To study the factors affecting green software development in India

Aniruddha N Pawgi

Senior Business Analyst
Accenture, Mumbai, India.

Abstract- **This study aims to identify, rank, and determine the significance of factors impacting green software development to ameliorate the sustainability dimension in the software assiduity. The field of sustainable and green software engineering is still youthful. The study reviewed the rearmost exploration and used an abstract model that demonstrated the consolidated life cycles of sustainable products and principle sustainable dimension confines. The use of software is growing due to numerous operations taking large and complex software. The greening software process can optimize energy consumption throughout all phases of development exertion. Integrating sustainability and waste rudiments is essential to ensure compliance with green process norms. The data required for this research was collected using a structured questionnaire which was filled in by the users of various IT professionals having experience. The research study surveys 120 IT professionals to explore the factors affecting adoption of green soft- ware technology in India's IT market. The research reveals the factors in SDLC-Requirement, Coding and Testing have been found significantly associated with adoption of green software technology. Furthermore, because of this study, businesses may adjust and focus on the most important phase in SDLC which could contribute to sustainable software.**

Keywords: **Green Software Development, Coding, Testing, Requirement.**

1 Background

The software industry is a significant contributor to the global economy, with extensive applications in almost all industries. In 2022, the global software market was valued at approximately 651 billion U.S. dollars and is projected to grow to over 800 billion U.S. dollars by 2026. However, the software industry's environmental impact is significant, with software production and use contributing to greenhouse gas emissions and resource depletion. India's software industry is a critical player in the global software market, with a market size of approximately 234 billion U.S. dollars in 2022. However, India's software industry faces several sustainability challenges, including high energy consumption and carbon emissions from software production and use. The energy usage of the Indian software sector is anticipated to rise quickly; predictions show that by 2025, energy usage would have tripled. Despite the environmental impact of software production and use,
The software industry has received relatively little attention in terms of sustainability. In recent years, there has been a growing interest in green and sustainable software development practices. However, the adoption of these practices is still in its early stages, particularly in developing countries such as India.

2 Introduction

The software development life cycle (SDLC) is a process used by software development teams to design, develop, test, deploy and maintain high-quality software. The SDLC typically consists of several phases, which are as follows: 1. Requirements Gathering: In this phase, the development team works closely with stakeholders to identify and gather the software requirements. The team will analyze the needs and goals of the stakeholders and document the functional and non-functional requirements of the software. 2. Design: In this phase, the development team will create a design that will meet the requirements. The design may include architecture diagrams, user interface design, database schema, and other design documents. This phase may involve several iterations to ensure that the design meets the requirements. 3. Coding: In this phase, the development team will write the code for the software based on the design. The code will be reviewed by the development team and may undergo several rounds of testing to ensure that it is of high quality. 4. Testing: During this phase, testing is performed on the software to validate that it is error-free and can satisfactorily meet the requirements. This may include functional, performance, and security testing. The testing phase is usually iterative, and the development team may fix any bugs found during testing and retest the software. 5. Implementation: In this phase, the software is deployed to the production environment. This may involve installing the software, configuring it for use, and training end-users on how to use it. The SDLC is an iterative process, which means that the development team may go back and forth between the phases until the software meets the requirements. The SDLC is essential to ensure that the software is developed in a structured, planned, and systematic way, and that it meets the needs of the end-users.

3 Literature Review

In the paper by Abdullah Green IT practices are becoming a key business component for many organizations, and their implementation can be applied by any organization is concluded. The study also emphasized the potential economic benefits of Green IT practices, such as saving energy, paper, water, transportation, physical space, maintenance, and waste, as well as improving the organization's image, respecting the environment, and enriching their employees. The study provided a useful guide for organizations to bring in greener technologies in the application of Information Technology and identified the largest gaps to be identified.

In this paper by Betz the need for accountability arrangements for designing and using digital systems in a responsible and sustainable way is discussed. The authors argue that a sustainability transformation requires new ways to elicit, analyze, and influence the

effects of sociotechnical systems in all spheres of society, including digital systems. They identify several niches where this is enabled, such as software engineering research and teaching, culture and public discourse, and policy activities that try to bring sustainability and digitalization together. However, they note that these efforts are still niche trends and that systemic efforts of engendering a transformation towards sustainability are not yet particularly strong. The authors argue that software engineering is a key site for enabling sustainability by design in digital systems and that multidisciplinary perspectives on sustainability and software engineering are needed. They use a social scientific sustainability transformation model to connect the state of research in their fields and explore accountability arrangements in software engineering. The paper concludes by discussing possible outlooks from here and research gaps that need to be addressed.

In this paper by Gil the importance of sustainable development and the role of information and communication technology (ICT) in achieving it is discussed. It highlights the impact of ICT on global CO2 emissions and the need for sustainable software development. The authors propose using AI techniques to identify key factors that predict program correctness and programmer performance. The paper also acknowledges the funding sources and contributions of the authors.

In this paper by Hejri the concept of green software development and the factors that influence it is discussed. The authors conducted semi-structured interviews with software developers, product managers, project managers, IT consultants, software architects, re- searchers, business analysts, and software test managers to gather their opinions on green software development and the factors that affect it. The interviews were analyzed using content analysis, and the identified factors were examined using a pairwise questionnaire and a fuzzy approach to deal with uncertainty and ambiguity. The paper concludes that sustainable and green software has become critical in the software engineering society, and the main goal is to promote stability and greening at any stage of the software development process.

In this paper Derbenev various software tools and techniques used in green chemistry, including predictors for yield, solvent selection, and other green metrics is discussed. It also discusses the use of electronic lab notebooks (ELNs) and retro-synthetic software in green chemistry. The paper highlights the potential of machine learning and AI in predicting and improving the environmental impact of chemical reactions. Overall, the paper emphasizes the importance of using software tools and techniques to promote sustainable and environmentally friendly practices in chemistry.

In this paper Yahaya presents a literature review on green software products, software sustainability, and related issues. The study identifies seven primary measurements as- sociated with green elements: productivity, cost reduction, usability, employee support, tool support, energy efficiency, and resource efficiency. The empirical analysis reveals that the green aspects are associated with sustainable dimensions: social, economic, and environmental. The study discloses the relationships between the dimensions and measurements. The paper concludes that a green assessment model is required to evaluate and assess the critical green measurements for software products. The authors suggest that a green software model (GSM) that includes the assessment process and criteria is needed to guarantee adequate business conditions in software development.

In this paper by R the concept of green software development, which aims to reduce car- bon emissions, save energy, and minimize waste in the software development process. The paper presents an enhanced model for the software development life cycle (SDLC) that incorporates measurements of carbon emissions and paper usage to help organizations move towards greener and sustainable software development. The paper also presents an empirical study conducted in Malaysia to investigate the characteristics of software developers and their experiences in software development activities. The study found that most respondents were software developers and engineers with 6 to 10 years of experience. The paper concludes that green software development is a relatively new study area in green computing and that there is a need for models that define how software can be developed and maintained in an environmentally friendly way.

In this paper by Kern the concept of green computing and green software, and the need for eco-labels for software products. It presents the results of a survey conducted to determine the awareness and interest of end-users in

environmental issues of software and their acceptance of eco-labels. The survey found that while there is an interest in environmental topics, this does not extend to environmental issues of software. However, there is a general interest in the presented aspects of green software, and the idea of an eco-label for software products is attractive, especially in terms of energy efficiency and hardware efficiency. The paper concludes that there is a need to better inform end-users about eco-labels for software products and to develop a certification for green software.
In this paper by Raisan the importance of sustainability in software engineering and the need for research in the field of green and sustainable software engineering is discussed. The paper presents a conceptual model for sustainable software engineering product and explores potential factors for improving sustainability in green software based on the life cycle of the software product process. The paper highlights the environmental, social, and economic impacts of software development, usage, distribution, and disposal phases. The paper concludes that sustainability in software engineering is a critical exploration subject that will be of great significance in the following years, and general work on its centrality is required.
In this paper by Kumarthe concept of Green Database Design in the Software Develop- ment Life Cycle (SDLC) phase is discussed. It proposes the use of green technologies to optimize energy utilization during the database design process. The paper presents formulas and processes for calculating energy usage and capacity planning in servers. The approach aims to reduce carbon emissions and increase the efficiency of software products without compromising the environment. The paper concludes that the initial step towards sustainable software engineering is to implement Green Database Design approaches using green technologies in each database design.
In this paper by Wolfram research on sustainable software development processes and models is discussed. It notes that while there have been efforts to identify relevant papers, there is still a possibility that more data sources and different search terms could yield more relevant papers. The paper identifies two categories of models related to sustainable software development processes and notes that there is a need for practical verification of these models. The paper also compares its findings to a previous literature review on sustainability in software engineering and notes that only a small number of papers in that review pertained to sustainable software solutions.

4 Conceptual Diagram

It helps to visualize how the solution of problem statement would look like

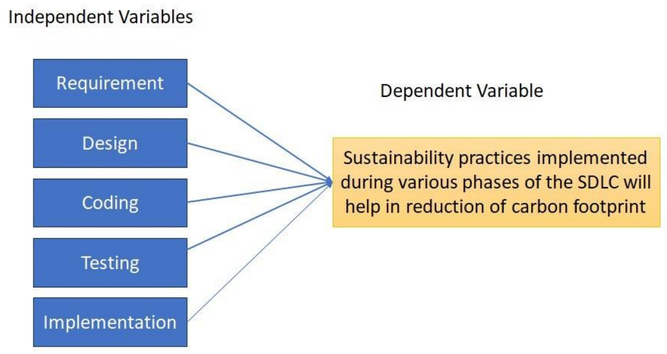

Independent Variables
Requirement
Design
Coding
Testing
Implementation
Dependent Variable
Sustainability practices implemented during various phases of the SDLC will help in reduction of carbon footprint

Figure 1: IV and DV relationship

4.1 Dependent and Independent Variables

Independent variables	Dependent Variable
Requirement	Sustainability practices implemented during various phases of the SDLC. will help in reduction of carbon footprint
Design	
Coding	
Testing	
Implementation	

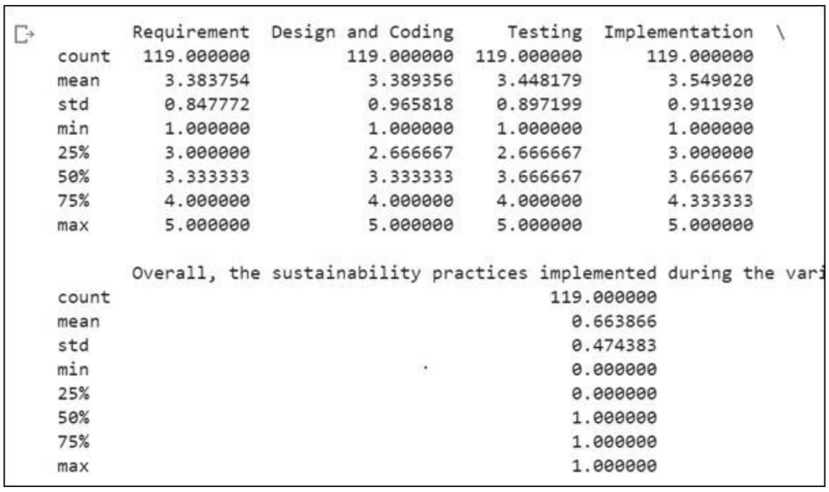
5 Research Objective

To identify the important factors(phases) in Software Development Life Cycle which contribute to, Green Software Development So a questionnaire is designed to for each of the variable and responses are taken on a 5-point Likert scale.

6 Research Methodology

The development of green software products and processes has become an increasingly important aspect of software engineering. Green software development focuses on reducing the environmental impact of software products and processes, by optimizing energy consumption and minimizing waste throughout the software development life cycle. The researcher has tried to understand the importance of factors affecting the green software development in SDLC (Software Development Life cycle). A google form questionnaire was circulated to get responses of the employees on various phase of SDLC and Sustainability dimensions. A dataset having the responses of 120 respondents has been considered. The dataset has in total 5 independent variables which are the phase of Software development life cycle.

7 Results and Conclusions



```
        Requirement  Design and Coding    Testing  Implementation  \
count    119.000000         119.000000 119.000000      119.000000
mean       3.383754           3.389356   3.448179        3.549020
std        0.847772           0.965818   0.897199        0.911930
min        1.000000           1.000000   1.000000        1.000000
25%        3.000000           2.666667   2.666667        3.000000
50%        3.333333           3.333333   3.666667        3.666667
75%        4.000000           4.000000   4.000000        4.333333
max        5.000000           5.000000   5.000000        5.000000

        Overall, the sustainability practices implemented during the vari
count                                          119.000000
mean                                             0.663866
std                                              0.474383
min                                              0.000000
25%                                              0.000000
50%                                              1.000000
75%                                              1.000000
max                                              1.000000
```

Figure 2: Software Development Sustainability Evaluation Dataset

This code snippet shows a summary statistic of a dataset with 119 observations and 5 variables related to sustainability practices implemented during the various phases of the software development life cycle (SDLC). The variables are:

- Requirement: mean score 3.38 and standard deviation 0.85
- Design and Coding: mean score 3.39 and standard deviation 0.97.
- Testing: mean score 3.45 and standard deviation 0.90
- Implementation: mean score 3.55 and standard deviation 0.91
- Overall, the sustainability practices implemented during the various phases of the SDLC will help in reduction of the carbon footprint and increase the sustainability of the software development process: mean score 0.66 and standard deviation 0.47.

The last variable is a binary variable with a value of 1 indicating that sustainability practices were implemented during the SDLC and 0 indicating otherwise. The mean of this variable is 0.66, suggesting that sustainability practices were implemented in most of the software development projects in the dataset. The standard deviation of this variable is 0.47, indicating that there is some variability in the implementation of sustainability practices across different projects. Overall, this dataset provides insights into the current state of sustainability practices in software development projects and can be used to identify areas for improvement in this domain.

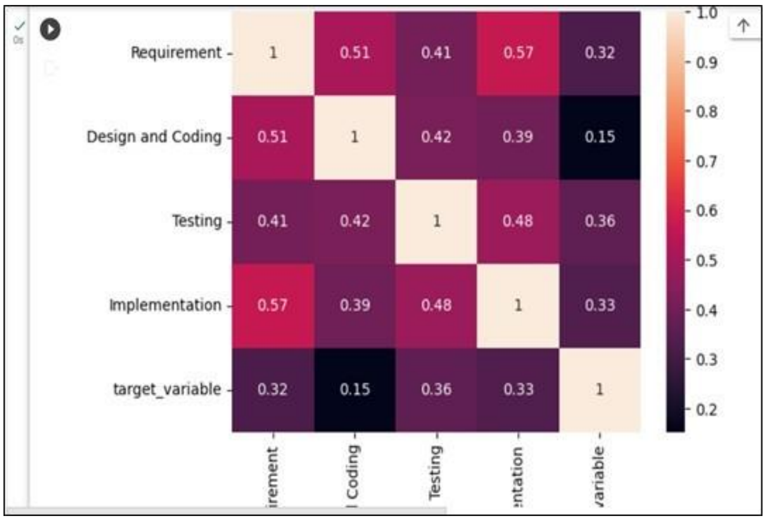
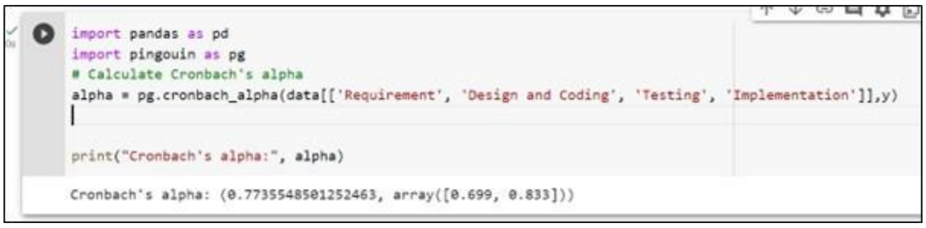


Figure 3: Heat Map

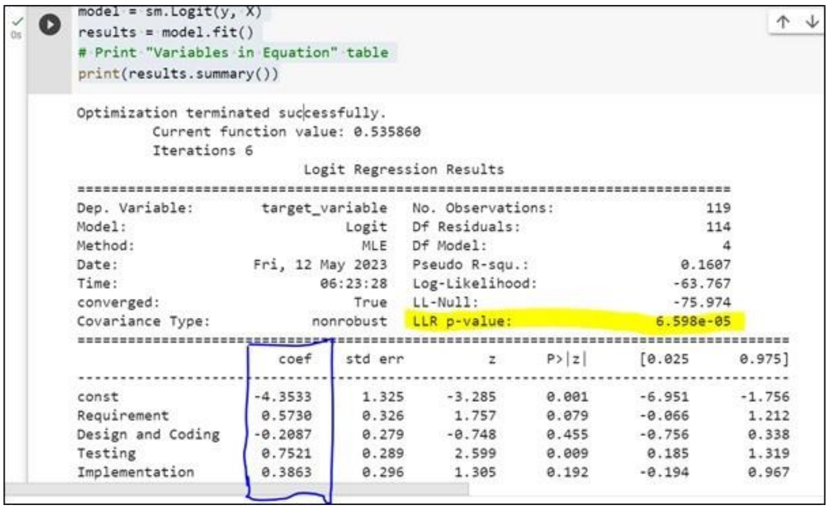
From this heat-map which we can say that values of correlation coefficients are less than 0.5 which so we can conclude that there exists no partial correlation between the independent variables which in this case is the phases in software development life cycle. So, this indicates that the questionnaire was properly designed for each of the independent variables.



```
import pandas as pd
import pingouin as pg
# Calculate Cronbach's alpha
alpha = pg.cronbach_alpha(data[['Requirement', 'Design and Coding', 'Testing', 'Implementation']],y)

print("Cronbach's alpha:", alpha)

Cronbach's alpha: (0.7735548501252463, array([0.699, 0.833]))
```

Figure 4: Cronbach's alpha test result

Cronbach's alpha is a measure of internal consistency reliability, often used to assess the reliability of psychometric tests or questionnaires. The alpha coefficient ranges from 0 to 1 and is a function of the average inter-item correlation and the number of items in the test. In general, a higher Cronbach's alpha indicates a greater degree of reliability and consistency in the test, suggesting that the items in the test are measuring the same underlying construct. A lower alpha coefficient may indicate that the items are not measuring the same construct or that there are other sources of error or inconsistency in the test. So, in this case Cronbach's alpha value is 0.77 so it suggests that there is a relatively high degree of internal consistency among the items in the measure. This means that the items in the measure are measuring the same construct and are likely to be reliable.



```
model = sm.Logit(y, X)
results = model.fit()
# Print "Variables in Equation" table
print(results.summary())

Optimization terminated successfully.
         Current function value: 0.535860
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:         target_variable   No. Observations:                  119
Model:                           Logit   Df Residuals:                      114
Method:                            MLE   Df Model:                            4
Date:                 Fri, 12 May 2023   Pseudo R-squ.:                  0.1607
Time:                         06:23:28   Log-Likelihood:                -63.767
converged:                        True   LL-Null:                       -75.974
Covariance Type:             nonrobust   LLR p-value:                 6.598e-05
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -4.3533      1.325     -3.285      0.001      -6.951      -1.756
Requirement           0.5730      0.326      1.757      0.079      -0.066       1.212
Design and Coding    -0.2087      0.279     -0.748      0.455      -0.756       0.338
Testing               0.7521      0.289      2.599      0.009       0.185       1.319
Implementation        0.3863      0.296      1.305      0.192      -0.194       0.967
```

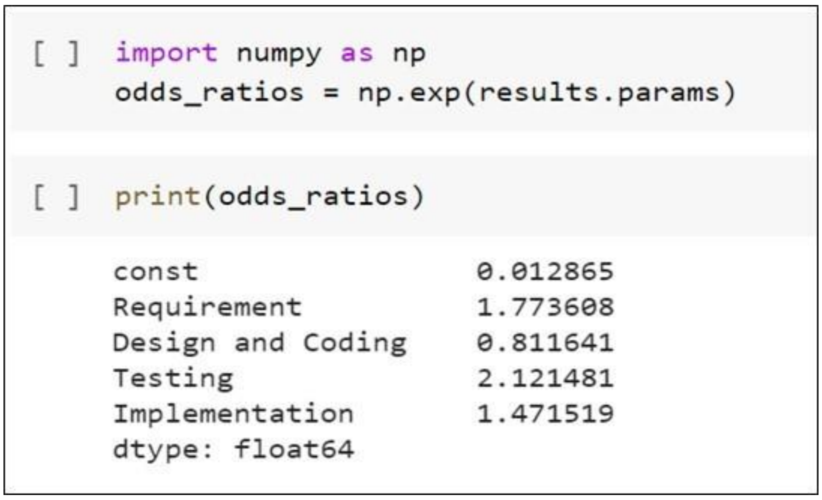
Figure 5: Variables in the equation

The coefficients of logit function are highlighted. The direction of independent variables is denoted by positive and negative signs in front of each coefficient. While almost all in- dependent variables have a positive coefficient (signaling direct association with dependent variable) but only one independent variable has negative coefficient (signaling indirect association with dependent variable).

1. Requirement: - The requirement phase is critical in green software development, as it helps identify and prioritize the features and functionalities of the software that are necessary for reducing its environmental impact. During this phase, the software development team can work with stakeholders to identify energy-efficient requirements that can be incorporated into the design and development phases. This is evident from the positive coefficient of this independent variable.
2. Design and Coding: - Design and coding do contribute to green software development, but their impact on the environment is typically considered to be smaller compared to other phases of software development such as requirement analysis, testing, and maintenance. This is because design and coding are mainly focused on creating the software itself and do not involve large-scale resource consumption or energy usage. However, it is important to note that the efficiency of code and design decisions can have a significant impact on the overall performance and energy consumption of the software, especially in resource-intensive applications. Therefore, it is important to consider environmental impact throughout the entire software development lifecycle, including the design and coding phases. This is evident from the negative coefficient of this independent variable.
3. Testing: - The testing phase is critical in green software development because its consumption that the

software is optimized for energy efficiency and minimal resource consumption. Through testing, developers can identify and address potential inefficiencies in the software's design or code, such as inefficient algorithms or resource-intensive processes. By optimizing the software's energy efficiency during the testing phase, developers can significantly reduce the amount of energy and resources required to run the software in production, resulting in a more sustainable and environmentally friendly product. Additionally, testing can help to identify and prevent issues that could lead to increased energy consumption or environmental harm, such as software bugs or vulnerabilities that could result in data breaches or other system failures. Overall, the testing phase plays a critical role in ensuring that green software development practices are implemented throughout the software development lifecycle. This is evident from the positive coefficient of this independent variable.

4. Implementation: - The implementation phase in software development involves the actual deployment of the software into the production environment. This phase is critical for green software development as it involves setting up the infrastructure and hardware required to run the software efficiently. During the implementation phase, the software developers need to ensure that the hardware and infrastructure are optimized to consume less energy and resources. This includes selecting energy-efficient servers, optimizing data storage and retrieval mechanisms, and implementing techniques to reduce energy consumption during software execution. Moreover, in the implementation phase, the developers can also consider using energy-efficient programming techniques such as code optimization, parallel processing, and memory management. These techniques can help reduce the energy consumption of the software during its execution. This is evident from the positive coefficient of this independent variable. Also, the LLR p-value is less than 0.05 which means the Overall Binomial Logistic Regression model is significant.

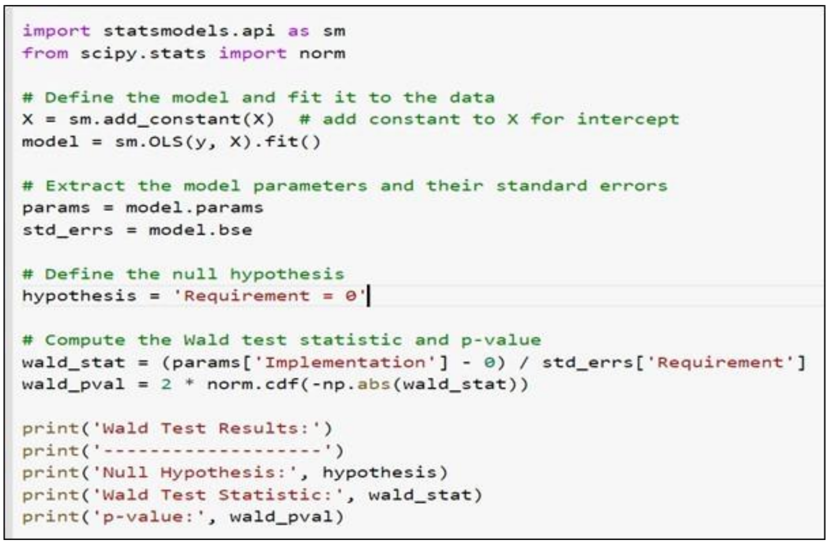


```
[ ] import numpy as np
    odds_ratios = np.exp(results.params)

[ ] print(odds_ratios)

    const                  0.012865
    Requirement            1.773608
    Design and Coding      0.811641
    Testing                2.121481
    Implementation         1.471519
    dtype: float64
```

Figure 6: Odds Ratio

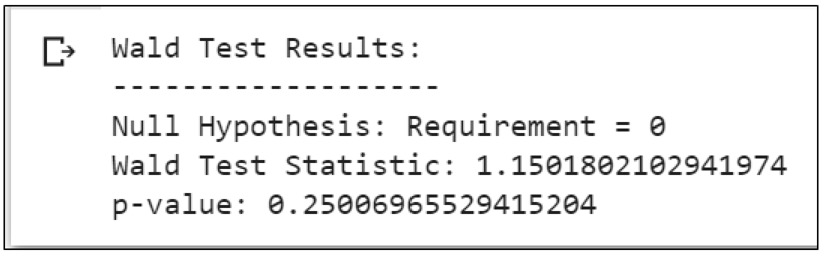
Odds ratio is a measure of the association between two binary variables. It is defined as the ratio of the odds of an event occurring in one group to the odds of the same event occurring in another group. In logistic regression, odds ratio is used to estimate the effect of an independent variable on the dependent variable. Suppose we have two groups, A and B, and we want to calculate the odds ratio of an event E occurring in these two groups. The odds of E occurring in group A is defined as the ratio of the number of individuals in group A who experience E to the number of individuals in group A who do not experience E. Similarly, the odds of E occurring in group B is defined as the ratio of the number of individuals in group B who experience E to the number of individuals in group B who do not experience E. The odds ratio is then calculated as the ratio of the odds of E occurring in group A to the odds of E occurring in group.
B. Odds Ratio = (a/b) / (c/d) where a is the number of individuals in group A who experience E, b is the number of individuals in group A who do not experience E, c is the number of individuals in group B who experience E, and d is the number of individuals in group B who do not experience E. Odds ratio is used in various fields, such as: Medical research: to study the risk factors associated with diseases, such as cancer, diabetes, and heart disease. - Economics: to study the factors influencing consumer behavior and purchasing decisions. - Social sciences: to study the factors influencing human behavior and attitudes.
Interpretation of odds ratio the odds ratio can take values from 0 to infinity. If the odds ratio is 1, it means that there is no association between the independent variable and the dependent variable. If the odds ratio is greater than 1, it means that the odds of the event occurrence are higher in group A than in group B. If the odds ratio is less than 1, it means that the odds of the event occurrence are higher in group B than in group A.
In the above example we want to study the effect of Requirement phase of SDLC on the Green Software Development. We divide it int 2 groups as Requirement phase has a significant impact on Green Software Development and Requirement phase does not have any significant impact on Green Software Development. The odds ratio of 1.777 for the requirement phase in green software development suggests that there is a positive association between performing a thorough requirement phase and achieving green software development goals. This means that the odds of achieving green software development goals are 1.777 times higher when a thorough requirement phase is performed compared to when it is not performed.
The odds ratio of 0.81164 for the design and coding phase in green software development indicates that the likelihood of the outcome (green software development) is less likely when compared to the reference category (not green software development) by a factor of 0.81164 in the design and coding phase. This means that the design and coding phase may not significantly contribute to achieving green software development.
The odds ratio of 2.1214 for the testing phase in green software development indicates that the odds of a successful and environmentally friendly software product increase by a factor of 2.1214 for each unit increase in the testing phase. This suggests that a greater emphasis on testing can lead to the identification and resolution of environmental issues, resulting in a more sustainable software product.
The odds ratio of 1.77 for the Implementation phase in green software development indicates that there is a positive association between the Implementation phase and green software development. It means that as the Implementation phase increases, the likelihood of achieving green software development also increases. However, it is important to note that the odds ratio only measures the strength of the association between two variables and does not establish causality.

Analysis of Wald statistic and Hypothesis Testing
The Wald statistic is a measure of the strength of the relationship between the independent variables and the dependent variable in logistic regression. The Wald statistic is calculated as the square of the estimate divided by its variance. A large Wald statis- tic indicates a strong relationship between the independent variables and the dependent variable. Calculation of Wald statistic The Wald statistic is calculated using the following formula: Wald statistic = (- 0) / SE () Where is the estimated coefficient for the independent variable, 0 is the hypothesized value of the coefficient, and SE () is the standard error of the coefficient.
Hypothesis Testing in Logistic Regression Hypothesis testing is a statistical method used to test a hypothesis about a population parameter using sample data. In logistic re- grissino, we use hypothesis testing to determine whether the relationship between the independent variables and the dependent variable is statistically significant. The Null Hypothesis (Ho)- The null hypothesis in logistic regression is that there is no significant relationship between the independent variables and the dependent variable. The Alter- native Hypothesis (H1) - The alternative hypothesis in logistic regression is that there is a significant relationship between the independent variables and the dependent variable. The Wald Test the Wald test is a statistical test used to determine the significance of the relationship between the independent variables and the dependent variable in logistic regression. The Wald test is based on the Wald statistic. If the Wald statistic is greater than the critical value, we reject the null hypothesis and conclude that there is a significant relationship between the independent variables and the dependent variable.
P-value in Logistic Regression The p-value is a measure of the strength of evidence against the null hypothesis. It represents the probability of observing a test statistic as extreme or more extreme than the observed test statistic, assuming the null hypothesis is true. In logistic regression, we use the p-value to determine whether the relationship between the independent variables and the dependent variable is statistically significant.
Calculation of P-value the p-value in logistic regression is calculated as follows: P-value.
= P (|Z| > |z|) Where Z is a standard normal distribution and z is the test statistic. The test statistic is calculated as the Wald statistic divided by its standard error. If the p-value is less than the significance level, we reject the null hypothesis and conclude that there is a significant relationship between the independent variables and the dependent variable.
Significance Level The significance level is the probability of rejecting the null hypothesis when it is true. In logistic regression, the most common significance level is 0.05, which means that we are willing to accept a 5After performing logistic regression and calculating the Wald statistic and p-value, we need to interpret the results. If the p-value is less than the significance level (usually 0.05), we can reject the null hypothesis and conclude that there is a significant relationship between the independent variables and the dependent variable. In other words, the independent variables are good predictors of the dependent variable. On the other hand, if the p-value is greater than the significance level, we fail to reject the null hypothesis, which means that there is not enough evidence to conclude that there is a significant relationship between the independent variables and the dependent variable. The Wald statistic measures strength of the relationship, while p value measures the probability of observing such a relationship by chance.
Now let us see code consisting of Wald statistic and p value for each independent variable.



```
import statsmodels.api as sm
from scipy.stats import norm

# Define the model and fit it to the data
X = sm.add_constant(X)  # add constant to X for intercept
model = sm.OLS(y, X).fit()

# Extract the model parameters and their standard errors
params = model.params
std_errs = model.bse

# Define the null hypothesis
hypothesis = 'Requirement = 0'

# Compute the Wald test statistic and p-value
wald_stat = (params['Implementation'] - 0) / std_errs['Requirement']
wald_pval = 2 * norm.cdf(-np.abs(wald_stat))

print('Wald Test Results:')
print('-------------------')
print('Null Hypothesis:', hypothesis)
print('Wald Test Statistic:', wald_stat)
print('p-value:', wald_pval)
```

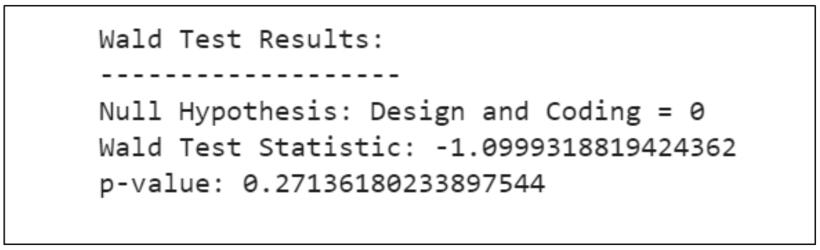
Figure 7: Python code for Wald test



```
Wald Test Results:
-------------------
Null Hypothesis: Requirement = 0
Wald Test Statistic: 1.1501802102941974
p-value: 0.25006965529415204
```

Figure 8: Wald test Result for Requirement

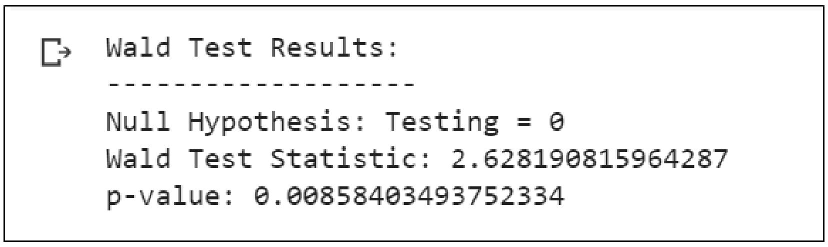
Here, the Wald statistic value is 1.15. To determine the statistical significance of this statistic we need to set up hypothesis and from p value to determine significance of this statistic. The Null Hypothesis (Ho)- There is no significant relationship between Requirement and the dependent variable. The Alternative Hypothesis (H1) -There is a significant relationship between the Requirement and the dependent variable. A p-value of 0.25 for the Wald statistic in the requirement phase indicates that the probability of observing such an extreme result, if the null hypothesis is true, is 0.25. In other words, there is not enough evidence to reject the null hypothesis at a significance level of 0.05, which is a commonly used threshold in hypothesis testing. Therefore, we cannot conclude that the requirement phase has a statistically significant effect on the greenness of software development.



```
Wald Test Results:
-------------------
Null Hypothesis: Design and Coding = 0
Wald Test Statistic: -1.0999318819424362
p-value: 0.27136180233897544
```

Figure 9: Wald test Result for Design and Coding

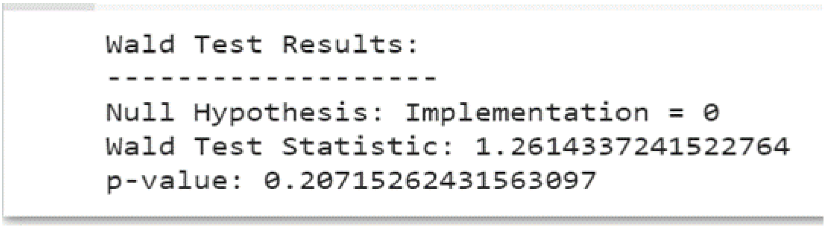
Here, the Wald statistic value is -1.099. To determine the statistical significance of this statis- tic we need to set up hypothesis and from p value to determine significance of this statistic. The Null Hypothesis (Ho)- There is no significant relationship between Design and Coding and the dependent variable. The Alternative Hypothesis (H1) -There is a significant relationship between the Design and Coding and the dependent variable. A p-value of 0.27 for the Wald statistic in the Design Coding phase suggests that there is not enough evidence to reject the null hypothesis that the coefficient for this phase is equal to zero. This means that the Design Coding phase may not have a significant effect on the outcome variable in the model. However, it is important to note that a p-value of 0.27 is relatively high, indicating that there may be some uncertainty in this conclusion and further investigation may be necessary.



```
Wald Test Results:
-------------------
Null Hypothesis: Testing = 0
Wald Test Statistic: 2.628190815964287
p-value: 0.008584034937523334
```

Figure 10: Wald test Result for Testing

Here, the Wald statistic value is 2.62. To determine the statistical significance of this statistic we need to set up hypothesis and from p value to determine significance of this statistic. The Null Hypothesis (Ho)- There is no significant relationship between Testing and the dependent variable. The Alternative Hypothesis (H1) -There is a significant relationship between the Testing and the dependent variable. A p-value of 0.00858 for the Wald statistic in the Testing phase indicates that there is strong evidence against the null hypothesis. In other words, the Testing phase has a statistically significant effect on the outcome (e.g., in the context of green software development). The lower the p-value, the stronger the evidence against the null hypothesis. In this case, the p-value is below the commonly used threshold of 0.05, suggesting that the effect of the Testing phase is likely to be meaningful and not due to chance.



```
Wald Test Results:
-------------------
Null Hypothesis: Implementation = 0
Wald Test Statistic: 1.2614337241522764
p-value: 0.20715262431563097
```

Figure 11: Wald test Result for Implementation

Here, the Wald statistic value is 1.26. To determine the statistical significance of this statistic we need to set up hypothesis and from p value to determine significance of this statistic. The Null Hypothesis (Ho)- There is no significant relationship between Implementation and the dependent variable. The Alternative Hypothesis (H1)-There is a significant relationship between the Implementation and the dependent variable.
A p-value of 0.207 for the Wald statistic in the Implementation phase means that there is not enough evidence to reject the null hypothesis that the coefficient for the Implementation phase is equal to zero. In other words, there is no significant association between the Implementation phase and the outcome variable in the model.

8 Implications Of Study

1. Adoption of energy-efficient hardware: Companies should invest in energy-efficient hardware like servers, storage devices, and other infrastructure components. This will reduce the energy consumption and carbon footprint of the software development process.
2. Use of renewable energy sources: Companies should use renewable energy sources like solar and wind power to power their data centers and other infrastructure components. This will not only reduce their carbon footprint but also help in cost savings.
3. Virtualization and Cloud Computing: Companies should adopt virtualization and cloud computing to reduce the number of physical servers required to host applications. This will reduce energy consumption and help in cost savings.
4. Green coding practices: Developers should follow green coding practices like using efficient algorithms and data structures, optimizing code for energy consumption, and reducing unnecessary computations. This will reduce energy consumption and improve software performance.
5. Green testing practices: Testing teams should adopt green testing practices like using energy-efficient test equipment and optimizing test cases for energy consumption. This will reduce energy consumption and help in cost savings.

9 Limitations of study

The study was based on just 120 responders from around the country, which is a relatively tiny sample size for producing a precise conclusion. The study may be subject to selection bias, as the participants may have been self-selected, which could limit the generalizability of the results. The findings of the study may not be generalizable to other countries or regions due to differences in culture, regulations, and technology adoption. Green software development practices may not be standardized across different companies, industries or regions, which could limit the comparability of results. The study may have been conducted over a limited timeframe, which could limit the ability to observe long-term trends in green software development practices. The study may be subjective in nature, as different researchers may have different interpretations of the data or different approaches to analyzing it.

REFERENCES:

1. Abdullah (2017). Systematic literature review on green it practices and executional fac- tors. Vol-6 issue-2,, International Journal of Supply Chain Management.
2. Betz (2022). Transformation2: Making software engineering accountable for sustainability. Technical report vol-1 issue-1,, Responsible Technology.
3. Derbenev, I. N. (2023). Software tools for green and sustainable chemistry. Volume-2, issue-11,, ScienceDirect ELSEVIER.
4. Gil, D. (2018). The effect of green software: A study of impact factors on the correctness of software. Technical report issue 1,, MDPI Sustainability.
5. Hejri, F. M. (2023). Analyzing the factors influencing green software development. Volume 2 no-3,, IJNAA.
6. Kern, E. (2018). Sustainable software products—towards assessment criteria for resource and energy efficiency. Volume 12, issue 1, Future Generation Computer Systems.
7. Kumar (2016). Green database design model in software development life cycle phase. Volume 1, issue 1,

Indian Journal of Science and Technology.

8. R, K. (2019). Green software process based on sustainability dimensions. Volume 1, issue 1,, INCITEST.
9. Raisan, K. (2016). Exploring potential factors in green and sustainable software products. Volume 2015, issue 1,, international Journal on Informatics Visualization.
10. Wolfram, N. (2017). Sustainability in software engineering. Technical report, ELE- SEVER.
11. Yahaya, J. (2022). Green measurements for software products based on sustainability dimensions. Volume 3, issue 4,, Computer Systems Science Engineering.