FROM RATINGS TO INSIGHTS: MACHINE LEARNING FOR LIKERT SCALE DATA INTERPRETATION

BY

ANDREW CHOONG KAH HOONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2026

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express thanks and appreciation to my supervisor, Ts Dr. Tong Dong Ling and my moderator, Ms Lye Kah Hooi who have given me a golden opportunity to involve in this project. Besides that, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

Likert-scale questionnaires are a cornerstone of research, yet traditional analysis often overlooks nuanced inter-question relationships, while standard machine learning models frequently suffer from overfitting and lack the generalizability required to handle diverse survey datasets. This project developed and validated a web-based, interpretable framework that automates the analysis of user-uploaded survey data using an enhanced C4.5 decision tree algorithm. The system's primary technical contribution is a robust preprocessing pipeline that executes automated heuristic cleaning of identifier columns and integrates Chi-Squared feature selection to statistically filter irrelevant predictors. To ensure model simplicity, the implementation utilizes Pessimistic Post-Pruning to algorithmically generalize the tree structure. A novel feature of the framework is the synthesis of quantitative decision rules with qualitative themes extracted from open-ended responses via Natural Language Processing (NLP), which are then interpreted by a Large Language Model (LLM) to provide narrative insights. System evaluation involved a real-world case study and a usability study with 40 respondents. Results from the Post-Study System Usability Questionnaire (PSSUQ) yielded an overall mean score of 1.84 (on a 1–7 scale), indicating exceptional user satisfaction. Expert validation further confirmed that the AI-generated interpretations successfully unveiled complex item relationships and mirrored manual research findings, effectively bridging the gap between complex survey data and actionable, explainable insights.

Area of Study (Minimum 1 and Maximum 2): Machine Learning, Survey Analytics

Keywords (Minimum 5 and Maximum 10): Likert Scale, C4.5 Algorithm, Decision Tree, Data Cleaning, Chi-Squared Feature Selection, Interpretability, Pessimistic Pruning, Preprocessing Pipeline, Streamlit, Large Language Models.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\chi 2$	Chi-Squared statistic used to test independence between variables
Oi	Observed frequency of a value in the Chi-Squared contingency table
Ei	Expected frequency of a value in the Chi-Squared contingency table
$\sum$	Summation operator (Sum of)
S	A set of training examples or data samples
Sv	A subset of S where attribute A has value v
p_i	The proportion (probability) of examples belonging to class i
$\log_2$	Logarithm to the base 2

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
C4.5	Classification Algorithm 4.5
CSV	Comma-Separated Values
EDA	Exploratory Data Analysis
FYP	Final Year Project
GUI	Graphical User Interface
ID3	Iterative Dichotomiser 3
LLM	Large Language Model
ML	Machine Learning
NFR	Non-Functional Requirement
PDF	Portable Document Format
PSSUQ	Post-Study System Usability Questionnaire
SPSS	Statistical Package for the Social Sciences
UI	User Interface
UX	User Experience
XAI	Explainable Artificial Intelligence

CHAPTER 1

Introduction

1.1 Problem Statement and Motivation

Survey research, using Likert scales, is a common way of collecting quantitative data in social science, education and marketing. Although the data collection methods are very well stipulated, the methods of data analysis are underperforming. The standard modes of analysis which are mostly done through statistical programs like SPSS are more inclined towards summary statistics, or hypothesis testing [1]. This method usually fails to portray the fine, non-linear relationships between items in the questionnaire such as how some patterns in the survey response (e.g., low score on "Service" plus high on "Price") influence a survey result. Machine Learning (ML) methods can have algorithms, capable of capturing such complex relationships, yet existing academic methods are often rigid, command-line scripts, which are applicable only to a rigid, pre-processed set of data. They lack advanced data preprocessing techniques required to deal with the messiness of raw survey data sets, so that it is hard to use them with non-technical users [2].

This project was inspired by the idea that advanced survey analytics should be democratized, and that the data to narrative analysis should be easily accessible. The instrument, not merely a computation engine, but an intelligent analytical assistant is desperately needed. The system enables the researcher to concentrate on the meaning of findings and not the mechanics of the analysis by automating technically challenging processes like data cleaning, statistical feature selection and processing of open-ended textual feedback. Moreover, due to the emergence of Generative AI, there is a one-of-a-kind chance to convert the lifeless statistical outputs into the readable explanations written in human language. This project seeks to address the interpretation bottleneck by combining Explainable AI (XAI) and Natural Language Processing (NLP) with Large Language Models (LLMs) so that users of the service can get immediate answers, understandable to them, based on their quantitative and qualitative data.

1.2 Project Objectives

The primary goal of the project is to come up with a generalizable and interpretable machine learning model that will automatically preprocess unstructured survey data to discover important decision-making paths and create narrative insights. This is done by the following objectives:

- **To Develop a Robust Data Preprocessing Pipeline:** To create and implement heuristics to automate the identification and removal of non-predictive columns (e.g., IDs, time/date stamps). This incorporates incorporating an obligatory Chi-Squared Feature Selection of quantitative data and Natural Language Processing (NLP) methods (lemmatization and term-frequency counts) of qualitative, open-ended answers. This also guarantees that only meaningful themes and important attributes are put in the model and the results derived.
- **To Implement Interpretable Machine Learning:** To implement the C4.5 decision tree algorithm, with Pessimistic Post-Pruning, in a program. This goal is to produce models that are accurate and simple (readable) to improve the "black box" issue in AI.
- **To Integrate Generative AI for Automated Interpretation:** To link the decision tree output (logic rules) and qualitative themes produced by NLP with a Large Language Model (LLM). This will enable the automatic generation of executive summaries, insight into relationships and recommendations, that translates the quantitative logic and qualitative themes into language.
- **To Develop a User-Centric Web Application:** To integrate the various components into a user-friendly Streamlit web interface that is easy to use. This involves the creation of a professional, downloadable PDF report, which integrates the result of exploratory data analysis (EDA) statistics, graphical visualizations, extracted rules and AI insights into a single report.

1.3 Project Scope and Direction

This project will have the following scope: design and development of a working prototype of a web-based software to analyze and interpret Likert-scale survey data automatically. The end product is a Web app programmed in Python with the Streamlit framework. The system will receive raw and structured survey data in common formats (CSV) and generate a detailed report of the analysis.

The solution includes an integrated end-to-end pipeline that involves automated preprocessing module to apply heuristics to clean data and eliminate identifiers, Natural Language Processing (NLP) module to extract themes in open-ended textual responses, statistical engine to use the Chi-Squared test of mandatory feature selection, a modelling engine with the implementation of the C4.5 decision tree algorithm with pessimistic post-pruning, and an interpretation

The system is specifically interested in ordinal data, categorical data, and short-textual data in the form of modern survey but specifically excludes unstructured data types such as images or audio. The main deliverable is a downloadable PDF report that summarizes descriptive statistics, variable distribution plots, visual decision trees, logic rules extracted, qualitative keyword frequencies and AI-generated narrative insights. Finally, the project can be reused as a framework to analyze a variety of datasets using a Graphical User Interface (GUI) and does not require writing any code, unlike a hardcoded script with a single particular study.

1.4 Contributions, Impact and Significance

This project provides both significant technical innovations and practical utility for the research community. The specific contributions of this work are:

1. **A Generalizable Preprocessing Framework:** The project provides a new, automated pipeline of data engineering with heuristic algorithms to dynamically clean diverse survey data. It enables the processing of varied formats of surveys without writing code by getting past hard-coded scripts and going directly to the dirty data issue by enabling non-technical researchers to process survey data.

2. **Synthesis of Quantitative and Qualitative Methods:** The system is the first of its kind to combine classical statistical filtering (Chi-Squared testing) with more recent Natural Language Processing (NLP) algorithms. This tool is based on a more comprehensive and accurate foundation of interpretation when compared with quantitative analysis alone as it simultaneously analyzes numeric Likert data and open-ended text responses.

3. **Accessible Explainable AI (XAI):** The project shows how XAI can be applied in practice, combining the algorithmic transparency of a post-pruned C4.5 decision tree with the semantic clarity of Large Language Models (LLMs). The LLM converts technical, mathematical decision rules into prose texts in plain English.

4. **End-to-End Automated Reporting:** The project provides a fully automated user-friendly web application, which handles the whole workflow, and ends up producing the final, professional, consolidated PDF report on the exploratory data analysis, the visual model, and the findings by the AI.

What is important about this project is that it can democratize the process of advanced data analysis among social scientists, educators and marketers. Cleaning, modelling and interpreting the research process can greatly help in saving time by automating the most challenging aspects of research. Moreover, the transformation of the complex decision tree logic into plain English stories makes the system such that the empirical findings are available instantly to the stakeholders and non-statistical decision-makers. Finally, the tool will enable organizations and academic institutions to make quick and efficient decisions that are made transparent and evidence based.

1.5 Background Information

To understand thoroughly the technical aspect of this project one needs to create background knowledge on the nature of the data and the exact algorithms used in the architecture of the system. The Likert scale is the most commonly used psychometric scale to measure attitudes and perceptions, and it most commonly entails the respondent being asked to rate his or her level of agreement with a statement on a symmetric, ordinal scale. An important feature of Likert data is that it is not continuous, the psychological distance between rating of 3 (Neutral) and 4 (Agree) cannot be mathematically equal to the distance between 1 and 2 [3]. Mathematical models that treat this data as purely continuous may give fundamentally misleading results and require certain algorithmic treatment that honors the ordinal boundaries.

The major machine learning algorithm used in this framework is the C4.5 decision tree algorithm that was first designed by J.R. Quinlan [4]. The choice of decision trees in this project is due to their being a white-box type of model whose internal decision-making rules can be represented visually as a flowchart of IF-THEN rules. The C4.5 is an improvement of previous tree algorithms since it uses a measure known as the Gain Ratio to choose the most informative survey questions with which to divide the data at each node. This normalization ensures that the algorithm is not biased towards the attributes that have numerous unique answers but no information. Moreover, C4.5 uses pessimistic post-pruning, which is used to cut branches of the tree that simply model random noise in the training data, thus ensuring that the resulting model captures general patterns, and not anomalies, which are unlikely to be similarly present in real-world data.

Lastly, Data Preprocessing is the most important process of transforming raw and dirty data that is usually full of errors, omissions and unwanted items to a clean dataset that can be trusted to do accurate modelling. The data in real world survey exports often includes irrelevant columns, missing values and unstructured text. Data preparation and cleaning, according to the literature, are up to 80% of the work in data mining projects [5], [12], [13]. There are two main methods that this project automates this crucial step. To begin with, it employs the Chi-Squared test of Selection of Features, a statistical tool that determines the independence of categorical variables and then only those survey questions that have significant impact to the target result are retained. Second, it proposes Natural Language Processing (NLP) methods, namely, lemmatization and counting of terms, in order to identify qualitative themes that have an important meaning in the responses to the unstructured, open-ended interview questions. Collectively, these preprocessing measures enhance the final model clarity, accuracy, and interpretability to a large extent.

1.6 Report Organization

This report is divided into seven chapters, explaining how the project evolved from the beginning to the final evaluation:

- Chapter 1: Introduction provides the setting of the research background, the problem statement of the rigidity of the existing survey tools, objectives of the project, and the importance of the suggested generalizable framework.
- Chapter 2: Literature Review The theoretical background of Likert scales and psychometrics are discussed, as well as a review of the technologies employed (Python, Streamlit, C4.5 algorithm), and comparison of current analytical solutions to determine the research gap.
- Chapter 3: System Methodology/Approach describes the Agile development methodology to be used to develop the prototype. It describes the system architecture and gives Use Case and Activity diagrams to describe the logic of the system.
- Chapter 4: System Design: It gives a technical detail of the system components. It contains the system block diagram and the description of the logic of algorithm to clean up the data, mine the text and construct the decision tree.
- Chapter 5: System Implementation records the actual implementation of the prototype. It elaborates on hardware and software installation, environment settings and step-to-step guide on how the system functions backed by screen shots.
- Chapter 6: System Evaluation and Discussion report on the results of the intensive testing phase. It includes a real life case study of the campus service prioritization, a test of the soundness of the missing data and a usability test of the PSSUQ.
- Chapter 7: Conclusion and Recommendation describes the conclusions of the project, whether the objectives were met or not and provides recommendations on how to improve in the future such as emotion analysis and longitudinal data processing.

CHAPTER 2

Literature Review

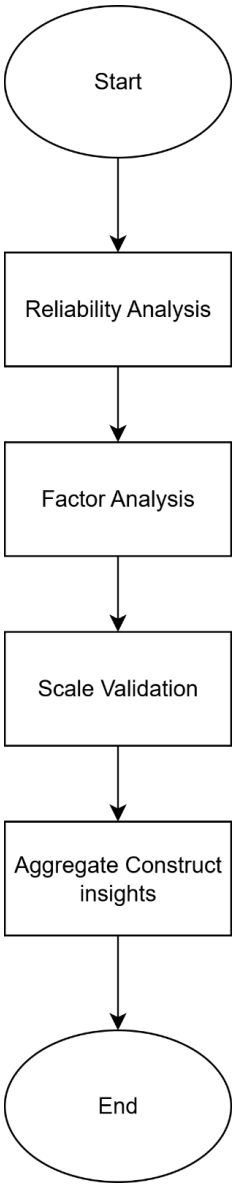
The chapter has provided a comprehensive review of the literature available that can be employed in the analysis of Likert scale survey data. It begins by critiquing the classic psychometric approaches that have dominated over decades and with their merits and flaws in construct validation, and in exposing complex non-linear interrelationships. The discussion then moves to the field of data mining and machine learning, namely the use of decision tree algorithms and the importance of data preprocessing that is so critical and is frequently ignored. Moreover, it discusses the recent development of incorporating Large Language Models (LLMs) into the data analysis processes in order to increase interpretability. Lastly, the critical analysis of the available solutions is provided and finally a comparative summary of the available solutions is provided that outlines the research gaps that will be addressed in this project.

2.1 Traditional Psychometric Approaches

The use of Likert scales in the measurement of attitudes and perceptions has long been the staple of measurement of psychology, education, and the social sciences [3] [17]. The typical analysis model of such data is highly based on Classical Test Theory (CTT). Most of the research investigations depend on known statistical software packages. One of the most common statistical packages was SPSS (Statistical Package for the Social Sciences) which aided them in validating the properties of their survey tools.

Reliability Analysis is often the main emphasis of traditional analysis. Methods like Cronbach's Alpha are universally used to determine internal consistency of a scale, so that various items that are supposed to measure the same construct give similar items. To supplement this is Factor Analysis, namely Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA), which is used to cluster together individual survey items into latent constructs (e.g. grouping questions Q1 through Q5 into a single factor Customer Satisfaction).

Although the methods are very effective in validating the survey instrument and testing special hypotheses concerning group variations, they are much more aggregate in nature. Traditional statistics are created to generalize data by means and standard deviations, but not elucidate the non-linear decision rules, which are unique to a particular outcome [4]. An example is that although a correlation analysis may tell you that there is a relationship between, say, "Service Quality" and "Loyalty" it does not usually give you granular, action-oriented, easy-to-understand, IF-THEN logic (e.g., "Under a rating of 3 of Service Quality Loyalty will reduce to zero") which can be easily comprehended and acted upon by a non-statistician.



.

Figure 2.1: Traditional Psychometric Analysis Workflow

2.2 Machine Learning in Survey Analysis

Recent studies have been attracted to the use of Machine Learning (ML) methods on survey data to overcome the shortcomings of the aggregate statistics and linear models. In contrast to the conventional linear regression, the ML algorithms can model high-dimensional, non-linear relationships between variables, which makes them highly appropriate in order to account the intricacy of human attitudes and behaviors.

2.2.1 Tree-Based Models and Interpretability

Decision Trees (like C4.5 and CART) are often mentioned in the literature as one of the most appropriate ML algorithms to use in the analysis of the survey because they are inherently interpretable [6] [14]. A decision tree is a representation of human logic that can be used to divide data into pieces according to a set of logical rules (e.g., Is Age > 25?). Such transparency is vital in such areas as social science, where it is often more important why a prediction was made than the prediction itself.

The advantages of the decision trees are numerous in that they can handle both categorical and numerical data, less data is needed to create them compared to neural networks, and they are presented in the form of visual models that are easy to understand by non-technical stakeholders. Nonetheless, there are flaws of standard decision tree implementations. They are likely to overfit, which means that the model generates an excessively complex tree that fits random noise in the training data instead of the underlying patterns [8]. In this project, I directly deal with this challenge by taking the Pessimistic Post-Pruning technique which simplifies the tree in an algorithmic manner to make the tree generalizable and readable to humans.

C4.5 algorithm, invented by J. R. Quinlan is a hallmark decision tree building algorithm [4]. It builds up a tree in recursive division of the data aided by the feature that yields the most information. The gist of this is the Entropy and Information Gain [19] [20] calculation. Entropy, which is a value of impurity in a set of examples (S), is obtained as:

$$Entropy(S) = -\Sigma p_{\mathrm{i}} * log_2(p_{\mathrm{i}})$$

Where:

p_i = the proportion of examples in class i

Based on entropy, the algorithm computes a quantity called Information Gain, which is the expected decrease in entropy of the split of the data across an attribute (A). It is defined as:

$$Gain(S, A) = Entropy(S) - \Sigma[(|S_v|/|S|) * Entropy(S_v)]$$

Where:

S = the original set of examples

A = the attribute to split on

v = each unique value of attribute A

S_v = the subset of examples where A has value v

Nonetheless, the Information Gain contains a bias and it is more inclined to support attributes that have numerous diverse values [4]. To eliminate this, C4.5 creates a normalization parameter known as Split Information (or Intrinsic Information) that is used to estimate the amount of potential information that can be produced by splitting the dataset itself:

$$SplitInfo(S, A) = -\Sigma[(|S_v|/|S|) * log_2(|S_v|/|S|)]$$

Lastly, the Gain Ratio is used as the final splitting criterion in C4.5. Dividing the Information Gain by the Split Information helps to reward attributes that have many values, making the tree more balanced and robust:

$$GainRatio(S, A) = Gain(S, A)/SplitInfo(S, A)$$

Each node uses the attribute with the largest Gain Ratio to be split on.

2.2.2 The Critical Role of Data Preprocessing

One such gap that has been found in scholarly literature is that there is no emphasis on Data Preprocessing in practice-based studies of ML. The majority of studies make use of pre-cleaned, toy, datasets (such as the notorious Iris or Titanic datasets) which are not representative of the reality of primary data collection. Raw survey exports are infamously noise, which includes timestamps, email addresses, haphazard column names, and mixed data types in practice.

A powerful Data Cleaning strategy is vital to effective analysis. It is proposed to use automated removal of high-cardinality identifier columns (Columns with many similar values, such as User IDs) because these do not provide predictive value and can confuse the algorithms [5] [12] [13]. Moreover, survey data are prone to the curse of dimensionality, with many more questions than are needed to make the prediction.
Selection of features is thus crucial. The Chi-Squared (χ^2) Test of Independence, as well as other approaches, are especially suitable with categorical survey data. Statistical dependence of each feature can be measured to filter the irrelevant questions, so before modelling, the irrelevant questions can be filtered [9]. This preprocessing step not only enhances the accuracy of the model, but also greatly decreases the complexity of the final tree, making it easier to understand. The following formula is used to compute Chi-Squared statistic [21]:

$$\chi^2 = \Sigma[(O_\mathrm{i} - E_\mathrm{i})^2 / E_\mathrm{i}]$$

Where:

O_i = the Observed Frequency in a cell

E_i = the Expected Frequency in a cell

Σ = the sum over all cells in the table

A p-value is then determined using the resulting χ^2 value. A low p-value (usually less than 0.05) suggests a statistically significant association between the feature and the target and allows the null hypothesis of independence to be rejected. By calculating this for all features, a system can rank them by importance and filter out those that are likely irrelevant.

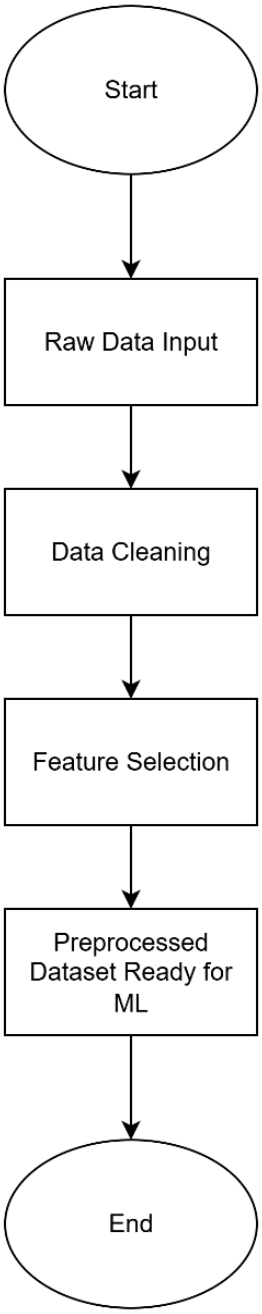


Figure 2.2: General Machine Learning Preprocessing Pipeline

2.2.3 Natural Language Processing in Survey Analysis

Although Likert scales offer more structured data which is quantitative, recent surveys often feature open-ended questions to allow qualitative responses. Historically, regarding this text, the researchers had to conduct thematic coding using their hands that is a time-consuming task and can be subjective. Over the past years, there was a growing use of Natural Language Processing (NLP) techniques in order to make the meaning of these unstructured responses automatic [22].

Text normalization is a basic process of NLP to analyze a survey. This includes methods like lowercasing text, eliminating punctuation and ignoring stop words (common, non-informative words such as the, is, and). Moreover, it is common that lemmatization is used to shorten words to the base form in the dictionary (e.g., replacing cards and carding with the word card). This is done to make sure that frequency algorithms get counted on underlying concepts as opposed to simple spelling differences [23].

Basic NLP uses are usually based on Term Frequency (TF) analysis, after normalization. Researchers using this tool can easily determine the themes prevailing in a huge amount of text by calculating the frequency of certain keywords or bigrams. This is normally represented in Word Clouds or frequency tables. The literature, however, points out that there are a huge number of gaps in mixed methods automated systems. Most of the tools available silo the quantitative Likert analysis and the qualitative text mining processes. A system that can synthesize the two streams of data by means of text themes to create a context of statistical rules is a big step towards holistic analysis of surveys.

2.3 Emerging Trends: Generative AI for Data Interpretation

The black-box models can also be seen to have ethical and practical issues due to their obfuscation, especially when making high-stakes decisions [11], [15], [16]. The latest and radical innovation of data analysis is the use of Generative AI and Large Language Models (LLMs). Whereas Explainable AI (XAI) has been dedicated to the trans parentification of the inner workings of algorithms (i.e., visualizing the decision tree), Generative AI aims at making the insights obtained by such models accessible.

Recent publications show that LLMs can serve as semantic reasoning engines. Systems can generate coherent and narrative explanations by feeding structured data outputs like the logic rules of a C4.5 tree or statistical summary table into an LLM [10] [18]. This is able to fill in the gap between technical outputs and human interpretation of the underlying problem, transitioning the paradigm from mere "Data Visualization" to an entire story about the data, referred to as Data Storytelling. It can be used to generate automated executive summaries and strategic recommendations; this is much lost in more traditional statistical software.

2.4 Critical Analysis of Previous Works

An in-depth analysis of the available solutions and scholarly literature presents four main constraints that delay the extensive use of advanced, comprehensive analytics by scientists:

1. **Lack of Generalizability:** The majority of current ML scripts in the literature are hard-coded to particular data sets (an example is students retention variables opening and closing a study). They do not have an actively running data engineering pipeline to upload, auto clean and analyze any survey data with varying structures and data types, and column names.

2. **The Siloing of Quantitative and Qualitative Data (Mixed-Methods Gap):** In the common surveys, both quantitative Likert scale and qualitative open-ended textual feedbacks are often gathered. In the past, researchers have had to examine these in silos by alternative methodologies and software (e.g., statistical software with numbers and manual thematic code with text). The particular shortage of automated mechanisms to triangulate statistical predictive models and use Natural Language Processing (NLP) to create a coherent interpretation is evident.

3. **The "Black Box" vs. "Glass Box" Trade-off:** There are two extremes of predictive tools used in existing. They are either highly predictive but low interpretability (e.g. Neural Networks, Deep Learning) or highly interpretable and but demand high-level manual work and statistical expertise (e.g. classic regressions in SPSS). The prompt need is a set of tools that brings together the high interpretability of Decision Trees (glass box) and automated narrative explanations.

4. **User Accessibility Barriers:** Higher level tools In general, high level analysis software is often highly technical (e.g. Python or R programming) or commercially expensive (e.g. SPSS, SAS). This puts a high barrier to entry to non-technical researchers and domain experts who would love to use the modern data science methods and have access to the computational power or the code know-how to implement it.

2.5 Comparative Analysis of the Proposed Solution

Table 2.1 summarizes the features of the proposed system against existing standard solutions in the research space. It highlights the novel contributions of this project in addressing the identified methodological and usability gaps.

Feature / Capability	**SPSS / Excel (Traditional)**	**Custom Python Scripts (Academic Standard)**	**Proposed System (Web-Based Tool)**
User Interface	Complex GUI	None (Command Line / Code)	**Simple Web GUI**
Automated Data Cleaning	☐ No (Manual)	☐ No (Hardcoded)	**☑ Yes (Heuristic Pipeline)**
Exploratory Data Analysis	☑ Yes (Manual triggers)	☐ No (Manual Code)	**☑ Yes (Automated)**
Statistical Feature Selection	☐ No (Manual)	☑ Yes (Manual Code)	**☑ Yes (Automated Chi-Squared)**
Qualitative Text Mining	☐ No	☐ No (Requires separate NLP script)	**☑ Yes (Automated Lemmatization)**
Model Interpretability	☐ Low (Aggregate Tables)	☑ High (Visual Tree)	**☑ High (Pruned Tree + Rules)**
Automated Narrative Insight	☐ No	☐ No	**☑ Yes (LLM Synthesis)**
Report Generation	☐ Static Export	☐ No	**☑ Dynamic PDF Report**
Cost / Accessibility	High Cost / High Skill	Free / High Skill Barrier	**Free / Low Skill Barrier**

Table 2.1: Comparative Analysis of Survey Analysis Solutions

2.6 Summary of Literature Review

The literature proves the effectiveness of decision trees to study surveys but states a strong lack of usability, generalizability and automated interpretation. Existing tools are either too manual, are treated as the black box that cannot be described, or do not even in asking quantitative and qualitative data to be considered as two different entities. The project fills these gaps by encircling a heavy, post-pruned C4.5 algorithm, in a web-accessible interface. It robots the resource intensive, critical stages of preprocessing and feature selection, incorporates NLP in text mining, and is the only application to use Generative AI to deliver easy to understand synthesized narrative insights.

CHAPTER 3

System Methodology/Approach

This chapter outlines the general approach and conceptual designs for the development of the AI-Powered Likert Scale Analyzer. It describes the lifecycles of the development and shows the system architecture, user interactions and dynamic operations of the system.

3.1 Development Methodology

This project will be undertaken following the Agile Development Methodology. Agile is an iterative and incremental development approach which places emphasis on flexibility, continuous testing and rapid delivery of working software increments. The short timeframe (2 semesters - 7 weeks for FYP 1 and 14 weeks for FYP 2) and the various technologies involved in this project (C4.5 algorithmic model, Streamlit framework and LLM APIs) do not easily lend itself to a Waterfall-based approach.

This Agile cycle for this project involves short sprints for each module of the system (e.g., Pre-processing Sprint, Algorithm Sprint, UI Sprint). This ensures frequent feedback and feature development based on the latest test results, and a stable and functional prototype at the end of each development iteration.

The main stages in this approach to this project are the following:

- Requirements: The initial high-level requirements, as outlined in Chapter 1, make up the project backlog. During the planning for each sprint, some of the requirements are chosen to be implemented.
- Design: In this stage, the technical details of the features to build in the current sprint are planned. This may involve designing the user interface (UI) elements in Streamlit, determining the signatures and logic of the underlying functions, and how data will be passed between components.
- Development: Here, the features are coded in Python, using the identified libraries.

- Testing: The new components are subjected to testing. These tests checks for the functionality of individual functions and the interaction between different components of the system.
- Deployment: In this project, "deployment" is running the application on a local web server using the Streamlit command to make the functional prototype available for demonstrations and user acceptance testing.
- Review: Each sprint concludes with a review of the sprint's stated goals. These reviews also help to update the project backlog and update the next set of requirements.

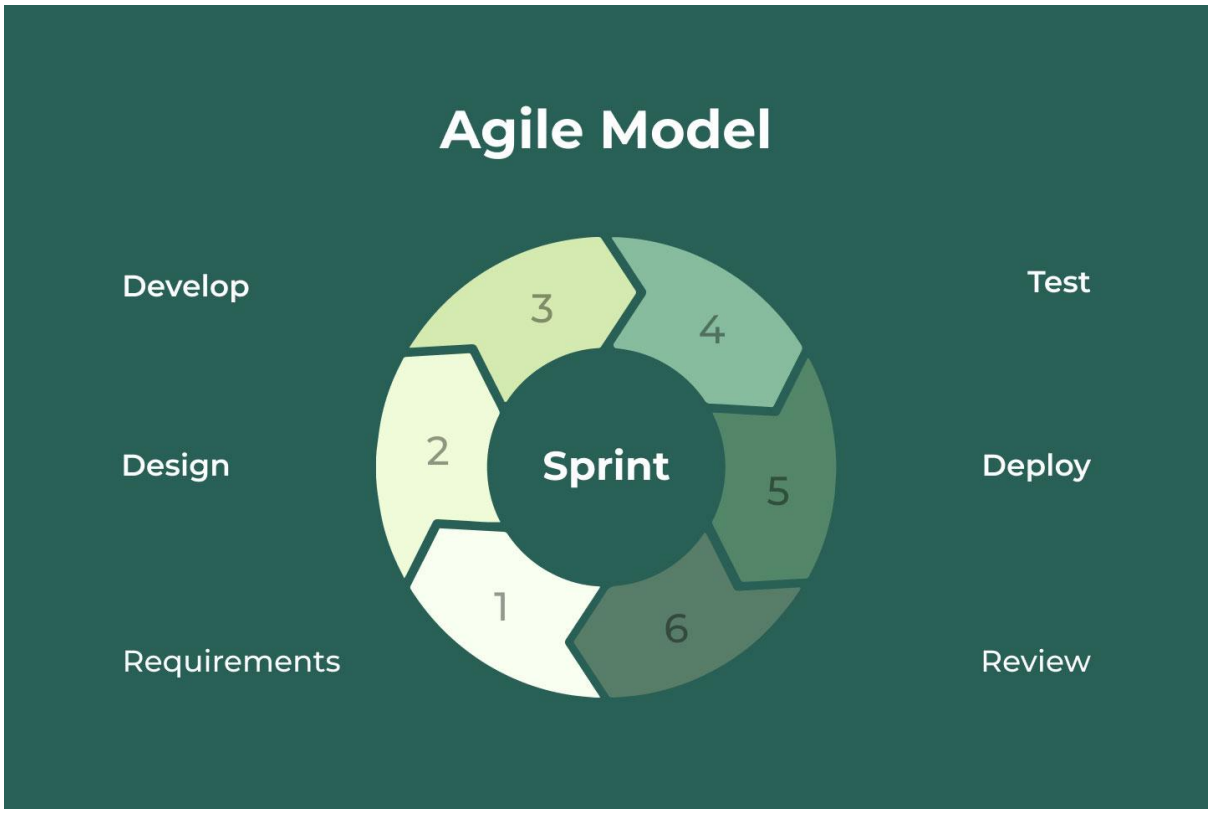


Figure 3.1: Agile Development Lifecycle

3.2 System Design Diagram/Equation

Our system uses a mixture of statistical and machine learning models for data analysis. The mathematical model used in the feature selection is the Chi-Squared (χ^2) Test of Independence which defines the independence of two categorical variables. It is defined as:

$$\chi^2 = \Sigma[(O_i - E_i)^2/E_i]$$

Where:

O_i = the Observed Frequency in a cell

E_i = the Expected Frequency in a cell

Σ = the sum over all cells in the table

The predictive modeling relies on the C4.5 algorithm's **Gain Ratio**, which utilizes **Entropy** to measure dataset impurity before and after a potential split:

$$Entropy(S) = -\Sigma p_i * log_2(p_i)$$

Where:

p_i = the proportion of examples in class i

$$GainRatio(S, A) = Gain(S, A)/SplitInfo(S, A)$$

The attribute that has highest Gain Ratio is selected as the split attribute.

The mathematical principles represented by the equations are the basis for the system architecture and decision-making.

3.2.1 System Architecture Diagram

The system is designed to adopt a modular Three-Tier Architecture. This is to clearly separate the concerns of the user interface from the complex logic and mathematical processes that are used in the app. This approach improves code maintainability and scalability of the system.

The system is built up of three layers:

1. The presentation layer (frontend) is built with Streamlit and is the client-side element. It offers user interface components for file import (CSV), dynamic rendering of widgets for variable selection and presentation (rendering of decision trees, EDA tables and AI narrative).

2. Application Controller Layer (Middleware): The code for this layer is in app.py and is the orchestrator. It is responsible for Session State Management, so that data is maintained between different points in the workflow that the user engages with the app. It communicates with the Computational Engine from the Presentation Layer.

3. Computational Engine (Backend): It is responsible for running the business logic. It is comprised of a Data Processing Module (Pandas/NumPy - heuristic cleaning), a Machine Learning Module (Scikit-Learn and C4.5 logic) and an External Integration Module (Google Gemini API).

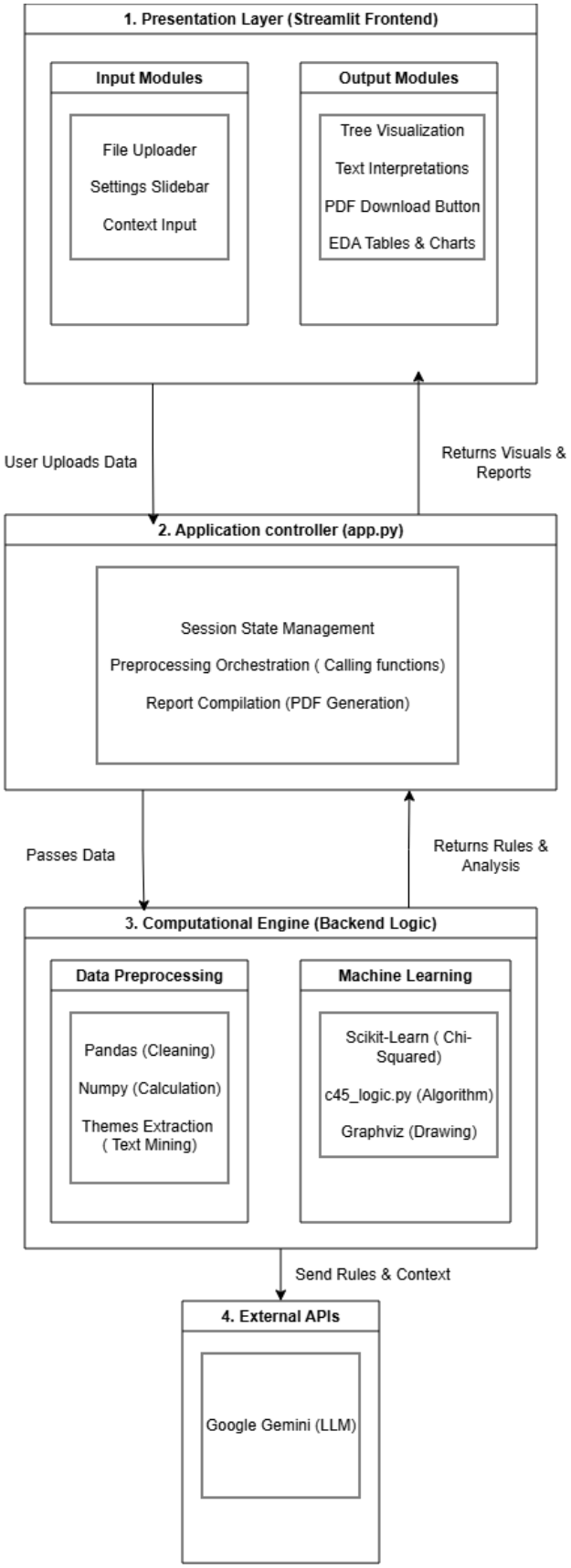
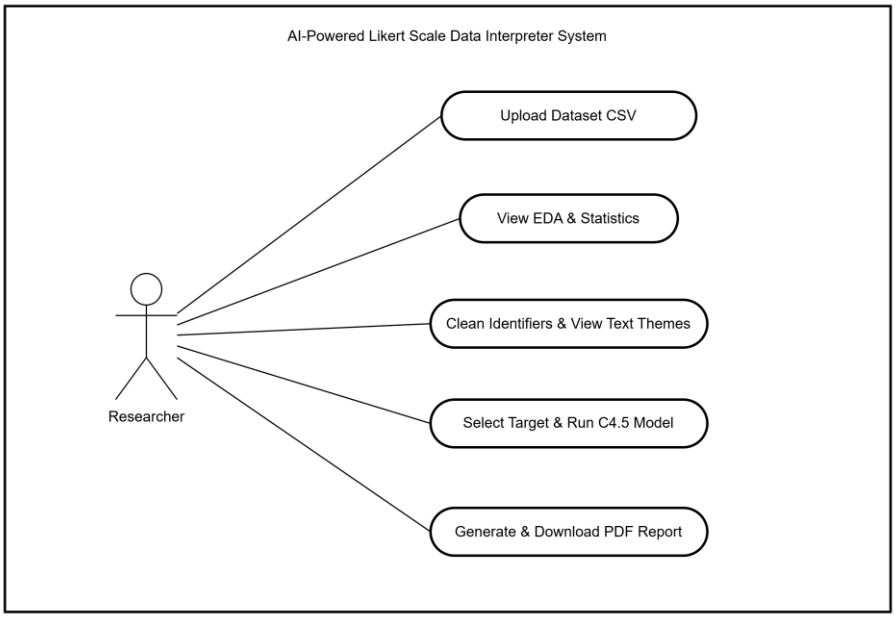


Figure 3.2: System Architecture Diagram

3.2.2 Use Case Diagram and Description

The Use Case diagram describes the main actor which can be a non-technical researcher and his/her interaction with the system.



.

Figure 3.3: Use Case Diagram

Description of Use Cases:

- **Upload CSV/Excel:** The user starts the process by importing survey data.
- **View EDA & Statistics:** The user examines descriptive statistic & data distribution table/chart for data integrity.
- **Clean Identifiers/Text Mining:** The user engages with the preprocessing module to approve the deletion of system-identified columns that the system considers as "noise" and views the themes generated by the text mining algorithms.
- **Select Target & Run C4.5:** The user defines the model (target variable and number of features) and invokes the main ML algorithms.
- **Generate & Download PDF Report:** The user concludes the process by generating a report that contains the streamlined tree, rules and AI-generated descriptions and uploads a PDF.

3.2.3 Activity Diagram

The Activity Diagram is a dynamic representation of the sequence of activities of the system, from the starting point (user input) to the end (user output), showing the simultaneous quantitative and qualitative data processing.

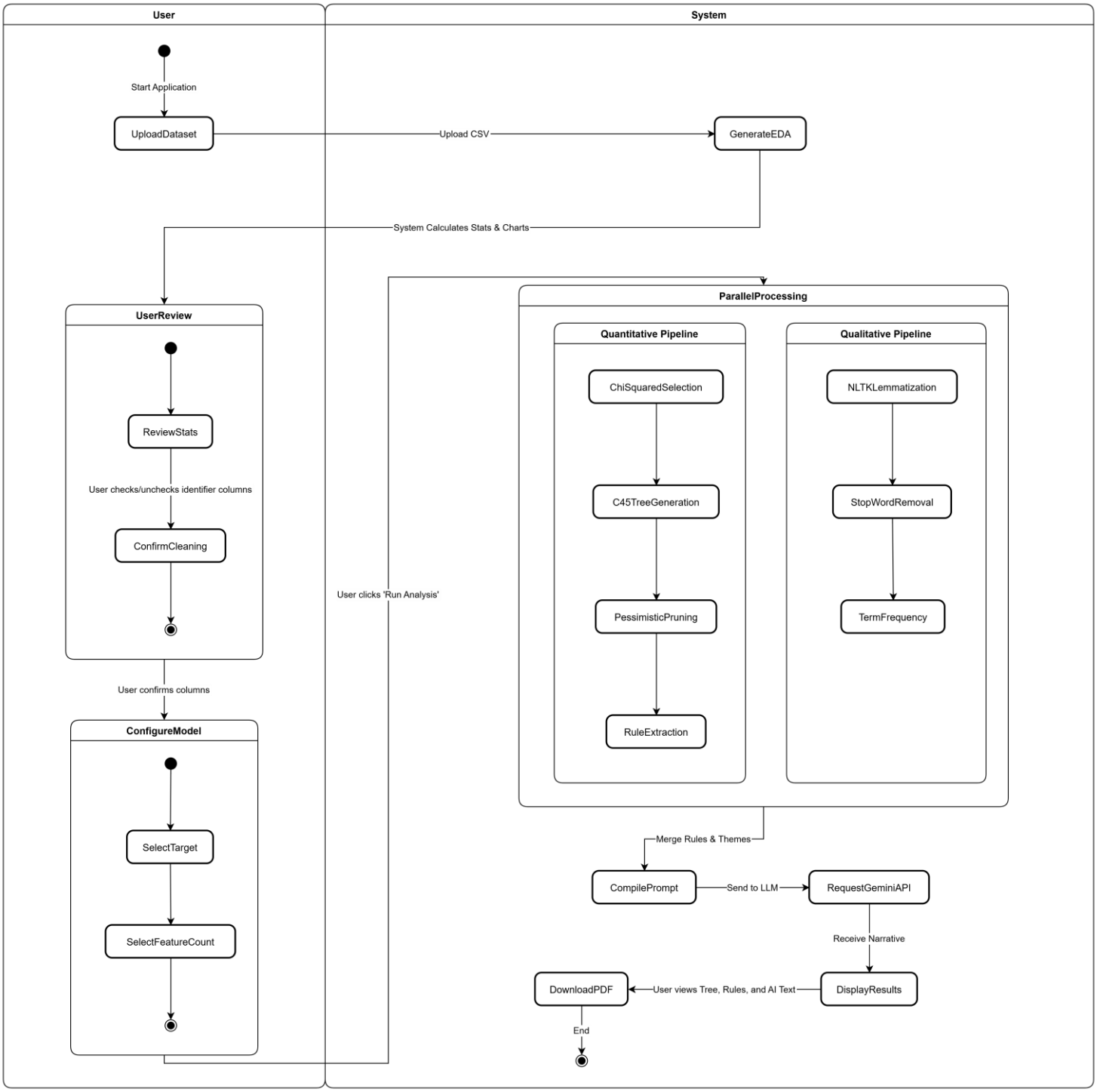


Figure 3.4: System Activity Diagram

3.3 Project Timeline

The project development is divided into two phases: FYP 1 (7 Weeks), involves the project initiation and the development of the initial prototype, while the FYP 2 Phase (14 Weeks) involves the implementation of the final features, testing and system delivery.

Task ID	Task Description	W1	W2	W3	W4	W5	W6	W7
1.0	**Project Inception & Research**							
1.1	Review Feedback from the Proposal Presentation	■						
1.2	Familiarize with Likert Data	■						
1.3	Preliminary Literature Review	■	■					
2.0	**Core System Development**							
2.1	Algorithm Development: C4.5 Entropy Logic			■				
2.2	Algorithm Development: Recursive Splitting				■			
2.3	Preprocessing Module: Data Cleaning Heuristics				■	■		
2.4	Interface Design: Basic Streamlit Layout					■		
3.0	**Documentation & Presentation**							
3.1	Drafting Interim Report						■	
3.2	Finalizing Interim Report							■
3.3	Preparation for Presentation Slides							■
3.4	**FYP 1 Final Presentation**							■

Table 3.1: FYP 1 Schedule (7 Weeks)

Task ID	Task Description	W1-2	W3-4	W5-6	W7-8	W9-10	W11-12	W13-14
4.0	**Advanced Feature Implementation**							
4.1	Integration of Chi-Squared Feature Selection	■						
4.2	Integration of Google Gemini LLM API		■					
4.3	Development of PDF Report Generation Module			■				
5.0	**System Refinement & Visualization**							
5.1	Enhancement of Graphviz Tree Visualization			■				
5.2	Implementation of Interactive EDA Charts				■			
6.0	**Testing & Validation**							
6.1	System Testing (Unit & Integration Testing)					■		
6.2	User Acceptance Testing (UAT) & Debugging					■		
7.0	**Final Documentation & Delivery**							
7.1	Drafting Final Report						■	
7.2	Final Report Proofreading & Formatting							■
7.3	Preparation for Final Presentation & Poster							■
7.4	**Final Project Submission & Presentation**							■

Table 3.2: FYP 2 Schedule (14 Weeks)

CHAPTER 4

System Design

In this chapter, we will explain in detail how the AI-Powered Likert Scale Analyzer was built. It breaks down the system into blocks, presents the requirements and provides insights into the internal workings of the Python code, including the integration of a C4.5 algorithm and the Google Gemini LLM. This chapter provides a roadmap for taking the survey data and turning it into storytelling.

4.1 System Block Diagram

The system is modular in design to manage and simplify the operations of the application. The System Block Diagram (Figure 4.1) shows the major modules and dataflow in the system. The system is broadly broken down into a Frontend (User Interface) block and a Backend (Processing Engine) block, which can be further broken down into Data Engineering, Machine Learning and Artificial Intelligence. The system block diagram is shown in Figure 4.1.

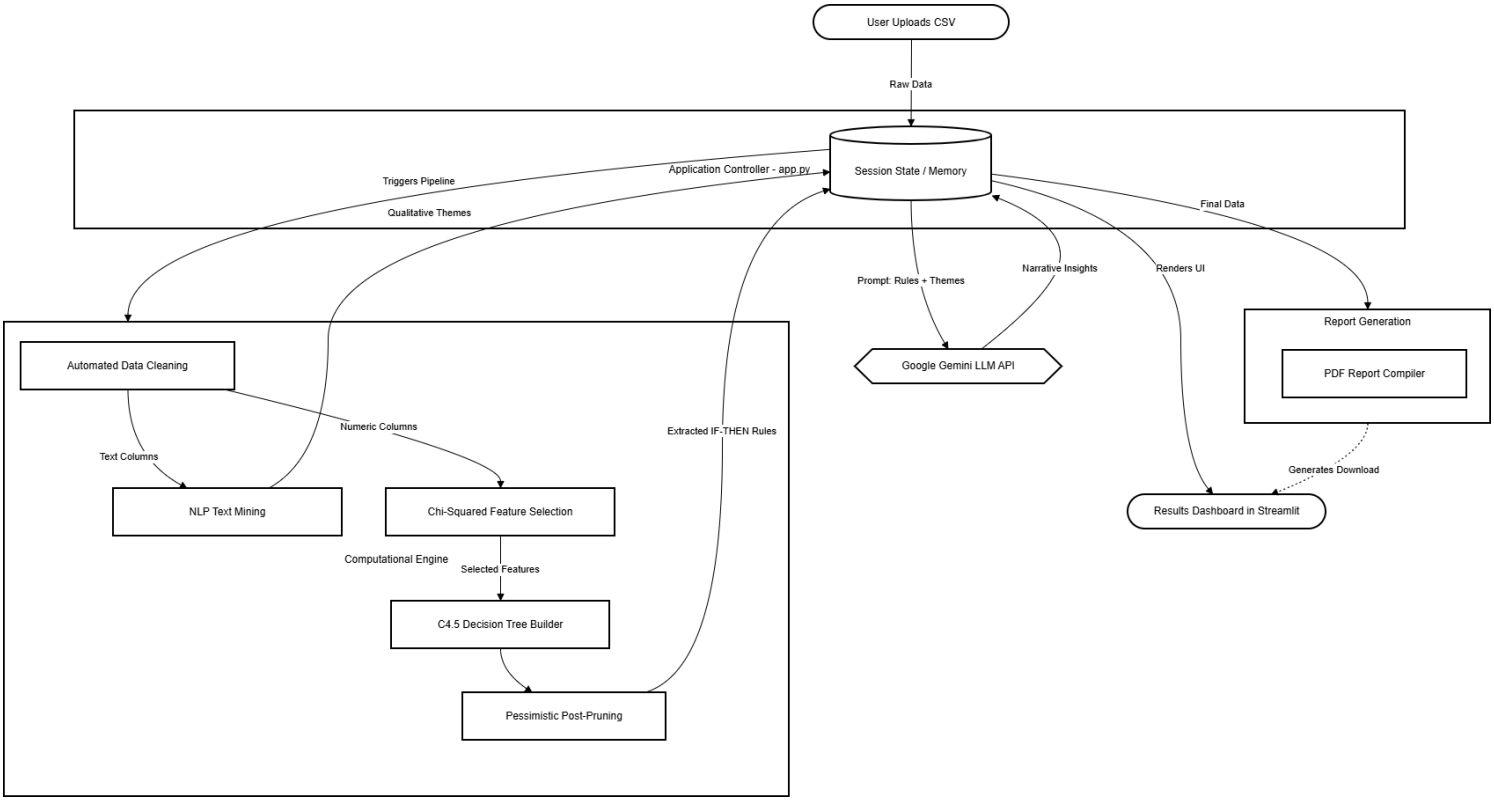


Figure 4.1: System Block Diagram

4.2 System Components Specifications

The design of the components of the system is determined by a set of functional and non-functional requirements, to ensure that the system can be used by non-technical researchers.

4.2.1 Functional Requirements

The software modules are built to perform the following core operations:

- **File Upload:** The system must allow users to upload their survey data files directly in .csv formats.
- **Data Cleaning:** The system must automatically scan the uploaded data to find columns that look like names, IDs, or emails, and suggest removing them.
- **Data Summary (EDA):** The system must calculate and display a statistical summary (mean, minimum, maximum, etc.) and draw basic charts (histograms, bar charts) so the user can understand their data before analyzing it.
- **Text Analysis:** If the survey contains long text answers (like interview comments), the system must extract the most common themes.
- **Feature Selection:** The system must use a statistical test (Chi-Squared) to automatically pick the most important survey questions and ignore the irrelevant ones.
- **Decision Tree Generation:** The system must build a decision tree model to show how the survey answers lead to the outcome, and display this as a clear picture.
- **AI Interpretation:** The system must send the rules from the decision tree and the text themes to an AI (Google Gemini) so it can write a plain-English summary of what the data means.
- **Report Download:** The system must combine the statistics, the tree picture, and the AI summary into a PDF file that the user can download.

4.2.2 Non-Functional Requirements

- **Usability:** The system should be user-friendly. It must have tabs (Upload, EDA, Model, Results) to let a user not familiar with code know what to do next.
- **Speed:** The analysis of the data and the generation of the tree and the AI text should be done within 1 minute.
- **Error Handling:** The system shouldn't crash if the user uploads a bad file or attempts to build a model without first uploading data.

4.3 Software Components Design (Algorithms and Logic)

This section explores the system's code (app.py and c45_logic.py) to understand how the system works.

4.3.1 The Heuristic Data Cleaning Component

In order to save the user time, the detect_identifier_columns(df) function was written to automatically detect "noisy" columns.

```
def detect_identifier_columns(df):
    id_cols = []
    for col in df.columns:
        if len(df) > 0 and df[col].nunique() / len(df) > 0.95: id_cols.append(col); continue
        if df[col].dtype == 'object' and df[col].str.contains(r'@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}', na=False).any(): id_cols.append(col); continue
        if re.search(r'id|name|email|phone|timestamp', col.lower()): id_cols.append(col)
    return list(set(id_cols))
```

Figure 4.2: Screenshot of detect_identifier_columns function

As shown in the code above, the logic checks every column against three rules:

1. **High Cardinality:** If over 95% of the rows in a column are unique (e.g., df[col].nunique() / len(df) > 0.95) then it is an ID column.
2. **Email Pattern:** It uses Regular Expressions (re) to check if the text looks like an email address.
3. **Keywords:** It checks if the column header contains words like 'id', 'name', or 'phone'.

4.3.2 The Text Mining (NLP) Component

The analysis of free-form survey questions is done by the analyze_text_responses(df) component.

```
def analyze_text_responses(df):
    text_columns_to_analyze = []
    potential_text_columns = df.select_dtypes(include=['object']).columns
    for col in potential_text_columns:
        if df[col].nunique() > 10 and df[col].dropna().astype(str).str.split().str.len().mean() > 2:
            text_columns_to_analyze.append(col)
    if not text_columns_to_analyze: return None, None, []
    all_words = []
    for col in text_columns_to_analyze:
        text_data = df[col].dropna().astype(str).tolist()
        for text in text_data:
            clean = re.sub(r'[^a-z\s]', '', text.lower())
            words = clean.split()
            meaningful_words = [lemmatizer.lemmatize(w) for w in words if w not in STOP_WORDS and len(w) > 2]
            all_words.extend(meaningful_words)
    if not all_words: return None, None, []
    word_counts = Counter(all_words)
    most_common = word_counts.most_common(15)
    summary_df = pd.DataFrame(most_common, columns=['Keyword/Theme', 'Frequency'])
    llm_summary_text = ", ".join([f"{word} ({count})" for word, count in most_common])
    return summary_df, llm_summary_text, text_columns_to_analyze
```

Figure 4.3: Screenshot of analyze_text_responses function

In the code, we first select columns containing text by filtering columns where the average response length is more than two words (to exclude plain "Yes/No" type of questions). It then uses the Natural Language Toolkit (NLTK) library to preprocess the text by:

1. Making text lower-case and removing punctuation.
2. Converting each word into its root word (eg "services" becomes "service") using WordNetLametizer.
3. Removing custom "stop words" (such as 'the', 'is', 'eg', 'ie') so only themes we are interested in are counted.

4.3.3 The C4.5 Decision Tree Engine

The underlying algorithm that makes predictions is a C4.5 algorithm in the c45_logic.py module. The aim of this component is to provide interpretability and reliably build a decision tree with survey data.

This algorithm recursively constructs the tree. At each step the engine will determine the best survey question (feature) to split the remaining data into the target classes. The choose_best_attribute method does this by computing the Entropy of the target, and the Information Gain Ratio of each attribute.

This is the mathematical implementation of this process (see Figure 4.4). The Information Gain is divided by the Split Information to punish attributes with a high number of values, avoiding bias of the tree towards unimportant columns.

```
base_entropy = compute_entropy(examples[target_attribute])

# Calculate Weighted Entropy and Split Information for the attribute
weighted_entropy = 0
split_info = 0

for value in unique_vals:
    subset = subset_not_missing[subset_not_missing[attribute] == value]
    weight = len(subset) / len(subset_not_missing)

    # Add to total weighted entropy
    weighted_entropy += weight * compute_entropy(subset[target_attribute])

    # Calculate intrinsic information (Split Info)
    if weight > 0:
        split_info -= weight * log2(weight)

info_gain = base_entropy - weighted_entropy

# Calculate Gain Ratio (Penalizes attributes with too many branches)
# Penalty multiplier applied for missing data handling
gain_ratio = (info_gain / split_info if split_info != 0 else 0) * (len(subset_not_missing) / len(examples))
```

Figure 4.4: Screenshot of choose_best_attribute function

In addition, the component also employs Pessimistic Post-Pruning. Decision trees are very susceptible to overfitting, particularly in our case in human survey data. Once the initial tree has been fully constructed, the tree is then pruned, beginning at the leaf nodes. It gives a pessimistic estimate of the error of each subtree. In the case that a reduced error can be obtained by replacing a subtree with a leaf node, the subtree will be pruned. This key design decision helps the resulting model to generalize to new data, while being comprehensible to human eyes.

4.4 System Components Interaction Operations

The different components of the code communicate via Streamlit's session_state. This is like the application's memory.

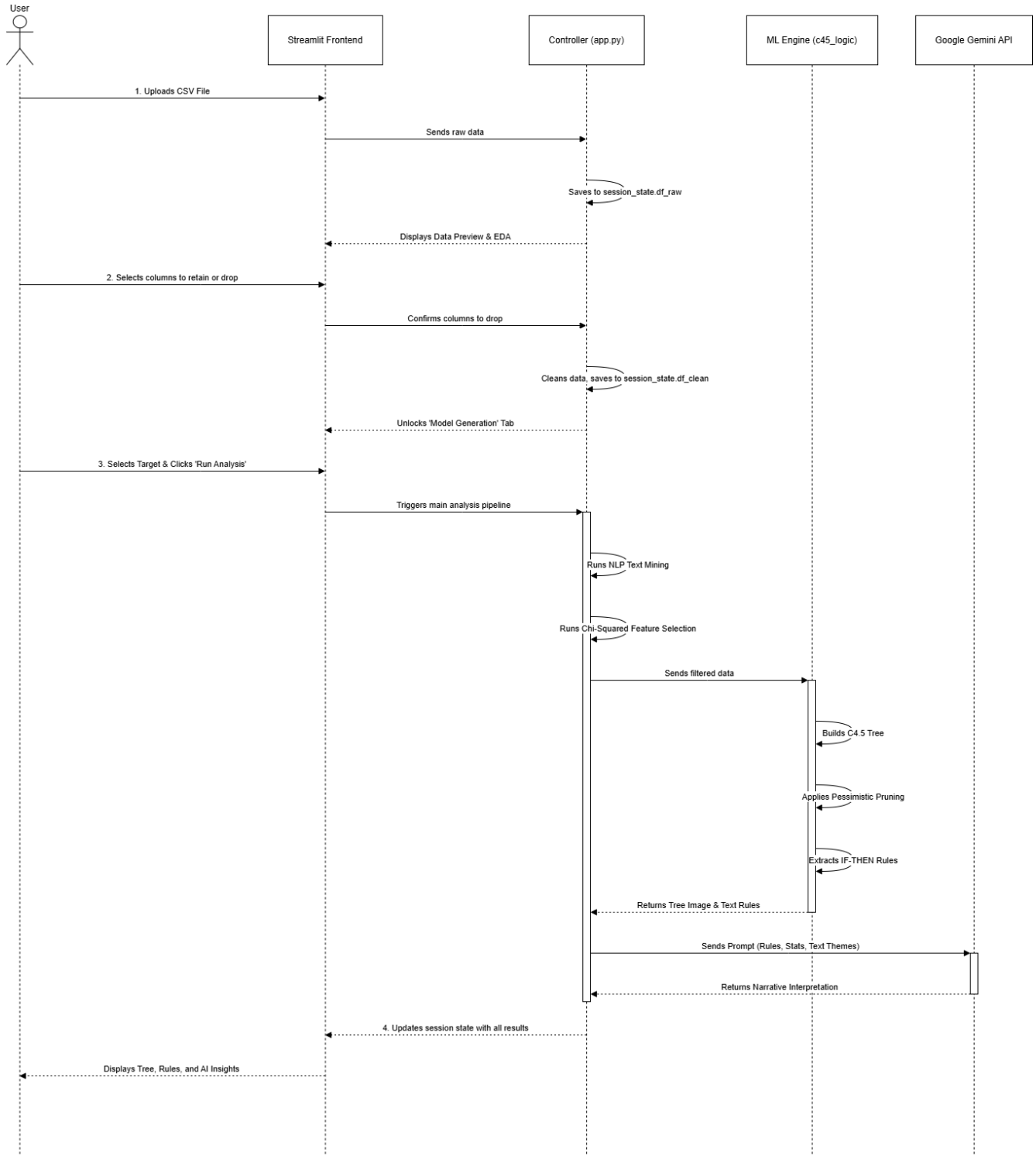


Figure 4.5: System Sequence Diagram

The Sequence Diagram (Figure 4.2) shows the interaction between the Frontend (what you click on), the Controller (app.py) and the Backend engines to give you the result.

As we can see in the diagram, to reduce the workload the model training and Google API call is only done when the user clicks the "Run Analysis" button. The session_state takes care of the fact that if the user changes the tab, the system doesn't need to be rerun.

CHAPTER 5

System Implementation

In this chapter, we describe how the AI-Powered Likert Scale Analyzer was implemented. It provides details of how the system architecture and approaches outlined in previous chapters are implemented as a software system prototype. It presents the hardware and software platforms used in development of the system, system configurations, a walk-through of the system's operation using screenshots of the interface and a critical discussion of issues that arose during the coding process.

5.1 Hardware Setup

The development, training and testing of the machine learning models, and the interactive web application were all from a local development machine. The system is an internet-based, client-server system, and therefore the hardware requirements of the end user are minimal.

The specifications of the development machine used to implement the prototype are as follows:

- **Device Model:** HP ProBook 440 G6
- **Processor:** Intel Core i7-8565U (8th Generation, Quad-Core)
- **Memory (RAM):** 16 GB DDR4 (Crucial for handling in-memory Pandas DataFrames and matrix operations during Chi-Squared testing).
- **Storage:** 512 GB Solid State Drive (SSD)
- **End-User Requirement:** Any standard device (PC, tablet, or smartphone) equipped with a modern web browser and internet access. All heavy computational tasks are handled by the host server.

5.2 Software Setup

The software ecosystem was developed using open-source, industry-standard tools for data science. The main programming language used was Python (Version 3.9+), due to its rich library ecosystem and ease of use. The code editor used was Visual Studio Code (VS Code).

The system's primary operation depends on the installation (using the pip package manager) of the following Python packages:

- streamlit: To create the interactive, tab-based Graphical User Interface (GUI).
- pandas and numpy: To ingest and manipulate data and work with arrays.
- scikit-learn: To use the SelectKBest and chi2 statistical feature selection algorithms.
- nltk (Natural Language Toolkit): To execute text mining, the WordNetLemmatizer to return the words' base dictionary form.
- graphviz and matplotlib: To visualise the decision tree and Exploratory Data Analysis (EDA) plots.
- google-generativeai: To connect with the Google Gemini Large Language Model (LLM).
- fpdf: To automatically generate analysis results into a downloadable PDF.

5.3 Setting and Configuration

For the system to work, some additional configurations were needed on the environment side, beyond the typical Python code:

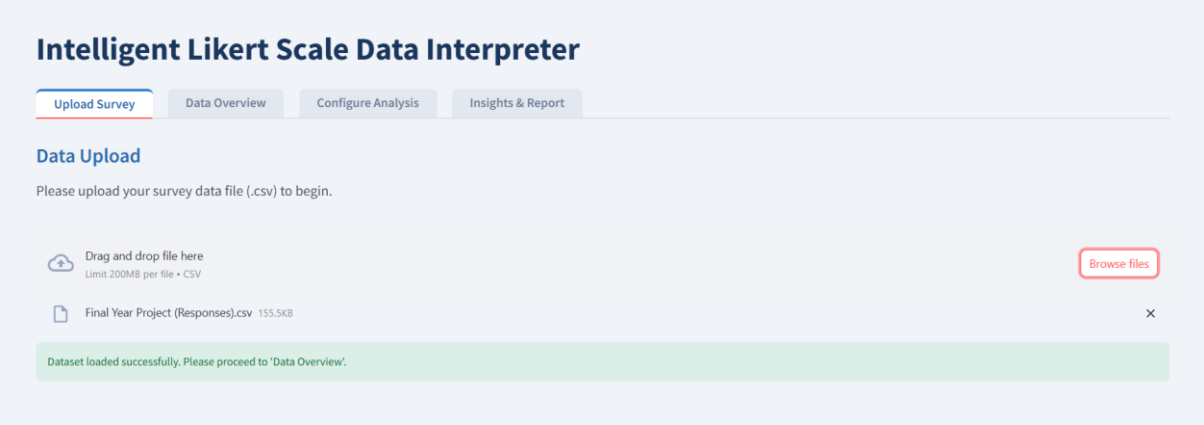
1. **API Key Authentication:** The system is integrated with the Generative AI model and needs to be authenticated. An API key was created from Google AI Studio. The system is set to transfer this key to the genai.configure() method to have the application access to generate responses if provided with a prompt to the gemini-1.5-flash model.
2. **Graphviz System Path Integration:** Although graphviz is a Python library, it is a wrapper for system-level C-binaries. For the Python script to be able to plot the decision trees, it was necessary to install the Graphviz software on the Windows 11 host system, and then to manually add the \bin directory to the System PATH variable.
3. **NLTK Corpora Download:** For the system to allow the Natural Language Processing (NLP) module to lemmatize, it requires linguistics databases. The system checks if the wordnet and omw-1.4 lexical databases are available on the system and automatically downloads them on the first startup through nltk.download().

5.4 System Operation

The system is designed with a linear, tab-based workflow that guides the non-technical researcher from raw data to final insights.

5.4.1 Step 1: Data Upload (Tab 1)

The workflow starts by uploading data in the "Upload Data" tab. It offers a file upload tool that accepts .csv files. The data is then processed by the backend load_data() method to a Pandas DataFrame and stored in the app's Session State to avoid reloading the data when navigating between tabs.



.

Figure 5.1: The Data Upload Interface

5.4.2 Step 2: Exploratory Data Analysis (Tab 2)

Once the data has been loaded the user changes tabs to the EDA. Our system would automatically categories the Likert data into numerical and categorical/text data. It automatically constructs a summary table of statistics (count, mean, standard deviation and missing values). It has also provided dynamic drop-down menu to dynamically generate histograms or bar charts to investigate data distributions

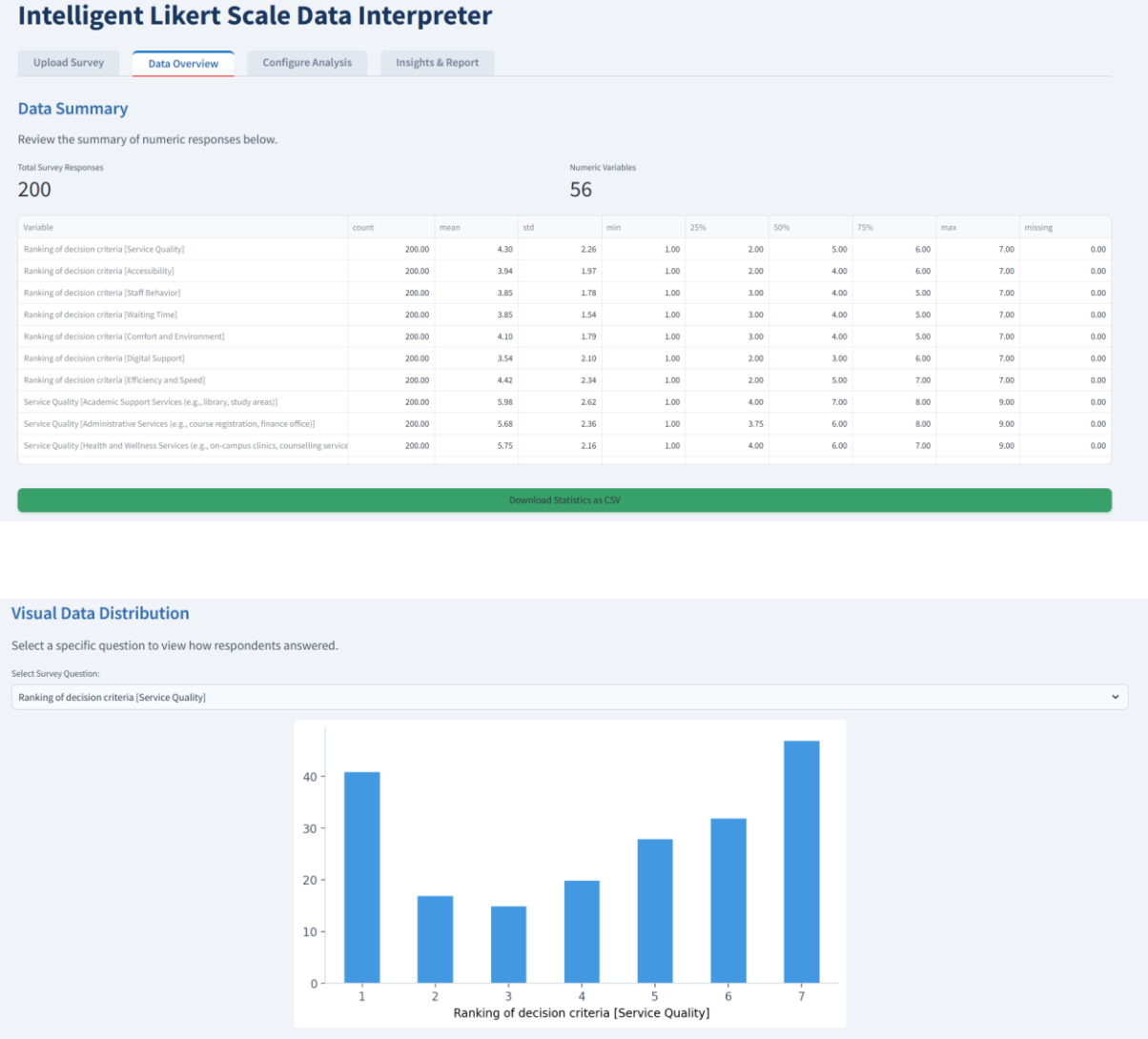


Figure 5.2: Automated Exploratory Data Analysis Dashboard

5.4.3 Step 3: Model Setup and Preprocessing (Tab 3)

The staple data engineering pipeline is found in this tab.

1. Data Cleaning: The system runs heuristics to discover high-cardinality columns that are ID variables (e.g., Respondent ID) and these are automatically selected in a multiselect box and can be deleted.

2. Text Analysis: It searches through free-form responses of an interview. When it is there it cleans the text, removes the stop-words, lemmatizes the words that are left and creates the highest frequencies of the themes.

3. Configuration: The user selects the Target Variable and enters the number of features that Chi-Squared algorithm is supposed to provide using a slider, and, optionally, adds a description of the survey.

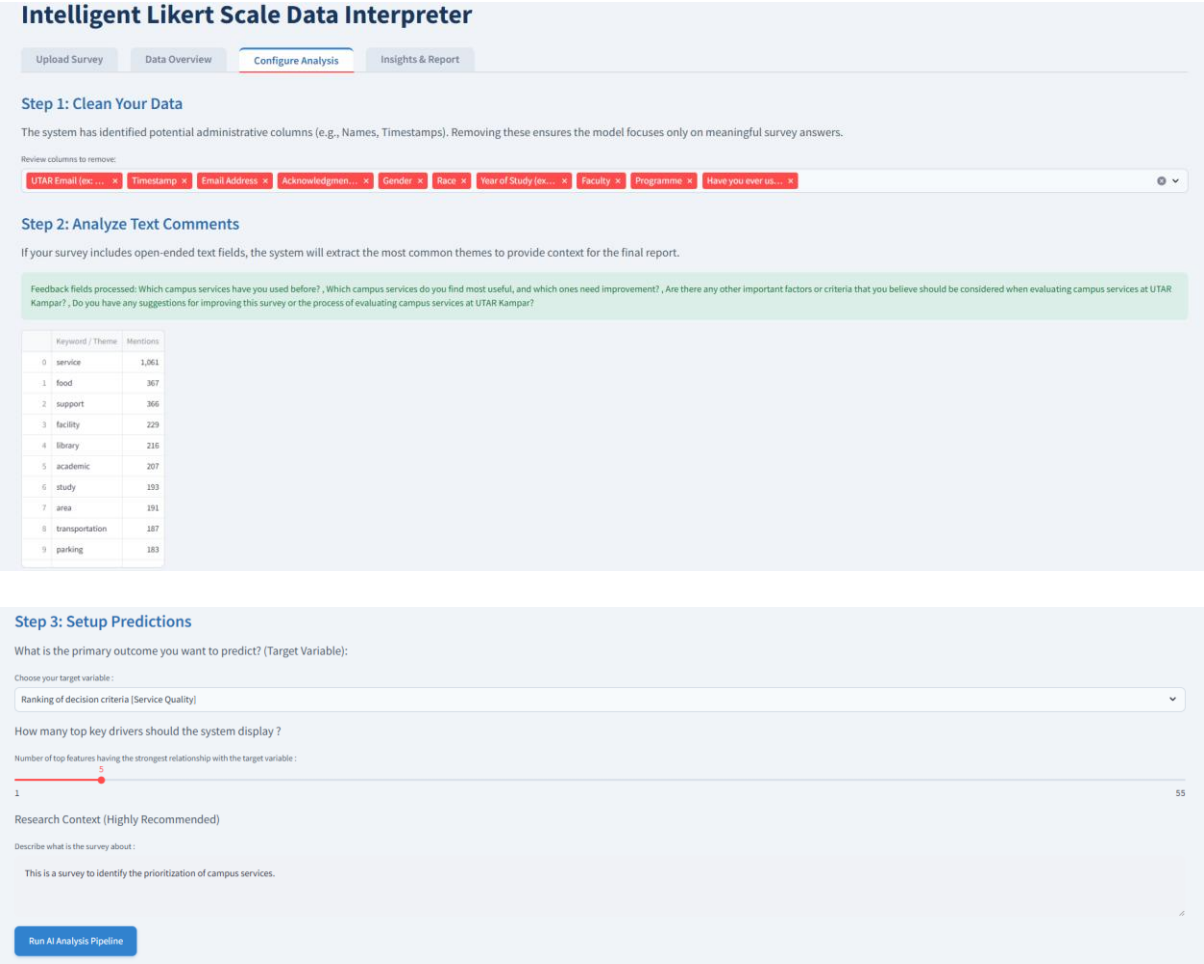


Figure 5.3: Model Configuration and Data Cleaning Interface

5.4.4 Step 4: Results and Interpretation (Tab 4)

To execute the Chi-Squared feature subset selection, construct and prune the C4.5 Decision Tree, and execute the LLM, the user clicks on Run AI Analysis Pipeline and this invokes the back-end to perform the analyses. The final tab shows the overall findings: the table of statistically chosen features, the decision tree diagram, the IF-THEN rules and the story told by AI. A big button allows one to download a report of these as a PDF.

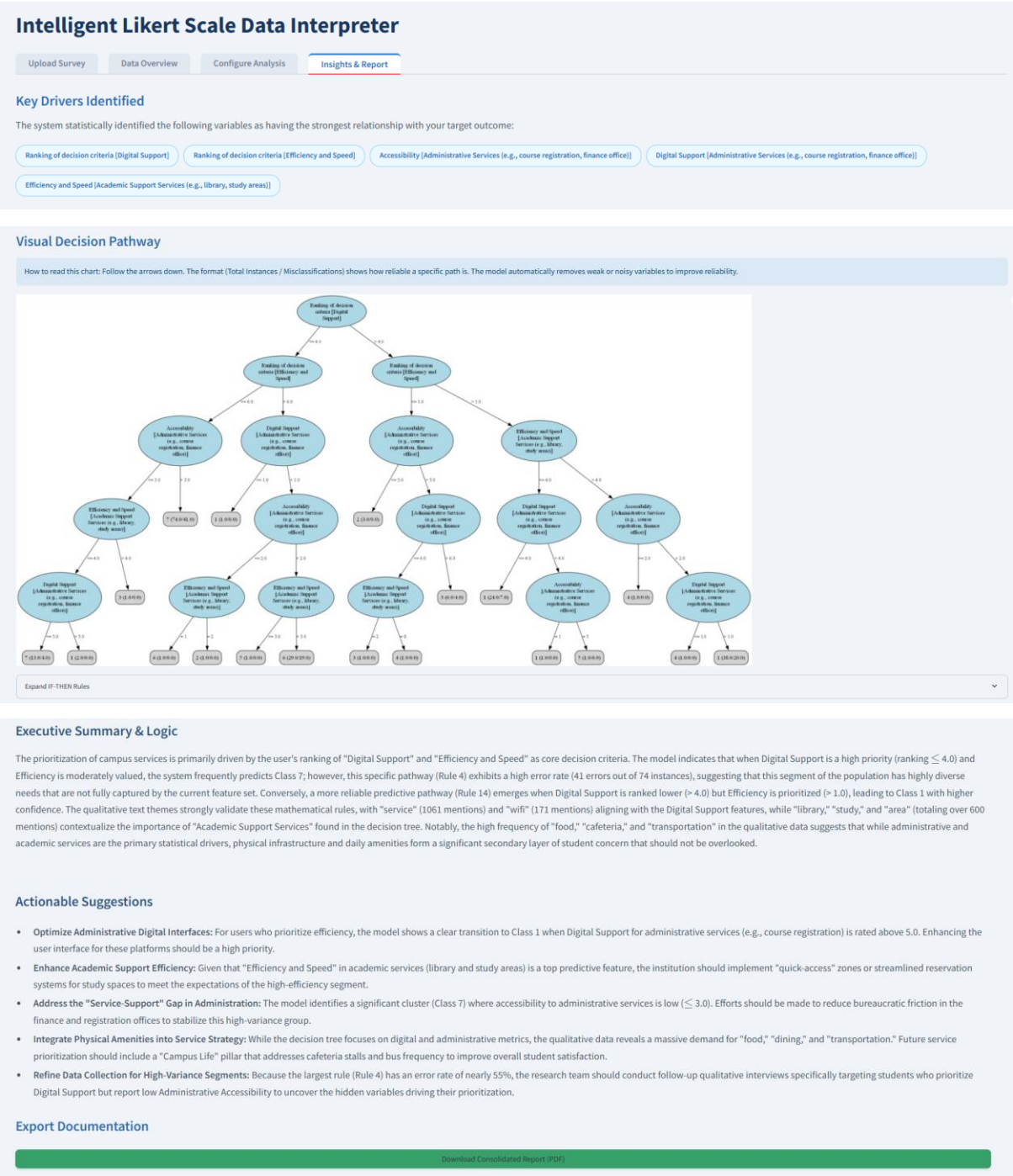


Figure 5.4: Results Dashboard and AI Interpretation

5.5 Implementation Issues and Challenges

During the software development lifecycle, several significant technical challenges were encountered and resolved.

1. Robustness of the Data Preprocessing Pipeline

Survey data is notoriously problematic to work with, with differing types of data, variable column headings, and extraneous columns within the files. The heuristics needed to correctly identify useful categorical data (e.g., "Gender") from non-useful indicators (e.g., "Student ID") required a large degree of experimentation. This challenge was addressed through a multi-pronged approach that checks the cardinality (uniqueness ratio >95) of the data as well as certain string patterns (using Regular Expressions) to ensure that useful data for analysis was not inadvertently lost in the automated data cleaning process.

2. Application State Management

One of the great challenges was the state of the web framework. Streamlit app uses a "rerun" model of execution where the Python script is re-executed from the beginning for each interaction (such as a checkbox click). If not carefully addressed, this resulted in the application "losing" uploaded datasets or "cleaning" the results of analysis, as the user navigated through different tabs - a highly unacceptable situation. This issue was addressed by implementing a solid st.session_state logic. This serves as a virtual storage in the server's memory, explicitly saving the raw dataset, the processed dataset, tree diagram, and AI-generated results at each step of the process.

3. AI Reliability and Hallucination Mitigation

The use of the Google Gemini API brought on the issue of AI "hallucination". Large Language Models are stochastic and may provide plausible but not necessarily correct explanations if not curbed. A rigorous Prompt Engineering approach was taken to prevent this. This comprised two technical safeguards. First, adjusting the temperature of the model to a very low value (0.2) to reduce the creative output and force the AI model to answer deterministically and based only on facts and second, explicitly limiting the AI context. By providing the AI with the exact, math-derived logical rules obtained from the C4.5 algorithm, and by limiting the AI's interpretation to the supplied statistics, the system guarantees that the narrative is academically rigorous, and based on the user's survey statistics.

5.6 Concluding Remark

The system implementation phase has translated the conceptual design into a user-friendly, highly interactive web system. Addressing issues of state management, unstructured text mining, and the API limitations, the prototype is able to seamlessly integrate data science pipelines into an interactive user interface, thereby achieving the project's core goal.

CHAPTER 6

System Evaluation and Discussion

This chapter provides an end-to-end assessment of the completed Intelligent Likert Scale Data Interpreter. The system is tested along two main lines: a real-world use case that uses a third-party survey data set to test the accuracy of the models and the quality of the interpretations, and a user-focused usability study with 40 participants to test the effectiveness of the user interface. Finally, anomalies detected during the testing process and subsequent system optimizations are presented.

6.1 System Testing and Performance Metrics

To test its technical and operational performance, the system was put through several stages of testing.

6.1.1 Testing Methodology

The system underwent three distinct levels of testing:

1. Unit Testing: Modules such as automatic removal of the non-deterministic identifiers and C4.5 entropy measure module were tested against manual calculations to ensure that mathematical principles are correctly applied.

2. Integration Testing: This tested the interaction between the Python system and external modules, namely the Google Gemini API and the Graphviz graph renderer.

3. System Testing: The system was tested end-to-end with raw survey data to determine if the process from insertion to PDF creation was error-free.

6.1.2 Performance Metrics

The system was judged based on:

- Mathematical Correctness: How the C4.5 output matches the expected outcome.
- Usability (PSSUQ): Assessed using the Post-Study System Usability Questionnaire; the lower the average score (out of 1-7), the better.
- Semantic Accuracy: Semantic correctness of the AI's integration of statistics and domain knowledge.

6.2 Testing Setup and Results

6.2.1 Real-World Case Study: Expert Validation

To test the system's capacity to manage the implementation of research, a data set was collected by a user on institutional preference for services. This data set had 200 rows and 70 columns.

Result: The system was able to determine key satisfaction drivers. The user was very satisfied after examining the AI-generated "Executive Summary" as it matched the relationships between human satisfaction behaviours the user had discovered and reported in the report. This demonstrates the system offers expert insights in minutes.

6.2.2 Evaluation of Missing Data Handling (Robustness Test)

An important experiment was undertaken to test the system's ability to handle missing data. Traditional Machine Learning models cannot deal with "blank" data and halt when encountering incomplete data. Hence, Imputation, which is filling the missing values with "fake" data (e.g. average, or most frequent value - mode) to make the dataset complete so that the system can run, is a common practice. To test if this practice is safe in analyzing surveys, a predictor variable in the dataset was artificially changed to have 80% missing values (a question that was not answered by most respondents).

- **Scenario A: Imputation (Filling with Mode)**
    - The Process: The 80% missing values were replaced with the "Mode" (the most frequent response).
    - The Result: The algorithm considered this column to be complete data. With 80% of the data the same (the Mode), the algorithm detected a very strong pattern. It chose this variable as the first tree splitter.
    - The Problem: This result is misleading. The system found a pattern in the artificial opinion (filler) questions, not among the real opinions. It would fool the user into believing this question is very important, even though the majority of respondents didn't answer it.
- **Scenario B: Native C4.5 Handling (Proposed Solution)**
    - The Process: The C4.5 system does not impute. The system's C4.5 implementation determines the "Information Gain" (predictive power) from the existing data.
    - The Logic: The system gives a mathematical penalty. If 80% of the data is not present, the calculated gain is multiplied by 0.2 (20% reliability).
    - The Result: This penalty significantly reduced the algorithm's score for this feature. The algorithm properly tagged the feature as "unreliable" and did not use it in the tree, preferring to skip it and instead use other, complete features.

Conclusion: The native handling (Scenario B) was much more reliable It doesn't fool the user into thinking something that's not there. Now the system behaves appropriately if the data is missing.

6.2.3 Impact of Contextual Input on AI Interpretation

The optional "Research Context" is a key aspect of the prototype design. A semantic accuracy test was performed to assess the effectiveness. The LLM interpretation engine was used to interpret the same set of extracted C4.5 rules. First with the context field empty, and then with the context.

Scenario A: Interpretation Without Context

Without the context, the AI simply gave a mathematical description of the rules. It indicated that the significant splitter was the biggest, but it could not offer any clarification, that is, Ranking of decision criteria [Digital Support]. As an example, regarding Rule 19 (38.0/20.0) the AI said that the pathway was with Class 1 having a 52.6% error rate. It viewed the large error rate as being a model issue and advised just to increase the size of the dataset.

The AI generated domain reasoning, and not a statistical storytelling which was created with the domain context. The AI could combine the quantitative tree logic that it had with the qualitative themes as shown in the "Executive Summary" to identify relationships.

- Logic Synthesis: This is the most significant interaction that the AI identified between the words Digital Support and Efficiency and Speed. It concluded that to digitally savvy individuals (Ranking > 4.0), the most important areas to be satisfied with are Library and Administrative Services.

- Understanding Pathways with High-Error: The AI took the 52% error in Rule 19 and converted it into behavioural understanding, as opposed to merely reporting it. It rationalized that this working-age cohort are presumably motivated by some external factors not realized by the existing tree and that digital support is probably a baseline expectation amongst this cohort.

- Mixed-Methods Triangulation: The AI was able to triangulate the quantitative emphasis on "Efficiency" with the qualitative emphasis on the word "library" (30 mentions). This enabled it to make a specific recommendation: "Optimize Academic Support Efficiency… focus on reducing wait times in physical and digital study areas."

Comparative Evaluation

The AI was able to identify a "hierarchy of needs" in the campus system due to the presence of context. As Table 6.3 shows, the quality of suggestions has improved considerably.

Feature	Without Context (Generic)	With Context (Domain-Specific)
Primary Driver	"Focus on Feature SI2."	"Enhance Administrative Digital Interfaces."
Logic Analysis	"Rule 14 leads to Class 1."	"Prioritize transformation of course registration and finance offices."
Qualitative Use	"The word 'library' appeared 30 times."	"Library themes suggest a user base focused on functional utility."
Strategic Fix	"Refine the model parameters."	"Address infrastructure fundamentals like food and parking to improve satisfaction."

Table 6.1: Impact of Context on Actionable Recommendations

Conclusion: This experiment confirms that the "Context" field is necessary for an "Intelligent" analytical tool. The decision tree offers the mathematical relationship of the changes (exposing how one item is related to another), but the contextual AI offers the research story needed to make these relationships relevant for the university administrator.

6.2.4 User-Centric Evaluation: PSSUQ Results

A study with 40 participants who used the system on a desktop was undertaken. PSSUQ has a 7-point scale with 1 as the best score [24].

Sub-scale	Items	Mean Score	Interpretation
System Usefulness (SYSUSE)	1 - 6	1.67	Excellent
Information Quality (INFOQUAL)	7 - 12	1.86	Excellent
Interface Quality (INTERQUAL)	13 - 15	2.25	Good
OVERALL USABILITY SCORE	**1 - 16**	**1.84**	**Excellent**

Table 6.2: PSSUQ Mean Scores (N=40)

Discussion: With an overall score of 1.84 out of 7, users are highly satisfied. The greatest satisfaction was with System Usefulness (1.67), confirming the tabbed interface successfully helped non-technical users to complete complex data science tasks. Although "Interface Quality" was slightly lower (2.25) due to initial concerns about font size, the later change to the CSS and Graphviz parameters has resolved the issue.

6.2.5 Feedback and Continuous Improvement

While the quantitative PSSUQ results were very positive, text feedback from the study identified areas where improvements could be made:

- **Font Sizes**: A number of users reported the default font sizes in the results tab and the decision tree image were too small to read clearly on a high-resolution screen.

- **Instructional Clarity**: A few users reported the instructions for uploading files and setting model parameters were slightly unclear and open to interpretation.

Based on this feedback, the prototype was improved. CSS was updated to use a larger font size for all tabs, and prompt engineering for the graphviz tree was modified to increase the size of the nodes in the tree. In addition, the guidance text that appeared in the app was modified to be less obscure. The incremental change is an example of the success of the Agile project methodology adopted.

6.3 Algorithmic Discussion: The Logic of Pruning

During the testing period, a particular scenario was noticed: despite being set to use the "Top 5" variables, the final tree would show a tree with only four variables. This is not an error, but the desired outcome of Pessimistic Error Post-Pruning, a fundamental feature in the C4.5 algorithm to avoid overfitting.

6.3.1 The Metric: Pessimistic Error Estimation

In a normal decision tree, the "training error" is usually 0 because the tree can grow until its leaves are all 100% pure. But this tree is too large and doesn't work. To avoid this problem, the system uses a Pessimistic Error Estimate for each node.

The system does not trust on the 0% error on the training data. Rather, it punishes any leaves. Assuming pessimistically that its error is (E+0.5)/N, in case a leaf contains N instances and E errors, the system would think that its error is (E+0.5)/N. Pessimistic Threshold = 0.50. This is a statistical benefit of the doubt that the rule is false, we are just yet to discover the counterexample.

6.3.2 The Decision Mechanism: "Subtree vs. Leaf"

The pruning occurs in a bottom-up way (Post-Order Traversal). It compares the value of two factor at every internal node:

1. **Subtree Error** (SE): The sum of all pessimistic errors of the branches below that node.
2. **Leaf Error** (LE): The error rate if all those branches were deleted and the node was turned into a single, simple leaf.

The Pruning Rule:
The system knows to prune **if and only** if the Estimated Error of a single leaf is less than or equal to the total Estimated Error of the complex subtree.

$$\text{If } LE \leq SE \rightarrow \text{Prune Subtree}$$

6.3.3 Why the 5th Variable was Removed

This means that the algorithm logic in the case study where the user requested 5 variables but only 4 of them were shown looked like the following:

- The Chi-Squared test showed the 5th variable was significant with the target.
- The C4.5 algorithm first created a split for the 5th variable at the lowest level in the tree.
- But during pruning, it was calculated that the Leaf Error (a simple outcome) was statistically more accurate than the Subtree Error (the outcome from the 5th variable).
- So at that point, the small amount of data (N was low) meant that the split on the 5th variable was "noise" and not a "trend."

6.3.4 Practical Benefit for the User

This is what makes the "Bigger Picture" interpretation. The system's algorithmically prune of variables that fail to pass the pessimistic test guarantees that:

- Simplicity: The researcher does not get distracted by minor variables that only apply to a very small number of people.
- Generalizability: Rules will be more likely to be true for the population, rather than just the sample in the case study.
- Comprehensibility: The "Final Decision Tree" is still intelligible, with only the "Critical Drivers" of the outcome shown.

6.4 Project Challenges

1. UI Feedback: Early users found the fonts in the tree and results were too small. The prototype was adapted to include custom CSS to increase font sizes and some Graphviz properties were tweaked to improve readability.
2. Instructions: Some found the "Synthesis" term confusing. The user interface was changed to "Reasoning" and "Key Drivers".
3. PDF Generation: The professional PDF generation to capture the AI's markdown code and the images required a cleaning script to remove non-standard characters (emojis), which crashed the library.

6.5 Objectives Evaluation

The project fulfilled all planned objectives as outlined in Chapter 1.

Objective	Status	Evidence of Fulfilment
1. Dynamic Preprocessing Pipeline	Met	Heuristic identifier removal and automated statistics logic.
2. Interpretable C4.5 Modelling	Met	Built C4.5 engine with post-pruning for model simplicity.
3. Generative AI Integration	Met	Synthesis of tree rules and text themes via Gemini API.
4. System Development & Validation	Met	Functional prototype validated by PSSUQ (1.84 score).

Table 6.3: Objective Evaluation

6.6 Concluding Remark

Evaluation stage indicates that system bridges the gap between information and human knowledge. The technical feasibility and user satisfaction of the Intelligent Analytical Assistant is verified by the PSSUQ survey (1.84/7) and the validation of the experts (see case study). The system maintains quality of data with no imputation, and provides an interpretable big picture view of the relationships of the survey.

CHAPTER 7

Conclusion and Recommendation

This is the last chapter of the current study that unites the technical innovations and analytical insights of the AI-Powered Likert Scale Data Interpreter. It provides a final outline on how the system is successful in addressing the research gaps by providing a transferable preprocessing workflow, interpretable C4.5 modelling and novel narrative generation using AI. This chapter confirms the success of the objectives of the project and suggests specific recommendations on further research to develop the system in terms of its greater analytic capabilities and scalability to other researchers.

7.1 Conclusion

This project has effectively addressed one of the biggest dilemmas of quantitative survey studies that are the ability to address the issue of how to extract useful and interpretable insights and actions out of raw Likert-scale data. Likert scales are the most popular method used globally to quantify human opinion, but traditional methods of analysing Likert-scale data only typically provide aggregate summaries, which conceal the decision-making routes that can be found between. Conversely, typical machine learning methods focus on predictive accuracy, rather than on interpretability, and they tend to produce black box models that cannot be used by stakeholders in the most important field such as health, education and social science.

The Intelligent Likert Scale Data Interpreter has fulfilled the major objective of this project: an easy-to-understand analyst in the hands of non-technical scientists. The tool has succeeded in eliminating technical data science jobs (cleaning, statistical filtering, modelling) in the user through sound, three tier system architecture and Python and Streamlit.

Several technical achievements distinguish this prototype from existing solutions:

1. **Generalizable Data Engineering:** The implementation of a dynamic preprocessing pipeline using cardinality-based heuristics ensures that the system is not restricted to a single dataset. It can ingest and clean diverse survey formats, a significant advancement over the hardcoded scripts commonly found in previous academic iterations.

2. **Methodological Integrity:** The system validates that a "white box" approach, utilizing the C4.5 decision tree algorithm, is highly effective for Likert data analysis. By employing a mathematical penalty for missing data rather than using artificial imputation, the system ensures that the generated insights are an honest reflection of genuine respondent data.

3. **The Semantic Breakthrough:** The novel integration of Generative AI (LLMs) to synthesize quantitative logic rules with qualitative NLP themes has successfully bridged the "last mile" of data analysis. The system moves beyond visualization to provide automated storytelling, explaining the "Why" behind the "What."

In summary, the prototype developed across FYP 1 and FYP 2 has proven that interpretable machine learning can be made accessible and powerful simultaneously. The system successfully transforms raw ratings into meaningful insights, fulfilling all project objectives and providing a reusable toolkit for the academic community.

7.2 Recommendations

While the current prototype is highly functional and robust, the following recommendations are proposed for future iterations to further enhance the system's analytical depth and operational scalability:

7.2.1 Advanced NLP Integration: Sentiment Analysis

The current qualitative module focuses on term-frequency analysis to identify themes. Future work could integrate Sentiment Analysis (utilizing libraries such as TextBlob or VADER) to automatically classify open-ended responses as Positive, Negative, or Neutral. Feeding these sentiment scores as additional features into the C4.5 algorithm could provide a deeper layer of triangulation between Likert scores and respondent emotion.

7.2.2 Algorithmic Comparison and Ensemble Methods

To provide even greater validation for the researcher, the system could be expanded to include a "Model Comparison" toggle. Allowing users to compare the C4.5 output with other interpretable models, such as CART (Classification and Regression Trees) or Logistic Regression, would allow for cross-model verification of results. Additionally, exploring the use of "Random Forests" while maintaining interpretability through Global Surrogate Models could offer higher accuracy for extremely large datasets.

7.2.3 Persistent Data Management and User Authentication

To move from a functional prototype to a production-ready research platform, the integration of a persistent database (e.g., PostgreSQL) is recommended. This would allow researchers to create secure accounts, save their analysis sessions, and compare historical surveys over time. Such a feature would enable trend analysis and longitudinal studies, which are critical in many areas of social science research.

7.2.4 Interactive PDF Customization

The current reporting module provides a comprehensive, static PDF. Future enhancements could include a "Report Builder" interface where users can drag and drop specific sections, select specific charts to include, and add their own manual annotations to the AI interpretation before finalizing the export. This would further empower the user to tailor the output to their specific stakeholder audience.

REFERENCES

[1] A. Field, *Discovering Statistics Using IBM SPSS Statistics*, 5th ed. London, U.K.: SAGE Publications, 2018.

[2] A. B. Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI," *Information Fusion*, vol. 58, pp. 82-115, Jun. 2020, doi: 10.1016/j.inffus.2019.12.012.

[3] S. S. Stevens, "On the Theory of Scales of Measurement," *Science*, vol. 103, no. 2684, pp. 677-680, Jun. 1946, doi: 10.1126/science.103.2684.677.

[4] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann Publishers, 1993.

[5] S. Dasu and T. Johnson, *Exploratory Data Mining and Data Cleaning*. Hoboken, NJ, USA: John Wiley & Sons, 2003.

[6] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Waltham, MA, USA: Morgan Kaufmann, 2012.

[7] R. Likert, "A Technique for the Measurement of Attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 1–55, 1932.

[8] H. S. Bloom, C. J. Hill, and J. A. Riccio, "Linking program implementation and effectiveness: Lessons from a pooled sample of welfare-to-work experiments," *Journal of Policy Analysis and Management*, vol. 22, no. 4, pp. 551–575, 2003, doi: 10.1002/pam.10164.

[9] Q. Zheng, M. Chen, and W. Chen, "A survey on machine learning methods for Likert scale data analysis," *PLoS One*, vol. 17, no. 4, p. e0266729, 2022, doi: 10.1371/journal.pone.0266729.

[10] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182, Mar. 2003. [Online]. Available: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf

[11] T. Brown *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877-1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[12] D. Pyle, *Data Preparation for Data Mining*. San Francisco, CA, USA: Morgan Kaufmann, 1999.

[13] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Boston, MA, USA: Kluwer Academic, 1998.

[14] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth, 1984.

[15] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Mach. Intell.*, vol. 1, no. 5, pp. 206–215, May 2019, doi: 10.1038/s42256-019-0048-x.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[17] G. M. Sullivan and A. R. Artino Jr., "Analyzing and interpreting data from Likert-type scales," *J. Grad. Med. Educ.*, vol. 5, no. 4, pp. 541–542, Dec. 2013, doi: 10.4300/JGME-5-4-18.

[18] W. X. Zhao *et al.*, "A Survey of Large Language Models," *arXiv preprint arXiv:2303.18223*, 2023. [Online]. Available: https://arxiv.org/abs/2303.18223

[19] GeeksforGeeks."Decision Tree Algorithms." GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/decision-tree-algorithms/

[20] A. Navlani. "Decision Tree Classification in Python." MachineLearningGeek.com. https://machinelearninggeek.com/decision-tree-classification-in-python/

[21] GeeksforGeeks. "Chi-Square Test for Independence" GeeksforGeeks. https://www.geeksforgeeks.org/r-language/chi-square-test-for-independence/

[22] F. V. Ordenes, B. Theodoulidis, J. Burton, T. Gruber, and M. Zaki, "Analyzing customer experience feedback using text mining: A linguistics-based approach," *J. Serv. Res.*, vol. 17, no. 3, pp. 278–295, Aug. 2014.

[23] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and I. Mathur, *Natural Language Processing: Python and NLTK*. Birmingham, UK: Packt Publishing Ltd, 2016.

[24] UIUX Trend, "PSSUQ – Post-Study System Usability Questionnaire," *UIUX Trend*. [Online]. Available: https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/

APPENDIX A: USABILITY TEST QUESTIONNAIRE (PSSUQ)

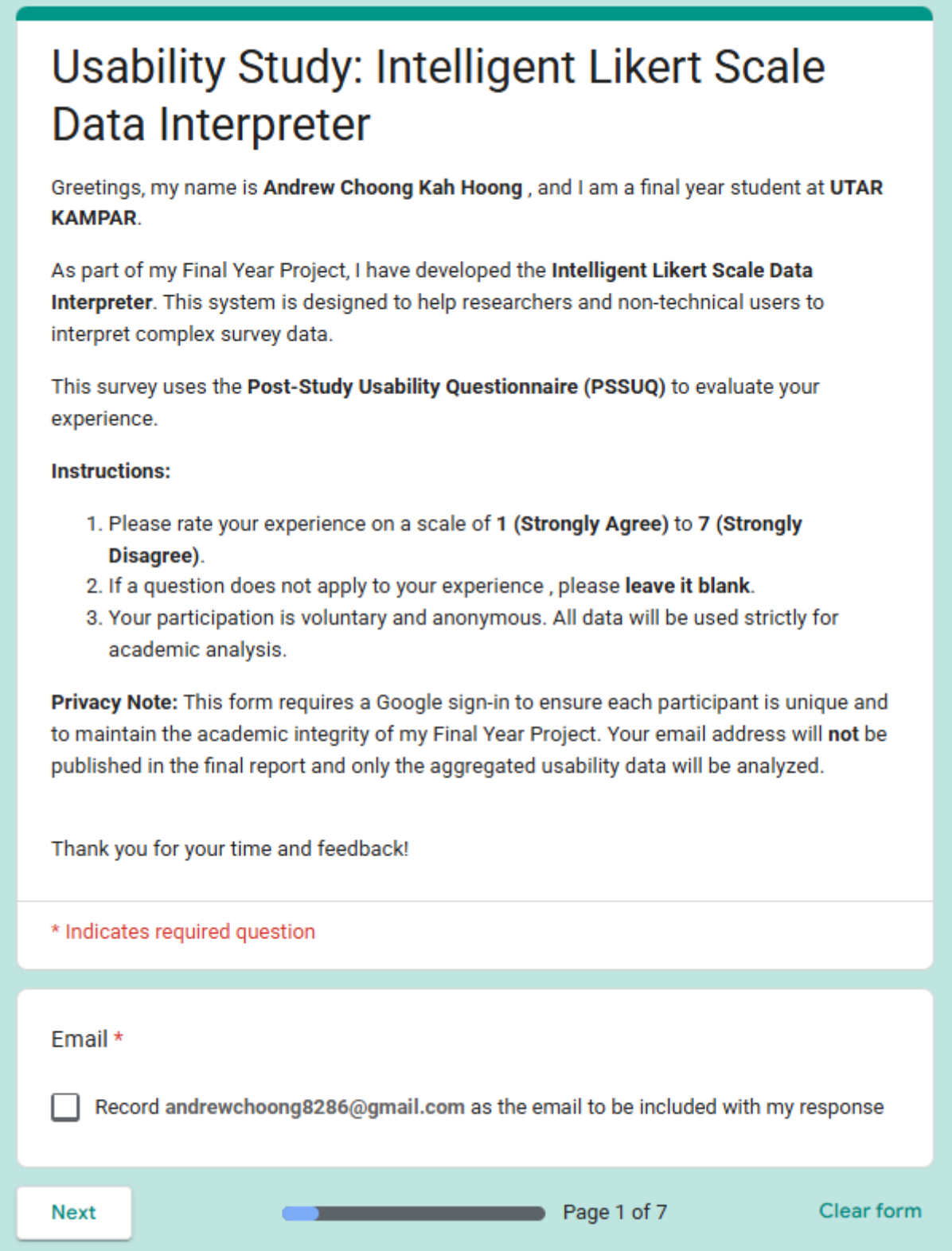


Figure A1

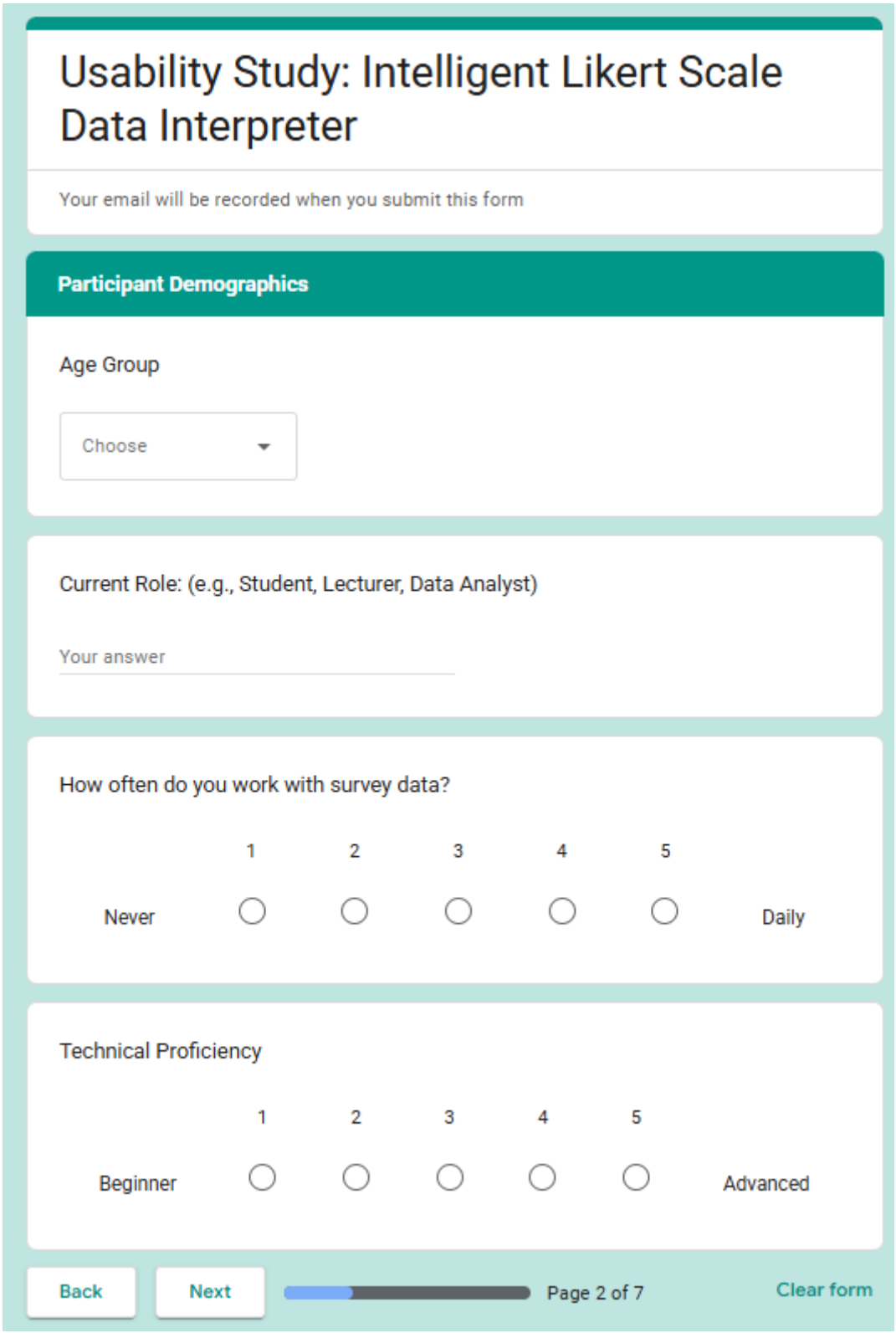

Usability Study: Intelligent Likert Scale Data Interpreter
Your email will be recorded when you submit this form
Participant Demographics
Age Group
Choose
Current Role: (e.g., Student, Lecturer, Data Analyst)
Your answer
How often do you work with survey data?
1 2 3 4 5
Never
Daily
Technical Proficiency
1 2 3 4 5
Beginner
Advanced
Back
Next
Page 2 of 7
Clear form

Figure A2

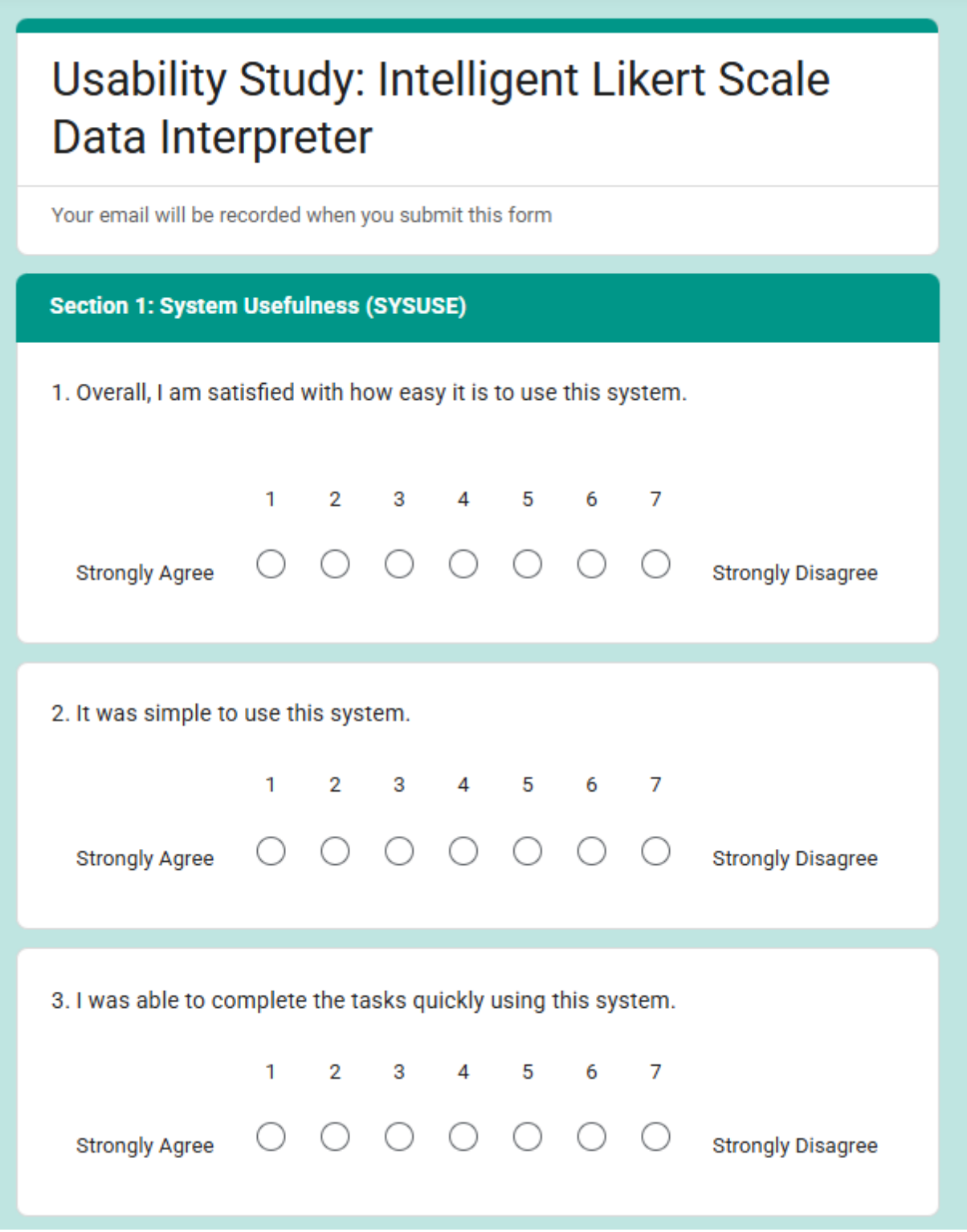

Usability Study: Intelligent Likert Scale Data Interpreter
Your email will be recorded when you submit this form
Section 1: System Usefulness (SYSUSE)
1. Overall, I am satisfied with how easy it is to use this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
2. It was simple to use this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
3. I was able to complete the tasks quickly using this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree

Figure A3

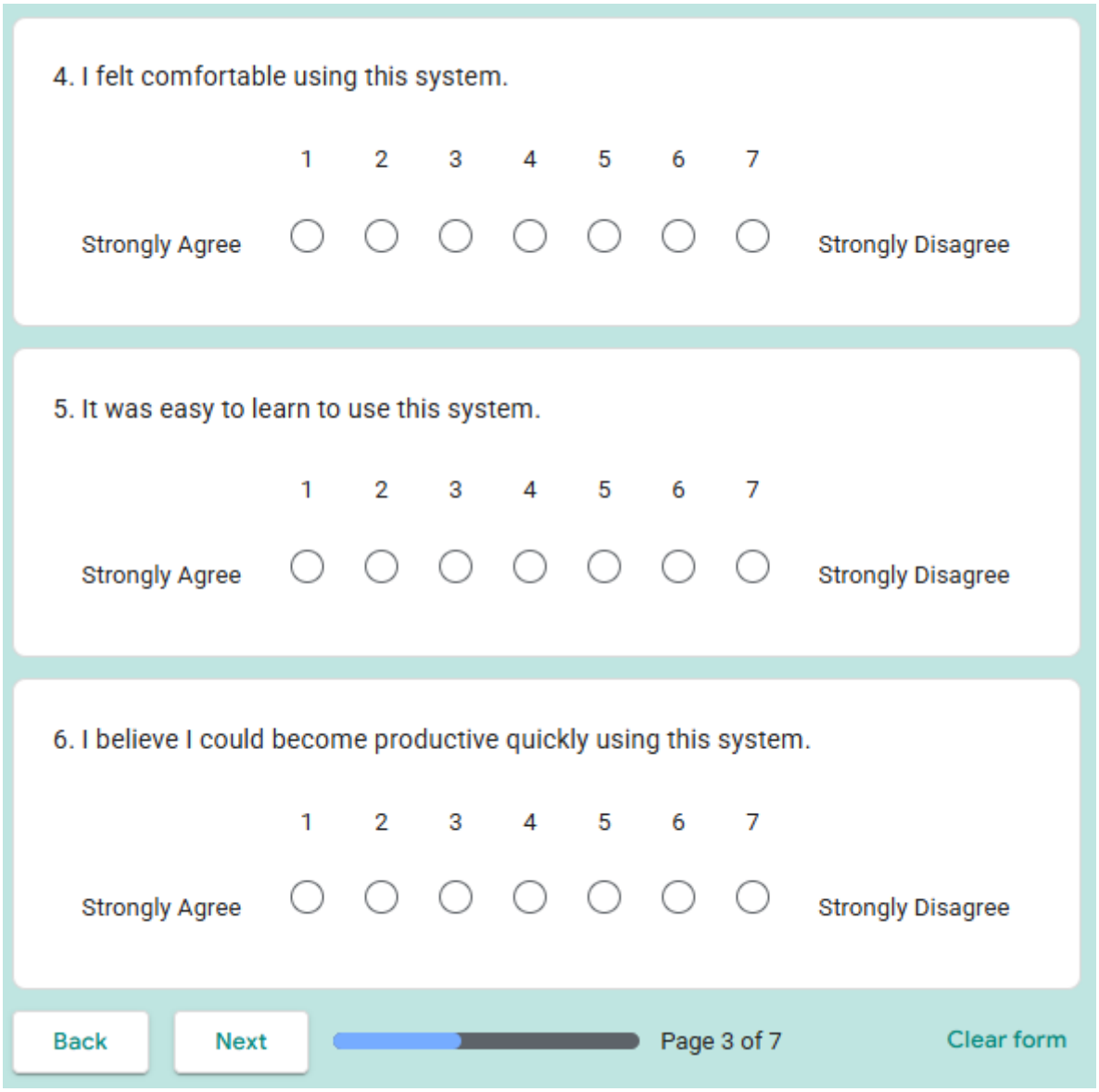
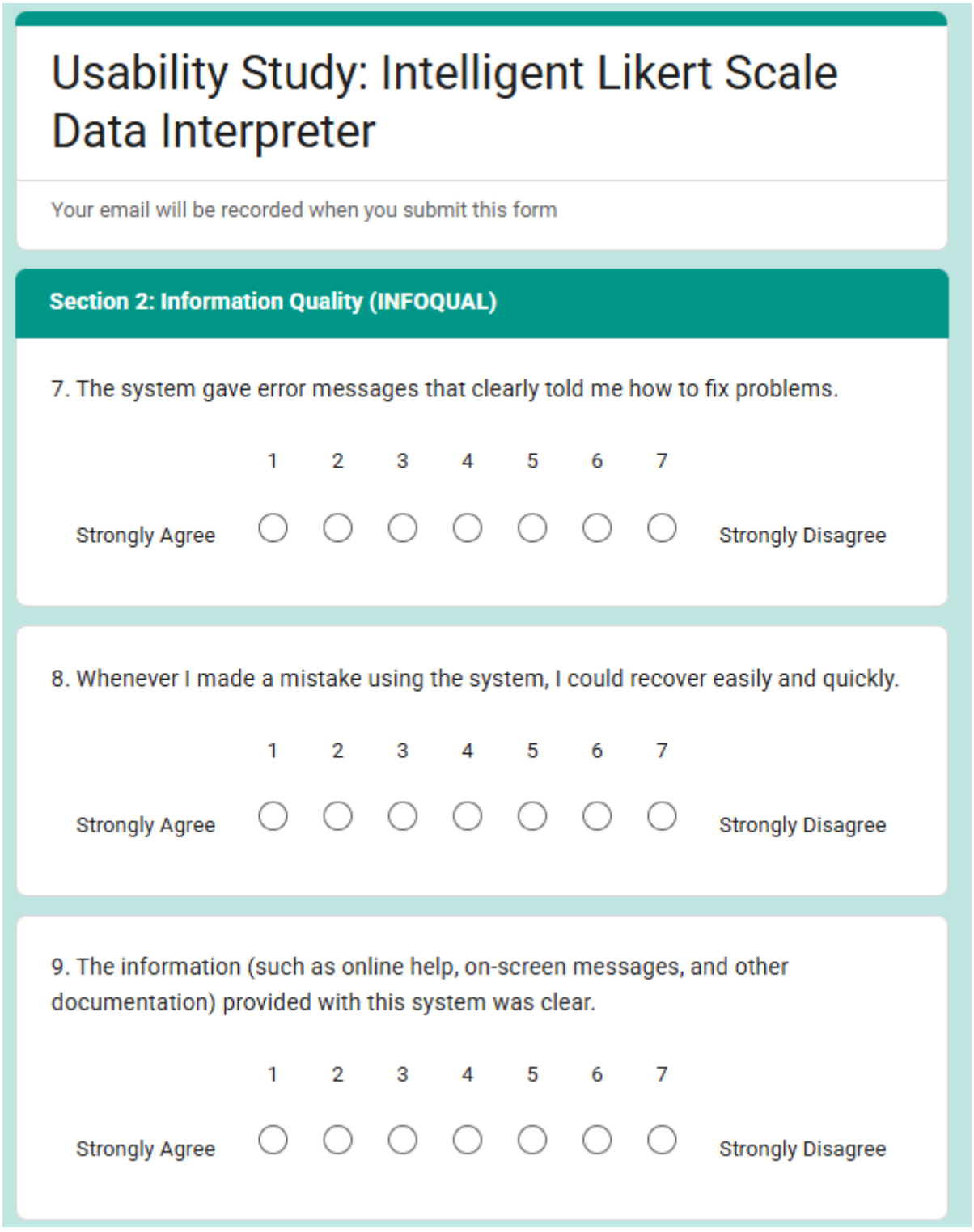

4. I felt comfortable using this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
5. It was easy to learn to use this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
6. I believe I could become productive quickly using this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
Back
Next
Page 3 of 7
Clear form

Figure A4



Usability Study: Intelligent Likert Scale Data Interpreter

Your email will be recorded when you submit this form

Section 2: Information Quality (INFOQUAL)

7. The system gave error messages that clearly told me how to fix problems.

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

8. Whenever I made a mistake using the system, I could recover easily and quickly.

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

9. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.

	1	2	3	4	5	6	7	
Strongly Agree	○	○	○	○	○	○	○	Strongly Disagree

Figure A5

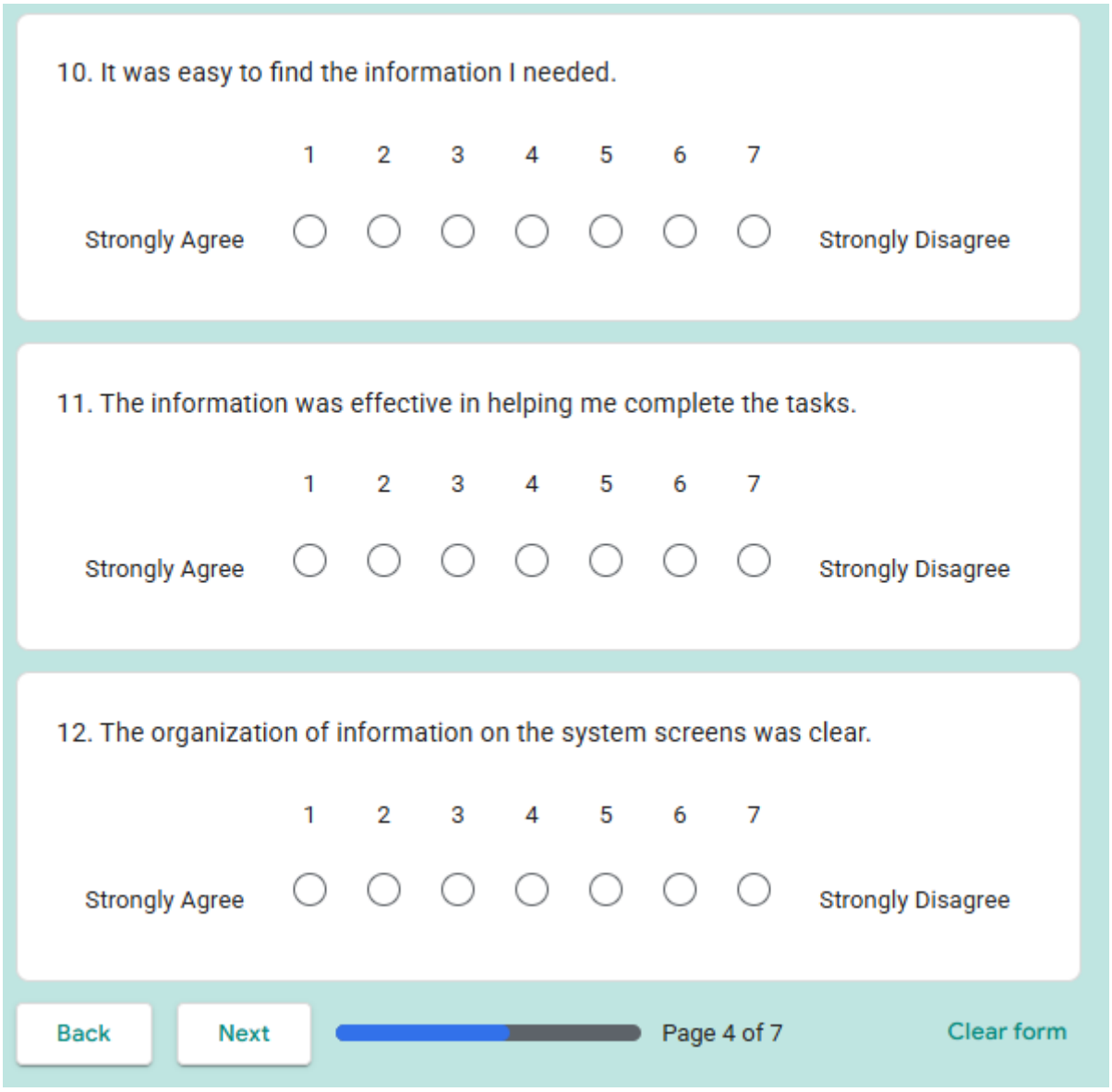

10. It was easy to find the information I needed.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
11. The information was effective in helping me complete the tasks.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
12. The organization of information on the system screens was clear.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
Back
Next
Page 4 of 7
Clear form

Figure A6

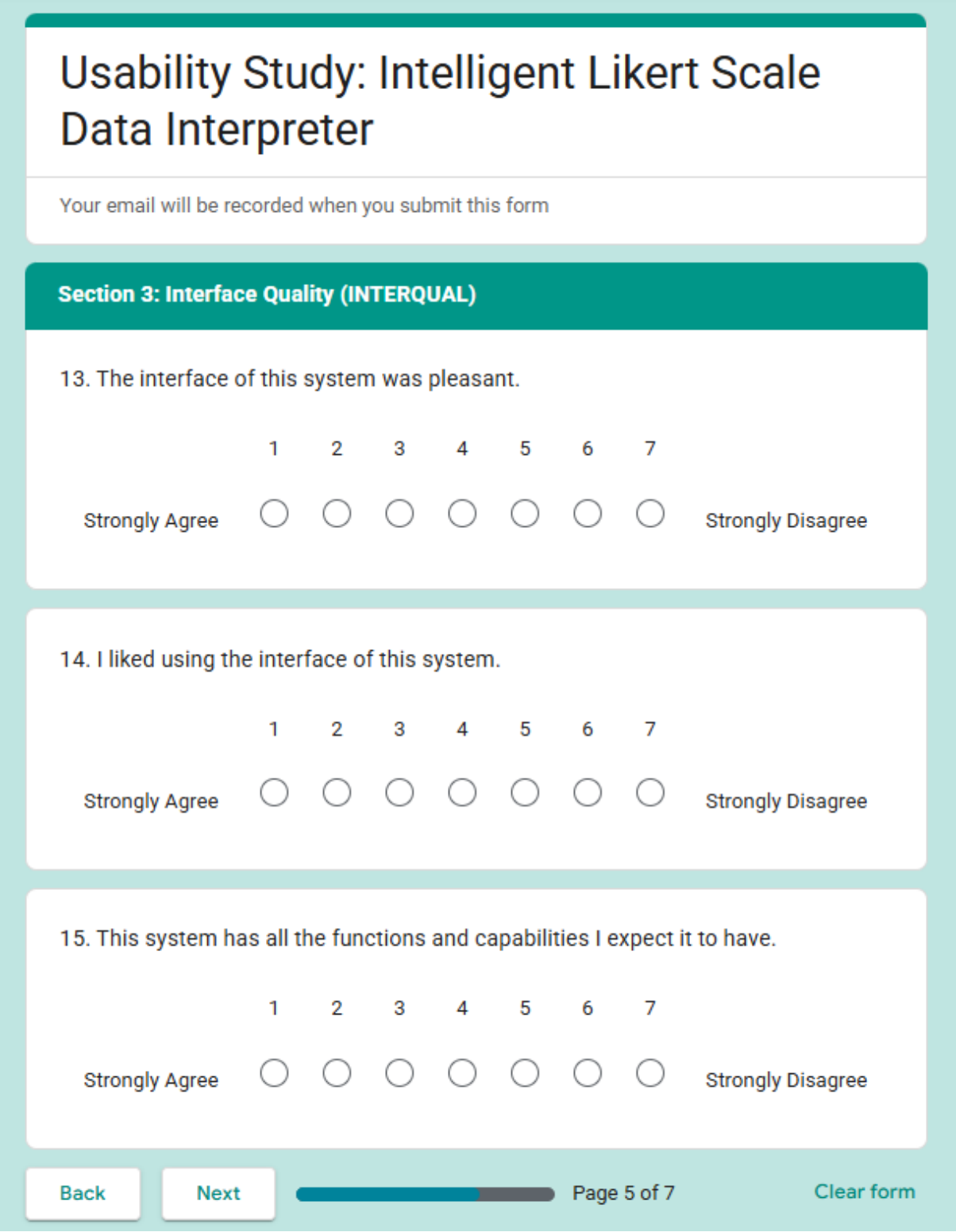

Usability Study: Intelligent Likert Scale Data Interpreter
Your email will be recorded when you submit this form
Section 3: Interface Quality (INTERQUAL)
13. The interface of this system was pleasant.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
14. I liked using the interface of this system.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
15. This system has all the functions and capabilities I expect it to have.
1 2 3 4 5 6 7
Strongly Agree
Strongly Disagree
Back
Next
Page 5 of 7
Clear form

Figure A7

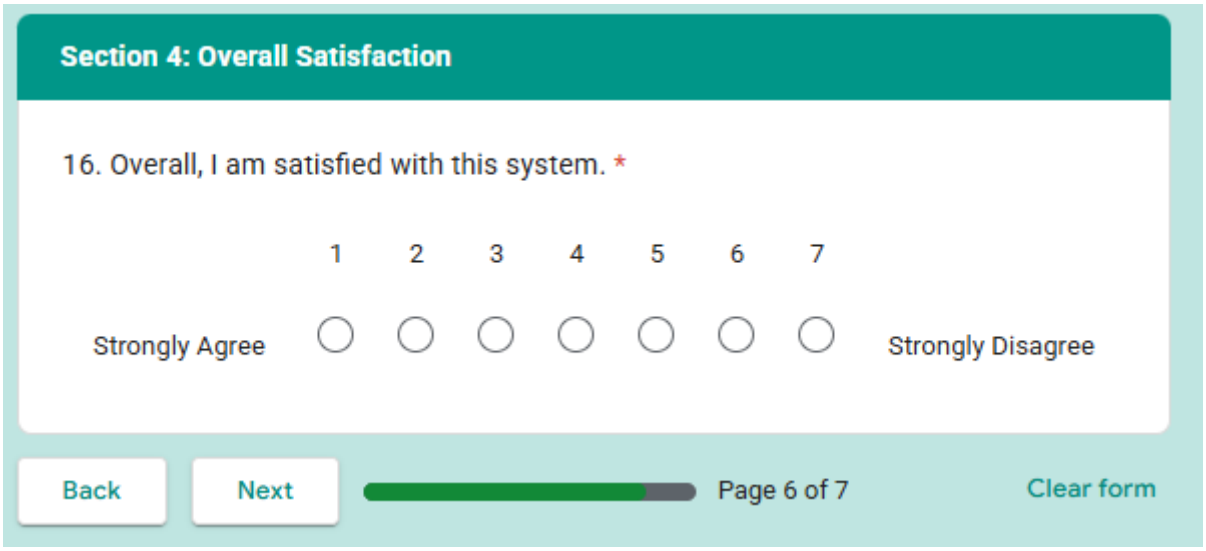


Figure A8

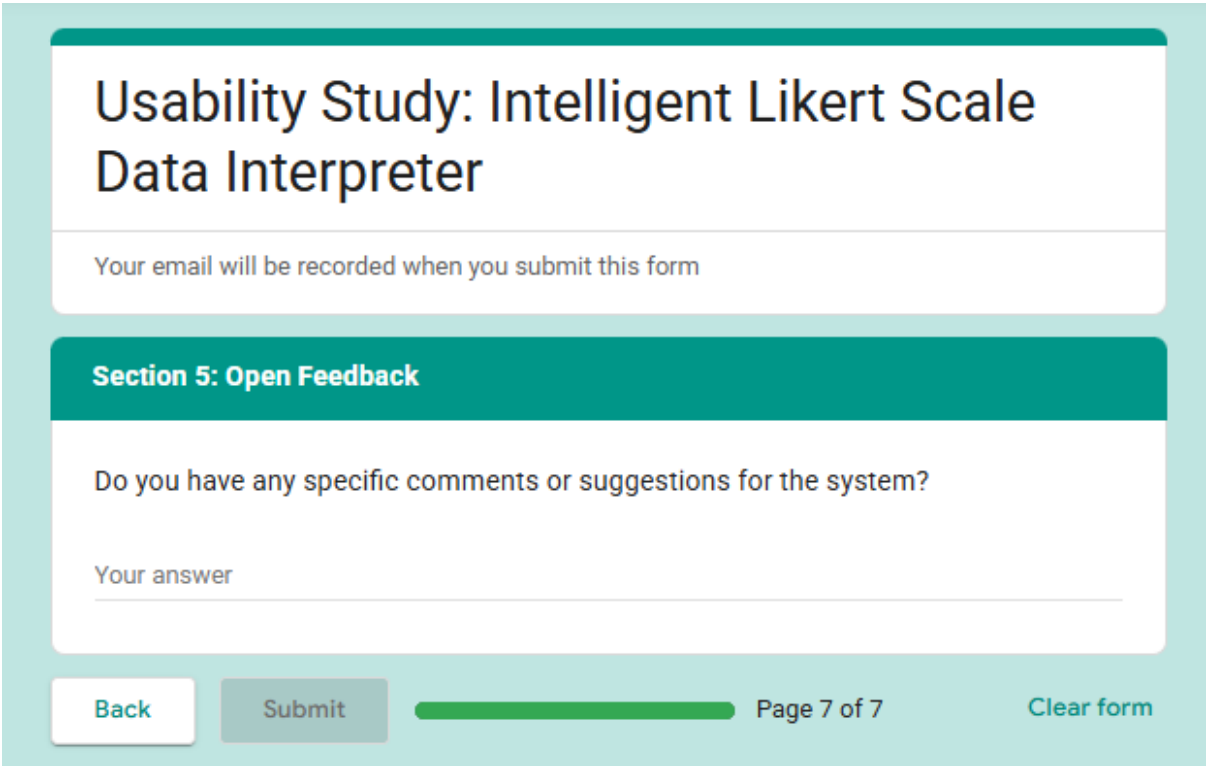


Figure A9

APPENDIX B: POSTER

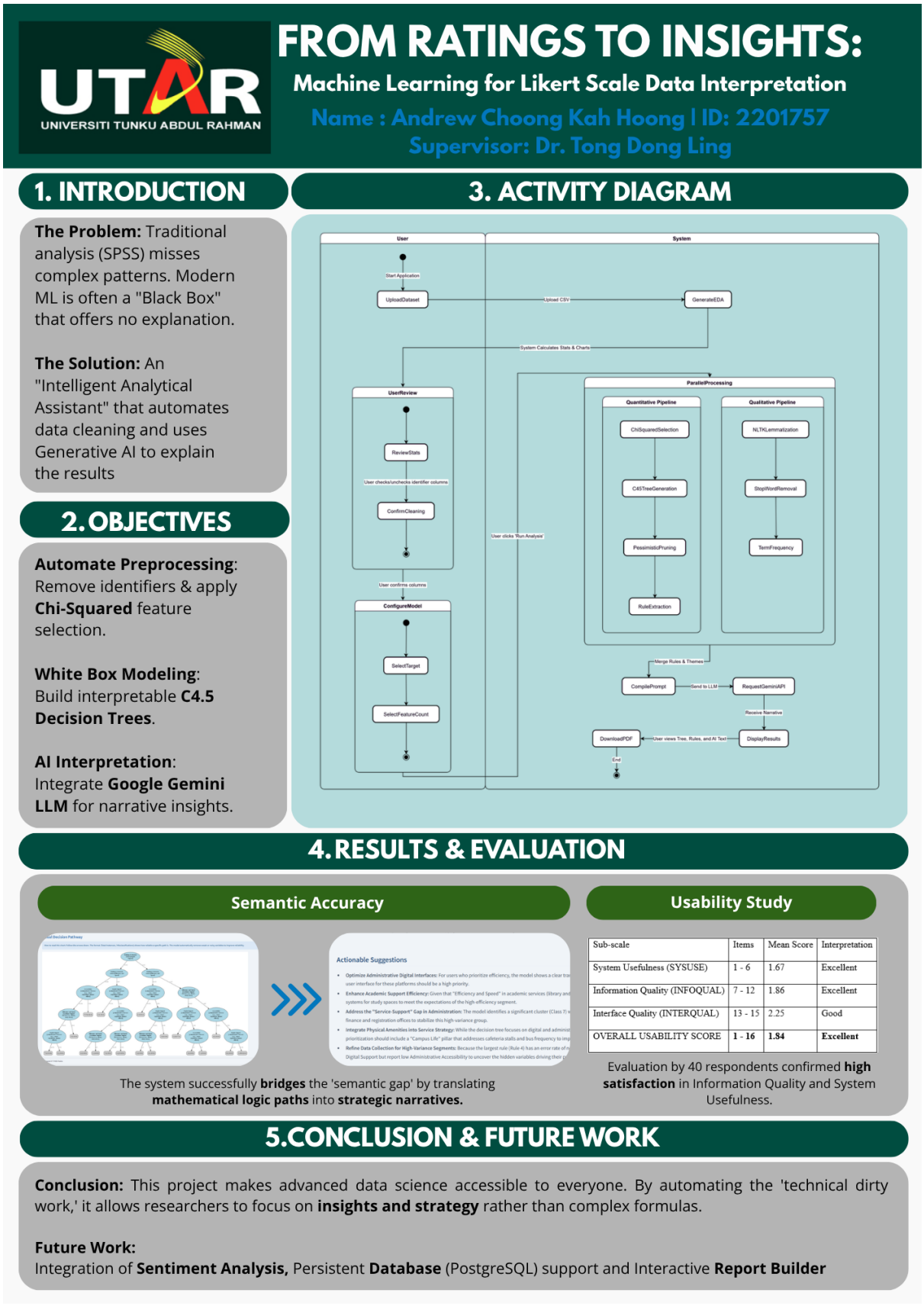


Figure B1