J. P. V. D. Meulen and M. A. Revilla. Correlations between internal software metrics and software dependability in a large population of small c/c++ programs. In *Proceedings of the The 18th IEEE International Symposium on Software Reliability*, ISSRE '07, page 203–208, USA, 2007. IEEE Computer Society.

[27] M. Mirzayanov. What is codeforces? `https://codeforces.com/blog/entry/1`, accessed 2020-12-01.

[28] M. Mirzayanov. What is codeforces api? `https://codeforces.com/apiHelp`, accessed 2020-12-01.

[29] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[30] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

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

[32] F. Naz and J. E. Rice. Sociolinguistics and programming. In *2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 74–79. IEEE, 2015.

[33] C. Ozgur, T. Colliau, G. Rogers, Z. Hughes, B. Myer-Tyson, et al. Matlab vs. python vs. r. *Journal of Data Science*, 15(3):355–372, 2017.

[34] G. Papageorgiou, P. Bouboulis, and S. Theodoridis. Robust linear regression analysis—a greedy approach. *IEEE Transactions on Signal Processing*, 63(15):3872–3887, 2015.

[35] M. M. H. Rafee. Computer program categorization with machine learning. Master's thesis, University of Lethbridge, Canada, 2017.

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

[37] P. Simon. *Too big to ignore: the business case for big data*, volume 72. John Wiley & Sons, 2013.

[38] A. P. Singh and A. W. Moore. *Finding optimal Bayesian networks by dynamic programming*. Citeseer, 2005.

[39] I. Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.

[40] P. R. Srivastava, P. Patel, and S. Chatrola. Cause effect graph to decision table generation. *ACM SIGSOFT Software Engineering Notes*, 34(2):1–4, 2009.

[41] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[42] N. Tasnim. Machine learning in the classification of computer code. Master's thesis, University of Lethbridge, Canada, 2020.

[43] R. Wardhaugh. *An introduction to sociolinguistics*, volume 28. John Wiley & Sons, 2011.

[44] A. H. Watson, T. J. McCabe, and D. R. Wallace. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*, volume 500. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1996.

[45] I. H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, 31(1):76–77, 2002.

[46] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[47] W. Wolf. *High-Performance Embedded Computing: Architectures, Applications, and Methodologies*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

Appendix A

Additional details

A.1 Detailed calculation on information gain

Let us assume that a data point in the training data set D belongs to a class C_i. The probability P_i of that data point is calculated using Equation A.1.

$$P_i = \frac{the\ number\ of\ records\ in\ D\ that\ belong\ to\ C_i}{the\ total\ number\ of\ records\ in\ D} \tag{A.1}$$

Using the values in Table 2.4 the probability value for each class can be calculated using Equation A.1.

P_Y: buys-car = "Yes" = $\frac{3}{5}$
P_N: buys-car = "No" = $\frac{2}{5}$

The expected information or entropy needed to classify a data record or instance is calculated using Equation A.2

$$\text{Info}(D) = -\sum_{i=1}^{m} P_i \log_2(P_i) \tag{A.2}$$

where m is the number of classes that D contains. Table 2.4 contains two classes. Therefore, in this example the value of $m = 2$.

Using Equation A.2, the entropy of this data set is calculated as follows:

$$\begin{aligned}\text{Info}(D) &= -P_Y \log_2(P_Y) - P_N \log_2(P_N) \\ &= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} \\ &= 0.971.\end{aligned}$$

The example data set D is partitioned according to each attribute value. Attribute 'Engine condition' has three values: Good, Moderate and Bad. The other attribute 'Mileage' has three values: Low, Medium and High.

From Table A.1, the expected information for attribute 'Engine condition' can be cal-

culated using Equation A.3.

$$\text{Info}_{Engine\ condition}(D) = \frac{|D_{Good}|}{|D|}\text{Info}(D_{Good}) + \frac{|D_{Moderate}|}{|D|}\text{Info}(D_{Moderate}) + \frac{|D_{Bad}|}{|D|}\text{Info}(D_{Bad}) \tag{A.3}$$

Replacing the values of $|D_{Good}|$, $|D_{Moderate}|$, $|D_{Bad}|$, $|D|$, Info (D_{Good}), Info $(D_{Moderate})$ and Info(D_{Bad}) in Equation A.3 gives

$$\begin{aligned}\text{Info}_{Engine\ condition}(D) &= \frac{2}{5}(-\frac{2}{2}log_2\frac{2}{2} - \frac{0}{2}log_2\frac{0}{2}) + \frac{2}{5}(-\frac{1}{2}log_2\frac{1}{2} - \frac{1}{2}log_2\frac{1}{2}) \\ &+ \frac{1}{5}(-\frac{0}{1}log_2\frac{0}{1} - \frac{1}{1}log_2\frac{1}{1}) \\ &= 0.4.\end{aligned}$$

From Table A.2, the value of expected information for attribute 'Mileage' can be calculated using Equation A.4.

$$\text{Info}_{Mileage}(D) = \frac{|D_{High}|}{|D|}\text{Info}(D_{High}) + \frac{|D_{Medium}|}{|D|}\text{Info}(D_{Medium}) + \frac{|D_{Low}|}{|D|}\text{Info}(D_{Low}) \tag{A.4}$$

Replacing the values of $|D_{High}|$, $|D_{Medium}|$, $|D_{Low}|$, $|D|$, Info (D_{High}), Info (D_{Medium}) and Info (D_{Low}) in Equation A.4 gives

$$\begin{aligned}\text{Info}_{Mileage}(D) &= \frac{2}{5}(-\frac{0}{2}log_2\frac{0}{2} - \frac{2}{2}log_2\frac{2}{2}) + \frac{1}{5}(-\frac{1}{1}log_2\frac{1}{1} - \frac{0}{1}log_2\frac{0}{1}) \\ &+ \frac{2}{5}(-\frac{2}{2}log_2\frac{2}{2} - \frac{0}{2}log_2\frac{0}{2}) \\ &= 0.\end{aligned}$$

The value of Gain for attribute 'Engine condition' can be calculated using Equation A.5.

$$\text{Gain}_{Engine\ condition} = \text{Info}(D) - \text{Info}_{Engine\ condition(D)} \tag{A.5}$$

From Equation A.5 the Gain of attribute 'Engine condition' can be calculated.

$$\text{Gain}_{Engine\ condition} = 0.971 - 0.4 = 0.571.$$

For attribute 'Mileage', the value of Gain can be calculated using Equation A.6.

$$\text{Gain}_{Mileage} = \text{Info}(D) - \text{Info}_{Mileage(D)} \tag{A.6}$$

Using Equation A.6, the value of Gain is calculated as follows:

$$\text{Gain}_{Mileage} = 0.971 - 0 = 0.971.$$

The attribute 'Mileage' has higher information gain than the other attribute 'Engine condition'. Thus the attribute 'Mileage' will be more likely selected in feature selection.

Table A.1: Frequency of Yes and No for Buy depending on Engine condition attribute.

Records	Engine condition	Buy?		Total
		Yes	No	
D_{Good}	Good	2	0	2
$D_{Moderate}$	Moderate	1	1	2
D_{Bad}	Bad	0	1	1

Table A.2: Frequency of Yes and No for Buy depending on Mileage attribute.

Records	Mileage	Buy?		Total
		Yes	No	
D_{High}	High	0	2	2
D_{Medium}	Medium	1	0	1
D_{Low}	Low	2	0	2

A.2 Gender-based data sets

Table A.3: Information on the gender-based data sets.

	Gender		
Data set	Male written programs	Female written programs	Total records
1	148	148	296
2	114	114	228
3	124	124	248
4	106	106	212
5	282	282	564
6	157	157	314
7	84	84	168
8	240	240	480
9	178	178	356
10	1433	1433	2866

Appendix B

T-test ρ values of features

Table B.1: T-test ρ values of features for gender-based (individual problem) classification.

Features	**ρ values**								
	dataset 1	dataset 2	dataset 3	dataset 4	dataset 5	dataset 6	dataset 7	dataset 8	dataset 9
Memory	$4.5x10^{-01}$	$7.7x10^{-02}$	$3.9x10^{-03}$	$7.1x10^{-03}$	$1.9x10^{-06}$	$5.5x10^{-09}$	$6.2x10^{-02}$	$4.9x10^{-02}$	$8.6x10^{-01}$
Cyclomatic complexity	$1.1x10^{-06}$	$2.4x10^{-04}$	$5.2x10^{-06}$	$1.2x10^{-04}$	$1.1x10^{-25}$	$1.6x10^{-14}$	$1.1x10^{-07}$	$1.0x10^{-13}$	$7.7x10^{-10}$
Time	$6.7x10^{-01}$	$5.3x10^{-01}$	$8.1x10^{-01}$	$5.3x10^{-02}$	$4.2x10^{-03}$	$5.3x10^{-04}$	$8.1x10^{-01}$	$9.0x10^{-01}$	$3.2x10^{-01}$
Difficulty	$3.1x10^{-07}$	$1.3x10^{-05}$	$6.2x10^{-03}$	$4.3x10^{-04}$	$5.4x10^{-19}$	$4.3x10^{-10}$	$1.0x10^{-06}$	$9.3x10^{-07}$	$1.1x10^{-03}$
Lloc	$2.1x10^{-07}$	$2.6x10^{-04}$	$6.9x10^{-06}$	$8.2x10^{-03}$	$1.9x10^{-18}$	$6.8x10^{-13}$	$1.7x10^{-09}$	$1.0x10^{-19}$	$1.5x10^{-18}$
Sloc-p	$6.3x10^{-10}$	$2.0x10^{-04}$	$1.7x10^{-07}$	$2.1x10^{-04}$	$1.1x10^{-27}$	$9.3x10^{-15}$	$2.0x10^{-11}$	$4.3x10^{-22}$	$6.7x10^{-21}$
Number of function	$1.6x10^{-06}$	$8.0x10^{-03}$	$7.7x10^{-06}$	$2.1x10^{-05}$	$1.9x10^{-07}$	$1.1x10^{-09}$	$1.9x10^{-05}$	$1.4x10^{-05}$	$3.4x10^{-03}$
Comments	$2.7x10^{-21}$	$1.0x10^{-04}$	$9.0x10^{-09}$	$2.3x10^{-12}$	$3.3x10^{-32}$	$4.9x10^{-21}$	$1.5x10^{-14}$	$1.4x10^{-20}$	$2.1x10^{-16}$
Program Length	$8.8x10^{-18}$	$3.3x10^{-06}$	$2.9x10^{-14}$	$2.6x10^{-13}$	$1.5x10^{-20}$	$1.8x10^{-18}$	$2.8x10^{-09}$	$5.5x10^{-16}$	$3.5x10^{-12}$
Calculated estimated program length	$2.6x10^{-10}$	$2.1x10^{-06}$	$5.4x10^{-05}$	$2.1x10^{-09}$	$6.3x10^{-25}$	$1.3x10^{-09}$	$1.5x10^{-10}$	$1.7x10^{-06}$	$6.7x10^{-06}$
Program vocabulary	$4.5x10^{-10}$	$1.6x10^{-06}$	$7.0x10^{-04}$	$3.5x10^{-08}$	$7.5x10^{-26}$	$4.4x10^{-08}$	$1.3x10^{-10}$	$2.8x10^{-05}$	$3.9x10^{-05}$
Loc	$2.6x10^{-20}$	$9.4x10^{-05}$	$2.1x10^{-18}$	$2.6x10^{-11}$	$3.6x10^{-45}$	$5.7x10^{-26}$	$3.0x10^{-17}$	$5.6x10\text{-}^{27}$	$1.9x10^{-23}$
Volume	$1.6x10^{-17}$	$5.6x10^{-05}$	$3.2x10^{-14}$	$4.1x10^{-13}$	$8.4x10^{-17}$	$3.7x10^{-17}$	$1.4x10^{-08}$	$3.0x10^{-14}$	$4.9x10^{-11}$
Blanklines	$7.4x10^{-06}$	$5.8x10^{-02}$	$3.1x10^{-04}$	$7.5x10^{-04}$	$3.6x10^{-12}$	$1.2x10^{-05}$	$4.7x10^{-08}$	$6.1x10^{-08}$	$3.9x10^{-07}$
Effort	$7.5x10^{-13}$	$1.5x10^{-03}$	$1.6x10^{-09}$	$1.3x10^{-07}$	$1.1x10^{-02}$	$2.2x10^{-10}$	$7.5x10^{-04}$	$2.5x10^{-07}$	$1.4x10^{-06}$

Table B.2: T-test ρ values of features for gender-based (combined problem) classification.

Features	ρ **value**
Loc	5.63×10^{-120}
Comments	1.82×10^{-119}
Program Length	2.03×10^{-85}
Volume	3.65×10^{-73}
Sloc-p	2.08×10^{-67}
Calculated estimated program length	2.67×10^{-62}
Program vocabulary	9.91×10^{-57}
Cyclomatic complexity	3.41×10^{-53}
Lloc	2.36×10^{-52}
Difficulty	1.09×10^{-45}
Blanklines	6.17×10^{-40}
Number of function	3.84×10^{-30}
Effort	1.21×10^{-11}
Memory	3.20×10^{-08}
Time	0.391

Table B.3: T-test ρ values of features for region-based classification.

Features	ρ **value**
Program vocabulary	8.44×10^{-78}
Time	2.99×10^{-76}
Calculated estimated program length	9.02×10^{-67}
Difficulty	1.27×10^{-22}
Volume	5.45×10^{-21}
Program Length	5.10×10^{-20}
Memory	1.10×10^{-11}
Comments	2.48×10^{-04}
Number of function	0.00131
Loc	0.00231
Effort	0.00236
Lloc	0.0328
Sloc-p	0.0509
Cyclomatic complexity	0.0529
Effort	0.281

Appendix C

Use of features

Table C.1: Use of features in gender-based (individual problem) classification.

Features	Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5		Dataset 6		Dataset 7		Dataset 8		Dataset 9	
	M	F	M	F	M	F	M	F	M	F	M	F	M	F	M	F	M	F
Cyclomatic complexity	44%	56%	32%	68%	43%	57%	43%	57%	37%	63%	37%	63%	28%	72%	40%	60%	43%	57%
Difficulty	45%	55%	43%	57%	47%	53%	45%	55%	38%	62%	42%	58%	31%	69%	46%	54%	46%	54%
Lloc	43%	57%	32%	68%	43%	57%	45%	55%	41%	59%	39%	61%	35%	65%	40%	60%	39%	61%
Sloc-p	42%	58%	33%	67%	42%	58%	44%	56%	39%	61%	39%	61%	34%	66%	39%	61%	39%	61%
Number of function	37%	63%	31%	69%	42%	58%	38%	62%	33%	67%	30%	70%	23%	77%	28%	72%	32%	68%
Comments	27%	73%	29%	71%	32%	69%	25%	75%	24%	76%	25%	75%	20%	80%	28%	72%	29%	71%
Program Length	41%	59%	34%	66%	43%	57%	39%	61%	34%	66%	36%	64%	26%	74%	40%	60%	41%	59%
Calculated estimated program length	45%	55%	43%	57%	47%	53%	43%	57%	40%	60%	43%	57%	36%	64%	46%	54%	46%	54%
Program vocabulary	46%	54%	45%	55%	48%	52%	45%	55%	42%	58%	45%	55%	39%	61%	47%	53%	47%	53%
Loc	38%	62%	32%	68%	38%	62%	37%	63%	35%	65%	34%	66%	28%	72%	36%	64%	36%	64%
Volume	40%	60%	32%	68%	42%	58%	38%	62%	32%	68%	34%	66%	23%	77%	39%	61%	39%	61%
Blanklines	35%	65%	39%	61%	37%	63%	35%	65%	30%	70%	36%	64%	29%	71%	34%	66%	33%	67%
Effort	34%	66%	22%	78%	37%	63%	31%	69%	16%	84%	24%	76%	7%	92%	32%	68%	32%	68%

Table C.2: Use of features in gender-based (combined problem) classification.

Features	**Male-written programs**	**Female-written programs**
Program vocabulary	45%	55%
Calculated estimated program length	43%	57%
Difficulty	42%	58%
Lloc	40%	60%
Sloc-p	39%	61%
Memory	38%	62%
Cyclomatic complexity	38%	62%
Program Length	37%	63%
Volume	36%	64%
Loc	35%	65%
Loc	34%	66%
Number of function	32%	68%
Effort	26%	74%
Comments	26%	74%

Table C.3: Use of features in region-based classification.

Features	**Eastern region**	**Western Region**
Memory	55%	45%
Blanklines	52%	48%
Lloc	49%	51%
Loc	49%	51%
Comments	48%	52%
Number of function	47%	53%
Program Length	45%	55%
Program vocabulary	45%	55%
Difficulty	44%	56%
Calculated estimated program length	44%	56%
Volume	44%	56%
Time	30%	70%

Appendix D

Correlation analysis of features

Table D.1: Correlation matrix of features for gender-based data set.

Features	time	memory	program vocabulary	program length	calculated estimated program length	volume	difficulty	effort	loc	sloc-p	lloc	comments	blank line	cyclomatic complexity	number of function
time	1														
memory	0	1													
program vocabulary	0	-0.07	1												
program length	-0.08	0.11	0.71	1											
calculated estimated program length	0	-0.06	0.99	0.75	1										
volume	-0.06	0.09	0.73	0.99	0.77	1									
difficulty	-0.05	0.06	0.54	0.62	0.53	0.6	1								
effort	-0.02	0.08	0.42	0.7	0.48	0.74	0.56	1							
loc	-0.06	0.14	0.28	0.66	0.32	0.64	0.19	0.24	1						
sloc-p	-0.08	0.18	0.18	0.6	0.21	0.57	0.22	0.22	0.88	1					
lloc	-0.08	0.18	0.15	0.55	0.17	0.52	0.2	0.2	0.78	0.97	1				
comments	0.01	0.01	0.3	0.45	0.33	0.45	0.07	0.17	0.72	0.31	0.15	1			
blankline	-0.02	0.05	0.14	0.3	0.16	0.29	0.12	0.11	0.54	0.31	0.06	0.64	1		
cyclomatic complexity	-0.18	0.29	0.26	0.67	0.3	0.65	0.37	0.32	0.67	0.76	0.73	0.24	0.24	1	
numberof function	-0.01	0.01	0.36	0.57	0.42	0.59	0.19	0.31	0.52	0.39	0.33	0.48	0.3	0.57	1

Table D.2: Strongly related pairs of features for gender-based data set.

Pair number	**Pairs of features**	
1.	Program vocabulary	Program Length
2.	Program vocabulary	Calculated estimated program length
3.	Program vocabulary	Volume
4.	Program vocabulary	Difficulty
5.	Program length	Calculated estimated program length
6.	Program length	Volume
7.	Program length	Difficulty
8.	Program length	Effort
9.	Program length	Loc
10.	Program length	Sloc-p
11.	Program length	Lloc
12.	Program length	Cyclomatic complexity
13.	Calculated estimated program length	Volume
14.	Calculated estimated program length	Difficulty
15.	Volume	Difficulty
16.	Volume	Effort
17.	Volume	Loc
18.	Volume	Sloc-p
19.	Volume	Lloc
20.	Volume	Cyclomatic complexity
21.	Volume	Number of function
22.	Difficulty	Effort
23.	Loc	Sloc-p
24.	Loc	Lloc
25.	Loc	Comments
26.	Loc	Blanklines
27.	Loc	Cyclomatic complexity
28.	Loc	Number of function
29.	Sloc-p	Lloc
30.	Sloc-p	Cyclomatic complexity
31.	Lloc	Cyclomatic complexity
32.	Comments	Blanklines
33.	Cyclomatic complexity	Number of function

Table D.3: Correlation matrix of features for region-based data set.

Features	numb-er of funct-ion	sloc-p	program length	blank line	loc	program vocabul-ary	comme-nts	memory	cyclo-matic compl-exity	volume	effort	calculated estimated program length	difficu-lty	time	lloc
numberof function	1														
sloc-p	0.53	1													
program length	0.7	0.62	1												
blankline	0.1	0.18	0.13	1											
loc	0.62	0.92	0.66	0.27	1										
program vocabulary	0.6	0.39	0.8	0.07	0.47	1									
comments	0.54	0.46	0.5	0.3	0.78	0.42	1								
memory	0.04	0.29	0.21	0.04	0.26	0.13	0.11	1							
cyclomatic complexity	0.7	0.8	0.75	0.13	0.76	0.56	0.42	0.35	1						
volume	0.72	0.58	1	0.12	0.63	0.82	0.49	0.19	0.74	1					
effort	0.22	0.14	0.64	0.03	0.14	0.32	0.1	0.03	0.22	0.65	1				
calculated estimated program length	0.65	0.4	0.83	0.07	0.48	0.99	0.43	0.13	0.59	0.85	0.35	1			
difficulty	0.26	0.27	0.72	0.05	0.26	0.52	0.13	0.1	0.37	0.72	0.85	0.51	1		
time	0	-0.04	-0.01	0	-0.05	0.05	-0.05	0.08	-0.07	0	-0.01	0.05	0.02	1	
lloc	0.52	0.99	0.61	0.06	0.9	0.39	0.42	0.29	0.8	0.57	0.14	0.4	0.27	-0.04	1

Table D.4: Strongly related pairs of features for region-based data set.

Pair number	Pair of features	
1.	Number of function	Sloc-p
2.	Number of function	Program length
3.	Number of function	Loc
4.	Number of function	Program vocabulary
5.	Number of function	Comments
6.	Number of function	Cyclomatic complexity
7.	Number of function	Volume
8.	Number of function	Calculated estimated program length
9.	Number of function	Lloc
10.	Sloc-p	Program length
11.	Sloc-p	Loc
12.	Sloc-p	Cyclomatic complexity
13.	Sloc-p	Volume
14.	Sloc-p	Lloc
15.	Program length	Loc
16.	Program length	Program vocabulary
17.	Program length	Cyclomatic complexity
18.	Program length	Volume
19.	Program length	Effort
20.	Program length	Calculated estimated program length
21.	Program length	Difficulty
22.	Program length	Lloc
23.	Loc	Comments
24.	Loc	Cyclomatic complexity
25.	Loc	Volume
26.	Loc	Lloc
27.	Program vocabulary	Cyclomatic complexity
28.	Program vocabulary	Volume
29.	Program vocabulary	Calculated estimated program length
30.	Program vocabulary	Difficulty
31.	Cyclomatic complexity	Volume
32.	Cyclomatic complexity	Calculated estimated program length
33.	Cyclomatic complexity	Lloc
34.	Volume	Effort
35.	Volume	Calculated estimated program length
36.	Volume	Difficulty
37.	Volume	Lloc
38.	Effort	Difficulty
39.	Calculated estimated program length	Difficulty