



Using Data Mining to Investigate the Effect of Cognitive Style on Programming Habits

Zahra Karimi ^{a,*}, S.Fatemeh Noorani ^b

^aDepartment of Computer Science, Faculty of Mathematical Science, Shahrekord University, Shahrekord, Iran.

^bFaculty of Technical and Engineering, Department of Information Technology and Computer Engineering, Payame Noor University (pnu), P.O.Box 19395-4697, Tehran, Iran.

ARTICLE INFO.

Article history:

Received: 28 August 2022

Revised: 13 September 2022

Accepted: 14 September 2022

Published Online: 25 October 2022

Keywords:

Computer Programming,
Programming Habits, Human
Factors, Cognitive Style, Data
Mining

ABSTRACT

Different programmers code in different ways. Knowing these habits and the human factors that affect them significantly impacts teaching and task assignments in programming. This article examines the effect of cognitive style on programming habits. We used a questionnaire to obtain data on cognitive style, programming experience, programming skill, interest in programming, and programming habits from 275 student programmers. After preprocessing and feature selection, we evaluated the effectiveness of different data mining techniques in estimating programming habits. Using the Support Vector Machine, the most effective method, we predicted each programming habit once without cognitive style and the second time with cognitive style. The results showed that cognitive style affects the programming habit of "systematic debugging" with Glass's Delta value = 0.22. Programmers with a median score in cognitive style, both analytic and Intuitive, more often debug their codes systematically than programmers with lower or higher scores in cognitive style. Thus assigning programmers with both Intuitive and analytic talent would be more effective when projects need systematic debugging. Moreover, trainers should pay more attention to only Intuitive or only analytical students when teaching systematic debugging. We recommend teachers, trainers, and managers consider cognitive style in programming.

1 Introduction

Cognitive style describes how individuals think, perceive information in the environment, remember them, and also patterns of thought they use to develop a knowledge base about the environment [1]. Cogni-

tive styles differ from cognitive ability (intelligence), yet they may influence how individuals do tasks and their outcomes. Psychologists disagree on the unified classification of cognitive styles. However, there are two basic cognitive styles: Field-Dependence/Field-Independence and Analytic/Intuitive.

A Field-Dependent individual has more difficulty finding a hidden geometric shape in the background than a Field-Independent one. In other words, Field-Dependent people need external helpers to remove the ambiguity when the situation is ambiguous. How-

* Corresponding author.

Email addresses: zahra.karimi@sku.ac.ir (Z. Karimi),
SF.Noorani@pnu.ac.ir (S.F. Noorani)

<https://dx.doi.org/10.22108/JCS.2022.134932.1105> ISSN:
2322-4460



ever, Field-Independent people function with greater autonomy from others under such conditions [2].

Intuitive individuals have a more global approach to problem-solving and do not identify the parts of a whole as readily as their peers. They also have difficulty with delayed gratification on tasks. Analytic individuals are more analytical and consider more alternatives in their problem-solving approach. In addition, Analytic individuals would typically take more time on a task and produce more accurate work than Intuitive ones [3].

Some researchers consider MBTI (Myers Briggs Type Indicator) a multi-dimensional cognitive style model [4]. The first dimension is the Extraversion/Introversion attitude. Extraverted ones seek breadth of knowledge and influence, while introverted ones seek depth of knowledge and influence. The second dimension, Sensing/Intuition, is the information-gathering function. Sensing people look for the present, concrete information, details, and facts. Intuitive people prefer information that is less dependent upon the senses, and that can be associated with other information, either remembered or discovered by seeking a broader context or pattern. Thinking/Feeling are the decision-making (judging) functions. Thinking people make decisions by matching a given set of rules. Those who prefer Feeling tend to make decisions by associating with the situation. The last dimension shows that people also have a preference for using either the judging function (Thinking or Feeling) or their perceiving function (Sensing or Intuition) [5].

Cognitive style is an essential concept in the areas of education and management. If an individual has an appropriate cognitive style, it is more probable that the individual will have a more positive experience in task completion [6]. Cognitive style is also essential in problem-solving and computer programming [7]. There are many pieces of research on the influence of cognitive style on problem-solving and programming.

For example, Akinola and Oluwatosin [8] investigated the effect of MBTI on the ability to code inspection among 50 students in Nigeria. They found that Extraverted, Sensing, Thinking Judging students have better code inspection results. Huang et al. [9] studied the effect of cognitive style on programming performance and the number of faults. They experimented with 70 Chinese students and reported that Intuitive subjects performed better than Sensory subjects and produced less fault density.

Theodoropoulos et al. [10] examined the effect of cognitive style on the attitude and performance of students learning computer programming through a game. In their experiment, 77 Greek students worked

for four weeks for 7 hours. In each puzzle, students wrote a program that got a specific character or created some geometrical shapes. They reported that Intuitive and judging students found the programming activities more motivating. They also found that Introverted and judging programmers achieved higher programming performance.

Abdelnour-Nocera et al. [11] investigated the relationship between the scores of cognitive style and the performance students achieved in design and evaluation tasks. They reported that Analytic students performed better in heuristic evaluation than Intuitive students.

Susilowati et al. [12] studied the effect of Field-Dependent/Field-Independent on student learning outcomes. They found that Field-Independent students achieved better computer programming and data structure grades. They also tested the impact of Field-Dependent/Field-Independent on computer programming in the context of collaborative programming with the assistance of an advanced organizer or without an organizer. They found that the Field-Independent programmers achieved a better result in both cases [13].

Yen and Liao [14] studied the effect of Field-Dependence/Field-Independence on 106 students in Taiwan during the Scratch programming course. The students were novices; however, they took the programming course for 40 minutes per week for eight weeks. Participants took a test on programming concepts in the last session to show their project performance. The test comprised 15 multiple-choice items on knowledge comprehension and knowledge application. They also graded the student projects in five aspects: correct programming, completeness of programming, content creativity, appealing interface, and creative performance. They reported that Field-Independent programmers are more effective in code comprehension than Field-Dependent programmers.

Pretorius et al. [15], who investigated 80 programmers, found that females who preferred more than one cognitive style (Intuition and rationality) produced the most novel software design.

Cognitive style might also affect problem-solving [16] since, for example, Setiawan et al. [17] found that Field-Independent students achieved better problem-solving ability. As another example, Khalid et al. [18] showed differences in critical thinking skills in cognitive styles in solving mathematics problems. Field-Independence Subjects did all of the critical thinking aspects such as interpretation, analysis, evaluation, inference, explanation, and self-regulation. At the same time, Field-Dependence ones did only several indica-



Table 1. Summary of Existing Research on the effect of cognitive style on computer programming and problem solving.

Year	N	Performance	Habits/strategies	Area
2013	50	scores in programming and code inspection task		computer programming
2014	70	failure and fault density, programming time		computer programming
2016	77	programming game score		computer programming
2017	100	evaluation score		HCI Design problem solving
2018	226	test score		computer programming
2019	82	test score		computer programming
2019	106	test score	gaming behavior	computer programming
2020	80	design score		computer programming
2020	109	essay score		mathematical problem-solving
2020	4		different aspects of critical thinking	mathematical problem-solving
2021	97		the manner of quantitative reasoning	mathematical problem-solving
2021	40		fixation count and duration	computer programming
2022	29			

tors in critical thinking aspects. Yusnaini et al. [19] found that the performance of independent-field students is higher than dependent-fields in answering unfamiliar types of questions on accounting. Nonetheless, there is no difference in performance between the two cognitive styles when faced with familiar types of questions. Also, Muzaini et al. [20] reported that cognitive style affects the appropriate problem-solving strategy. Hung and Wang [21] monitored a debugging task of 40 students who studied a C++ programming course. They found that Field-Independent students applied more fixation counts in longer durations than Field-Dependent students.

We summarized the existing recent research on the influence of cognitive style on programming and problem-solving in Table 1. As seen in Table 1, some studies investigated the effect of cognitive style on programming performance. Some other studies investigated the impact of cognitive style on problem-solving habits in other areas like mathematics. However, the study [14] investigated programming habits but in a gaming context, and the study [20] investigated the effect of cognitive style on only debugging habits. Thus, to the extent we know, there are few studies on the influence of cognitive style on programming habits and there is no study on the impact of cognitive style in programming in Iran. We need more research to clarify the effect of cognitive style on computer programming habits. To advance this research area, in

this paper, we investigated this issue: the influence of cognitive style on programming habits among Iranian Programmers.

Different programmers have different habits in coding [16]. For example, some programmers respond immediately to Interactive-Development-Environment (IDE) alerts when coding and fix the error as soon as they see it. In contrast, some fix the errors after a while. Some run the program after writing two or more lines of code, while others write more code before running.

As another example, some programmers behave systematically during debugging, read the code carefully, and as long as they are not sure of the cause of the error, do not change the code. While some programmers scan the code, and as soon as they find the first guess, they change the code and test their guesses [22, 23].

Studying the habits of different programmers not only helps programmers to know themselves better but is also effective in teaching and managing programmers. Teachers can consider the different habits of different programmers in teaching and assessment. Managers can consider programming habits to hire appropriate programmers and assign appropriate tasks.

The existing research in software psychology cannot be simply summarized. Some researchers confirmed the existence of a relationship between psychological variables and programming, and some denied the existence of this relationship. Some reported a direct



relationship between the two variables and some reported an inverse relationship. At the same time, the influence of psychological variables is confirmed by researchers, and the new research in this field is increasing. Programming is complicated; thus, we can not simply analyze the linear relationship between psychological variables and programming. Thus in this paper, we used estimation methods in data mining to better investigate the influence of psychological variables, and cognitive style in programming.

We collected data on cognitive style and programming habits. We also collected some other individual differences like programming experience, and interest in programming, etc. Then, we explored data mining techniques and estimated programming habits.

In summary, the objective of our study is to investigate the influence of cognitive style on programming habits through data mining techniques. Specifically, the following questions are considered:

Research Question One-Which data mining technique is more effective for estimating programming habits considering cognitive style?

Research question two-Which of the programming habits is influenced by Cognitive style?

It is better to mention that both individual differences and situation affects programming habits [16]. But in this research, the same situation was considered for all participants since they were all computer and information technology students at Shahrekord University.

In Section 2, the details of the research method are presented. Section 3 describes the findings and Sections 4 and 5 discuss the results and present conclusions. Our findings showed that the Support Vector Machine is the best method for predicting programming habits and cognitive style influences the habit of “systematic debugging.

2 Research Methodology

This study aims to use data mining to investigate the effect of cognitive style on programming habits. For this purpose, the following steps have been planned (Figure 1). First, we selected several programming habits and prepared the questionnaire to collect the data. After collecting data, in data preparation: we filled in missing data with the cluster average, we removed outliers, and we did feature selection. Then, the effectiveness of several learning techniques was evaluated to select the most appropriate technique (research question one). Finally, we investigated the effect of cognitive style on programming habits (re-

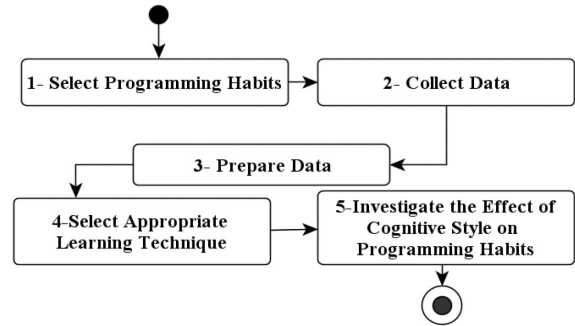


Figure 1. Steps of Present Research.

search question 2). In the next section, the details of these steps are explained.

2.1 Programming Habits

We classified programming habits based on previous research and expert opinions. To find relevant research, we applied the snowball review method starting from cox and fisher work [16]. Snowballing is tracking down citations in documents. The snowball method is a way of finding literature by using a key document on the subject of interest as a starting point. Then it continues by consulting the citations in the key document (article) and citations of the citations to find other relevant titles on the subject.

In the found articles, we considered all items named “programming habit”, “programming style” and “programming strategy”. Then, the authors of this article, and two computer students, had several meetings to classify programming habits. We also had a meeting with a psychologist and an expert software engineer to discuss and revise the programming habits. Finally, considering expert opinions, we chose the following categories for programming habits:

- Learning a new programming language: Programmers might take programming classes, study programming books, browse web pages or even ask questions from experts to learn a new programming language.
- Indentation and Spacing: Programmers might follow consistent indentation and spacing patterns or leave spacing as is.
- Comments: Programmers differ in how often they write in-line comments, header-blocks documentation or even use a different file for code. documentation.
- Compile and Run: Programmers differ in the frequency of recompilation and running.
- Debugging: Some programmers often debug systematically: they read the code carefully to find the bug. Others prefer to scan the code and test the initial guesses or search the web and ask



their colleagues. Programmers are different in how much they insist on code.

- Code Reuse: Some programmers usually copy-paste code where needed. Others define new methods or classes to reuse code or even reference the entire existing project.

2.2 Data Collection and Data Preparation

The sample needed to include a certain number of people which accurately reflect the conditions of the overall population it's meant to represent. To calculate our necessary sample size, we used Cochran's formula [24] (see relation 1).

$$n = \frac{Nz^2pq}{Nd^2 + z^2pq} \quad (1)$$

Where n is the sample size, z is the selected critical value of desired confidence level, p is the estimated proportion of an attribute that is present in the population, $q = 1-p$, and d is the desired level of precision. There were 628 students which study Computer or Information Technology in Sharekord universities in Fall 2020 ($n_0 = 628$). Assuming the maximum variability, which is equal to 50% ($p = 0.5$, $q = 0.5$) and taking 95% confidence level with $\pm 5\%$ precision ($d = 0.05$, z is 1.96) the calculation for required sample size will be as follows: (see relation 2).

$$\begin{aligned} n &= \frac{628(1.96)^2 \times 0.5 \times 0.5}{628(0.05)^2 + (1.96)^2 \times 0.5 \times 0.5} \quad (2) \\ &= \frac{603.1312}{2.5304} = 238.35 \end{aligned}$$

Therefore we needed 239 questionnaires (see relation 2). To be sure, we collected more samples: 275 students were selected via convenience sampling. The questionnaire (uploaded in Zenodo¹) was designed on Porsline² and shared on social media like WhatsApp and eLearning websites.

The questionnaire has three sections: Background, cognitive style, and programming habits. In the Background, we asked respondents about programming experience, skills, and level of interest in programming.

In the second part of the questionnaire, we used the Alinson and Hayes index[16] to measure cognitive style. This index is a questionnaire with 38 items. The answer to each item is: "True", "Between True and False" and "False". Numbers 2, 1, and 0 are assigned to each answer, respectively which will be summed for each respondent. The closer the score is to the maximum score (76), the more analytical the

respondent is, and the closer it is to the minimum score (0), the more Intuitive the respondent is.

The validity of this index has been shown through significant correlations with different personality dimensions and different job levels [3] and [25]. Alinson and Hayes[25] analyzed the reliability of this structure in Cronbach's alpha in 60 different samples of published studies and reported an average value of 0.84 which indicates a very good level. The authors translated the questions of this index into Persian. Then, the translation was validated by the experts in psychology and especially cognitive style. In the collected data, Cronbach's alpha was 0.868 which indicates a good level of reliability.

In the third part, the programming habits were asked with 24 items and its content validity was confirmed by expert software engineers. The Cronbach's alpha value of programming habit items, was 0.861, which indicates a good level of reliability.

To prepare data, first, the missing data were filled by the cluster average. Then, outliers were detected considering the Mahalanobis distance[26, 27]. Unlike the Euclidean distance, the Mahalanobis distance is not sensitive to correlated variables and outliers. Finally, 257 samples were considered for further analysis.

In feature selection, we used the Pearson correlation coefficient and selected 12 programming habits that were more correlated with CSI. We also selected the attributes which were correlated significantly with at least 3 selected programming habits. In the flowing, we listed the dataset columns:

- age, gender number of terms
- Cognitive Style Index
- Level of interest in programming
- Programming skills and experience
 - self-evaluated programming skill
 - self-evaluated programming skills in comparison with friends
 - the worst grade in programming exams
 - number of programming languages that the respondent knows
 - number of taken programming courses except for university courses
 - size of the largest project in which the programmers had a code contribution in it
- 12 programming habits

2.3 Data Analysis

In the first experiment, programming habits were assessed using individual characteristics of skill, experience, and interest in programming, and without cognitive style (see the right part of Figure 2). . In the second experiment, the programming habit was esti-

¹ <https://zenodo.org/record/6972500#.YvBJmXVBzcs>

² <https://survey.porsline.ir/s/jrYZOxO/>



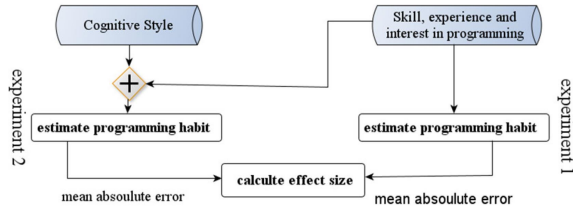


Figure 2. Experiment Design in Present Research to answer Research Question Two.

ated by considering the cognitive style along with the variables of experience of programming skills and interest (see the left part of Figure 2).

Then, Glass's Delta has been used to determine how much the prediction accuracy improves. Glass's Delta is a method of obtaining the effect size [26, 27]. Effect size is a concept for evaluating the effectiveness of improving prediction accuracy. In this way, the value of the effect is compared with a reference value. If it is greater than the reference value, it is said that increasing the accuracy of the prediction is effective.

The method for calculating Glass's Delta is shown in Equation (3). In this regard, $M(e2)$ is the average accuracy of experiment 2, and $M(e1)$ is the average accuracy of the e1 method. $s(e1)$ is the standard deviation of the accuracy in experiment 1. The more scattered the accuracy of the e1, the accuracy of the e2 needs to be better to be effective.

$$\Delta = \frac{M_{p2} - M_{p1}}{sd_{p1}} \quad (3)$$

3 Research Findings

In this section, at first, the detailed data set and feature selection are described. Then we describe our findings on the appropriate learning technique and the effect of cognitive style on programming habits. Finally, the results are summarized in the form of answers to research questions.

3.1 Dataset

The data set is described in Table 2. As can be seen in Table 2, about a quarter of programmers in our sample, had high experience (skill) and about a quarter of programmers had low experience (skills0). About one-third of programmers were very interested in programming and planned to pursue programming in the future, and about one-third were more interested in non-programming jobs such as analysis and design (see Figure 3). The sample represents a real diversity of programmers in terms of programming experience and skills.

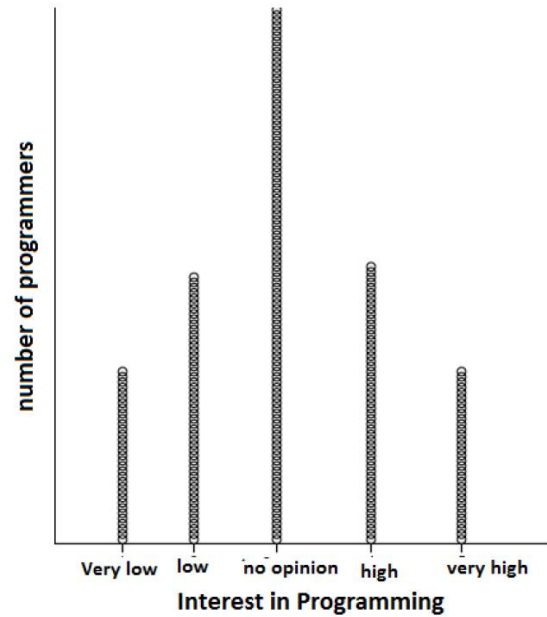


Figure 3. Distribution of programmers in "interest in programming".

Table 2 also shows that one-third of programmers in our sample had an analytical score in cognitive style and about half had a semi-analytical score. It means, more than 75% of programmers in our sample tend to an analytical style and the findings indicate that analytical style is common in the sample of this study.

As mentioned earlier, the Spearman correlation between cognitive style on the one hand and programming habits variables on the other was used to select the feature (Table 3), and 12 programming habits that were more correlated with cognitive style were selected.

As can be seen in Table 2, at least one habit has been selected from each type except the "Code Reuse" type. The frequency of the values of the selected habits is shown in Figure 3. As can be seen, most programmers fix the bugs by getting the help of other programmers (Habit 8), and few programmers write comments for important lines (Habit 5).

Then, the correlation between the selected habits on the one hand, and other individual variables, on the other hand, was obtained. The individual variables that had a significant correlation with at least three selected habits were also selected (see the selected variables in section 3.2).

3.2 Selection of Learning Technique

As mentioned earlier, different learning techniques were used to predict each of the 12 habits. 10-layer validation was used and the prediction of each habit



Table 2. Description of Dataset.

Programming Experience	62 (24%) had participated in two or more non-university programming courses and 97 (37%) had experience writing large programs.
Programming Skill	65 programmers (25%) had good programming skills in their own opinion, and 70 programmers (27%) had more programming skills than their friends in their own opinion. 65 programmers (25%) never got a bad grade in university programming courses. 143 (55%) considered themselves to have fewer programming skills and 128 (49%) had fewer programming skills than their friends. 68 (26%) programmers got bad grades in programming lessons.
Interest in Programming	80 programmers (31%) were very interested in programming and 77 programmers (29%) preferred to choose non-programming jobs like analyst and designer in the future.
Cognitive Style	The minimum score in cognitive style was 37, the maximum was 63, the mean was 50.57 and the standard deviation was 4.983. The skewness of this variable was 0.192.

Table 3. Summary of Existing Research on the effect of cognitive style on computer programming and problem solving.

	Programming habit	correlation		
Learning a	Learning by taking a programming course	0.171	✓	1
	Learning by studying a programming book	0.175	✓	2
	Learning by browsing web pages	0.060	✗	
	Learning by asking questions from experts	0.095	✓	3
Indentation and	adjusting spacing and indentation while writing code	0.115	✓	4
	adjusting spacing and indentation while reviewing code	0.007	✗	
	adjusting spacing and indentation before finalizing the code	0.036	✗	
Comments	Writing comments only for important lines	0.127	✓	5
	Writing comments per each method or class	0.009	✗	
	preferring to write descriptions out-of-code	0.028	✗	
Compile and Run	solving compile errors as soon as they see it	0.165	✓	6
	solving compile errors after writing a few lines	0.061	✗	
	executing code after writing one logical unit	0.024	✗	
	executing code after writing several lines	0.094	✓	7
Debugging	scanning the code to find a bug	0.023	✗	-
	getting the help of other colleagues	0.134	✓	8
	reading the code carefully to find bugs	0.209	✓	9
	Searching the web to find clues	0.089	✓	10
	Testing the initial guesses to remove	0.035	✗	-
	Insisting on code to find bugs	0.076	✓	11
	Deeper examination of guesses before implementation	0.156	✓	12
Code Reuse	copying and pasting in the middle of the code	0.007	✗	-
	copying and pasting in a separate method and calling the method	0.059	✗	-
	adding a reference to needed projects	0.006	✗	-

was repeated 500 times in a new random replacement of the data. In other words, each learning technique

was tested 6,000 times.



Table 4. Mean absolute error in prediction techniques.

	Error sd	Error mean
Linear Regression	0.131	0.707
Perceptron Neural Network	0.240	1.000
Support Vector Machine	0.144	0.688
Random Forest	0.132	0.720
Regression Tree	0.140	0.727

Table 4 shows the mean and standard deviation of mean-absolute-error in each of the learning techniques. As can be seen in Table 4, except Perceptron neural network, the means and standard deviations in other learning techniques, are very close. Therefore, to select the best learning technique, the mean errors were compared statistically.

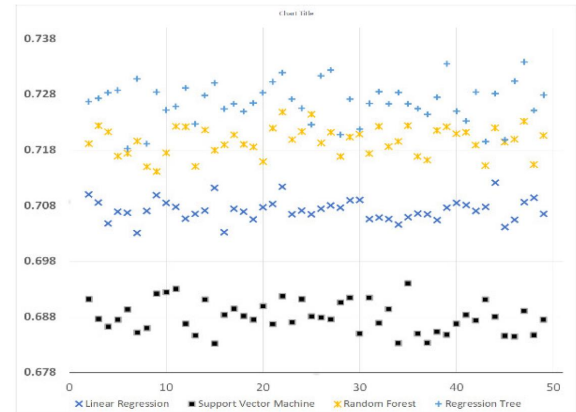
One-way analysis of variance was used to statistically compare the mean-absolute-error in different techniques. One-way analysis of variance is used to determine whether there is a significant difference between the means of three or more

levels of the independent variable. In the present dataset, the assumption of homogeneity of variance was not confirmed (significance level of 0.000 in Levin test), and so Welch statistic was analyzed. The level of significance of the Welch statistic (0.000) indicated that the means were not the same and therefore, there is a significant difference between at least two techniques.

The results of the Dant-C post-analysis showed that the mean error in each technique was significantly different from the other techniques. Therefore, the results presented in Table 4 are also statistically confirmed: The Support Vector Machine with the lowest mean-absolute-error is the best prediction technique for our data set. Then linear regression, random forest, regression tree, and Prospectron neural network techniques are in the next positions, respectively.

Mean-error-absolute is shown in Figure 4. The superiority of the support vector method (lowest mean-error-absolute) is shown in Figure 4. Since the mean-error-absolute in the Prospectron technique was much higher than the other techniques, it is not shown in this figure.

The Support Vector Machine technique was further analyzed and the RBFKernel core with a gamma of 0.05 was considered for further analysis in which the mean-absolute-error was reduced to 0.6762 and the standard deviation to 0.1342.

**Figure 4.** Mean absolute error in different data mining techniques.

3.3 The influence of cognitive style on programming habits

As mentioned earlier, to investigate the influence of cognitive style on programming habits, the Support Vector Machine technique was used to predict each of the 12 selected habits: in experiment one without cognitive Style and experiment two with cognitive Style. In each experiment, the habit was predicted 500 times.

In Table 5 the mean and standard deviation of mean-error-absolute are reported in experiments one and two. Cognitive style influences habits in which consideration of mean-error-absolute cognitive style is reduced. As can be seen in Table 5, cognitive style influences the habits of "3-Learning by asking questions from experts", "5-Writing comments only for important lines" and "9-reading the code carefully to find bugs".

Figure 5 shows the mean-absolute-error chart for habit 3-Learning by asking questions from experts. As can be seen, the mean-absolute-error is reduced by considering the cognitive style, in other words, the prediction accuracy is increased.

Figure 6 shows the mean-absolute-error chart for the habit of 5-writing comments for important lines. As can be seen, the mean-absolute-error is reduced



Table 5. Mean and standard deviation of mean-absolute-error by programming habit.

#	Programming Habits	Without CS		With CS	
		Mean	Sd	Mean	Sd
1	Learning by taking a programming course	0.689	0.1	0.693	0.13
2	Learning by studying a programming book	0.695	0.1	0.699	0.11
3	Learning by asking questions from experts	0.663	0.1	0.65	0.09
4	adjusting spacing and indentation while writing code	0.721	0.1	0.724	0.11
5	Writing comments only for important lines	0.899	0.1	0.881	0.11
6	solving compile errors as soon as they see it	0.61	0.1	0.612	0.11
7	executing code after writing several lines	0.787	0.1	0.788	0.14
8	getting the help of other colleagues	0.523	0.1	0.524	0.13
9	reading the code carefully to find bugs	0.61	0.1	0.59	0.09
10	Searching the web to find clues	0.682	0.1	0.683	0.1
11	Insisting on code to find bugs	0.677	0.1	0.678	0.12
12	A deeper examination of guesses before implementation	0.592	0.1	0.592	0.11

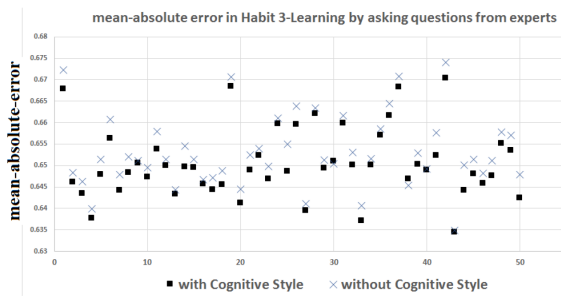


Figure 5. mean-absolute-error in Habit 3.

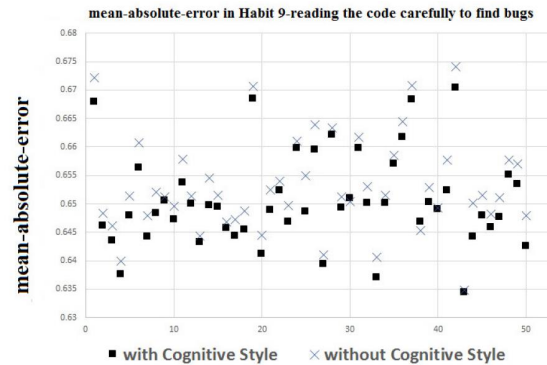


Figure 7. mean-absolute-error in Habit 9.

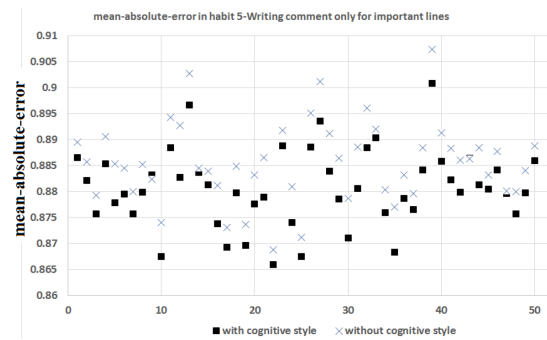


Figure 6. mean-absolute-error in Habit 5.

by considering the cognitive style, in other words, the prediction accuracy is increased.

Figure 7 shows the mean-absolute-error chart for habit 9-reading the code carefully to find bugs. As can be seen, the mean-absolute-error is reduced by considering the cognitive style, in other words, the prediction accuracy is increased.

As mentioned earlier, in calculating the effect size

in addition to the difference in means, the standard deviation is also taken into account. The effect size of cognitive style on habit-3 is 0.13935, on habit-5 is 0.15295 and on habit-9 is 0.21764.

In social science research, the effect size with a value of at least 0.2 is acceptable. Therefore, only the influence on habit-9 (with an effect size of 0.21764) is not negligible. Examining Table 3, it is clear that the correlation between cognitive style and habit, 9 is direct. This means that the higher the cognitive style and the more Analytical the programmer, the more prone he prefers to carefully read the code in finding bugs.

Figure 8 shows the scatter diagram of habit 9 based on cognitive style. As can be seen, with the decrease in cognitive style, the use of habit 9 has decreased and with the increase of cognitive style, the use of habit 9 also increased.



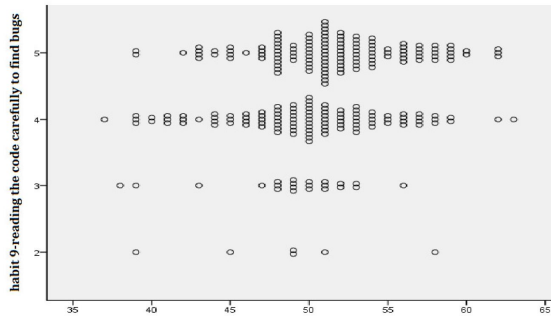


Figure 8. Scatter Plot of the relation between habit 9 and cognitive style.

3.4 Answers to Research Questions

In this section, research questions are answered briefly:

Research Question One - Which data mining technique is more effective for estimating programming habits in terms of cognitive style? Perceptron neural network, linear regression, regression tree, random forest, and support vector machine were tested and the mean-absolute-error was compared. Among them, the support vector machine significantly had less mean-absolute-error and, therefore, is more effective.

Research question two-Which of the programming habits is influenced by Cognitive style? Cognitive style affects the habit 9-reading the code carefully to find bugs with Glass's Delta about 0.22. It means Analytical programmers more often read code carefully when finding bugs than Intuitive programmers.

4 Discussion

In this section, we discussed the main contributions of the present research.

1-Analytical cognitive style is common among programmers. More than of third forth of programmers are Analytic or semi-Analytic in our sample. This result is in line with the theory of Allinson and Hayes [25] and the findings of Daraghmeh [28] and Abdelnour-Nocera[10].

Allinson and Hayes [25] claimed that jobs such as engineering and accounting, which require a structured approach, are more analytic. The findings of Daraghmeh [29] showed that more than half of the students who plan to study programming are Analytic or semi-Analytic. Abdelnour-Nocera's[11] findings also showed that software engineering and computer science students are more Analytic than other engineering students.

2-Based on our findings, we conclude having both an Analytic and Intuitive style is an advantage in

programming. Since in one direction, the best-found estimation method, Support Vector Machine with RFBKernel, is a descending parabolic curve. In another direction, the relationship between cognitive style and habit 9-systematic debugging fits a parabolic curve (see Fig 8). It means programming habits might need both Analytic and Intuitive cognitive styles. This claim is consistent with the findings of Piritorius et al. [15] in software design. They found that women who can benefit from both types of cognitive styles have the most innovation in design [15]. Baldacchino et al. [?] also showed that experienced people can use both Intuitive and Analytical styles.

4.1 Limitations

First, we used a self-developed questionnaire to assess the programming habits. There is the threat that questions do not reflect programming habits. We mitigated this threat by relying on the previous literature for formulating the indicators and piloting and revising the questionnaire several times.

Second, we used a questionnaire to collect the required data. Some of the participants may not have readily shared their views for fear of being victimized for disclosing sensitive information. To decrease this limitation, the participants and the respondents were assured that the information which was collected was for the research purpose and that their privacy will be respected.

Third, like many other studies of this kind, studies of this kind, this study suffers from volunteer bias. People who are willing to volunteer their opinion may differ from those who decline to take part. It might be volunteers tended to a specific cognitive style. To decrease this limitation, we comprehensively broadcast the questionnaire link and waited for about two months in collecting data.

Forth, the participants of this study were student programmers and our results might not be true for the professional population. This result as just conjectures about the habits of professional programmers was interesting for us. Moreover, this result is also interesting for computer science students and their supervisors. We wanted to give them a clue about how their habits are different.

5 Conclusions

This study aimed to use data mining to investigate the effect of cognitive style on programmers' habits. 24 programming habits, cognitive style, and some important characteristics of programmers such as programming experience, programming skills, and level of



interest in programming were collected from 275 programmers. The obtained data were preprocessed and some irrelevant habits were removed. 12 programming habits and 257 samples were considered for further analysis.

To determine the appropriate learning technique, each programming habit was estimated 50 times with linear regression, perceptron neural network, support vector machine, regression tree, and random forest in 50-fold cross-validation. Accuracy in different techniques was compared with a one-way analysis of variance and the support vector machine was considered as the most effective technique for further analysis.

To examine the effect of cognitive style on programming habits, each programming habit was estimated once by considering cognitive style and other individual variables, and once without cognitive style and only with the variables of age, gender, experience, skill, and interest in programming. Cognitive style affects habits in which the accuracy of prediction is sufficiently better when considering cognitive style than when not considering cognitive style. In this study, the effect of cognitive style was measured with Glass's Delta. According to the results, the cognitive style affects the habit of "reading the code carefully during debugging". A careful reading of the code is a systematic habit versus code scanning, which is an opportunistic habit [23].

Further analysis showed that there is a direct relationship between cognitive style and the habit of systematic debugging. The more analytic the cognitive style of the programmer, the greater the use of the habit of systematic debugging than programmers without the analytical style.

These results are useful in both training programmers and selecting the right programmers. For the non-analytic programmer, it is necessary to train the style of reading-the-code-carefully-during-debugging. In selecting programmers in situations where careful reading of the code is required, it is recommended to assign analytical programmers.

This study offers some suggestions for future researchers: It is suggested that the research be replicated in a larger community of more professional programmers. New psychological variables such as motivation or emotional variables can be studied. New programming habits can be gathered by analyzing software repositories or observing programmers.

References

- [1] M. Kozhevnikov. Cognitive styles in the context of modern psychology: Toward an integrated

framework of cognitive style. *Psychological Bulletin*, 133(3):464–481, 2007. doi:10.1037/0033-2909.133.3.464.

- [2] R. T. Pithers. Cognitive Learning Style: a review of the field dependent-field independent approach. *Journal of Vocational Education & Training*, 54(1):117–32, 2002. doi:10.1080/13636820200200191.
- [3] C. Allinson and J. Hayes. The Cognitive Style Index - Technical Manual and User Guide. *Retrieved January*, 13, 2014.
- [4] J. R. Hough and D. Ogilvie. An Empirical Test of Cognitive Style and Strategic Decision Outcomes. *Journal of Management Studies*, 42(2):417–448, 2005.
- [5] I. B. Myers. *A Guide to the Development and Use of the Myers-Briggs Type Indicator: Manual*. Consulting Psychologists Press, 1985.
- [6] M. J. Kirton. *Adaption-innovation: In the context of diversity and change*. Routledge, 2004.
- [7] G. L. White and M. P. Sivitanides. A theory of the relationships between cognitive requirements of computer programming languages and programmers' cognitive characteristics. *Journal of Information Systems Education*, 13(1), 2002.
- [8] O. S. Akinola and O. Oluwatosin. A Study on the Interplay of Cognition in Computer Programming and Code Inspection Skills in an Academic Environment. *Journal of Science Research*, 12(1):167–178, 2013.
- [9] F. Huang, B. Liu, Y. Song, and S. Keyal. The links between human error diversity and software diversity: Implications for fault diversity seeking. *Science of Computer Programming*, 89:350–373, 2014. doi:10.1016/j.scico.2014.03.004.
- [10] A. Theodoropoulos, A. Antoniou, and G. Lepouras. How Do Different Cognitive Styles Affect Learning Programming? Insights from a Game-Based Approach in Greek Schools. *ACM Transactions on Computing Education (TOCE)*, 17(1): 1–25, 2016. doi:10.1145/2940330.
- [11] J. Abdelnour-Nocera, T. Clemmensen, and T. G. Guimaraes. Sequence and Structure based Protein Peptide Binding Residue Prediction. In *IFIP Conference on Human-Computer Interaction*, pages 198–217. Springer, 2017. doi:10.1007/978-3-319-68059-0_13.
- [12] D. Susilowati, D. Susilowati, N. S. Degeng, N. S. Degeng, P. S. P. Setyosari, and S. U. Ulfa. The Role of Cognitive Styles in Computer Programming Learning. In *2nd International Conference on Learning Innovation*. Springer, 2018. doi:10.5220/0008410102160220.
- [13] D. Susilowati, I. N. S. Degeng, P. Setyosari, and S. Ulfa. Effect of collaborative problem solving assisted by advance organisers and cognitive style



- on learning outcomes in computer programming. *World Transactions on Engineering and Technology Education*, 17(1):35–41, 2019.
- [14] J. Yen and W. Liao. Effects of Cognitive Styles on Computational Thinking and Gaming Behavior in an Educational Board Game. *International Journal of Learning Technologies and Learning Environments*, 2(2):1–10, 2019. doi:10.52731/ijltle.v2.i2.477.
- [15] C. Pretorius, M. Razavian, K. Eling, and F. Langerak. Combining cognitive styles matters for female software designers. *IEEE Software*, 38(2):64–69, 2020. doi:10.1109/ms.2020.3043663.
- [16] A. Cox and M. Fisher. Programming style: Influences, factors, and elements. In *2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 82–89. IEEE, 2009. doi:10.1109/ACHI.2009.48.
- [17] A. Setiawan, I. Degeng, C. SA'DIJAH, and H. Praherdhiono. The Effect Of Collaborative Problem Solving Strategies And Cognitive Style On Students' Problem Solving Abilities. *Journal for the Education of Gifted Young Scientists*, 8(4): 1618 – 1630, 2020. doi:10.17478/jegys.812781.
- [18] M. N. Kholid, P. S. Hamida, L. N. Pradana, and S. Maharani. Students' Critical Thinking Depends On Their Cognitive Style. *International Journal of Scientific and Technology Research*, 9(1):1045–1049, 2020. ISSN 2277-8616. doi:10.17478/jegys.812781.
- [19] Y. Yusnaini, K. Dewi, and A. Novriansa. Cognitive Style and Cognitive Mapping: Experimental Study in Accounting Decision Making. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 17(6):7151–7168, 2020.
- [20] M. Muzaini, M. Hasbi, and N. Nasrun. The Role of Students' Quantitative Reasoning in Solving Mathematical Problems Based on Cognitive Style. *Vygotsky: Jurnal Pendidikan Matematika dan Matematika*, 3(2):87–98, 2021. ISSN 2656-5846. doi:10.30736/voj.v3i2.380.
- [21] J. C. Hung and C. Wang. The Influence of Cognitive Styles and Gender on Visual Behavior During Program Debugging: A Virtual Reality Eye Tracker Study. *Human-centric Computing and Information Sciences*, 11(22):1–21, 2021.
- [22] T. Michaeli and R. Romeike. Improving Debugging Skills in the Classroom – The Effects of Teaching a Systematic Debugging Process. In *Proceedings of the 14th Workshop on Primary and Secondary Computing Education*, pages 1–7. ACM, 2019. ISBN 978-1-4503-7704-1. doi:10.1145/3361721.3361724.
- [23] Z. Karimi, A. Baraani-Dastjerdi, N. Ghasem-Aghaee, and S. Wagner. Links between the personalities, styles and performance in computer programming. *Journal of Systems and Software*, 111:228–241, 2016. doi:10.1016/j.jss.2015.09.011.
- [24] J. Kotrlik and C. Higgins. Organizational research: Determining appropriate sample size in survey research appropriate sample size in survey research. *Information technology, learning, and performance journal*, 19(1):43–50, 2001.
- [25] C. W. Allinson and J. Hayes. The Cognitive Style Index: A Measure of Intuition-Analysis For Organizational Research. *Journal of Management Studies*, 33(1):119–135, 1996. doi:10.1111/j.1467-6486.1996.tb00801.x.
- [26] G. J. McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999.
- [27] M. Shepperd and S. MacDonell. Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8):820–827, 2012. doi:10.1016/j.infsof.2011.12.008.
- [28] H. A. R. Daraghme. The impact of learning computer programming on the development of high school students cognitive abilities in the uae. *The British University in Dubai (BUiD)*, 2019.
- [29] L. Baldacchino, D. Ucbasaran, and L. Cabantous. Linking Experience to Intuition and Cognitive Versatility in New Venture Ideation: A Dual-Process Perspective. *Journal of Management Studies*, 2022. doi:10.1111/joms.12794.



Zahra Karimi has been an assistant professor in the Department of Computer Science at Shahrekord University since 2017. She received her B.Sc degree in software engineering from the Isfahan University of Technology, an MSc degree in software engineering from the Iran University of Science and Technology in 2001, and a Ph.D. degree in software engineering from Isfahan University in 2016. Zahra Karimi worked as a professional developer in the industry from 2002–2009. She was a visiting Ph.D. scholar at the Institute of Software Technology, the University of Stuttgart, from 2012–2014. Her research interests are data mining and empirical and behavioral methods in software engineering.



S. Fatemeh Noorani received her B.Sc. in Applied Mathematics from the Iran University of Science & Technology and her M.Sc. in Computer Engineering from Shahid Beheshti University, and her Ph.D. degree in computer engineering from Isfahan University of Technology, IRAN, in 2018. She is currently an Assistant Professor at Payame Noor University. She is involved in User Modeling, Data Mining, Educational Data Mining, and Game Theory research.

