THE UNIVERSITY OF ZAMBIA

SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTING AND INFORMATICS

FINAL YEAR PROJECT REPORT

ACADEMIC YEAR 2026

CREDIT SCORING SYSTEM FOR MULTICLASS LOAN APPLICANT CLASSIFICATION USING MACHINE LEARNING

A report submitted to the Department of Computing and Informatics in partial fulfilment of the requirements for the award of the degree of Bachelor of Science in Computer Science

BY

JOSHUA MVULA

COMPUTER NUMBER: 2021424014

SUPERVISOR: Dr Lighton Phiri

CO-SUPERVISOR: Mr. Mwangala Mwiya

DATE OF SUBMISSION: 22nd October 2026

DECLARATION

I, Joshua Mvula, Computer Number 2021424014, declare that this report, titled "Credit Scoring System for Multiclass Loan Applicant Classification using Machine Learning", is my own original work, undertaken individually. It has not been submitted previously, in whole or in part, for any other degree or examination at this or any other university. All sources of information used have been acknowledged through citations and references. I understand that plagiarism is an academic offence and I accept the consequences of any breach.

Name: _____

Computer number: _____

Signature: _____

Date: _____

APPROVAL

This final year project report has been submitted for examination with the approval of the project supervisor.

Supervisor Name: _____

Signature: _____

Date: _____

Departmental/Examiner Approval (if applicable): _____

ABSTRACT

Credit assessment at many lenders in Zambia still depends on manual judgement and on formal credit histories that large parts of the population do not have. This project developed a credit scoring system that classifies loan applicants into four ordered risk categories (low, medium, high and very high) using machine learning. Loan and repayment records from the PostgreSQL database of Hobbiton Investment Management were extracted, consolidated at borrower level and transformed into six behavioural features: average loan amount, average number of instalments, loan count, average days late, maximum days late and overdue loan count. An audit of the database showed that demographic, employment, education and alternative-data fields were absent or unreliable, which limited the feature set to repayment behaviour.

The study followed CRISP-DM and Scrum. Four classifiers (Logistic Regression, Decision Tree, Random Forest and XGBoost) were trained on 293 borrowers and tested on 126 under identical conditions. XGBoost performed best, with a test accuracy of 85.71%, a macro-averaged F1-score of 85.11%, a micro-average ROC-AUC of 0.98 and a cross-validated accuracy of 87.59% (±3.06%). The selected model was packaged as a Flask REST service hosted on Render and is called through a PHP REST gateway and a PHP MVC web interface hosted on InfinityFree. For each applicant the prototype returns a risk category with class probabilities and reports failures with distinct error codes. Because the risk labels were derived from repayment rules rather than observed defaults, future work should validate the model against real default outcomes and extend the feature set when richer data become available.

Keywords: credit scoring; multiclass classification; XGBoost; buy-now-pay-later; financial inclusion; REST API

ACKNOWLEDGEMENTS

I thank my supervisor, Dr Lighton Phiri, and my co-supervisor, Mr. Mwangala Mwiya, for their guidance, patience and constructive criticism throughout this project. I am grateful to Hobbiton Investment Management for granting access to the data that made this study possible, and to the Department of Computing and Informatics at the University of Zambia for the training that prepared me for it.

I also thank my family and friends for their encouragement and support, and my classmates for the discussions that sharpened my thinking. Above all, I give thanks to God for the health and strength to complete this work.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION **1**
1.1 Background and Context 1
1.2 Problem Statement 1
1.3 Aim 2
1.4 Objectives 2
1.5 Research Questions 2
1.6 Scope 3
1.7 Significance of the Study 3
1.8 Limitations of the Study 4
1.9 Organisation of the Report 4
CHAPTER 2: LITERATURE REVIEW **5**
2.1 Introduction 5
2.2 Background to Credit Scoring 5
2.2.1 From Manual Judgement to Statistical Models 5
2.2.2 Risk Management and Financial Inclusion 6
2.2.3 The Rise of Digital Finance and Alternative Data 6
2.3 Feature Integration and Diverse Data Sources 7
2.3.1 Financial Inclusion and Credit Access Challenges in Zambia 7
2.3.2 Limitations of Traditional Credit Scoring Models 10
2.3.3 Alternative Data Sources for Credit Scoring 11
2.3.4 Feature Extraction from Raw Transactional Data 12
2.4 Multi-Class Credit Scoring 14
2.4.1 Machine Learning Approaches in Credit Scoring 14
2.4.2 Multi-Class Classification in Credit Risk 15
2.4.3 Class Imbalance and Structural Challenges in African Contexts 15
2.4.4 Justification for a Multi-Class Approach in Zambia 16
2.4.5 Ordinal Classification in Credit Risk 17
2.5 Model Evaluation and Decision Support 17
2.5.1 Evaluation Metrics in Credit Scoring 17
2.5.2 Automated Decision Support and Market Validation 19
2.5.3 Justification for a Multi-Metric Evaluation Pipeline 19
2.6 System Integration via Web Service 20
2.6.1 Web Service Integration as a Model Serving Infrastructure 20
2.6.2 Data Governance and Regulatory Challenges in Integration 21
2.6.3 Integration Framework for the Zambian Credit Scoring Context 21

2.6.4 Justification for Integration with Existing Web Service 22
2.6.5 Feature Alignment and Real-Time Preprocessing 22
2.7 Summary of Literature and Gaps in the Study 24
CHAPTER 3: METHODOLOGY 27
3.1 Introduction 27
3.2 Research Design 27
3.3 Risk Analysis 28
3.3.1 Data Quality and Availability Risk 28
3.3.2 Asymmetric Misclassification Risk 28
3.3.3 System Integration Risk 29
3.3.4 Data Drift and Model Staleness Risk 29
3.3.5 Privacy and Ethical Risk 29
3.4 Methodological Framework 29
3.5 CRISP-DM Framework 30
3.6 Business Understanding 31
3.7 Data Understanding 32
3.7.1 Database Review Procedure 32
3.7.2 Feature Audit Approach 33
3.7.3 Document Data Review 33
3.7.4 Target Variable Derivation 34
3.8 PostgreSQL Data Pipeline 34
3.9 Data Preparation 35
3.10 Modelling 35
3.11 Evaluation 37
3.12 Deployment 37
3.13 Scrum Development Methodology 38
3.14 System Architecture 38
3.14.1 Existing System Constraints and Integration Context 39
3.15 Tools and Technologies 40
3.16 Security and Ethical Considerations 41
3.17 Chapter Summary 41
CHAPTER 4: SYSTEM ANALYSIS AND DESIGN 42
4.1 Introduction 42
4.2 Stakeholder Analysis 42
4.3 Functional Requirements 42
4.4 Non-Functional Requirements 43

4.5 Use Case Model 44
4.6 System Architecture 45
4.7 Data Design 46
4.8 Interface and Interaction Design 48
4.9 Technical Design 48
4.9.1 Feature construction 48
4.9.2 Risk label derivation 49
4.9.3 Model configuration 49
4.9.4 Class structure and interaction 50
4.10 Chapter Summary 52
CHAPTER 5: SYSTEM IMPLEMENTATION 53
5.1 Introduction 53
5.2 Development Environment and Tools 53
5.3 System Components 54
5.4 Key Implementation Details 54
5.4.1 Data extraction and feature engineering 54
5.4.2 Model training and persistence 55
5.4.3 Flask scoring service 55
5.4.4 PHP gateway 55
5.4.5 MVC web interface 56
5.5 Integration 56
5.6 Deployment and Hosting 56
5.7 Challenges and Solutions 58
5.8 Chapter Summary 58
CHAPTER 6: TESTING, RESULTS AND EVALUATION 59
6.1 Introduction 59
6.2 Dataset and Modelling Population 59
6.2.1 Target Variable Distribution 60
6.3 Feature Engineering Results 61
6.4 Justification for the Final Feature Set 63
6.5 Model Training Results 65
6.6 Comparative Model Performance 66
6.7 Per-Class Model Performance 68
6.8 Confusion Matrix Results 68
6.9 ROC-AUC Results 69
6.10 Feature Importance Analysis 70

6.11 Cross-Validation Results 71
6.12 Selection of the Final Model 72
6.13 System Testing 74
6.14 Discussion of Results 76
6.15 Limitations Observed in the Results 76
6.16 Chapter Summary 77
CHAPTER 7: DISCUSSION 78
7.1 Introduction 78
7.2 Interpretation of the Findings 78
7.3 Comparison with Existing Work 78
7.4 Contributions 79
7.5 Limitations and Implications 79
7.6 Chapter Summary 80
CHAPTER 8: CONCLUSION AND RECOMMENDATIONS 81
8.1 Introduction 81
8.2 Summary of the Project 81
8.3 Conclusions 81
8.4 Achievement of Objectives 82
8.5 Recommendations 82
8.6 Future Work 82
8.7 Chapter Summary 83
REFERENCES 84
APPENDICES 90
Appendix A: API Specification and Sample Payloads 90
Appendix B: Postman Test Collection 90
Appendix C: Deployment Configuration 91
Appendix D: Feature Engineering Source Code 91

LIST OF FIGURES

Figure 2.1: Adults with a mobile money account (%) 2014-2021 [3] 7
Figure 2.2: Financial Access Strand by Rural/ Urban [2] 8
Figure 2.3: Financial Access Strand by Gender in 2020 [2] 9
Figure 2.4: Financial Access Strand by Age(percent) in 2020 [2] 9
Figure 2.5: Percentage of Firms with Various Obstacles to Access to Credit [2] 10
Figure 3.1: CRISP-DM and Scrum relationship in the project 30
Figure 3.2: Business problem and project objectives 32
Figure 3.3: System architecture 39
Figure 4.1: UML use case diagram of the credit scoring system 44
Figure 4.2: UML deployment diagram of the hosted system 45
Figure 4.3: Entity-relationship diagram of the source and derived data 47
Figure 4.4: UML class diagram of the PHP frontend and gateway 50
Figure 4.5: UML class diagram of the Python scoring service 51
Figure 4.6: UML sequence diagram of a scoring request 51
Figure 5.1: Render web service hosting the scoring service 57
Figure 5.2: InfinityFree hosting account for the PHP components 57
Figure 6.1: Distribution of borrowers across credit-risk categories 61
Figure 6.2: Distribution of engineered borrower-level credit-risk features 63
Figure 6.3: Outcome of the Hobbiton database feature audit 65
Figure 6.4: Model training process in Google Colab 66
Figure 6.5: Comparison of model accuracy 67
Figure 6.6: Confusion matrix for the XGBoost model 69
Figure 6.7: ROC Curve 70
Figure 6.8: Feature importance of the XGBoost credit-scoring model 71
Figure 6.9: Example credit-risk prediction input 73
Figure 6.10: Example credit-risk prediction output 73

LIST OF TABLES

Table 2.1: Summary of Literature and Gaps in the Study 24
Table 4.1: Stakeholders and their needs 42
Table 4.2: Functional requirements 43
Table 4.3: Non-functional requirements and design responses 43
Table 4.4: Use case specification: request credit risk score 44
Table 4.5: Definition of the model features 47
Table 4.6: REST endpoints exposed by the system 48
Table 4.7: Rule-based risk points used to derive the target 49
Table 4.8: XGBoost configuration 49
Table 5.1: Tools and technologies used 53
Table 5.2: Components and their responsibilities 54
Table 5.3: Deployment summary 56
Table 5.4: Implementation challenges and how they were resolved 58
Table 6.1: Summary of the Final Modelling Dataset 59
Table 6.2: Distribution of Borrowers by Credit-Risk Category 60
Table 6.3: Descriptive Statistics of the Final Model Features 62
Table 6.4: Summary of Feature Availability in Hobbiton Database 64
Table 6.5: Comparative Performance of the Credit-Scoring Models 66
Table 6.6: Per-Class Performance of the XGBoost Model 68
Table 6.7: Cross-Validation Performance of the Evaluated Models 72
Table 6.8: System test cases and results 74
Table B.1: Requests in the Postman collection 90

LIST OF ABBREVIATIONS

Abbreviation	Meaning
API	Application Programming Interface
AUC	Area Under the Curve
BNPL	Buy-Now-Pay-Later
CRISP-DM	Cross-Industry Standard Process for Data Mining
CSS	Cascading Style Sheets
CV	Cross-Validation
ER	Entity-Relationship
F1	Harmonic mean of precision and recall
FR / NFR	Functional / Non-Functional Requirement
FTP	File Transfer Protocol
HTTP(S)	Hypertext Transfer Protocol (Secure)
JSON	JavaScript Object Notation
ML	Machine Learning
MVC	Model-View-Controller
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
ROC	Receiver Operating Characteristic
SQL	Structured Query Language
TC	Test Case
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
XGBoost	Extreme Gradient Boosting

CHAPTER 1: INTRODUCTION

1.1 Background and Context

Credit scoring is the process of estimating the likelihood that a borrower will repay a loan, and it underpins lending decisions in banks, microfinance institutions and digital lenders [5], [6]. Traditional scorecards depend on formal credit histories, yet a large share of adults in Zambia has little or no such history, and access to formal credit remains uneven across rural and urban areas, between men and women and across age groups [2], [25]. Lenders that serve these customers need ways of judging risk from the data they already hold.

Buy-now-pay-later (BNPL) and other digital lending products generate rich transactional records: every disbursement, maturity date and repayment is stored with a timestamp. Machine learning methods can learn repayment patterns from such records and have been shown to outperform traditional statistical scorecards on many lending datasets [7], [8], [20]. Ensemble methods such as Random Forest and gradient boosting are particularly suited to this task because they capture non-linear relationships and interactions between behavioural variables that linear models cannot represent [19], [20], [34].

Hobbiton Investment Management operates a BNPL lending product and maintains a PostgreSQL database of members, loan disbursements and repayments. Hobbiton needs a reliable and automated way of grading borrowers by risk that its existing web service can call. This project responds to that need by developing, evaluating and deploying a multiclass credit scoring system built on Hobbiton's own historical data.

1.2 Problem Statement

Credit assessment at Hobbiton, as at many Zambian lenders, relies on manual review that is slow, inconsistent between officers and difficult to scale as the number of applications grows. Much of the published machine learning work on credit scoring treats the problem as binary (default or non-default), uses simulated or public datasets from other markets, or relies on variables that are not available in a lender's operational systems [22], [29]. A binary outcome

gives loan officers little guidance on how much risk an applicant carries, whereas graded risk categories support differentiated decisions such as loan limits and monitoring intensity.

There is therefore a need for an automated, data-driven and multiclass credit scoring system that is built on real lender data, evaluated against alternative algorithms and deployed in a form that Hobbiton's existing web service can consume.

1.3 Aim

To develop and deploy an automated, data-driven, multiclass credit scoring system that classifies Hobbiton loan applicants into risk categories to support timely and consistent credit risk assessment.

1.4 Objectives

1. To identify and integrate relevant features from the data available in Hobbiton's systems into a unified credit scoring feature framework, originally targeting insurance, investment, payment-transaction, demographic and psychometric sources.
2. To preprocess and analyse the Hobbiton dataset and to develop a multiclass model that classifies borrowers into four risk categories.
3. To evaluate the effectiveness of the model using several performance measures and to compare it with baseline classifiers.
4. To deploy the selected model as a web service with a REST interface and a web front end that Hobbiton's existing web service can integrate.

1.5 Research Questions

1. Which credit-relevant features can be derived from the data that Hobbiton actually holds?
2. How do Logistic Regression, Decision Tree, Random Forest and XGBoost compare when classifying borrowers into four risk categories on the same data?

3. Can the selected model be served through a REST architecture so that a web interface can obtain a risk category and class probabilities for a new applicant?

1.6 Scope

The study is limited to the following.

- BNPL loan and repayment records held in Hobbiton's staging PostgreSQL database, accessed through a read-only extract.
- A four-class ordinal risk target (low, medium, high, very high) derived from repayment behaviour.
- Four candidate algorithms: Logistic Regression, Decision Tree, Random Forest and XGBoost.
- A working prototype made up of a Flask scoring service, a PHP REST gateway and a PHP MVC web interface, hosted on free-tier platforms.

The study does not cover changes to Hobbiton's production database or application logic, fully automated lending decisions without human review, real-time retraining, regulatory certification of the model, or data domains that the feature audit found to be absent from the database.

1.7 Significance of the Study

- **Hobbiton and its loan officers** obtain a consistent, repeatable way of grading applicants, which reduces reliance on individual judgement and shortens assessment time.
- **Borrowers** benefit indirectly, because behaviour-based scoring can serve people without formal credit histories and applies the same rules to everyone.
- **Researchers** gain evidence from a real Zambian lender on which features are available and how common algorithms compare on them, a setting that the literature reviewed in Chapter 2 shows to be under-studied.

- **The Department and future students** gain a documented, extensible reference architecture for deploying a machine learning model behind a REST interface.

1.8 Limitations of the Study

- Demographic, employment, education and alternative-data fields were absent or unreliable in the database, so the model uses repayment behaviour only (Section 6.3).
- The risk labels were derived from repayment rules because observed default outcomes were not available, so the model learns to reproduce that rule-based grading rather than actual defaults.
- The final dataset contains 419 borrowers, which limits statistical power and the generalisability of the results.
- The prototype runs on free hosting tiers with sleeping services and usage limits, and is not hardened for production use.

1.9 Organisation of the Report

Chapter 2 reviews the literature on credit scoring, machine learning, alternative data and model deployment, and identifies the research gap. Chapter 3 describes the methodology, which combines CRISP-DM with Scrum. Chapter 4 presents the system analysis and design, including requirements and UML models. Chapter 5 describes the implementation and deployment. Chapter 6 reports the data audit, model results, system testing and an objective-by-objective evaluation. Chapter 7 discusses the findings, and Chapter 8 concludes the report with recommendations and future work. References and appendices follow.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter reviews literature on credit scoring as a core instrument in modern lending, tracing its evolution from rule-based and statistical models to machine learning and alternative data approaches. By synthesising existing literature, this chapter identifies the structural limitations of traditional credit assessment methods, particularly their dependence on formal bureau records and the extent to which those limitations are acute in Zambia and the broader Southern African region. The chapter further examines the state of multi-class credit risk modelling, the evaluation frameworks appropriate to such systems, and the technical requirements for integrating a trained model into an operational web service. This synthesis is organised around the four research objectives of the study and culminates in a summary table mapping each cited work to its methods, findings, and gaps.

2.2 Background to Credit Scoring

Credit scoring arose as a systematic response to the limitations of manual lending decisions, which relied on credit officers' subjective judgement. While flexible, such decisions were inherently inconsistent, susceptible to individual bias, and impossible to scale as loan volumes grew. By transforming measurable borrower characteristics into a single numerical risk estimate, credit scoring introduced efficiency, consistency, and a degree of auditable objectivity into the lending process.

2.2.1 From Manual Judgement to Statistical Models

Early formalised scoring systems drew on logistic regression and linear discriminant analysis, both of which were valued for their transparency and their compatibility with the interpretive requirements of financial regulators. These methods quantified the relationship between borrower attributes such as, income, employment status, repayment history and the probability of default, producing scorecards that loan officers could inspect and apply consistently. Their

adoption was rapid in mature credit markets supported by comprehensive bureau infrastructure, where the assumption of rich, complete financial histories was broadly valid.

2.2.2 Risk Management and Financial Inclusion

Applied responsibly, credit scoring simultaneously supports risk management and financial inclusion. By standardising the assessment of creditworthiness, well-designed scoring systems can extend credit to borrowers who might otherwise be declined on subjective grounds. However, the dependence of traditional models on formal bureau records imposes a structural barrier to inclusion in developing economies. In Zambia, bureau penetration remains below 30 percent of adults, and the Bank of Zambia [1] reports that 52 percent of loan applications were rejected in 2025 due to the absence of sufficient credit data. This dynamic creates a self-reinforcing cycle: borrowers excluded from formal credit cannot build the histories that would make them scoreable under conventional frameworks.

2.2.3 The Rise of Digital Finance and Alternative Data

The proliferation of mobile money and digital transaction platforms has materially expanded the range of signals available for credit assessment. In Zambia, mobile money penetration rose from 14 percent in 2015 to 58.5 percent in 2020, with transaction values exceeding ZMW 300 billion by 2023 [2]. The National Financial Inclusion Strategy II (2024–2028) [2] identifies these trends as key drivers of financial inclusion and frames the integration of alternative data into scoring frameworks as a national policy priority. This shift is not unique to Zambia: across sub-Saharan Africa, mobile money ecosystems are generating rich longitudinal records of financial behaviour for populations that hold no formal credit products, creating an empirical foundation for more inclusive scoring models.

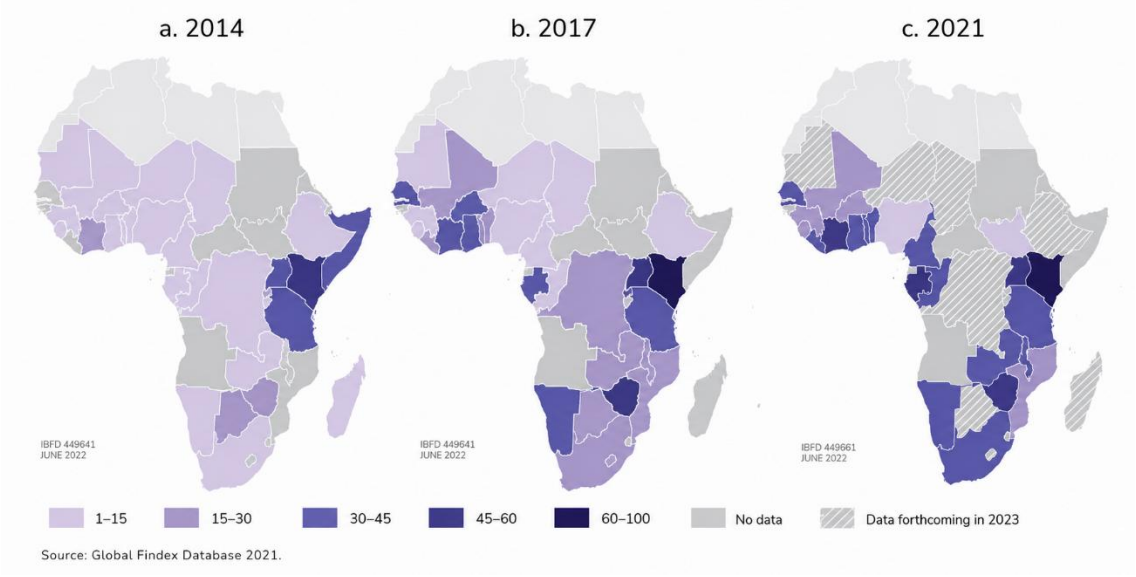


Figure 2.1: Adults with a mobile money account (%) 2014-2021 [3]

2.3 Feature Integration and Diverse Data Sources

2.3.1 Financial Inclusion and Credit Access Challenges in Zambia

The structural exclusion of large segments of the adult population from formal credit markets in sub-Saharan Africa has been extensively documented in the financial inclusion literature. Demirguc-Kunt, Klapper, Singer, Ansar, and Hess [4] draw on the Global Findex Database to show that across low- and middle-income countries, the absence of formal account ownership is the primary barrier preventing individuals from building the credit histories on which conventional scoring models depend. In Zambia specifically, the Bank of Zambia [1] reports that 52 percent of loan applications were rejected in 2025 due to insufficient credit data, a figure that reflects both the thinness of bureau coverage, estimated at below 20 to 30 percent of adults by the World Bank [3] and the structural mismatch between traditional scoring inputs and the financial realities of the majority population.

The FinScope Zambia surveys provide the most granular longitudinal evidence of these dynamics. The Bank of Zambia's National Financial Inclusion Strategy II [2] reports that overall inclusion rose from 59.3 percent in 2015 to 69.4 percent in 2025, with formal inclusion increasing from 38.2 to 61.3 percent over the same period. However, informal inclusion declined from 37 to 32 percent, indicating that while more individuals are entering formal financial systems, a significant segment remains entirely absent from bureau records.

Demirguc-Kunt et al. [4] identify similar patterns across comparable economies and argue that digital financial services, particularly mobile money, represents the most scalable mechanism for transitioning informally active individuals into scorable formal channels. Geographic and demographic disparities compound the challenge, the Bank of Zambia [2] reports urban financial inclusion at 83.8 percent against 56.9 percent in rural areas, with women (67.9 percent) and senior citizens (43.3 percent excluded) structurally disadvantaged relative to men (71.2 percent). These disparities directly motivate the need for scoring frameworks that draw on alternative data sources capable of capturing financial behaviour outside the bureau system.

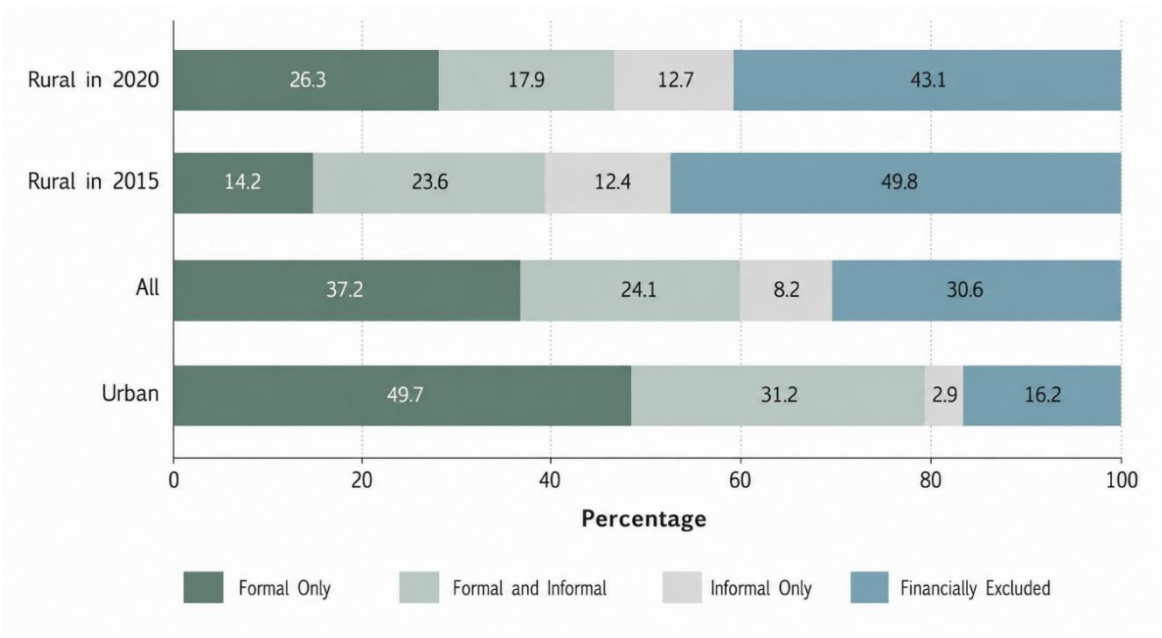


Figure 2.2: Financial Access Strand by Rural/ Urban [2]

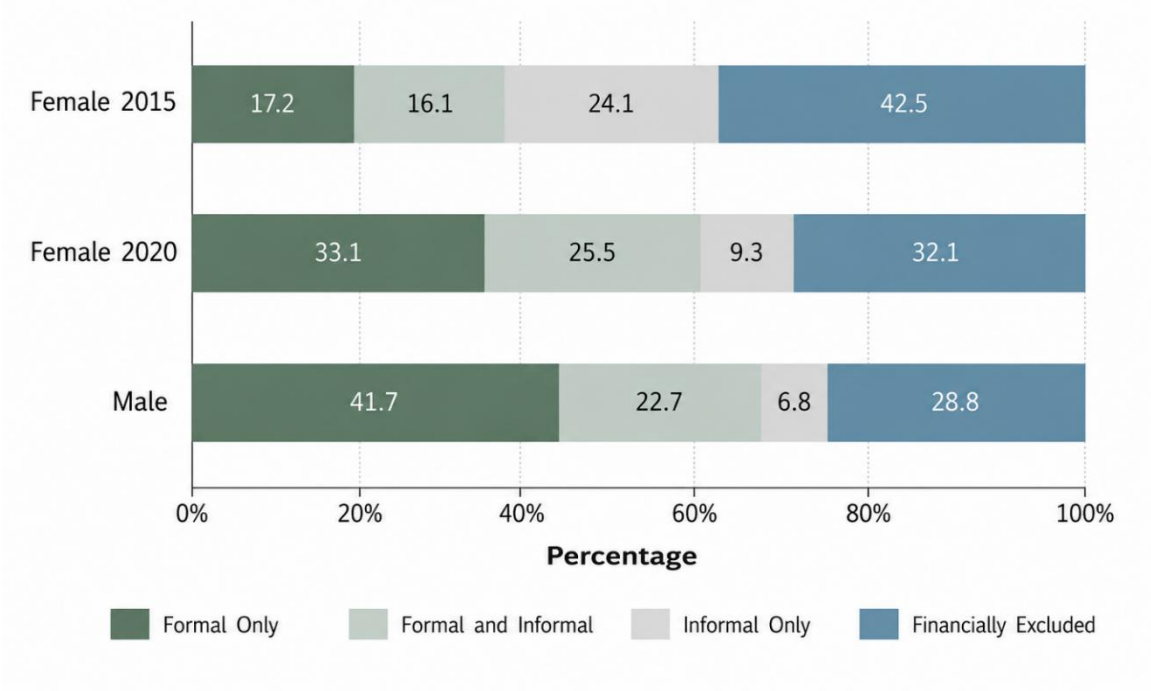


Figure 2.3: Financial Access Strand by Gender in 2020 [2]

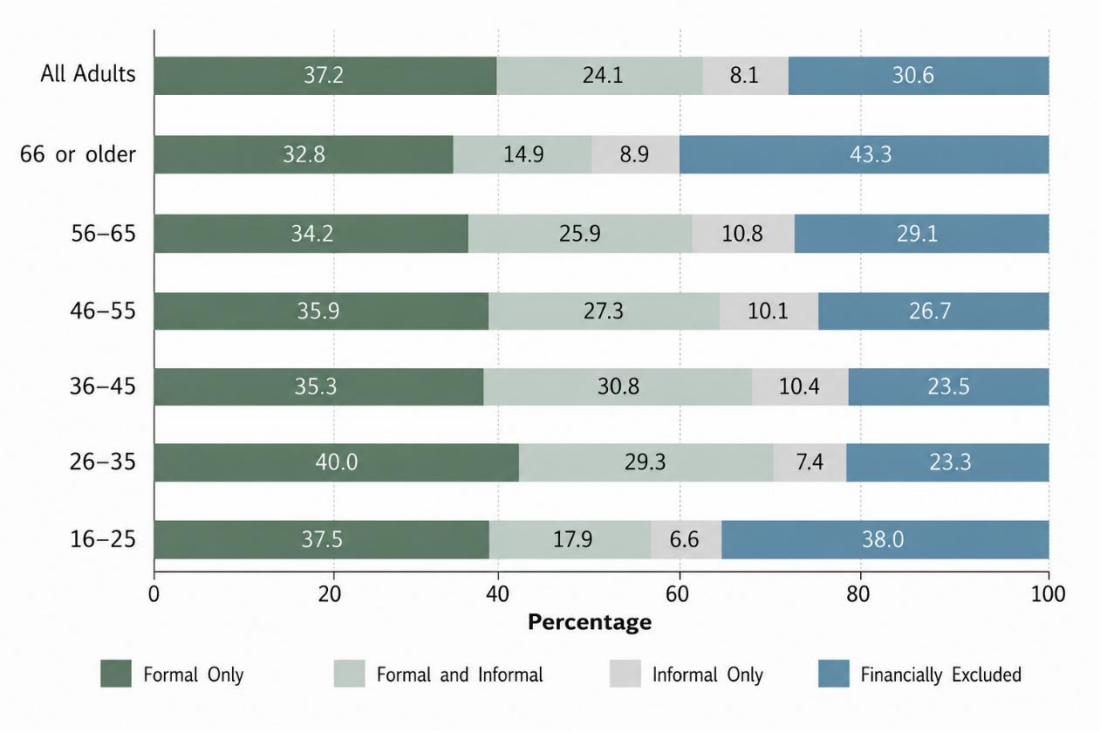


Figure 2.4: Financial Access Strand by Age(percent) in 2020 [2]

The World Bank's [3] Financial Sector Assessment for Zambia further confirms that micro, small, and medium enterprises (MSMEs) cite access to finance as their foremost obstacle, with high collateral requirements and data scarcity compounding the challenge.

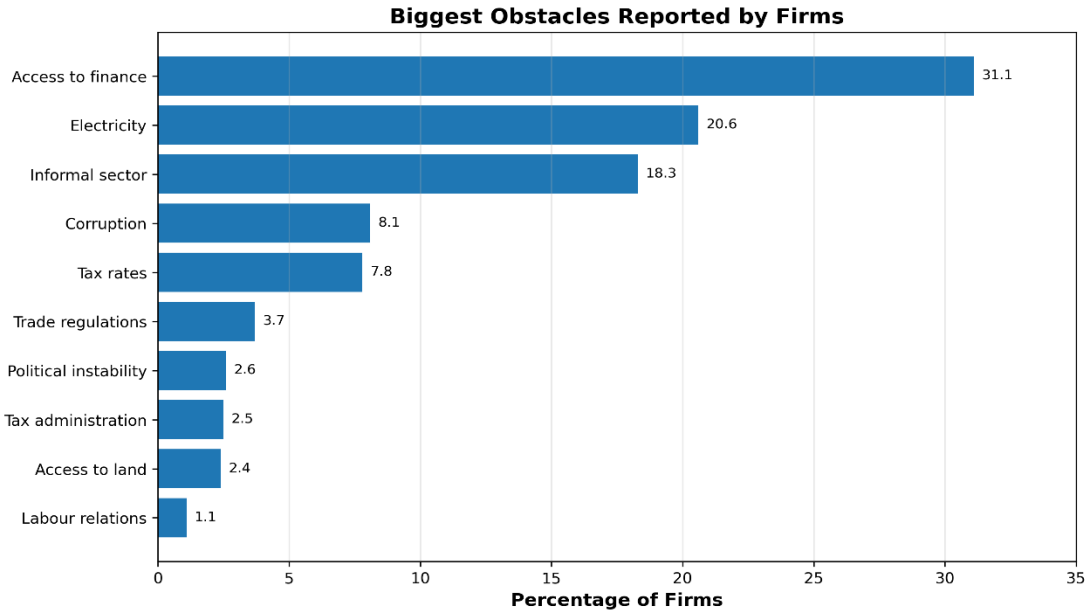


Figure 2.5: Percentage of Firms with Various Obstacles to Access to Credit [2]

2.3.2 Limitations of Traditional Credit Scoring Models

Thomas, Edelman, and Crook [5] provide the most comprehensive treatment of classical credit scoring methodology, tracing the evolution of statistical scorecards from linear discriminant models through logistic regression to survival analysis. Their foundational work establishes that while these methods are transparent and regulatorily accepted, they are fundamentally constrained by the quality and completeness of bureau data. Where bureau infrastructure is thin as it remains in Zambia and much of Southern Africa these models cannot be applied without substantial loss of population coverage. Hand and Henley [6] similarly argue that statistical classification methods in credit scoring assume a degree of historical data richness that does not obtain in emerging market contexts, and that the binary framing of most classical models obscures the continuous and heterogeneous nature of credit risk across diverse borrower populations.

Baesens, Van Gestel, Viaene, Stepanova, Suykens, and Vanthienen [7] benchmark a wide range of classical and early machine learning methods across real-world credit datasets and find that traditional logistic regression models, while competitive on well-structured bureau data, are significantly outperformed in settings that include non-linear feature interactions or missing data conditions that are pervasive in Zambian lending. The authors identify feature completeness as the dominant determinant of scorecard performance, reinforcing the case for alternative data integration as a prerequisite for effective scoring in data-scarce environments. The structural weaknesses of traditional models are therefore threefold, they assume complete financial histories, they model predominantly linear relationships between features and the probability of default, and they ignore alternative signals such as mobile money flows, insurance behaviour, and psychometric characteristics.

2.3.3 Alternative Data Sources for Credit Scoring

The literature on alternative data for credit scoring has expanded substantially over the past decade, driven by the need to score thin-file and no-file borrowers in emerging markets. Gambacorta, Huang, Qiu, and Wang [8] demonstrate, using data from a Chinese fintech firm, that machine learning methods applied to non-traditional borrower information including digital transaction records and behavioural signals improve predictive performance substantially compared to models trained on conventional bureau features alone. Their findings are particularly significant because they establish the predictive validity of alternative data even when traditional credit records are available, suggesting that the two data types are complementary rather than substitutable.

Payment transaction data has attracted the most empirical attention. Bjorkegren and Grissen [9] demonstrate, using mobile phone usage and transaction records from Rwanda and Colombia, that digital footprints are strongly predictive of loan repayment behaviour and can achieve scoring accuracy comparable to bureau-based models for previously unscored borrowers. This finding is directly relevant to Zambia, where the Bank of Zambia [2] reports that mobile money transaction values exceeded ZMW 300 billion in 2023, generating longitudinal behavioural records for millions of individuals who hold no formal credit products. Insurance data represents a further underexplored source, policyholders' premium payment regularity, claims frequency, and coverage levels are argued in the actuarial literature

to reflect attitudes toward financial risk and fiscal discipline that are plausibly correlated with creditworthiness. Investment data encompassing portfolio composition, trading frequency, and asset diversification may similarly proxy for financial sophistication and long-term wealth orientation. Psychometric data, derived from structured assessments of cognitive ability, risk attitude, and business acumen, has also been validated as a credit signal; the Alliance for Financial Inclusion [10] cites randomised evidence from South Africa showing predictive power comparable to financial statement data for small business applicants.

The integration of these heterogeneous data streams into a unified scoring framework is, however, technically non-trivial. Baesens et al. [7] observe that features drawn from different sources vary substantially in dimensionality, sampling frequency, measurement scale, and completeness, and that sophisticated preprocessing pipelines involving imputation, normalisation, categorical encoding, and temporal aggregation are required to produce a consistent feature space for model training. The current study addresses this challenge directly through its first objective: to identify and integrate relevant features from insurance, investment, payment transaction, demographic, and psychometric data into a unified credit scoring framework suited to the Zambian lending context.

2.3.4 Feature Extraction from Raw Transactional Data

Feature integration in credit scoring is not only about collecting alternative data; it also requires transforming raw records into predictive variables that a model can use. In transaction-based credit scoring, researchers commonly convert event level logs into customer-level summary features using aggregation methods such as recency, frequency, and monetary value analysis. Wang, Zhou, and Huang [11] show that the key problem in bank-card transaction credit scoring is feature construction, and they propose the Recency, Frequency, Monetary, Standard Deviation (RFMS) approach, where the standard Recency, Frequency, Monetary (RFM) framework is extended by adding standard deviation to capture transaction variability. This is important because raw transaction logs are usually too granular and inconsistent to be used directly in machine learning models, so they must first be summarised into tabular indicators that reflect spending regularity, cash-flow intensity, and behavioural stability.

Feature engineering from raw transaction logs typically involves converting timestamps, amounts, and transaction counts into structured variables that represent applicant behaviour over a fixed time window. For example, recency measures how recently a borrower has transacted, frequency measures how often transactions occur, and monetary value captures the size of those transactions. In a credit-scoring context, these features can be further expanded into variables such as average monthly inflow, cash-out volatility, transaction growth trends, and dormant-account periods, which help the model detect both activity level and financial discipline.

Researchers also emphasise that feature extraction is not a single-step operation but a pipeline that includes cleaning, temporal aggregation, transformation, and selection. Raw records may contain duplicate entries, inconsistent categories, missing timestamps, or extreme values, all of which must be handled before the data can support modelling. Feature engineering in credit scoring therefore acts as the bridge between unstructured behavioural data and the structured feature matrix required by machine learning algorithms. Aggelis [12] highlights this in the context of RFM analysis, showing that preprocessing and transformation are essential before clustering can meaningfully classify customers.

Recent studies such as Yan, Zhang and Shen [13] and Zheng and Casari [14] show that integration is usually achieved through feature engineering rather than by feeding raw alternative data directly into the model. In particular, raw transaction logs and mobile usage records are commonly converted into structured numerical variables through aggregation, where multiple observations are summarised into counts, means, totals, maxima, minima, and variance measures over a defined period. Ionescu, Diaconita and Radu [15] emphasises that scaling and standardisation are critical in credit risk, especially when variables are measured on different numerical ranges, since normalisation helps models learn from heterogeneous signals without allowing large-valued features to dominate smaller ones.

Zheng and Casari [14] and Shukla and Gupta [16] note that where data are highly dimensional or redundant, researchers apply dimensionality reduction techniques such as principal component analysis, which compresses correlated variables into a smaller set of informative components while preserving most of the variance in the original data. Systematic reviews further highlight that feature selection methods such as recursive feature elimination,

univariate selection, information value, and tree-based importance ranking are used to retain the most predictive variables while removing noisy or weak features that could reduce model performance.

Teng, Lin and Lu [17] also show that other common transformations include one-hot encoding for categorical variables, log transformation for skewed financial variables, interaction features for capturing combined effects, and recency-frequency measures for behavioural data. These techniques collectively demonstrate that the value of alternative data lies not only in its source but also in how raw observations are engineered into structured inputs that machine learning models can learn from effectively.

In this regard, alternative data such as payment transaction histories, insurance payment behaviour, investment activity, demographic attributes, and psychometric scores must be encoded through a consistent preprocessing workflow before they can be integrated into a unified credit scoring framework.

2.4 Multi-Class Credit Scoring

2.4.1 Machine Learning Approaches in Credit Scoring

The application of machine learning to credit scoring has expanded significantly over the past two decades, driven by demonstrated performance improvements over statistical baselines. Khandani, Kim, and Lo [18] show that consumer credit risk prediction can be substantially improved by machine learning methods, particularly where borrower behaviour is not adequately captured by the linear assumptions of logistic regression. Dastile and Celik [19] further demonstrate that ensemble models such as Random Forest and XGBoost outperform logistic regression in predictive accuracy across multiple benchmark datasets, and show that explainability techniques including SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-agnostic Explanations) significantly enhance the interpretability of complex credit scoring models, particularly those based on deep learning architectures.

Nerella et al. [20] provide a comparative evaluation showing that ensemble models, specifically Random Forest and Gradient Boosting Machines achieve 87.6 percent accuracy compared to 72.3 percent for logistic regression on FinTech lending datasets. The authors

argue that in environments where data complexity is high and decision speed is commercially important, machine learning models are increasingly preferred over classical scorecards. Taken together, these studies indicate that traditional scorecard methods are increasingly inadequate for credit environments characterised by heterogeneous borrower profiles and incomplete financial histories, the precise conditions that obtain in Zambia.

2.4.2 Multi-Class Classification in Credit Risk

The predominant framing of credit scoring as a binary classification problem, that is, a borrower either defaults or does not, limits the actionability of model outputs for lending institutions. In practice, lenders require granular risk stratification to support portfolio management, risk-based pricing, and targeted collections strategies. A borrower assessed as medium risk warrants different treatment from one assessed as very high risk, even if both would be declined under a binary threshold. Song, Wang, Ye, Zaretzki, and Liu [21] present a rating-specific and multi-objective ensemble learning scheme for loan default prediction and demonstrate that risk-sensitive modelling can improve performance under severe class imbalance. Their study is significant because it moves beyond the default-versus-non-default formulation and recognises that loan applications may belong to qualitatively different risk strata requiring distinct modelling strategies.

Umagba, Abara, Isa, Okoro, and Yahaya [22] propose a multi-class machine learning model for predicting credit default and show that unbalanced multi-classification can be handled effectively when the model is carefully designed and the evaluation framework accounts for class-level performance. These findings collectively support the argument that binary classification is too coarse an instrument for institutions that manage diverse borrower portfolios and require nuanced risk differentiation to operate responsibly.

2.4.3 Class Imbalance and Structural Challenges in African Contexts

Although multi-class machine learning models are promising, their deployment in African credit markets is constrained by several practical challenges. The most significant is class imbalance, default cases are substantially fewer than performing loans, making minority risk classes difficult to learn without corrective intervention. Anis and Ali [23] demonstrate that SMOTE-based oversampling can improve the performance of classifiers trained on

imbalanced financial datasets, though the authors note that the technique's effectiveness varies across algorithms and dataset characteristics. Extensions of SMOTE to multi-class settings, operating through one-versus-rest decompositions, have been proposed but not systematically validated in credit scoring applications.

Interpretability constitutes a second structural challenge. Dastile and Celik [19] demonstrate that although XGBoost and Random Forest perform well on predictive metrics, explainability methods such as SHAP and LIME are essential to make resulting credit decisions transparent to borrowers and regulators alike. Schmitt [24] similarly emphasises that SHAP-based interpretation strengthens institutional trust in AI-driven credit decisions, an observation of particular relevance in Zambia where financial institutions are expanding into previously unserved markets and must build borrower confidence in algorithmic outputs.

2.4.4 Justification for a Multi-Class Approach in Zambia

The Zambian credit environment calls for a model that captures borrower diversity more effectively than a binary classifier. The Bank of Zambia [1] continues to monitor credit market conditions, while the FinScope Zambia 2025 survey [25] reports that financial inclusion has risen to 80.1 percent. However, broader inclusion does not automatically imply that borrowers have rich formal credit records; lenders therefore still require nuanced methods of evaluating risk. Mwela [26] notes that digital and alternative financial services can expand access, but converting access into usable credit information for underwriting remains the central challenge.

In this context, a four-class risk system comprising low, medium, high, and very high risk is more appropriate for Zambia because it supports differentiated lending decisions for the heterogeneous borrower population, including salaried employees, informal traders, first-time borrowers, and SMEs. Such a model offers a practical mechanism to improve portfolio control, extend credit more responsibly, and support financial inclusion objectives without relying exclusively on a binary default label. However, the risk bands in this study are not only multiple classes but also ordered categories, which makes ordinal classification an important extension to the discussion**.**

2.4.5 Ordinal Classification in Credit Risk

The risk categories used in this study, namely low, medium, high, and very high, are not merely separate labels; they are ordered outcomes, which makes the problem one of ordinal classification rather than ordinary multiclass classification. In ordinal problems, the relative distance between classes matters because misclassifying a very high-risk borrower as low risk is more serious than misclassifying the same borrower as high risk. This distinction is important in credit scoring because lending decisions are inherently ranked, and the model should respect the natural progression of borrower risk rather than treating every class as unrelated.

Ordinal regression has long been used in scoring contexts where the dependent variable has a meaningful order. The present study can be justified as an ordinal credit classification problem because the target labels represent increasing levels of lending risk rather than independent categories. Hirk, Hornik, and Gür [27] demonstrate this approach in their analysis of corporate credit ratings using multivariate ordinal regression.

Desprez, Zawada, and Ramp [28] emphasise that ordinal imbalanced datasets require methods that preserve ordering information, since ignoring class order can substantially impact predictive performance. The literature on ordinal classification also shows that standard multiclass methods are often insufficient when class order is ignored. Because credit risk classes are ordered, models that preserve rank information are more appropriate than models that only optimise for exact label prediction. This is particularly relevant in imbalanced datasets, where the majority class can dominate performance if the order structure is not explicitly modelled.

2.5 Model Evaluation and Decision Support

2.5.1 Evaluation Metrics in Credit Scoring

Evaluating machine learning models in credit scoring requires metrics that reflect the asymmetric cost of classification errors. Misclassifying a defaulter as low risk is far more costly to a lender than incorrectly declining a creditworthy applicant, which means that global accuracy alone is an inadequate performance indicator. Paz et al. [29] note that precision,

recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC) are the most widely reported metrics in credit risk literature because they provide clearer insight into model discrimination and class-level performance than accuracy. This is especially important in imbalanced datasets, where a model that predicts the majority class exclusively may achieve high accuracy while entirely failing to identify risky borrowers.

For multi-class credit risk systems operating across low, medium, high, and very high-risk bands, class-specific confusion matrices and macro-averaged F1-scores are more appropriate than binary evaluation alone. Paz et al. [29] confirm that machine learning and deep learning models are commonly evaluated using this combination of metrics to capture both predictive strength and robustness across classes. Osei, Mpinda, Sadefo-Kamdem, and Fadugba [30] further argue that performance should be assessed using both discrimination metrics and class-sensitive indicators, particularly in imbalanced datasets where a model that performs well on the majority class may nonetheless expose lenders to significant losses by misclassifying high-risk borrowers.

For ordinal credit-risk systems, evaluation should also account for the ordered nature of the target labels. Amigó, Gonzalo, Mizzaro, and Carrillo-de-Albornoz [31] say that standard metrics can ignore the distance between classes, which is a problem when low, medium, high, and very high are risk ranks rather than unrelated labels. Galdran [32] say that distance-sensitive measures such as quadratic-weighted kappa (QWK), expected cost, and ranked probability score are more suitable for ordinal classifiers because they penalize predictions more heavily when they are further from the true class. In the same line, Silva, Pinto, and Cardoso [33] say that ordinal models with imbalanced classes need performance indices that reflect both ranking and skewed class distributions.

This is especially relevant in credit scoring because misclassifying a high-risk borrower as low risk has a much greater business cost than making a small error between adjacent risk bands. Brown and Mues [34] say that imbalance-aware metrics are necessary in credit scoring because accuracy alone can hide poor minority-class detection, while the work of Amigó et al [31]. and Galdran [32]. shows that ordinal-sensitive measures help capture whether a model respects the risk hierarchy. For that reason, the present study should combine standard discrimination metrics with ordinal and imbalance-sensitive metrics rather than relying on accuracy alone

2.5.2 Automated Decision Support and Market Validation

Automated decision support has become a defining feature of modern credit scoring systems because it materially improves the speed and consistency of lending decisions. By reducing manual underwriting steps, automated models lower operational costs and enable institutions to reach borrowers who would otherwise remain excluded through the delays and inconsistencies of human review. The International Finance Corporation [35] reports that alternative data combined with AI can expand financial inclusion by improving the assessment of thin-file borrowers who lack conventional credit records.

This approach is of particular relevance in emerging markets where many applicants lack bureau histories. The Alliance for Financial Inclusion [10] notes that mobile usage records, transaction histories, and digital footprints are increasingly incorporated into credit scoring in developing economies. Gambacorta et al. [8] corroborate this finding, demonstrating that machine learning methods incorporating non-traditional borrower information improve predictive performance in precisely the settings where conventional data is most limited. Local evidence from Zambia further substantiates this direction, Chamangwa [36] found that automation improves loan recovery rates and strengthens lending processes in Zambian commercial banking, suggesting that automated decision systems are not only technically feasible but also directly aligned with the operational realities of the local market.

2.5.3 Justification for a Multi-Metric Evaluation Pipeline

The use of a multi-metric evaluation pipeline is justified by the direct financial and social consequences of credit decisions. A model that performs well on the majority class may still expose a lender to material losses if it systematically fails to identify high-risk borrowers. The World Bank [3] and the International Finance Corporation [35] jointly emphasise that credit scoring becomes more effective and more defensible when supported by transparent, multi-dimensional evaluation and the responsible use of alternative data. Model evaluation in this study is therefore conceived not only as a statistical exercise but as a business safeguard that helps manage default risk, support growth, and extend credit responsibly to underserved borrowers.

2.6 System Integration via Web Service

2.6.1 Web Service Integration as a Model Serving Infrastructure

System integration is a critical step in transforming an offline trained credit scoring model into a usable, production-grade decision support tool. In modern deployments, the trained model is wrapped inside a web service layer so that inference can be accessed through secure APIs from external applications. This architecture enables real-time predictions while maintaining a clean separation between the user interface, business logic, and scoring computation. Adekunle et al. [37] demonstrated this approach by deploying Python-based credit risk models through microservices and REST APIs to achieve scalable, real-time credit assessment in an embedded lending system, establishing a directly applicable precedent for the current study.

Gajwani [38] explains that traditional batch-based credit systems suffered from latency and reduced responsiveness, whereas microservices and API-based integration enable real-time data processing and improved scalability. Abdulsatar, Ahmad, Goel, and Ullah [39] corroborate this finding, demonstrating that microservice architectures enhance responsiveness compared to monolithic systems, though they require careful cybersecurity risk assessment. In practice, the web service also acts as a feature alignment layer, transforming incoming borrower data into the same structure used during model training, thereby preventing schema mismatches that could degrade prediction reliability. Machado, Chen, and Osterrieder [40] note that preprocessing and integration steps of precisely this kind are essential to align heterogeneous borrower data drawn from bureau records, mobile transactions, and utility histories into a uniform schema for reliable credit risk assessment.

The modular, layered architecture adopted in this study draws direct conceptual precedent from Mukuka and Mwape [41], whose semi-automated radiology reporting system at the University of Zambia deployed an AI inference module as a separate Python service communicating with a React.js front-end through a RESTful Express.js API. The separation of concerns achieved in that design with the AI component independently updateable without disrupting the broader platform, is an architectural principle directly applicable to the integration of a credit scoring model through existing web services.

2.6.2 Data Governance and Regulatory Challenges in Integration

Deploying machine learning credit systems introduces legal and operational challenges that must be addressed at the integration layer. Data governance is a primary concern, especially where personal financial data is collected from alternative sources such as mobile money platforms and utility providers. In Zambia, the Data Protection Act No. 3 of 2021 [42] provides the principal legal framework for lawful and transparent data processing, including rules governing cross-border data transfers. Compliance with this legislation is therefore central to any integration that aggregates borrower data from multiple sources for scoring purposes.

Interoperability represents a second challenge. Financial data ecosystems in emerging markets frequently depend on fragmented, heterogeneous sources, making real-time ingestion and schema alignment technically demanding. The Alliance for Financial Inclusion [10] notes that while alternative data can improve credit access, it requires structured ingestion pipelines and robust governance frameworks. The World Bank [3] further emphasises that credit scoring is only effective when data sources are trustworthy and consistently formatted, a standard that places significant demands on the preprocessing and validation components of the integration layer.

Explainability is also a critical requirement at the deployment stage. Lenders are increasingly required to justify adverse credit decisions to applicants and regulators in transparent, human-interpretable terms. A deployed system must therefore provide not only a risk classification but also interpretable feature contributions, most commonly via SHAP values, that can support audit and regulatory review. Finally, model drift poses an ongoing operational risk: credit behaviour changes over time in response to inflation, macroeconomic shocks, and shifts in borrower composition. Continuous logging, monitoring, and periodic retraining are required to maintain model relevance once deployed.

2.6.3 Integration Framework for the Zambian Credit Scoring Context

In Zambia, system integration must align with national financial inclusion goals. The National Financial Inclusion Strategy II (2024–2028) [2] identifies MSME credit scoring as a policy priority, which justifies integrating scoring models through service layers designed to evaluate underserved borrowers quickly, consistently, and at scale. The World Bank [3] and the

International Finance Corporation [35] note that non-traditional data strengthens inclusion by improving risk assessment where bureau histories are absent, a function that the web service layer can operationalise by acting as middleware, connecting mobile money records, investment data, and psychometric signals to the scoring model.

Modular deployment structures are increasingly recognised as the most sustainable approach for machine learning platforms in production. Abdulsatar et al. [39] demonstrate that microservice-style integration improves maintainability and resilience, allowing updates to preprocessing logic, model versions, and monitoring configurations without disrupting the user-facing platform. In this way, the integration layer becomes the operational mechanism that links prediction, compliance, monitoring, and borrower experience into a coherent system.

2.6.4 Justification for Integration with Existing Web Service

Using an existing web service as the deployment layer provides both technical and practical advantages. Technically, it allows the trained credit scoring model to be exposed through a secure, versioned API without requiring a redesign of the underlying platform architecture [17, 19]. Centralised preprocessing within the service layer ensures that production features remain consistent with those used during training, reducing the risk of prediction errors arising from schema drift [40]. Practically, the service layer supports regulatory and audit requirements by enabling detailed logging, SHAP-based explanation outputs, and consent-aware data processing, consistent with Zambia's Data Protection Act [42] and the inclusion objectives of the National Financial Inclusion Strategy II [2]. It also improves long-term sustainability: model updates and retraining cycles can be managed within the service layer without disrupting the borrower-facing application.

2.6.5 Feature Alignment and Real-Time Preprocessing

Feature alignment is a necessary extension of web service integration because the model-serving layer must reproduce the same feature transformations used during training. In this study, that means the API should not merely receive a raw JSON payload; rather, according to Zhou, Aryal and Bouadjenek [43] it should validate the incoming fields, map them to the correct feature names, and apply the same preprocessing rules before inference is made. This

is important because if the features served online do not match the training schema, the model may produce unstable or misleading outputs.

Missing-value handling is part of that alignment process. Xu et al [44] Real-time payloads often contain absent, null, or malformed fields, and the API layer must determine whether to impute, reject, or replace those values with predefined defaults before passing the data to the model. For numeric inputs, missing values may be replaced by training-set statistics such as the mean or median, while categorical variables may be assigned an unknown class or fallback category. These procedures are essential because live borrower requests rarely arrive in the same clean state as the training data.

Modak and Avula [45] explain that feature stores support this process by separating offline feature generation from online feature retrieval. In the feature-store literature, updates are coordinated so that the online store can serve the most recent valid feature values to the API layer during prediction.

Hutagaol, Sitorus, and Simanjuntak [46] emphasize that in practical terms, when a JSON payload reaches the API gateway, the service must first verify the structure, then align the request with the expected schema, and finally apply preprocessing before the request is passed to the ML model.

Horchidan, Kritharakis, Kalavri, and Carbone [47] show that preprocessing can be embedded directly in the serving stack so that transformation occurs at prediction time rather than being pushed to the client application. This approach reduces schema mismatch and improves consistency between offline training and online prediction, especially where inputs come from heterogeneous sources such as mobile transactions, demographic forms, and external financial records.

Derakhshan, Mahdiraji, Rabl, and Markl [48] argue that the deployment layer should therefore be understood as a controlled preprocessing and feature-alignment layer that receives incoming borrower data, handles missing or inconsistent values, aligns the request with the model's feature schema, and only then forwards the processed record to the classifier.

2.7 Summary of Literature and Gaps in the Study

Table 2.1 below consolidates the literature reviewed in this chapter, mapping each work to the methods employed, the key findings reported, and the gaps identified that the current study is positioned to address.

Table 2.1: Summary of Literature and Gaps in the Study

Authors/ Ref	Year	Title / Focus	Key Findings	Methods	Gaps Identified
Republic of Zambia. [42]	2021	Data Protection Act No. 3 of 2021	Provides a legal framework for lawful data processing, including cross-border transfers	Legislative instrument	Legal framework only; no technical guidance on ML compliance implementation
Song et al. [21]	2023	Multi-Objective Ensemble for Loan Default Prediction	Risk-sensitive ensemble modelling improves performance under severe class imbalance	Multi-objective ensemble learning	Does not address ordinal risk bands or integrate diverse alternative data sources
Umagba et al. [22]	2022	Multi-Class ML for Credit Default Prediction	Multi-class formulation is feasible and effective when model design accounts for imbalance	Multi-class machine learning classifier	No Zambian context; no system integration or web service deployment addressed
World Bank. [3]	2022	Financial Sector Assessment: Zambia	Bureau coverage below 20–30% of adults; MSMEs cite finance access as top obstacle	Sector assessment and financial indicator analysis	No ML-based scoring solution; no integration of psychometric or insurance features
Mukuka & Mwape. [41]	2025	Semi-Automated Cervical	Layered API architecture with a	ReactJS front-end, Express.js	Medical domain; not directly a credit

		Cancer Radiology Reporting	separate AI service module supports modular, maintainable integration	API, Python AI module	scoring study, but the integration architecture is directly analogous
Authors/ Ref	**Year**	**Title / Focus**	**Key Findings**	**Methods**	**Gaps Identified**
Dastile & Celik. [19]	2021	Deep Learning for Explainable Credit Scoring	XGBoost and Random Forest outperform logistic regression; SHAP and LIME improve interpretability	Ensemble models with XAI post-hoc explanation	No multi-class formulation; no deployment or API integration discussed
Gambacorta et al. [8]	2019	ML and Non-Traditional Data in Credit Scoring	Non-traditional data improves prediction where conventional records are limited	Machine learning on Chinese fintech data	Focused on a single market; not adapted for sub-Saharan or multi-class contexts
Anis & Ali. [23]	2017	SMOTE for Class Imbalanced Credit Scoring	SMOTE-based oversampling significantly improves minority-class performance	SMOTE resampling on credit datasets	Binary datasets only; multi-class SMOTE extensions not evaluated
Paz et al. [29]	2025	ML and Metaheuristics for Credit Risk Assessment	Precision, recall, F1 and AUC-ROC are the standard multi-class evaluation suite	Systematic literature review of ML credit models	Review paper; no empirical model built; no alternative data or Zambian application
Adekunle et al. [37]	2021	Python and Microservices	REST API microservice	Microservices and Python-	No multi-class output; no

		for Real-Time Credit Assessment	architecture enables scalable real-time credit risk deployment	based REST API deployment	Zambian regulatory or data governance discussion

The literature reviewed reveals three significant gaps that the current study is positioned to address. First, while individual alternative data sources have been investigated in isolation, no published work has constructed a unified feature framework integrating all five modalities targeted in this study: insurance, investment, payment transactions, demographic, and psychometric data. Second, the overwhelming majority of credit scoring literature frames the problem as binary classification; there remains a gap for rigorous, empirically validated multi-class approaches that provide lenders

with actionable, granular risk stratification calibrated to the Zambian context. Third, while API-based model deployment is standard practice in software engineering, the specific challenges of integrating a multi-class credit scoring model into an existing fintech web service, with the attendant requirements for data governance, explainability, and regulatory compliance under Zambian law, have received limited systematic treatment. The current study seeks to address all three gaps through its design, implementation, and evaluation objectives.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter presents the methodology adopted in the design, development, and evaluation of the proposed machine learning-based credit scoring system for Hobbiton. It describes the research design, the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework, the Scrum development process, the PostgreSQL data extraction pipeline, the data preparation procedures, the modelling approach, the evaluation methods, the deployment strategy, and the ethical considerations that guided the study. The chapter explains how structured borrower data was converted into a predictive credit scoring model capable of supporting lending decisions.

The study aimed to develop a credit scoring model that predicts the risk level of loan applicants using both conventional financial information and alternative data sources. The target variable was treated as a multiclass classification problem with four categories: low risk, medium risk, high risk, and very high risk. This classification structure was selected to provide a more detailed risk profile than a simple good-or-bad credit outcome.

3.2 Research Design

This study adopted a Design Science Research (DSR) approach supported by quantitative machine learning techniques. Design Science Research was suitable because the study focused on the development and evaluation of a practical artefact, namely a machine learning-based credit scoring system intended to address a real lending problem. The quantitative component involved analysing historical customer data stored in Hobbiton's PostgreSQL database in order to identify patterns associated with repayment behaviour and borrower risk.

The study consisted of two related activities. The first was the development of a machine learning credit scoring model capable of classifying loan applicants by risk level. The second was the integration of this model into Hobbiton's existing web service, so that risk predictions could support real lending decisions made by loan officers.

3.3 Risk Analysis

Before adopting the frameworks and procedures described in the remainder of this chapter, the study identified the principal risks associated with developing a credit scoring model on top of an existing production platform, and defined mitigation strategies for each. Identifying these risks at the design stage was necessary because the project did not involve building a database or web service from scratch; rather, it involved extracting data from, and ultimately integrating predictions back into, infrastructure that Hobbiton had already built and continued to operate independently of this study.

3.3.1 Data Quality and Availability Risk

Because Hobbiton's PostgreSQL database was designed to support day-to-day lending operations rather than machine learning research, there was a risk that key variables conceptually desirable for credit scoring, such as, employment history, education, or socio-economic indicators would be missing, incomplete, or stored in a form not directly usable for modelling. This risk was mitigated by conducting the structured feature audit described in Section 3.7, which verified the availability of every candidate feature against the actual schema before any modelling work began, rather than assuming a feature existed because it appeared in a generic credit-scoring feature list. Where a feature was found to be partially available or missing entirely, this was documented and treated as a constraint on the model rather than allowed to stall the project.

3.3.2 Asymmetric Misclassification Risk

Credit scoring is a high-stakes prediction task in which the cost of different errors is not equal. Classifying a high-risk borrower as low risk (a false negative on default risk) exposes Hobbiton to direct financial loss through default, whereas classifying a low-risk borrower as high risk (a false positive) results in a missed lending opportunity and potential harm to financial inclusion. This asymmetry was treated as a risk to the validity of the evaluation approach: a model could appear accurate overall while systematically underperforming on the minority high-risk classes that matter most to the business. This risk was mitigated by selecting evaluation metrics, described in Section 3.11, that explicitly account for per-class performance rather than relying

on overall accuracy alone, and by setting success criteria around discrimination power and minority-class precision-recall performance from the outset, in Section 3.6.

3.3.3 System Integration Risk

Because the credit scoring model was intended to be integrated into Hobbiton's existing, live web service rather than deployed as a standalone application, there was a risk that integration work could disrupt existing functionality or introduce failure points into a system already in production use. This risk was mitigated by scoping the model's integration footprint narrowly, as a discrete scoring component that the existing backend could call and receive a structured response from, rather than requiring changes to existing application logic. This integration approach is described further in Section 3.14.1.

3.3.4 Data Drift and Model Staleness Risk

Borrower behaviour and the composition of Hobbiton's applicant population were expected to change over time, meaning that a model trained on a single historical extract could degrade in accuracy as new data patterns emerged. This risk was acknowledged at the design stage and addressed through the deployment strategy described in Section 3.12, which allowed for future monitoring of model performance and periodic retraining using newer data from PostgreSQL.

3.3.5 Privacy and Ethical Risk

Because the study involved analysing real customer records held by Hobbiton, there was a risk of mishandling personally identifiable or sensitive information during extraction, processing, or storage. This risk was mitigated through the access controls, anonymisation practices, and data-handling principles described in Section 3.16.

3.4 Methodological Framework

The study used two complementary frameworks. CRISP-DM was used to guide the machine learning lifecycle, while Scrum was used to manage system development. CRISP-DM structured the analytical process into business understanding, data understanding, data preparation, modelling, evaluation, and deployment. Scrum, on the other hand, organised

implementation work into two-week iterations that allowed continuous refinement of the database pipeline, feature engineering logic, and model integration.

This combination was appropriate because the project involved both data science and software engineering tasks. The machine learning workflow required systematic data preparation and experimentation, while the development workflow required regular reviews, stakeholder feedback, and iterative refinement. Together, the two frameworks supported a disciplined and practical approach to implementation.

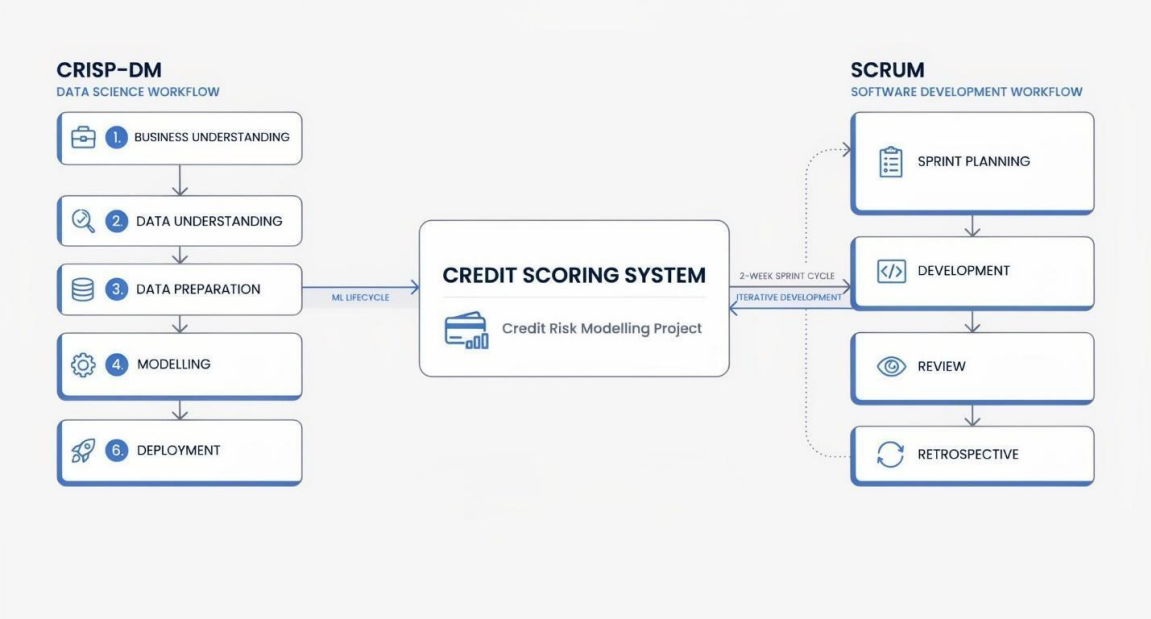


Figure 3.1: CRISP-DM and Scrum relationship in the project

3.5 CRISP-DM Framework

CRISP-DM was adopted as the principal framework for developing the credit scoring model. It was selected because it provides a structured, iterative, and widely recognised approach to data mining and machine learning projects. The framework consists of six stages: business understanding, data understanding, data preparation, modelling, evaluation, and deployment.

The process was applied iteratively, allowing results from one stage to inform improvements in earlier stages. This was especially important in the present study because the performance of the model depended heavily on the quality of the data extracted from PostgreSQL and the effectiveness of the derived features.

3.6 Business Understanding

The business understanding phase focused on defining the problem and clarifying the objectives of the project. Hobbiton required a reliable way of assessing the creditworthiness of loan applicants. Traditional credit assessment approaches often depend on formal credit histories and financial records, yet many applicants have limited or no formal credit history. This creates difficulty in determining borrower risk using conventional methods alone.

To address this challenge, the proposed system was designed to incorporate both conventional financial data and alternative data sources to build a more comprehensive credit scoring mechanism. The main business objectives were to improve credit risk assessment, reduce loan default rates, enhance financial inclusion, support loan officers with data-driven decisions, and automate parts of the loan evaluation process.

The project was considered successful if the model demonstrated acceptable predictive performance and effectively distinguished between borrowers in different risk categories. At this stage, the study defined the problem context, expected benefits, and practical requirements of the solution. Success criteria were further expressed in terms of machine learning evaluation metrics: achieving high discrimination power (an AUC-ROC score above 0.70), balanced accuracy across classes, and strong precision-recall performance for minority risk groups. These thresholds were intended to ensure that the model was not only statistically sound but also business-relevant, reflecting the asymmetric costs of misclassification in credit scoring, as identified in the risk analysis in Section 3.3.2.

Figure 3.2: Business problem and project objectives

3.7 Data Understanding

The data understanding phase involved systematically reviewing the customer information available within Hobbiton's PostgreSQL database in order to establish what data could realistically support a credit scoring model. Rather than assuming that all conceptually desirable credit-scoring variables existed in the source system, this stage was treated as an empirical audit: each candidate feature was checked against the actual database schema before any feature engineering or modelling work began. This approach was necessary to ensure that the eventual feature set was grounded in verified data availability rather than in an idealised feature list.

3.7.1 Database Review Procedure

An initial database review was conducted to identify the tables, relationships, data types, and attributes relevant to credit risk assessment. The database, comprising 73 tables in total, was reviewed systematically by examining table and column names, primary and foreign key relationships, and sample records, in order to understand how borrower-related information was organised across the schema.

This review made it possible to distinguish tables representing the applicant population from those representing internal staff, merchants, or platform settlement processes. For the purpose of this study, the primary borrower population was defined as the client records held in the relevant applicant table, while internal staff tables were excluded from the modelling population. Merchant-related tables were treated as transaction sources rather than applicant-profile tables, since they record business activity occurring on behalf of clients rather than independent borrower attributes.

3.7.2 Feature Audit Approach

A feature-to-table mapping exercise was conducted to assess whether each candidate credit-scoring feature, drawn from an initial conceptual feature list spanning demographic, financial, employment, business, agricultural, and mobile-usage data, could be located within the PostgreSQL schema. Each feature was classified into one of three categories:

- Available: the field existed in the schema and could be used directly, or with minimal processing such as a join on a known key.
- Partial: a related or approximate field existed but required further interpretation, joining across tables, or disambiguation before it could be used.
- Not Found: no matching field existed anywhere in the 73-table schema.

This classification scheme was applied consistently across six feature domains identified in the conceptual feature list: personal and demographic data, education and professional data, employment history, financial and loan data, mobile and alternative data, and business and agricultural data. Each feature was checked against the schema individually, and where a candidate match was found, the relevant table and column were recorded together with any joins required to resolve it to the borrower level. This audit procedure directly addresses the data quality and availability risk identified in Section 3.3.1.

3.7.3 Document Data Review

In addition to the structured feature audit, the schema review also examined document-related tables, including client documents, loan documents, and guarantor document tables, to assess whether document presence could be treated as a usable feature. This was reviewed as part of

the same audit procedure described above, and the specific outcome of this review is likewise reported in Chapter 6.

3.7.4 Target Variable Derivation

The target variable for model training was derived from historical repayment records held in the PostgreSQL database, rather than from self-reported borrower attributes. This approach was adopted to anchor the model to verify borrower performance and to make the credit scoring process more evidence-based and operationally relevant. Depending on the quality and completeness of the repayment records available, the target was defined either as a four-category multiclass risk label (low, medium, high, very high risk) or, where necessary, as a simplified binary good-risk versus bad-risk outcome. The final target definition adopted for modelling, together with the supporting class distribution, is reported in Chapter 6.

3.8 PostgreSQL Data Pipeline

The PostgreSQL database served as the source of borrower records for the study. Hobbiton provided authorised access to the database, and relevant variables were extracted using SQL queries. The extraction process began by identifying the tables containing applicant details, transaction histories, repayment outcomes, and other features relevant to credit assessment, based on the feature audit described in Section 3.7.

The next step involved joining related tables using primary and foreign keys so that borrower-level records could be consolidated into a single analytical dataset. This was necessary because important risk indicators were spread across multiple database tables rather than stored in one place. After extraction, the records were reviewed for completeness and consistency within Google Colab before further processing

The pipeline also supported the creation of derived features from raw database fields, including repayment trends, transaction frequency, loan count, and mobile money activity measures. Such derived features were necessary because raw operational records are not always directly suitable for machine learning, but meaningful predictive variables can be constructed from them through structured transformation.

3.9 Data Preparation

Data preparation transformed the raw PostgreSQL extract into a clean, structured dataset suitable for machine learning. The first step involved data cleaning, during which duplicate records were removed, inconsistent entries were corrected, and data integrity issues were addressed. Missing values were then handled using techniques such as mean imputation, median imputation, mode substitution, predictive imputation, or record exclusion, with the choice of technique depending on the nature and distribution of the variable concerned.

Feature engineering was carried out to generate new variables expected to improve predictive performance. Candidate engineered features included customer age, transaction frequency, number of previous loans, and mobile money activity measures, derived from the fields confirmed as available during the data understanding phase. These engineered features were intended to provide a more meaningful representation of borrower behaviour than the raw database fields alone.

Feature selection techniques were used to identify the most informative variables from the resulting feature set. Methods considered included correlation analysis, information gain, chi-square testing, recursive feature elimination, and feature importance ranking. Categorical variables were encoded using label encoding or one-hot encoding as appropriate, while numerical variables were normalised or standardised where required by the modelling algorithm. The prepared dataset was then split into training and testing subsets.

3.10 Modelling

The modelling phase involved developing and training machine learning models capable of predicting borrower risk. Google Colab was used as the modelling environment, and Python libraries, including Pandas, NumPy, and scikit-learn, were used for data processing, model training, and evaluation. This environment was suitable because it supports flexible experimentation and reproducible model development.

The prepared dataset was divided into training and testing sets, with stratified sampling used to preserve the class distribution across both subsets. This was particularly important given the expected class imbalance between the four risk categories, where very high-risk borrowers

were anticipated to form a small minority of the dataset relative to low and medium-risk borrowers.

Four algorithms were evaluated, each selected for a specific, justified reason relevant to the credit scoring context rather than as an arbitrary comparison set:

- Logistic Regression was included as an interpretable baseline. Its coefficients can be directly related to individual features, which is valuable in a lending context where loan officers and, potentially, regulators may need to understand why a particular risk classification was made.
- Decision Tree was included because its rule-based structure is similarly transparent and can be visualised as an explicit decision path, offering a second interpretable point of comparison against the baseline.
- Random Forest was included to capture non-linear relationships and interactions between features, such as the combined effect of repayment history and transaction frequency, that a linear model like logistic regression cannot represent, while remaining more robust to overfitting than a single decision tree.
- XGBoost was included as a gradient-boosted ensemble method, included to establish an upper bound on achievable predictive performance on this structured, tabular dataset, against which the interpretability of the simpler models could be weighed.

This combination of models allowed the study to compare interpretable methods against more powerful ensemble approaches, and to make a deliberate, evidence-based trade-off between predictive performance and explainability rather than defaulting to the single best-performing algorithm regardless of transparency.

To ensure model generalization and robustness, the final models were evaluated using Stratified 5-Fold Cross-Validation rather than optimizing on a single train-test split. Instead of unconstrained hyperparameter tuning which often leads to overfitting on smaller sample sizes, comparable complexity constraints, specifically restricting tree models to a shallow maximum depth of 3, were standardly applied. This approach deliberately prioritized model interpretability, robustness, and suitability for deployment. Since credit scoring is a high-stakes task with direct financial consequences for applicants, this study aimed to balance predictive

performance with structural simplicity and transparency rather than optimizing for accuracy alone, aligning with the asymmetric misclassification risks identified in Section 3.3.2

3.11 Evaluation

The developed models were evaluated using unseen test data withheld from the training process. Because the target variable was multiclass, the evaluation metrics used included accuracy, precision, recall, F1-score, the confusion matrix, and ROC-AUC. These metrics were selected because, taken together, they provide a more complete view of model performance across all risk categories than accuracy alone, which can be misleading on an imbalanced multiclass target.

The choice of metrics was directly informed by the asymmetric misclassification risk identified in Section 3.3.2. Overall accuracy was deliberately not treated as the primary decision criterion, because a model could achieve high accuracy simply by performing well on the majority low-risk and medium-risk classes while failing to identify high-risk and very high-risk borrowers, who are the smaller classes but the ones whose misclassification carries the greatest financial cost to Hobbiton. For this reason, per-class precision and recall, and recall on the high-risk and very high-risk classes in particular, were treated as more decision-relevant than aggregate accuracy when comparing candidate models, in line with the success criteria defined in Section 3.6.

Cross-validation was also performed to assess the stability and generalisability of each candidate model across different subsets of the data. The best-performing model, judged against the success criteria defined in Section 3.6 and the cost-sensitive metric priorities described above, was selected for deployment. In the context of credit scoring, this evaluation step is important because a model must not only be accurate, but also dependable enough to support real lending decisions.

3.12 Deployment

The deployment phase focused on integrating the selected model into the proposed credit scoring system. The deployed system was designed to support applicant data capture,

preprocessing, credit score generation, risk categorisation, and loan decision support. The model was designed to generate a probability of default for each applicant, which would then be transformed into a numerical score and an associated risk class.

The deployment strategy also allowed for future monitoring and retraining as new borrower data became available in PostgreSQL. This was considered important because borrower behaviour and underlying data patterns may change over time, requiring periodic model updates to maintain predictive performance, consistent with the data drift risk identified in Section 3.3.4.

3.13 Scrum Development Methodology

The software component of the project was developed using Scrum. Scrum was selected because it supports iterative delivery, regular stakeholder feedback, flexibility, and continuous improvement, all of which were necessary given the exploratory nature of the database integration work. Development was organised into two-week sprints, with each sprint focusing on specific implementation tasks such as database integration, feature extraction, model implementation, and interface development.

Each sprint followed four main activities: sprint planning, sprint development, sprint review, and sprint retrospective. During sprint planning, tasks were prioritised and assigned to the sprint backlog. During sprint development, the planned tasks were implemented. During sprint review, completed work was demonstrated to stakeholders and feedback was collected. During the sprint retrospective, the team reflected on lessons learned and identified areas for improvement.

This iterative approach enabled continuous refinement of the PostgreSQL pipeline and the machine learning component, and helped ensure that the project remained aligned with the study objectives throughout development.

3.14 System Architecture

The proposed system adopted a layered architecture consisting of four main components: a presentation layer, an application layer, a data layer, and a machine learning layer. The

presentation layer provides the user interface for applicants, loan officers, and administrators. The application layer contains the business logic, authentication, API services, and workflow management functions.

The data layer stores customer information, transaction records, model outputs, and system logs within PostgreSQL. The machine learning layer is responsible for data preprocessing, feature engineering, model training, prediction, and credit score generation. This architecture was adopted because it supports modular development, maintainability, and scalability.

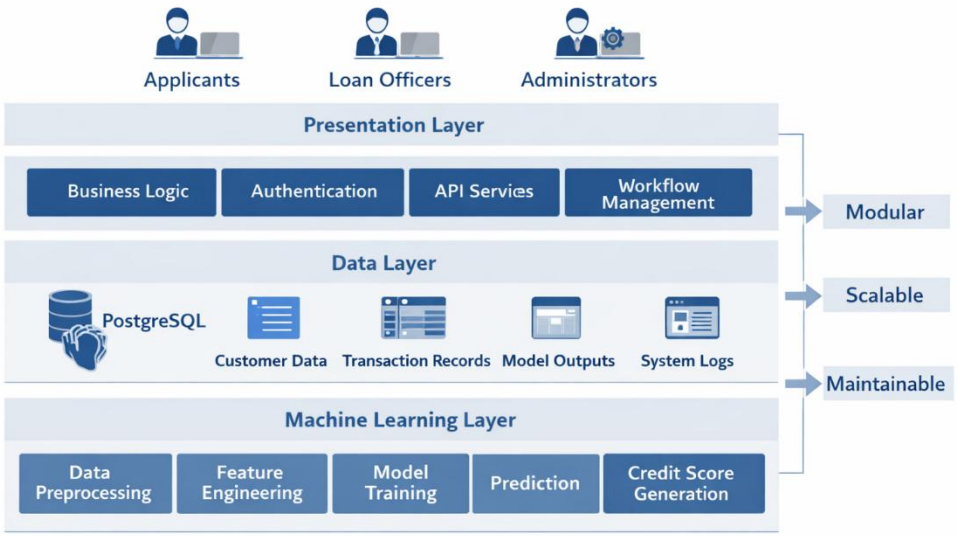


Figure 3.3: System architecture

3.14.1 Existing System Constraints and Integration Context

Unlike a system built entirely from scratch, the credit scoring model developed in this study was required to operate within the boundaries of Hobbiton's existing PostgreSQL database and web service, both of which were designed, built, and operated independently of this project. This had direct implications for the system design described above, and these implications are made explicit here rather than assumed.

For the data layer, the project treated the existing PostgreSQL schema as a fixed, read-accessed source rather than as a structure to be redesigned. Access for the purposes of this study was limited to extracting historical borrower, transaction, and repayment records for training, as

described in Section 3.8; no changes were made to Hobbiton's production schema, and no write access to operational tables was required or used.

For the application layer, the machine learning component was scoped as a discrete, self-contained scoring service rather than as a modification to Hobbiton's existing backend logic. The intended integration point was an inference interface: the existing web service would submit a borrower's relevant feature values to the scoring service and receive a structured response containing a risk score and a risk category, which the existing loan evaluation workflow could then incorporate into its decision process. This narrow integration footprint was adopted specifically to address the system integration risk identified in Section 3.3.3, by minimising the extent to which the new machine learning component could interfere with Hobbiton's existing, live application logic.

This integration approach also clarified the boundary of what this study was responsible for designing. The presentation layer, the existing application logic for loan processing, and the underlying database infrastructure were treated as given constraints inherited from Hobbiton's platform; the contribution of this study lay in the feature extraction pipeline, the trained model, and the scoring interface connecting the two to the existing system.

3.15 Tools and Technologies

The study used Python as the primary programming language for data preparation and model development. Google Colab served as the execution environment for the machine learning workflow, while scikit-learn was used for building and evaluating models. PostgreSQL was used as the relational database from which the data was extracted. Pandas and NumPy were used for data manipulation and analysis.

These tools were considered appropriate because they support end-to-end machine learning development, from database extraction through to model evaluation, while also making the workflow practical, reproducible, and suitable for a project of this scope.

3.16 Security and Ethical Considerations

Because the study involved sensitive customer information, security and privacy considerations were necessary throughout. Access to the PostgreSQL database was restricted to authorised use, and sensitive information was handled carefully during extraction and modelling. Personally identifiable information, such as names, National Registration Card (NRC) numbers, identity document details, and photographs, was not used directly as predictive features.

The study also recognised that credit scoring decisions can significantly affect borrowers. For this reason, the model was designed to support human decision-making rather than to replace it entirely. The data used in this study was applied solely for academic research and system development purposes, in accordance with data protection principles, addressing the privacy and ethical risk identified in Section 3.3.5.

3.17 Chapter Summary

This chapter has presented the methodology used in the development of the proposed machine learning-based credit scoring system. It discussed the research design, the principal risks associated with building on Hobbiton's existing platform and their mitigations, the CRISP-DM framework, the PostgreSQL data review and extraction procedures, the data preparation methods, the modelling process and the justification for each algorithm considered, the cost-sensitive evaluation approach, the Scrum development methodology, the system architecture and its integration with Hobbiton's existing infrastructure, and the ethical considerations guiding the study. Together, these methods provide the foundation for the results presented in Chapter 6 and for the credit scoring solution developed in this project.

CHAPTER 4: SYSTEM ANALYSIS AND DESIGN

4.1 Introduction

This chapter documents the analysis and design activities that shaped the implementation. It identifies the stakeholders, states the functional and non-functional requirements, models the system with UML diagrams, and describes the architecture, data design, interface design and technical design of the scoring service. The design follows the Scrum and CRISP-DM approach described in Chapter 3 and the layered architecture introduced in Section 3.14. UML notation follows the OMG specification [49].

4.2 Stakeholder Analysis

Stakeholders were identified from the Hobbiton lending context described in Section 3.14.1. Table 4.1 summarises each group and what it needs from the system.

Table 4.1: Stakeholders and their needs

Stakeholder	Role	Main needs
Loan officers	Primary users who assess applications	A fast, understandable risk category with supporting probabilities; simple data entry
Hobbiton management	Business owner of the lending product	Consistent risk grading, lower default exposure, a process that scales
Hobbiton IT and system owners	Own the database and web service	Minimal impact on production systems; a stable REST interface for integration
Model administrator	Maintains the model	Reproducible training, versioned artefacts, the ability to add features
Borrowers (indirect)	Affected by lending decisions	Fair, consistent and privacy-respecting treatment
Regulators	Oversee data protection	Compliance with the Data Protection Act [42]

4.3 Functional Requirements

The requirements in Table 4.2 were derived from the project objectives (Section 1.4), the integration constraints in Section 3.14.1 and the data that the audit showed to be available. Each is prioritised as Must or Should.

Table 4.2: Functional requirements

ID	Requirement	Priority
FR1	The system shall accept the six borrower behavioural features through a web form.	Must
FR2	The interface shall check that each input is present and numeric before submission.	Must
FR3	The web interface shall forward each request to the scoring service through the REST gateway.	Must
FR4	The scoring service shall return one of four risk categories: low, medium, high or very high.	Must
FR5	The scoring service shall return the probability assigned to each risk category.	Must
FR6	The result page shall display the predicted category and the class probabilities.	Must
FR7	The system shall report invalid requests, model failures and an unreachable scoring service with distinct error responses.	Must
FR8	The scoring service shall expose a health-check endpoint that reports the model version.	Should
FR9	The model shall be trainable from a reproducible pipeline and saved as a versioned artefact.	Should
FR10	The feature list shall be defined in one place so that new features can be added with minimal change.	Should

4.4 Non-Functional Requirements

Table 4.3: Non-functional requirements and design responses

Category	Requirement	Design response
Performance	A prediction is returned within about two seconds once the service is running.	Model loaded once at start-up; small JSON payloads.
Usability	A loan officer can complete a request in under a minute without training.	Six labelled fields, native browser validation and a colour-coded result.
Security	Data in transit is protected, inputs are validated and no personal identifiers are processed.	HTTPS on hosted platforms; server-side checks; only behavioural aggregates are used [42].
Reliability	Failures are reported rather than hidden.	Distinct 400, 500 and 502 responses; health endpoint.
Extensibility	New features and model versions should not require changes in every layer.	Single feature schema in the Flask configuration; gateway forwards JSON unchanged; versioned /v1 endpoints.
Portability	The system runs locally and on hosted platforms.	XAMPP and a Python virtual environment locally; Render and InfinityFree when hosted.
Cost	Hosting should cost nothing.	Free tiers of Render and InfinityFree.

4.5 Use Case Model

The use case model in Figure 4.1 shows two human actors and one external system. The loan officer enters borrower features, requests a score and views the result. The model administrator extracts and prepares data, trains and evaluates models and deploys the scoring service. Hobbiton's PostgreSQL database takes part only in the training workflow, through a read-only extract; the live scoring service does not connect to it. Table 4.4 specifies the main use case.

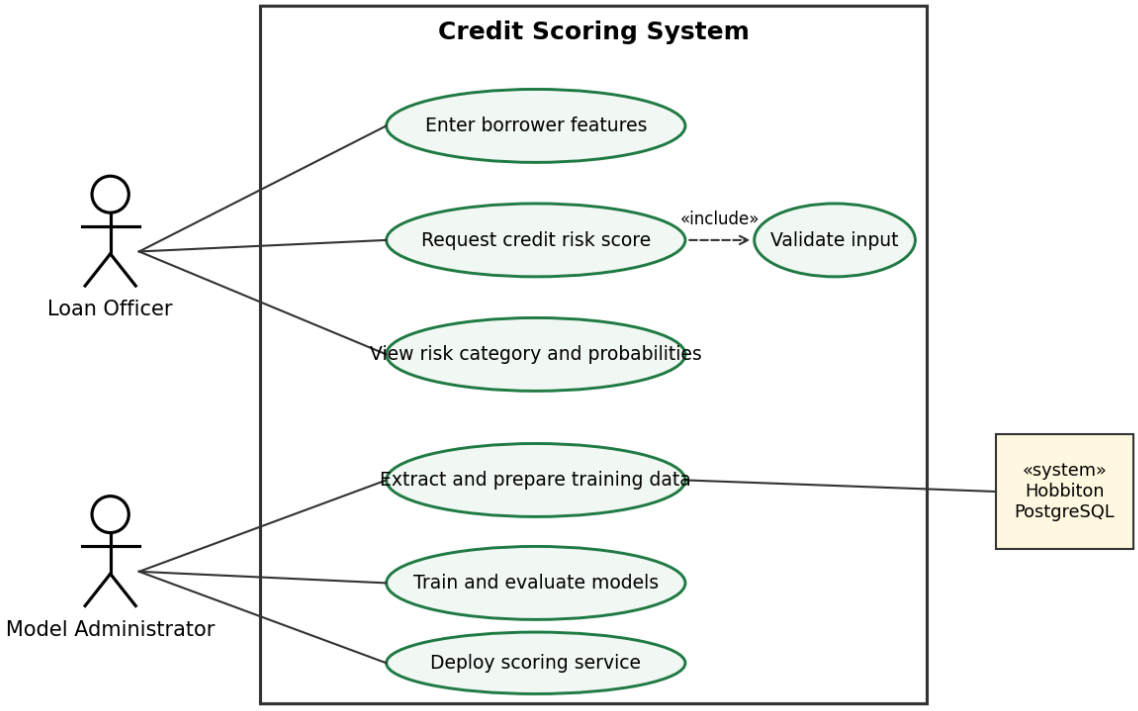


Figure 4.1: UML use case diagram of the credit scoring system

Table 4.4: Use case specification: request credit risk score

Field	Description
Use case	UC-02 Request credit risk score
Actor	Loan officer
Goal	Obtain a risk category and class probabilities for one borrower.
Preconditions	The web interface, gateway and scoring service are available.

Field	Description
Main flow	1. The officer opens the form. 2. The officer enters the six feature values and submits. 3. The interface validates the values and calls the gateway. 4. The gateway forwards the data to the scoring service. 5. The service predicts the risk category and probabilities. 6. The interface displays the result.
Alternative flows	A1. A value is missing or not numeric: the browser blocks submission, or the service returns 400 and the interface shows the message. A2. The scoring service cannot be reached: the gateway returns 502 and the interface shows an error. A3. The hosted service is asleep: the first response is delayed but succeeds.
Postcondition	The predicted category and probabilities are displayed; no applicant data are stored.

4.6 System Architecture

The system uses the four-layer architecture shown in Figure 3.3 and realises it as three independently deployed services, as the UML deployment diagram in Figure 4.2 shows. A PHP MVC web interface (presentation layer) calls a PHP REST gateway (application layer), which calls a Python Flask service (machine learning layer) that serves the trained XGBoost model. Training takes place offline in Google Colab against a read-only extract of Hobbiton's PostgreSQL database (data layer), and the resulting model files are committed to GitHub, from which Render deploys the scoring service.

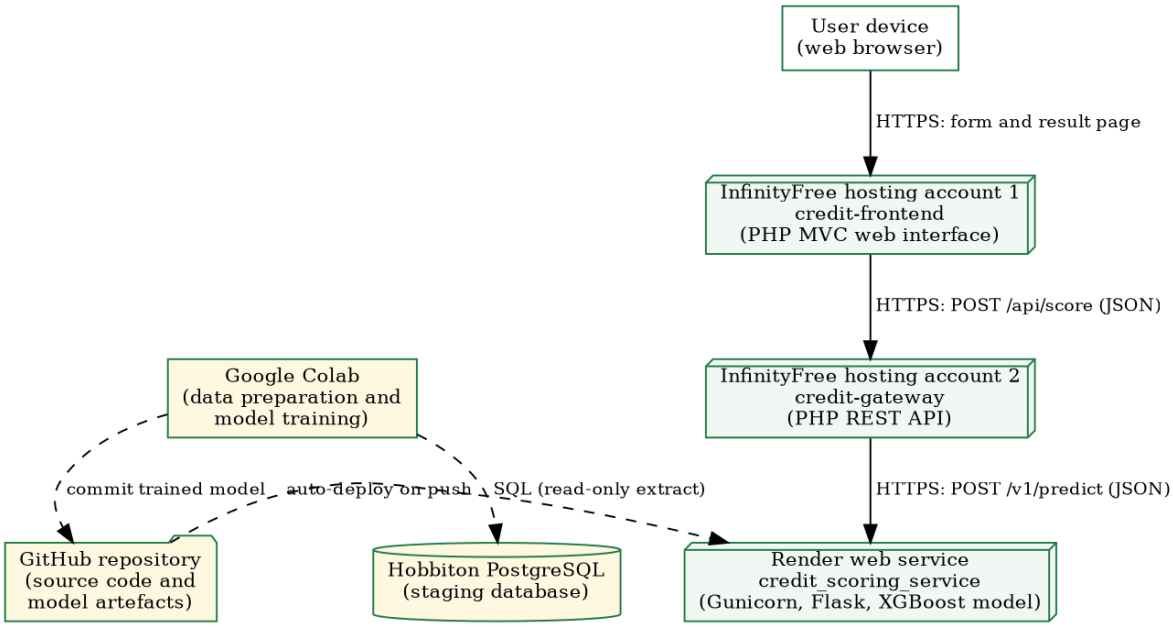


Figure 4.2: UML deployment diagram of the hosted system

Three decisions shaped this architecture. First, the model is exposed as a separate REST service rather than embedded in the PHP code, following the stateless client-server style described by Fielding [50]. The model can then be retrained and redeployed without touching Hobbiton's application code, in line with the microservice approaches discussed in [37] and [38]. Second, the gateway sits between the interface and the model so that validation, error translation and later concerns such as authentication, rate limiting and logging have a single home [46]. Third, the service is stateless and loads the model once at start-up, so each request pays only the cost of inference [47].

4.7 Data Design

Figure 4.3 shows the source tables used in this study and the derived borrower-level table. Only the columns needed for feature construction were extracted. The loan table is joined to the repayment table on the loan identifier, and both are aggregated by borrower to produce one row of features per member. The scoring service itself stores no applicant data; the trained model and label encoder are kept as files (risk_model_v1.joblib and label_encoder_v1.joblib).

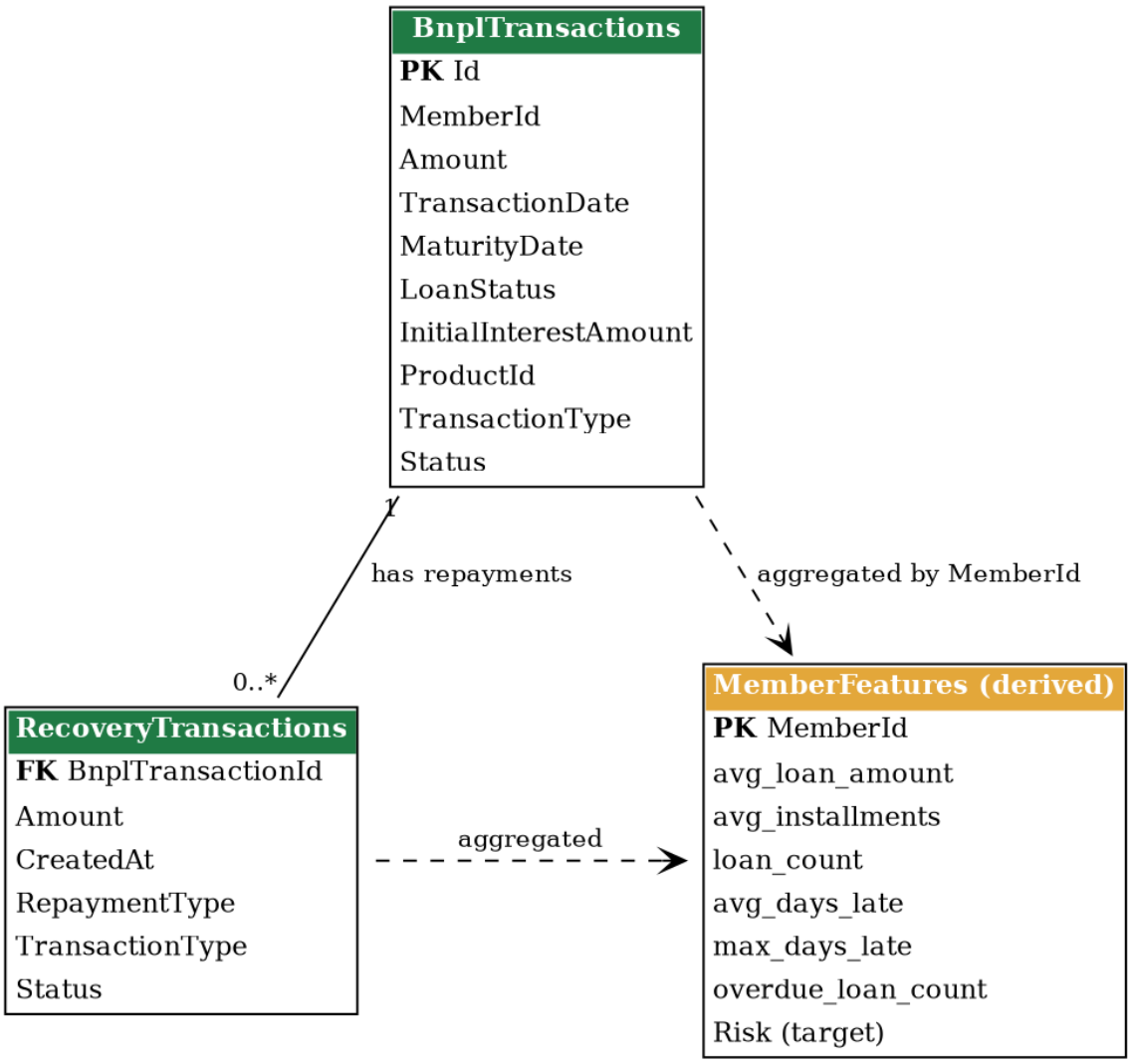


Figure 4.3: Entity-relationship diagram of the source and derived data

Table 4.5: Definition of the model features

Feature	Definition	Source
Average loan amount	Mean disbursed amount over the borrower's loans that have repayment history.	BnplTransactions.Amount
Average instalments	Mean number of successful repayment transactions per loan.	RecoveryTransactions
Loan count	Total number of loans the borrower has taken.	BnplTransactions (per MemberId)
Average days late	Mean number of days between the most recent repayment and the maturity date; negative values mean early payment.	MaturityDate, CreatedAt
Maximum days late	Largest of those differences for the borrower.	MaturityDate, CreatedAt

Feature	Definition	Source
Overdue loan count	Number of the borrower's loans with repayment history that are flagged overdue.	BnplTransactions.LoanStatus
Risk (target)	Four-class label obtained from the rule-based points scheme in Table 4.7.	Derived (Section 3.7.4)

4.8 Interface and Interaction Design

The web interface uses a split-card layout. A branded panel carrying the Hobbiton logo sits on the left, and the right panel holds a two-column grid of the six labelled inputs with a single submit button. The result view repeats the layout, shows the predicted category in a colour that runs from green (low) to red (very high), and draws each class probability as a horizontal bar. Examples of the input and output screens appear in Figure 6.9 and Figure 6.10.

Components communicate through the REST interface in Table 4.6. The gateway wraps the scoring service's output in a small envelope containing a success flag and the prediction, so the interface receives one consistent JSON shape whichever path was taken. Sample payloads are given in Appendix A.

Table 4.6: REST endpoints exposed by the system

Method	Endpoint	Purpose	Success	Errors
GET	/v1/health (Flask)	Service liveness and model version	200	None
POST	/v1/predict (Flask)	Predict risk from six features	200	400, 500
GET	/ (gateway)	Gateway welcome message	200	None
POST	/api/score (gateway)	Forward applicant data and relay the prediction	200	400, 405, 502

4.9 Technical Design

4.9.1 Feature construction

Feature construction follows six steps, implemented in the features module.

1. Collapse repayments to one row per loan: total repaid, number of instalments and most recent repayment date.
2. Count the loans each borrower has taken.

3. Remove loans with no repayment history, because lateness cannot be computed for them.
4. Compute days late as the most recent repayment date minus the maturity date, with both parsed as UTC.
5. Count, for each borrower, the loans flagged overdue.
6. Aggregate to one row per borrower using means, maxima and counts.

4.9.2 Risk label derivation

Observed default outcomes were not available, so the four-class target was derived from repayment behaviour with the points scheme in Table 4.7. The total points are mapped to a category as follows: 0 or fewer points is low, 1 to 2 is medium, 3 to 4 is high and 5 or more is very high.

Table 4.7: Rule-based risk points used to derive the target

Component	Condition	Points
Repayment timeliness	Average days late of 0 or less / up to 7 / up to 30 / above 30	0 / 1 / 2 / 4
Overdue share	Share of repayment-history loans flagged overdue: 0 / below one third / below two thirds / otherwise	0 / 1 / 2 / 3
Loan history depth	20 or more loans / 3 or fewer loans	-1 / +1
Loan size	Average loan amount of 5,000 or more	+1

4.9.3 Model configuration

XGBoost [51] was implemented through its scikit-learn interface [52] with the settings in Table 4.8. Shallow trees and a moderate learning rate were chosen to limit overfitting on a dataset of 419 borrowers. Labels are encoded in the fixed order low, medium, high, very high so that probabilities can always be mapped back to category names.

Table 4.8: XGBoost configuration

Parameter	Value	Rationale
n_estimators	100	Enough boosting rounds for a small dataset
max_depth	3	Shallow trees limit overfitting
learning_rate	0.1	Moderate step size

Parameter	Value	Rationale
eval_metric	mlogloss	Multiclass log loss
random_state	42	Reproducible results

4.9.4 Class structure and interaction

Figure 4.4 shows the classes of the two PHP tiers and Figure 4.5 shows the modules of the Python service. In the PHP tiers, ScoreController depends on ScoringGatewayClient, which wraps ApiClient; the gateway's ScoreHandler uses its own copy of ApiClient to call Flask. In the Python service, FlaskApp depends on RiskModel and Config, while the training pipeline combines FeatureBuilder, RuleLabeller and RiskModel offline.

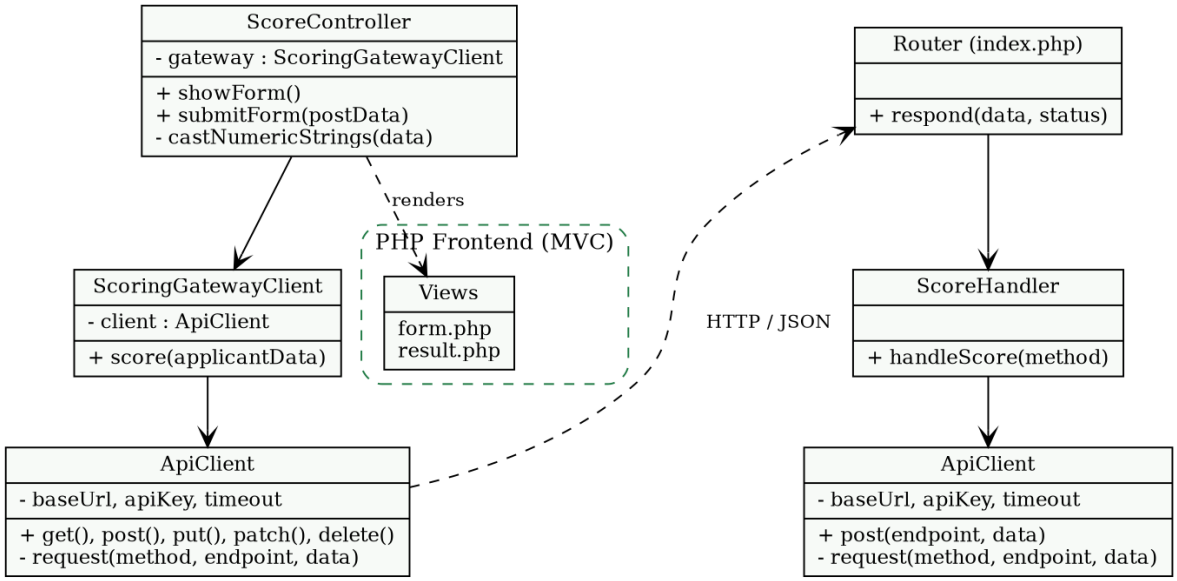


Figure 4.4: UML class diagram of the PHP frontend and gateway

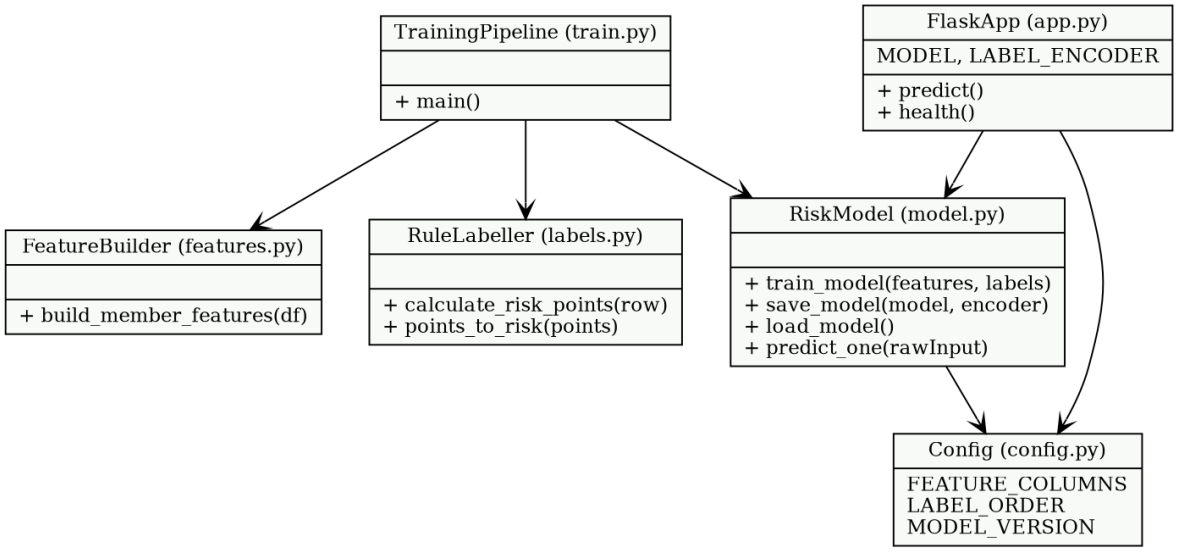


Figure 4.5: UML class diagram of the Python scoring service

Figure 4.6 shows the interaction for one scoring request, including the alternative path taken when the scoring service cannot be reached.

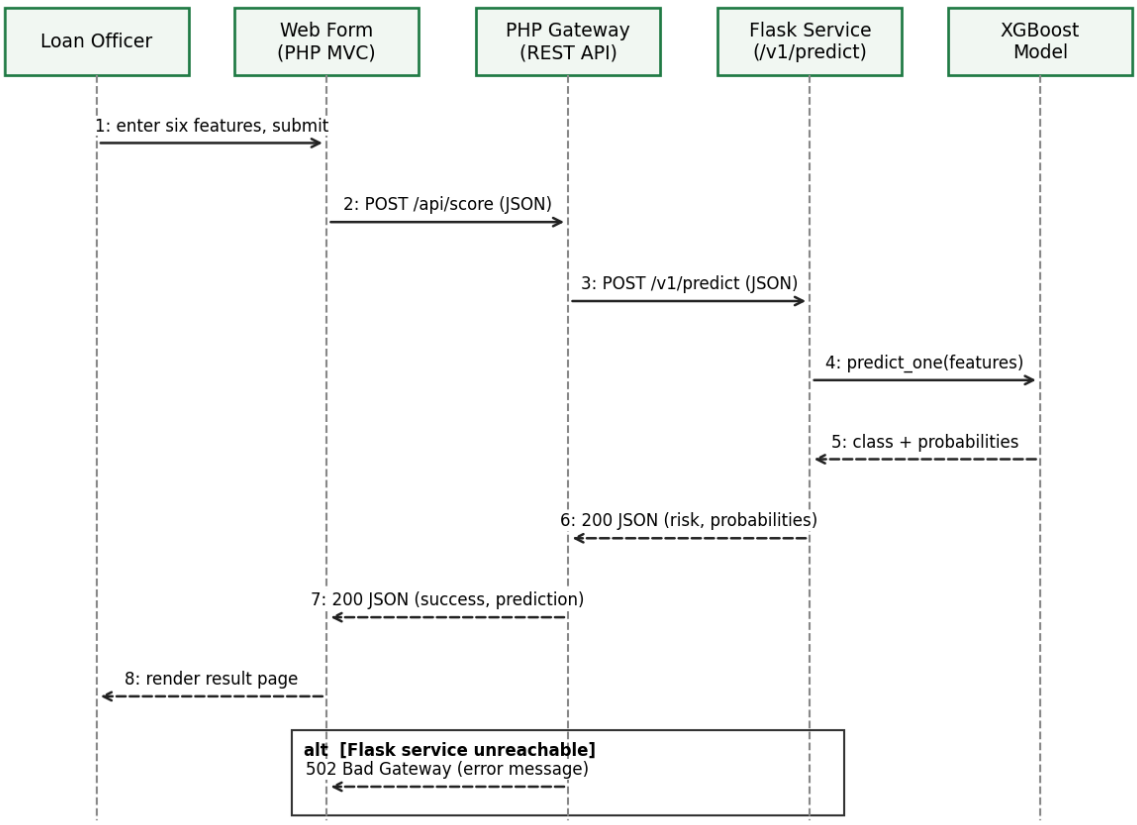


Figure 4.6: UML sequence diagram of a scoring request

4.10 Chapter Summary

This chapter specified ten functional and seven non-functional requirements, modelled the system with use case, deployment, entity-relationship, class and sequence diagrams, and described the feature construction, risk labelling and model configuration. The design separates the interface, gateway and model into independent services connected only by REST calls, which is what allows each to be built, tested, deployed and replaced independently. Chapter 5 describes how this design was implemented.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 Introduction

This chapter describes how the design in Chapter 4 was implemented and deployed. The project was carried out by a single student, so every component described here, from data extraction to deployment, is the author's own work. Implementation proceeded in Scrum sprints, building and testing each layer in isolation, starting with the model and moving outward, before connecting it to the next layer.

5.2 Development Environment and Tools

Table 5.1: Tools and technologies used

Area	Technology	Purpose
Machine learning	Python 3, pandas, scikit-learn [52], XGBoost [51]	Feature engineering, training and evaluation
Notebook environment	Google Colab	Exploratory analysis and model comparison
Data access	SQLAlchemy with a PostgreSQL driver	Read-only extraction from Hobbiton's staging database
Scoring service	Flask [53] served by Gunicorn	REST interface to the trained model
Gateway and interface	PHP with cURL; Apache (XAMPP) locally	REST gateway and MVC web interface
Styling	HTML5 and CSS3	Split-card interface
Testing	Postman	API test collection
Version control	Git and GitHub	Source control and deployment source for Render
Hosting	Render [54] and InfinityFree [55]	Free-tier hosting of the three services

A Python virtual environment isolated the dependencies of the scoring service, and Visual Studio Code was used as the editor on Windows.

5.3 System Components

Table 5.2: Components and their responsibilities

Tier	Files	Responsibility
Scoring service	credit_scoring/config.py	Feature schema, label order, model version and file paths
	credit_scoring/features.py	Builds borrower-level features from loan and repayment rows
	credit_scoring/labels.py	Rule-based risk points and category mapping
	credit_scoring/model.py	Training, saving, loading and prediction
	train.py and app.py	Training pipeline; Flask routes /v1/predict and /v1/health
PHP gateway	index.php, config.php, ApiClient.php, handlers/score.php	Routing, calling Flask and translating errors
PHP interface	index.php, controllers/, models/, views/, css/, images/	MVC web interface

5.4 Key Implementation Details

5.4.1 Data extraction and feature engineering

A single SQL query joined BnplTransactions to RecoveryTransactions on the loan identifier. It kept successful loan disbursements and successful loan repayments, and used a left join so that loans without repayments were retained. Repayment rows were then collapsed to one row per loan, and loans with no repayment history were removed. Days late were computed from the most recent repayment date and the maturity date, parsing both as UTC to avoid errors from mixed ISO-8601 timestamp formats. Loans were then aggregated by borrower to produce the six features, as in Listing 1.

```
features = with_repayment.groupby("MemberId").agg(
    avg_loan_amount=("LoanAmount", "mean"),
    avg_installments=("NumberOfInstallments", "mean"),
    loan_count=("Loan_count", "max"),
    avg_days_late=("DaysLate", "mean"),
    max_days_late=("DaysLate", "max"),
    overdue_loan_count=("LoanStatus", lambda s: (s == "overdue").sum()),
).reset_index()
```

Listing 1. Borrower-level feature aggregation.

5.4.2 Model training and persistence

All four classifiers were fitted on the same 70% training split. The XGBoost model and a label encoder were saved with joblib, so the service reproduces the label order exactly (Listing 2).

```
model = XGBClassifier(n_estimators=100, max_depth=3, learning_rate=0.1,
                      random_state=42, eval_metric="mlogloss")
model.fit(X_train, y_train_enc)
joblib.dump(model, "models/risk_model_v1.joblib")
joblib.dump(label_encoder, "models/label_encoder_v1.joblib")
```

Listing 2. Training and saving the selected model.

5.4.3 Flask scoring service

The service loads the model once at start-up and exposes /v1/predict. The route checks the request body against the feature list in the configuration module, so adding a feature to that list is enough to make the service require it (Listing 3).

```
MODEL, LABEL_ENCODER = mdl.load_model()   # loaded once at start-up

@app.route("/v1/predict", methods=["POST"])
def predict():
    body = request.get_json(silent=True)
    if body is None:
        return jsonify({"error": "Request body must be valid JSON"}), 400
    missing = [f for f in config.FEATURE_COLUMNS if f not in body]
    if missing:
        return jsonify({"error": f"Missing required fields: {missing}"}), 400
    result = mdl.predict_one(body, model=MODEL, label_encoder=LABEL_ENCODER)
    return jsonify(result), 200
```

Listing 3. The prediction endpoint.

5.4.4 PHP gateway

The gateway reads the request body, checks that it is valid and non-empty JSON, and forwards it unchanged to Flask using the ApiClient class. It separates three outcomes: an unreachable service becomes 502, an error response from Flask is relayed with its own status code, and a successful prediction is wrapped in a success envelope (Listing 4). The gateway does not know the feature names, which is why the feature list can change without editing it.

```
$client = new ApiClient(PYTHON_SERVICE_URL, null, PYTHON_SERVICE_TIMEOUT);
$result = $client->post('/v1/predict', $applicantData);
if ($result['status'] === 0) {            // upstream unreachable
    respond(['error' => 'Could not reach the credit scoring service'], 502);
}
if (!$result['success']) {                // relay Flask's own error
```

```
    respond(['error' => $result['error'], 'detail' => $result['data']],
$result['status']);
}
respond(['success' => true, 'prediction' => $result['data']], 200);
```

Listing 4. Gateway handling of the three outcomes.

5.4.5 MVC web interface

ScoreController shows the form on a GET request and processes a submission on a POST request. On submission it converts numeric strings to numbers, passes the data to ScoringGatewayClient, which calls the gateway's /api/score endpoint, and selects the result view or the error view. Only the form view names the six fields; the controller, model, gateway and Flask service treat the payload as opaque JSON apart from Flask's required-field check.

5.5 Integration

The layers communicate only through HTTP and JSON. Each address is a single constant, PYTHON_SERVICE_URL in the gateway and GATEWAY_URL in the interface, and changing those two constants is the only edit needed to move from localhost to hosted deployment. Every Flask response carries the model version, so it is always possible to tell which model produced a score. Integration was verified with the Postman collection described in Section 6.15.

5.6 Deployment and Hosting

The three services were deployed to two free platforms, as summarised in Table 5.3.

Table 5.3: Deployment summary

Component	Platform	Deployment method	Notes
Scoring service	Render web service (free plan)	Connected to GitHub; build: pip install -r requirements.txt; start: gunicorn app:app	HTTPS address; sleeps when idle, so the first request is slower
PHP gateway	InfinityFree, hosting account 2	Files uploaded through the File Manager	Calls Render over HTTPS
PHP interface	InfinityFree, hosting account 1	Files uploaded through the File Manager	Public entry point for users

On Render the service is built from the GitHub repository. The trained model files are committed with the code because the platform sees only repository contents, and Gunicorn replaces Flask's development server, which is not intended for production use [53]. On InfinityFree, one hosting account was created for each PHP project and its files were uploaded through the browser-based File Manager [55]. In both PHP projects the constants in config.php were changed from localhost to the public HTTPS address of the next layer.

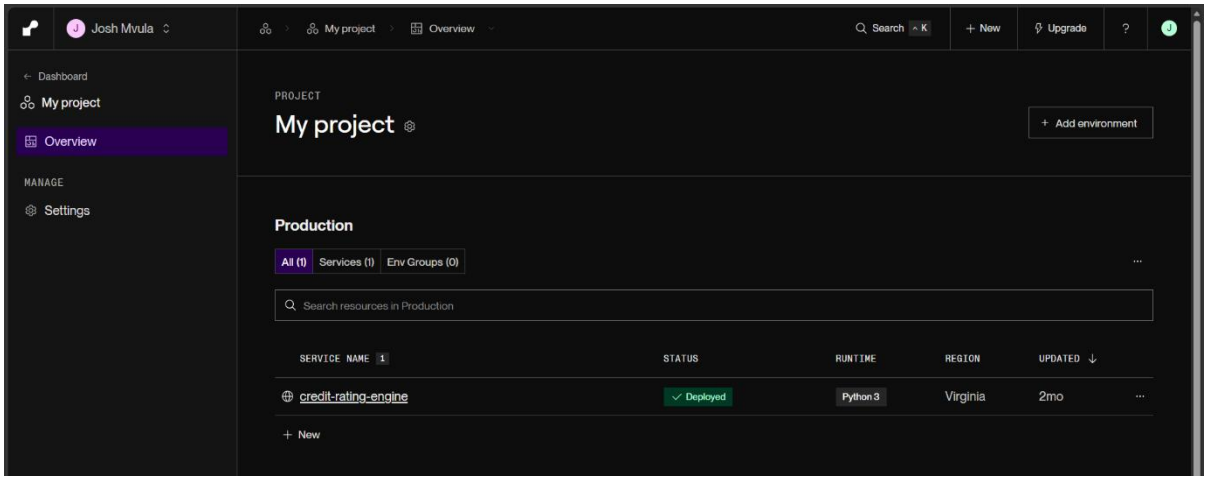


Figure 5.1: Render web service hosting the scoring service

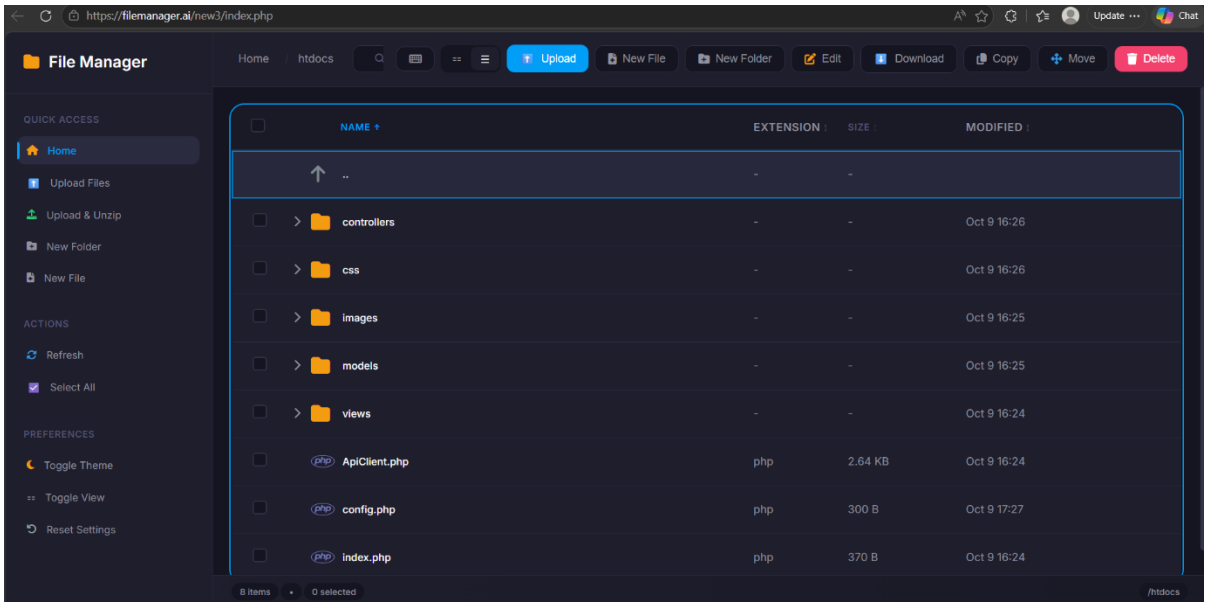


Figure 5.2: InfinityFree hosting account for the PHP components

Free hosting brings trade-offs. Render's free web services sleep after a period of inactivity, so the first request after a pause takes noticeably longer while the service restarts [54].

InfinityFree limits daily usage and provides no command-line access, so deployment is file-based. These limits are acceptable for a prototype and demonstration, but a production deployment would need always-on hosting.

5.7 Challenges and Solutions

Table 5.4: Implementation challenges and how they were resolved

Challenge	Solution
The feature list risked being duplicated in the interface, gateway and service.	The Flask configuration holds the only feature schema; the gateway and controller forward JSON unchanged.
A manually typed overdue percentage was error-prone and could not be derived from the other inputs.	The form collects the raw overdue loan count, which the model uses directly.
Different failures needed different treatment.	The gateway distinguishes 400 (bad input), 500 (prediction failure) and 502 (service unreachable).
Timestamps with mixed formats caused errors when computing days late.	Both timestamps are parsed as ISO-8601 in UTC before subtraction.
The free Render plan puts the service to sleep.	A health endpoint can be called to wake it before a demonstration, and the delay is documented.
InfinityFree offers no command-line access.	Deployment uses file upload, and all environment-specific addresses live in config.php.
The database lacked demographic and alternative data.	A feature audit documented the gap and scoped the model to repayment behaviour.

5.8 Chapter Summary

This chapter showed how the design was implemented as a Python scoring service, a PHP gateway and an MVC web interface, and how the three were deployed to Render and InfinityFree. Chapter 6 reports the results of the data audit, the model comparison and the testing of the deployed system.

CHAPTER 6: TESTING, RESULTS AND EVALUATION

6.1 Introduction

This chapter presents and discusses the results obtained from the development and evaluation of the machine learning-based credit scoring system. The results are organised according to the study objectives and the CRISP-DM stages described in Chapter Three. The chapter begins with the characteristics of the final modelling dataset and the results of feature engineering, followed by model training, comparative model evaluation, per-class analysis, and interpretation of the selected model.

Four classification models were evaluated: Logistic Regression, Decision Tree, Random Forest, and XGBoost. Model performance was assessed using accuracy and macro-averaged F1-score, with additional analysis based on per-class performance, confusion matrices, ROC-AUC, feature importance, and cross-validation. These measures were selected to provide a more complete assessment of the models' ability to distinguish among the four borrower risk categories.

6.2 Dataset and Modelling Population

The final modelling dataset was constructed from borrower, loan, and repayment records extracted from Hobbiton's PostgreSQL database. The records were consolidated at borrower level and transformed into behavioural credit-risk features derived from historical loan and repayment activity.

Table 6.1 summarises the final modelling population.

Table 6.1: Summary of the Final Modelling Dataset

Item	**Value**
Total borrower records extracted	2,837 (unique loans extracted) / 419 unique borrowers
Records removed during cleaning	1,198 (loans without repayment history dropped)

Final modelling records	419 (unique borrowers aggregated)
Number of input features	6
Number of risk classes	4(low, medium, high, very high)
Training-set records	293
Testing-set records	126
Train-test split	70:30

The dataset was divided into training and testing subsets using stratified sampling to preserve the distribution of the four risk categories across both subsets. This was important because an unstratified split could result in under-representation of minority risk classes in either subset and could therefore produce an unreliable estimate of model performance.

6.2.1 Target Variable Distribution

The target variable classified borrowers into four ordered risk categories: low risk, medium risk, high risk, and very high risk. The distribution of borrowers across these categories is presented in Table 6.2 and Figure 6.1.

Table 6.2: Distribution of Borrowers by Credit-Risk Category

Risk Category	**Number of Borrowers**	**Percentage of Dataset**
Low risk	63	15 %
Medium risk	192	45.8 %
High risk	96	22.9 %
Very high risk	68	16.2 %
Total	419	100.00%

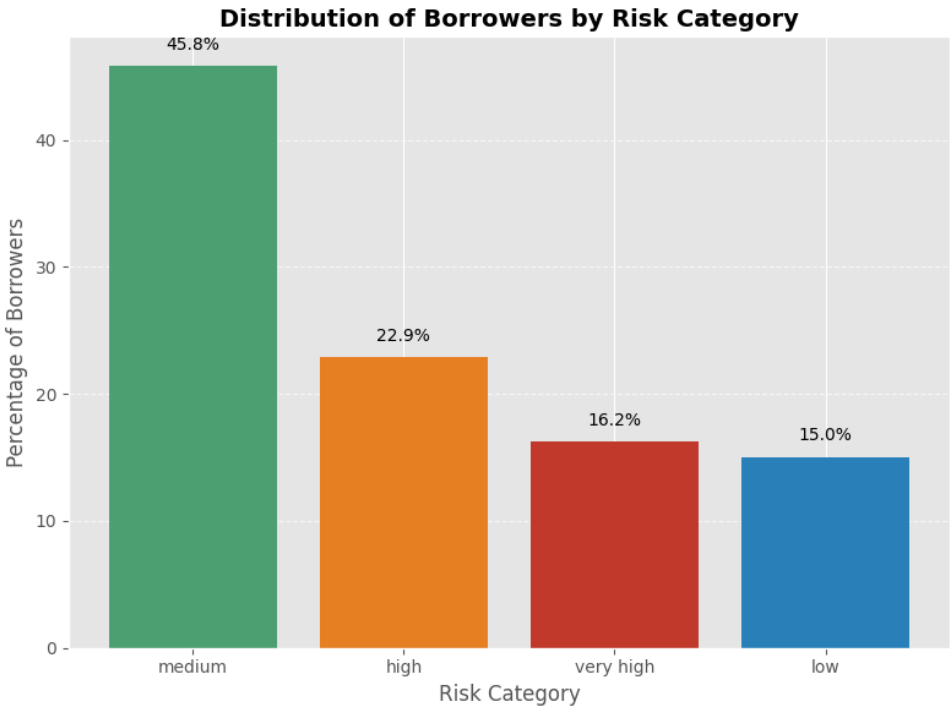


Figure 6.1: Distribution of borrowers across credit-risk categories

The class distribution indicates the relative representation of the four risk categories in the modelling dataset. The distribution was considered when interpreting model performance because overall accuracy can be misleading when classes are imbalanced. Consequently, macro-averaged F1-score and class-level precision, recall, and F1-score were considered alongside accuracy.

6.3 Feature Engineering Results

Six borrower-level features were retained for model development:

- Average loan amount
- Average number of instalments
- Loan count
- Average days late
- Maximum days late
- Overdue loan count

These features were derived from Hobbiton's historical loan and repayment records and represent three complementary dimensions of borrower behaviour: borrowing exposure, repayment structure, and repayment performance. Average loan amount and loan count capture borrowing exposure; average instalments capture repayment structure; while average days late, maximum days late, and overdue-loan count capture repayment discipline and historical delinquency.

Table 6.3: Descriptive Statistics of the Final Model Features

Feature	**Mean**	**Standard Deviation**	**Minimum**	**Maximum**
Average loan amount	1850.501720	2122.176580	10.000000	11666.666667
Average instalments	1.273393	0.617679	1.000000	8.000000
Loan count	27.868735	24.400946	1.000000	88.000000
Average days late	24.518011	39.599618	-26.787000	319.659636
Maximum days late	43.843100	62.360589	-24.641155	319.659636
Overdue loan count	0.338902	0.633733	0.000000	7.000000

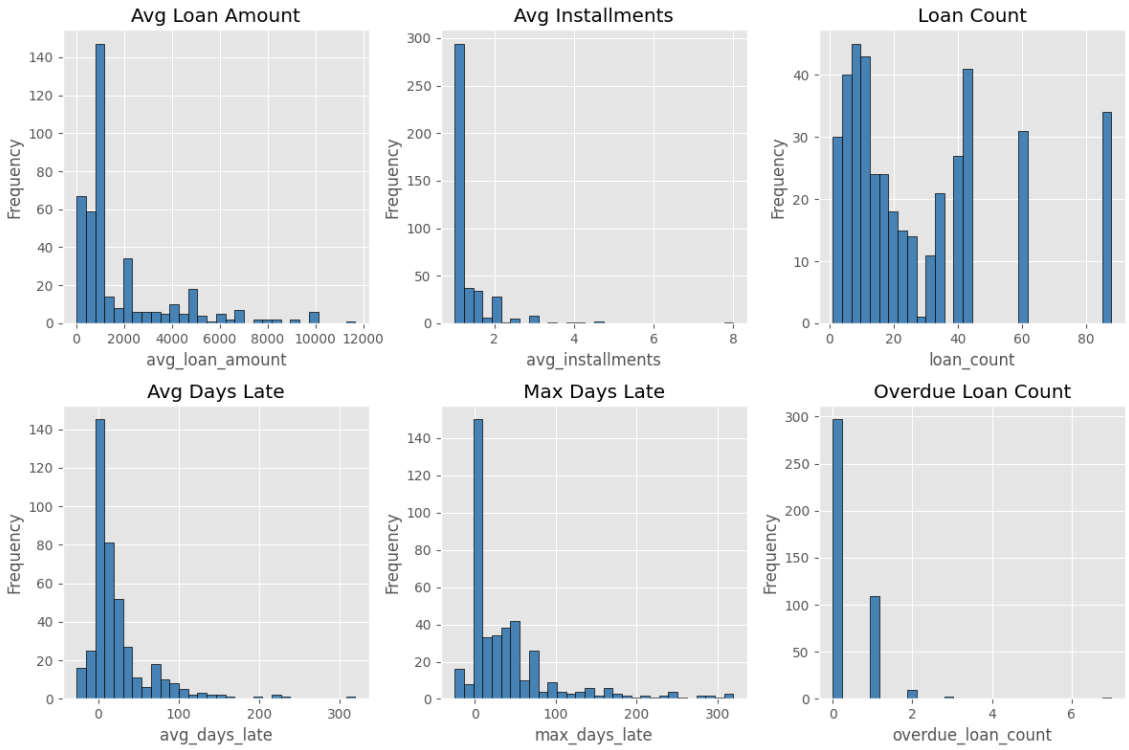


Figure 6.2: Distribution of engineered borrower-level credit-risk features

The descriptive statistics and feature distributions should be interpreted in relation to borrower repayment behaviour. In particular, average and maximum days late provide complementary information: the average reflects general repayment behaviour, while the maximum captures the severity of a borrower's worst observed repayment delay.

6.4 Justification for the Final Feature Set

Although the study initially considered a broader set of demographic, employment, financial, mobile-usage, business, and agricultural variables, the final model used six behavioural features. This decision was based on a structured feature audit in which candidate variables were verified against the Hobbiton PostgreSQL database rather than assumed to be available.

The audit found that several conceptually desirable variables, including employment history, education, socio-economic indicators, and other demographic attributes, were absent, incomplete, inconsistently recorded, or not available at a borrower level suitable for modelling. Including variables with substantial missingness or inconsistent representation could introduce noise and reduce model reliability. The final feature set therefore prioritised variables that were

verifiably available and could be consistently derived from historical loan and repayment records.

This finding represents a limitation of the existing operational data environment rather than a failure of the broader research design. Hobbiton's database primarily supports lending operations, and the six retained features provided the strongest directly observable behavioural signals available for the present study.

The retained features are appropriate for credit scoring because repayment behaviour is directly related to credit risk. Loan count and average loan amount capture borrowing exposure, average instalments capture repayment structure, and average days late, maximum days late, and overdue-loan count capture repayment discipline and historical delinquency.

Table 6.4: Summary of Feature Availability in Hobbiton Database

Feature Domain	**Examples of Candidate Features**	**Audit Outcome**	**Use in Final Model**
Demographic data	Age, gender, marital status, location	Not available	No
Employment and education	Employment status, employer, education level	Not available	No
Financial and loan data	Loan amount, instalments, loan count	Available / derivable	Yes
Repayment behaviour	Days late, overdue loans	Available / derivable	Yes
Mobile and alternative data	Mobile-money activity, transaction frequency	Not available	No
Business and agricultural data	Business type, agricultural activity	Not available	No

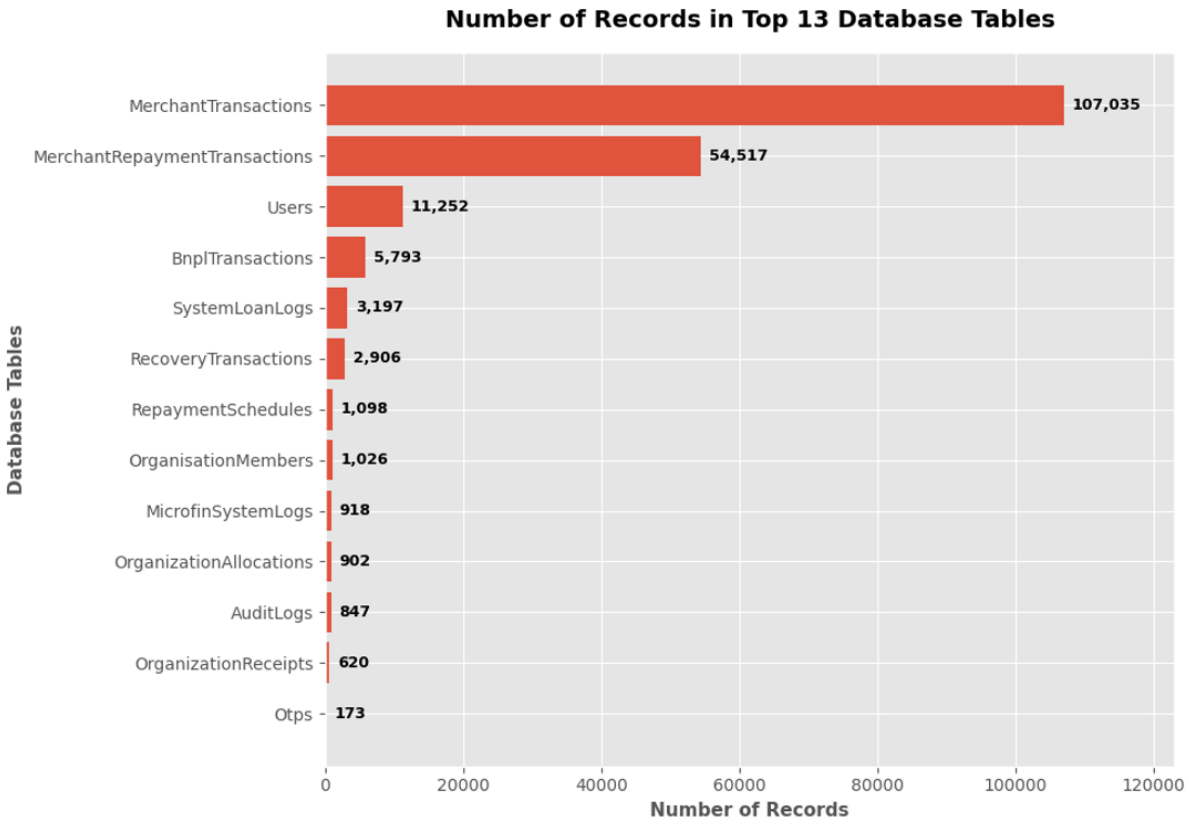


Figure 6.3: Outcome of the Hobbiton database feature audit

6.5 Model Training Results

All four models were trained using the same prepared dataset and evaluated on an identical, unseen test set representing 30% of the total records. Rather than tuning each model independently, comparable complexity constraints (such as a maximum depth of 3 for tree-based estimators) were standardly applied to establish a controlled baseline comparison. Model selection prioritized predictive performance, where XGBoost demonstrated superior accuracy (85.7%) and macro F1-score (85.1%), alongside model interpretability, robustness, and suitability for credit risk deployment.

Figure 6.4: Model training process in Google Colab

6.6 Comparative Model Performance

The four models were compared using overall accuracy and macro-averaged F1-score. Accuracy measures the proportion of correctly classified borrowers, while macro-averaged F1-score gives equal weight to each risk category and is therefore useful where class frequencies differ.

Table 6.5: Comparative Performance of the Credit-Scoring Models

Model	**Accuracy**	**Macro F1-Score**
XGBoost	0.857143	0.851112
Random Forest	0.761905	0.725463
Logistic Regression	0.746032	0.700605
Decision Tree	0.714286	0.666981

XGBoost achieved the highest performance, with an accuracy of 0.857143 (85.71%) and a macro-averaged F1-score of 0.851112 (85.11%). It outperformed Random Forest by 9.52 percentage points in accuracy and 12.56 percentage points in macro F1-score. Random Forest was the second-best model, followed by Logistic Regression and Decision Tree.

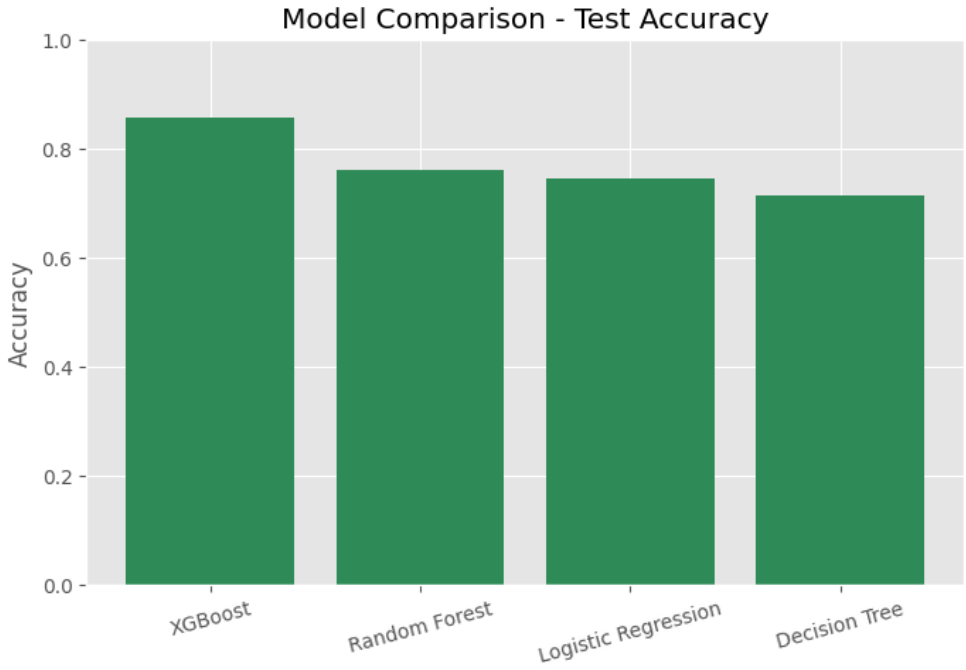


Figure 6.5: Comparison of model accuracy

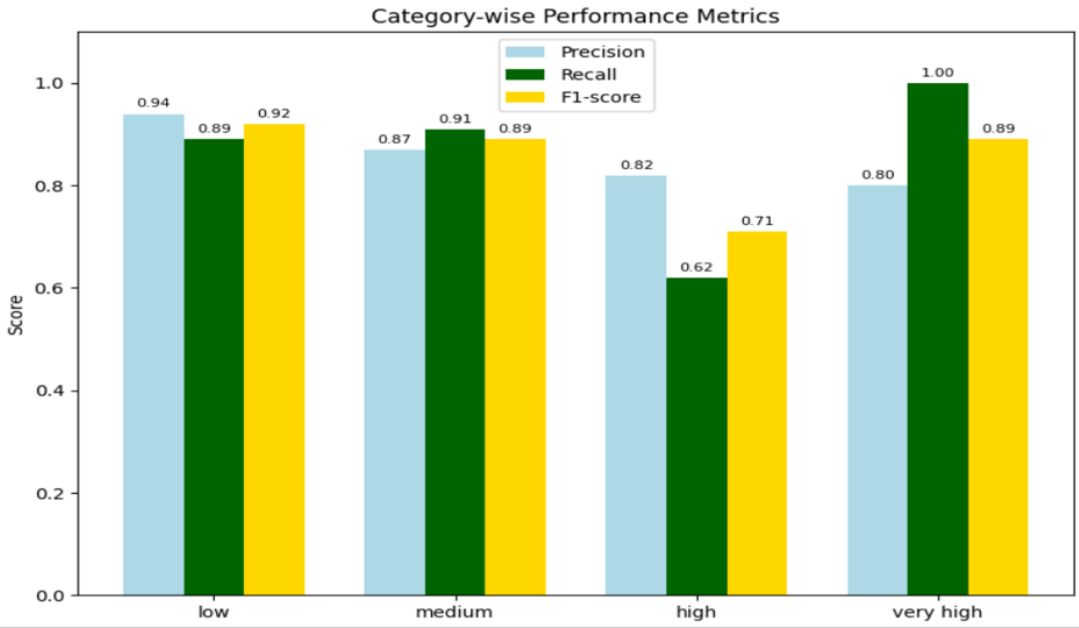
The results show that the ensemble methods evaluated in this study performed better than the individual Logistic Regression and Decision Tree models. XGBoost may have benefited from its ability to represent non-linear relationships and interactions among the behavioural features. However, this interpretation should be considered in conjunction with the per-class results and cross-validation results presented in subsequent sections.

Logistic Regression recorded lower performance than the ensemble methods, while the single Decision Tree produced the lowest accuracy and macro F1-score among the four models. These results indicate that, for the available feature set and modelling population, the ensemble approaches provided stronger predictive performance.

6.7 Per-Class Model Performance

Because the study used a four-class risk target, overall accuracy alone was insufficient to determine whether the models were suitable for credit-risk classification. Precision, recall, and F1-score were therefore examined for each risk category, with particular attention to the high-risk and very high-risk classes because incorrect classification of these borrowers may have greater financial consequences.

Table 6.6: Per-Class Performance of the XGBoost Model



6.8 Confusion Matrix Results

Confusion matrix was used to examine the specific types of classification errors made by the model. It showed both the number of correctly classified borrowers and the risk categories into which misclassified borrowers were placed.

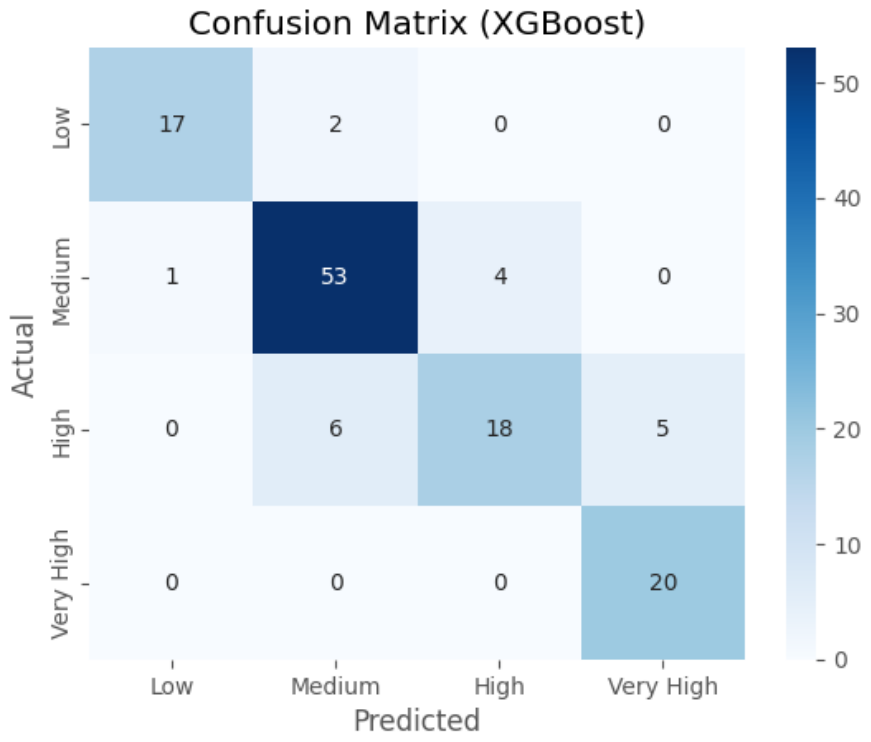


Figure 6.6: Confusion matrix for the XGBoost model

6.9 ROC-AUC Results

To evaluate the classification performance across the four borrower risk categories, Receiver Operating Characteristic (ROC) curves were generated for each class, together with a micro-average curve summarizing overall performance. The ROC curves plot the true positive rate against the false positive rate, with the area under the curve (AUC) serving as a quantitative measure of discriminative ability.

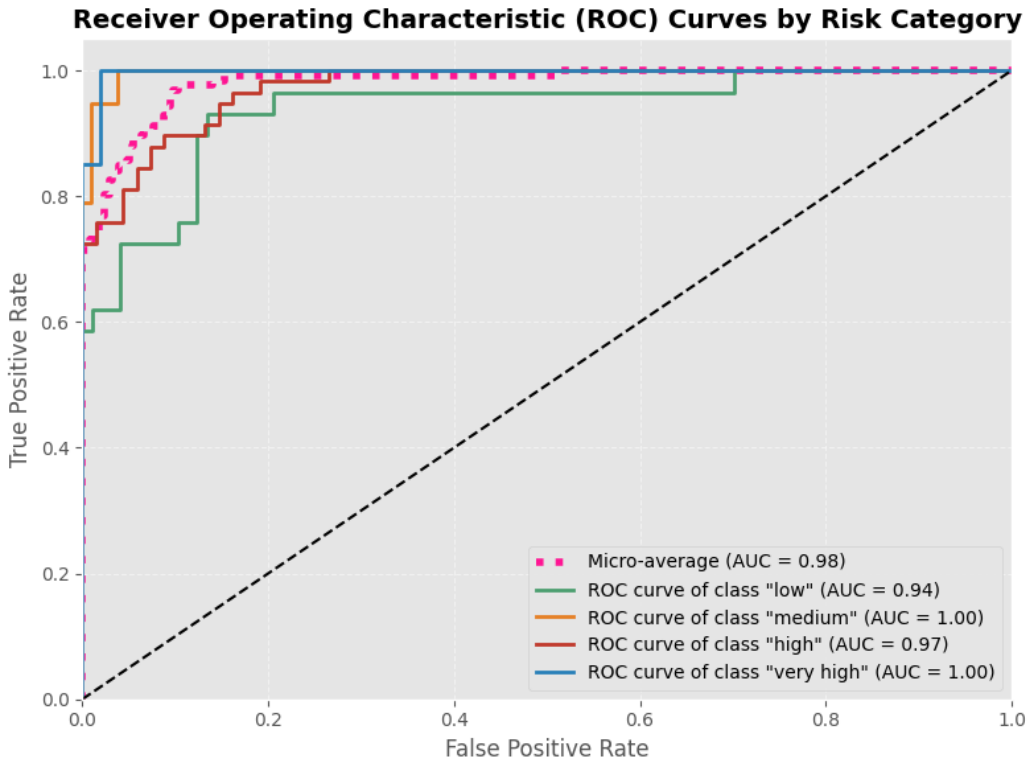


Figure 6.7: ROC Curve

As shown in Figure 6.7, the model achieved excellent results across all categories. The micro-average AUC was 0.98, indicating strong overall predictive capability. Class-specific AUC values were similarly high: 0.94 for *low* risk, 1.00 for *medium* risk, 0.97 for *high* risk, and 1.00 for *very high* risk. These values demonstrate that the model can reliably distinguish between borrower risk levels, with particularly strong separation for the medium and very high categories.

Overall, the ROC analysis confirms that the selected model exhibits robust classification performance, making it suitable for deployment in risk assessment applications.

6.10 Feature Importance Analysis

Feature importance analysis was conducted to identify the variables that contributed most strongly to the predictions of the selected XGBoost model. This analysis provides an additional

level of interpretability by showing which behavioural variables contributed most to the model's decisions.

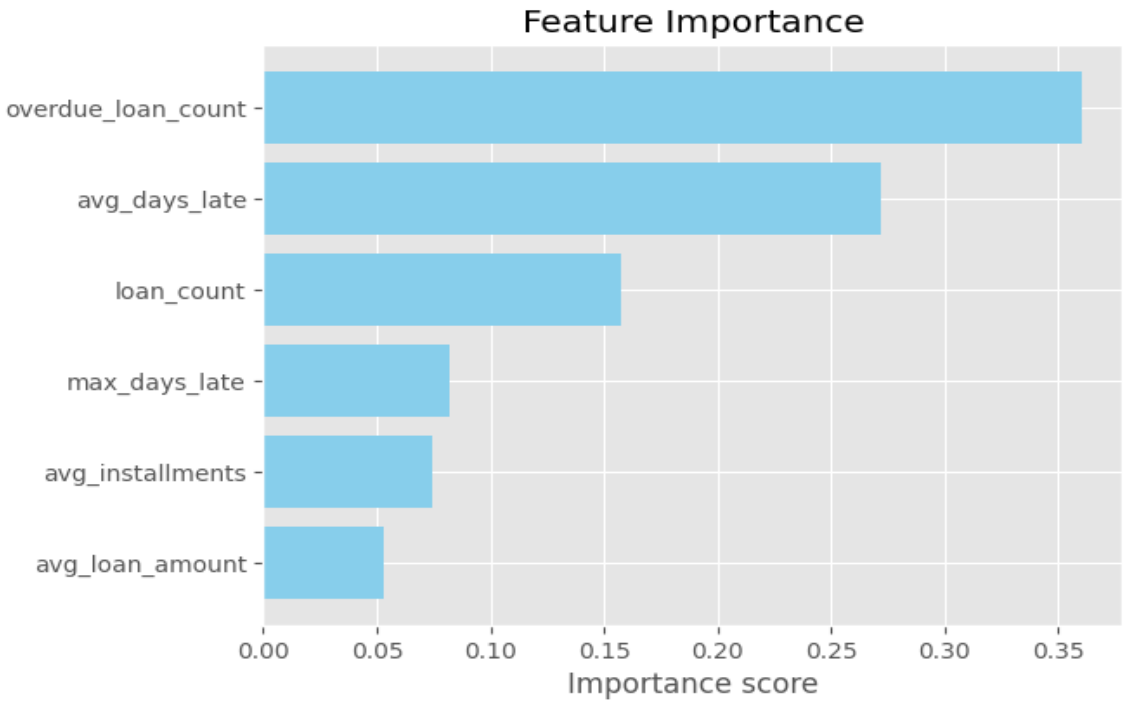
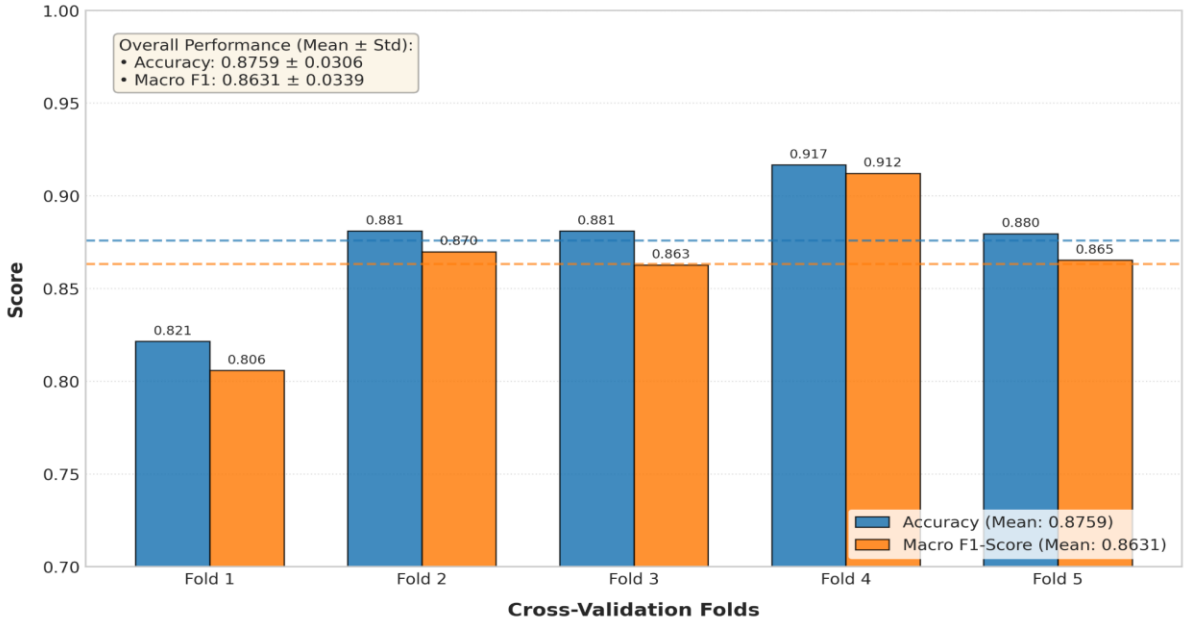


Figure 6.8: Feature importance of the XGBoost credit-scoring model

6.11 Cross-Validation Results

Cross-validation was performed to assess whether model performance remained stable across different subsets of the training data. This reduces the possibility that the reported test-set performance was caused by a particularly favourable train-test split.

Table 6.7: Cross-Validation Performance of the Evaluated Models



The XGBoost classifier demonstrated exceptional stability and consistency, achieving an average accuracy of 87.59% with a low standard deviation of ±3.06%. This indicates that the model is robust and does not depend on any particular random split of the data. Furthermore, the macro-averaged F1-score of 86.31% (±3.39%) confirms strong generalizability across all four borrower risk categories, maintaining balanced precision and recall even for the less represented classes.

6.12 Selection of the Final Model

Based on the available test-set results, XGBoost was selected as the final credit-scoring model. It achieved the highest accuracy, 0.857143 (85.71%), and the highest macro-averaged F1-score, 0.851112 (85.11%), among the four evaluated models.

Although XGBoost is less inherently interpretable than Logistic Regression or a single Decision Tree, its superior predictive performance on the available test set makes it the most suitable candidate.

The selected model can be integrated into Hobbiton's web service as a scoring component that receives the required borrower-level feature values and returns a predicted risk category and associated score. This supports the proposed architecture in which the machine learning model

operates as a discrete scoring service without replacing Hobbiton's existing loan-processing workflow.

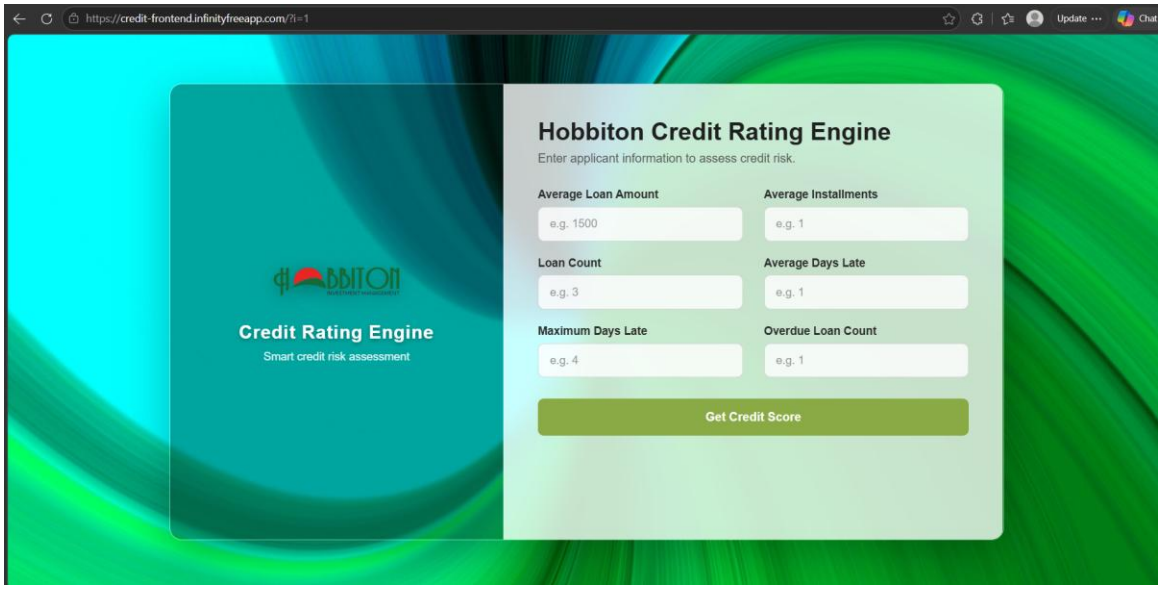


Figure 6.9: Example credit-risk prediction input

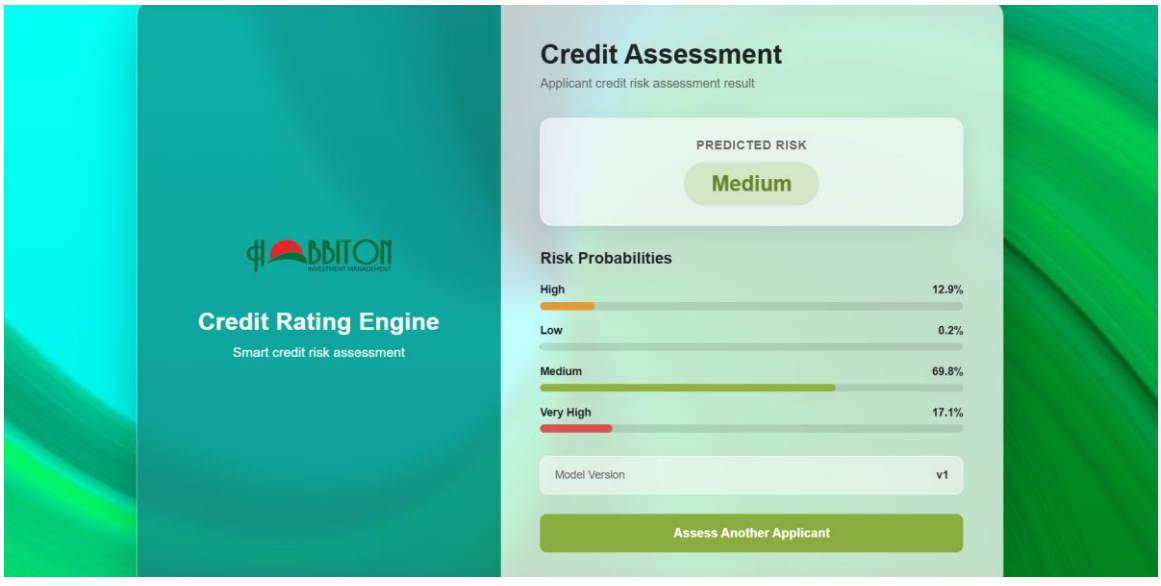


Figure 6.10: Example credit-risk prediction output

6.13 System Testing

The deployed system was tested through the Postman collection listed in Appendix B. The collection exercises the scoring service and the PHP gateway directly and uses two environment variables, flask_base and gateway_base, so the same requests can be run against local or hosted services. Table 6.8 lists each test case, the behaviour expected from specification table 4.6 and the requirement it checks.

Table 6.8: System test cases and results

ID	Request	Input	Expected result	Requirement	Result
TC1	GET /v1/health (Flask)	None	200 with the model version	FR8	Pass
TC2	POST /v1/predict	Six valid features	200 with a risk category, four class probabilities and the model version	FR4, FR5	Pass
TC3	POST /v1/predict	One or more features omitted	400 listing the missing fields	FR7	Pass
TC4	GET / (gateway)	None	200 welcome message	Table 4.6	Pass
TC5	POST /api/score	Six valid features	200 with success flag	FR3, FR4, FR5	Pass

			and prediction		
TC6	POST /api/score	Malformed JSON	400 "Request body must be non-empty valid JSON"	FR7	Pass
TC7	POST /api/score	Features omitted	400 relayed unchanged from Flask	FR7	Pass
TC8	POST /api/score	Valid features, scoring service unreachable	502 "Could not reach the credit scoring service"	FR7	Pass

For the sample applicant in Appendix A (average loan amount 1,500, one instalment, three loans, average lateness of one day, maximum lateness of four days and no overdue loans), the service returned medium risk with a probability of 0.722. The web interface requirements (FR1, FR2 and FR6) were checked manually through the form shown in Figures 6.9 and 6.10.

All eight requests behaved as specified. Distinct 400 and 502 responses separate invalid input from an unreachable scoring service, and the gateway relays the scoring service's validation errors unchanged, which is the behaviour required by FR7. The 405 response listed for /api/score in Table 4.6 was not part of the collection.

The testing was functional. Response time was not formally benchmarked, so the two-second target in Table 4.3 is unverified, and no load, security or user acceptance testing was carried out. These tests should be added before any production use.

6.14 Discussion of Results

The results demonstrate that a multiclass credit-scoring model can be developed using behavioural loan and repayment data available in Hobbiton's PostgreSQL database. Among the four evaluated models, XGBoost produced the strongest observed test-set performance, with an accuracy of 85.71% and a macro-averaged F1-score of 85.11%.

The stronger performance of XGBoost compared with Logistic Regression suggests that non-linear relationships and interactions among the available behavioural features may be useful for distinguishing borrower risk categories. For example, the number of previous loans may have a different risk implication depending on whether those loans were repaid on time or experienced repayment delays.

The use of macro-averaged F1-score was particularly important because it gives equal weight to the four risk categories rather than allowing performance on the most frequent class to dominate the evaluation. The per-class results should therefore be considered alongside the overall accuracy when determining whether the model is appropriate for deployment.

The results should nevertheless be interpreted with caution. The model was trained using historical Hobbiton data and may be affected by changes in borrower behaviour, lending products, or broader economic conditions. Continuous monitoring and periodic retraining using newly available records will therefore be important if the model is deployed operationally.

6.15 Limitations Observed in the Results

The main limitation observed during the modelling process was the limited availability of demographic and alternative-data features in Hobbiton's database. Although the initial research design considered demographic, employment, mobile-usage, business, and agricultural variables, the feature audit found that several of these variables were not sufficiently complete, consistent, or structured at borrower level for reliable modelling.

A second limitation was the relatively small number of final input features. Although the six behavioural features are directly related to repayment performance, additional variables such as income, employment stability, transaction regularity, mobile-money activity, and business

performance could potentially improve the model if reliable borrower-level data become available.

Finally, the model's performance should be validated over time using newly originated loans. The current results demonstrate predictive ability on historical records, but operational effectiveness depends on whether the model continues to perform well as new borrowers, lending products, and repayment patterns enter the system.

6.16 Chapter Summary

This chapter presented the results of the development and evaluation of the proposed multiclass credit-scoring system. It described the final modelling dataset, the six behavioural features retained after the feature audit, the distribution of borrowers across the four risk categories, and the comparative performance of Logistic Regression, Decision Tree, Random Forest, and XGBoost.

Based on the verified test-set results, XGBoost was identified as the best-performing model, achieving an accuracy of 85.71% and a macro-averaged F1-score of 85.11%. The results provide evidence that behavioural repayment features can support multiclass credit-risk classification despite the limited availability of demographic and alternative data in Hobbiton's existing database.

CHAPTER 7: DISCUSSION

7.1 Introduction

This chapter interprets the results reported in Chapter 6, places them in the context of the literature reviewed in Chapter 2, and considers what they mean for Hobbiton and for future work.

7.2 Interpretation of the Findings

XGBoost gave the strongest results among the four classifiers, with a test accuracy of 85.71% and a macro F1-score of 0.851, ahead of Random Forest (76.19%), Logistic Regression (74.60%) and the Decision Tree (71.43%). Its micro-average ROC-AUC of 0.98 and its cross-validated accuracy of 87.59% (±3.06%) indicate that the advantage did not depend on one favourable split. All four models were trained and tested under identical conditions, so the differences reflect the algorithms rather than the data.

The ordering of the models fits the nature of the problem. The risk label combines timeliness, overdue share, history depth and loan size through thresholds, and tree-based ensembles represent such threshold interactions directly, whereas Logistic Regression can only combine features linearly. Because overdue loan count is informative mainly in relation to loan count, a model that learns interactions has a structural advantage.

The feature audit is itself a finding. The database was designed for operating a lending product, not for research, and its demographic and alternative-data fields were absent or unreliable. Repayment behaviour was therefore the only dependable signal, and that is a realistic picture of what a small Zambian digital lender can offer a scoring model today.

7.3 Comparison with Existing Work

The finding that ensemble methods outperform Logistic Regression agrees with the studies reviewed in Chapter 2. Nerella et al. reported that Random Forest and gradient boosting clearly beat Logistic Regression on fintech lending data [20], and Dastile and Celik reached the same conclusion while stressing the need for explanations of ensemble predictions [19].

Benchmarking studies of classification algorithms for credit scoring point the same way [7], [34]. The use of behavioural transaction data is consistent with work showing that non-traditional and transactional data can support credit assessment where bureau data are thin [8], [9]. The accuracy figures should not be compared directly with published ones, because the datasets, targets and class definitions differ.

7.4 Contributions

- **Practical.** A working, hosted prototype that returns a four-class risk grade and probabilities through a REST interface that Hobbiton's web service can call.
- **Technical.** A layered architecture in which the feature schema lives in one place, so new features need no changes to the gateway or the controller.
- **Empirical.** A documented audit of which feature domains a Zambian BNPL lender's database can support, and a like-for-like comparison of four classifiers on real lender data.
- **Educational.** A documented, reproducible reference for deploying a machine learning model behind a REST interface using free platforms.

7.5 Limitations and Implications

Four limitations qualify the results. First, the target was derived from repayment rules applied to the same behavioural features, so part of the model's accuracy reflects its ability to reproduce that grading rather than to anticipate real defaults. Validation against observed default outcomes is the most important next step. Second, 419 borrowers is a small sample and the test set contains 126, so individual metrics carry uncertainty. Third, the data come from one lender over one period, so the model may not transfer to other lenders or to later conditions without monitoring and retraining. Fourth, XGBoost is harder to interpret than Logistic Regression or a Decision Tree; feature importance mitigates this but does not remove it, and explanation methods such as SHAP would help loan officers trust and challenge individual scores [19].

Excluding demographic variables narrows what the model can learn but also lowers the risk of discriminating on protected characteristics, which matters under the Data Protection Act [42]. Any future extension should weigh predictive gain against fairness and privacy.

7.6 Chapter Summary

The results support the choice of XGBoost on this dataset, agree with the direction of earlier studies, and show that a useful scoring prototype can be built from repayment behaviour alone. They are limited by the rule-derived labels, the small single-lender sample and the narrow feature set, which Chapter 8 addresses in its recommendations.

CHAPTER 8: CONCLUSION AND RECOMMENDATIONS

8.1 Introduction

This chapter summarises the project, states the conclusions that follow from the evidence, assesses the objectives, and gives recommendations and directions for future work.

8.2 Summary of the Project

The project set out to build an automated multiclass credit scoring system for Hobbiton Investment Management. Loan and repayment records were extracted from Hobbiton's PostgreSQL database, consolidated into 419 borrower profiles and described by six behavioural features. Four classifiers were compared, and XGBoost was selected and deployed as a Flask REST service. A PHP gateway and an MVC web interface make the service usable by loan officers, and the whole system was hosted on free platforms and tested end to end.

8.3 Conclusions

- Repayment behaviour held in Hobbiton's database is sufficient to build a useful four-class risk model, although richer domains were not available.
- XGBoost outperformed Random Forest, Logistic Regression and the Decision Tree on accuracy and macro F1, and its advantage held under cross-validation.
- A layered REST architecture with a single feature schema allows the model, gateway and interface to change independently.
- A multiclass risk grade with probabilities gives loan officers more guidance than a single approve-or-reject output.
- The prototype works end to end on free hosting, but its labels and sample size mean it should be validated before it influences real lending decisions.

8.4 Achievement of Objectives

Objective 1 was partially achieved: a feature audit was completed and six behavioural features were engineered, but demographic, insurance, investment and psychometric data were not available. Objectives 2 and 3 were achieved through the data preparation, the four-model comparison and the multi-metric evaluation. Objective 4 was achieved as a prototype, with a hosted REST service, gateway and web interface, while integration into Hobbiton's own production web service remains to be done. Figure 6.9 and 6.10 give the evidence.

8.5 Recommendations

- **Capture richer data.** Begin recording demographic, employment and alternative data at onboarding, with consent and data-protection safeguards, so that future models can use them.
- **Validate on real outcomes.** Record final default and write-off outcomes and retrain the model against them before relying on it for lending decisions.
- **Keep a human in the loop.** Use the score to support loan officers rather than to decide automatically, and monitor agreement between scores and officer decisions.
- **Monitor and retrain.** Track model performance over time and retrain when repayment patterns drift; the versioned artefacts and pipeline support this.
- **Harden the deployment.** Add authentication between the layers, logging and rate limiting at the gateway, and move to always-on hosting before production use.

8.6 Future Work

- Extend the feature set with demographic, psychometric and other alternative data as it becomes available.
- Evaluate with ordinal-aware metrics and tune hyperparameters more extensively, including cost-sensitive thresholds.

- Add explanation methods such as SHAP so each score can be justified to officers and applicants.
- Integrate the scoring service into Hobbiton's production web service and pilot it alongside manual assessment.

8.7 Chapter Summary

The project achieved its main aims within the limits of the available data. It produced a tested, deployed prototype and a documented design, and it identified the data and validation work needed to take the system from prototype to production.

REFERENCES

[1] Bank of Zambia, "Bank of Zambia. (2025). Credit market monitoring report, Q3 2025," Lusaka, Dec. 2025. Accessed: Jun. 14, 2026. [Online]. Available: https://www.boz.zm/publications/credit-market-monitoring-reports

[2] Republic of Zambia, "National Financial Inclusion II 2024-2028," Lusaka, Mar. 2024. Accessed: Jun. 14, 2026. [Online]. Available: https://www.mofnp.gov.zm/rfu/wp-content/uploads/2025/09/NFIS_II_2024_-_2028.pdf

[3] World Bank, "Financial Access," Washington, D.C, 2021. Accessed: Jun. 15, 2026. [Online]. Available: https://documents1.worldbank.org/curated/en/099833507072223098/pdf/IDU0f917b4120aa0f042270bbd10c8b257c99a54.pdf

[4] A. Demirguc-Kunt, L. Klapper, D. Singer, S. Ansar, and J. Hess, *The Global Findex Database 2017: Measuring Financial Inclusion and the Fintech Revolution*. Washington, DC: World Bank, 2018. doi: 10.1596/978-1-4648-1259-0.

[5] L. Thomas, D. Edelman, and N. Crook, *Credit scoring and its applications*, 2nd ed. Society for Industrial and Applied Mathematics, 2017.

[6] D. J. Hand and W. E. Henley, "Statistical Classification Methods in Consumer Credit Scoring: A review," *J. R. Stat. Soc. Ser. A Stat. Soc.*, vol. 160, no. 3, pp. 523–541, 1997.

[7] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," *Journal of the Operational Research Society*, vol. 54, no. 6, pp. 627–635, Jun. 2003, doi: 10.1057/palgrave.jors.2601545.

[8] L. Gambacorta, Y. Huang, H. Qiu, and J. Wang, "How Do Machine Learning and Non-Traditional Data Affect Credit Scoring? New Evidence from a Chinese Fintech Firm," 834, Dec. 2019. [Online]. Available: www.bis.org

[9] D. Björkegren and D. Grissen, “Behaviour revealed in mobile phone usage predicts credit repayment,” *World Bank Econ. Rev.*, no. 3, Art. no. 9074, Oct. 2020, doi: https://doi.org/10.1093/wber/lhz006.

[10] Alliance for Financial Inclusion (AFI), “Alternative Data for Credit Scoring,” Jan. 2025. Accessed: Jun. 22, 2026. [Online]. Available: https://afi-global.org/wp-content/uploads/2025/02/Alternative-Data-for-Credit-Scoring.pdf

[11] D. Huang, J. Zhou, and H. Wang, “RFMS Method for Credit Scoring Based on Bank Card Transaction Data,” *Stat. Sin.*, vol. 28, no. 4, Art. no. 28, 2018, Accessed: Jun. 23, 2026. [Online]. Available: https://www.jstor.org/stable/26511243

[12] V. Aggelis, “RFM analysis with Data Mining,” Patras, 2004.

[13] C. Yan, X. Zhang, and J. Shen, “Credit Score Classification Using Advanced Machine Learning: A Comprehensive Approach,” *Journal of Software Engineering and Applications*, vol. 18, no. 03, pp. 98–112, 2025, doi: 10.4236/jsea.2025.183007.

[14] A. Zheng and A. Casari, *Feature Engineering for Machine Learning*. O’Reilly Media, Inc, 2018.

[15] S. A. Ionescu, V. Diaconita, and A. O. Radu, “Engineering Sustainable Data Architectures for Modern Financial Institutions,” *Electronics (Basel).*, vol. 14, no. 8, p. 1650, Apr. 2025, doi: 10.3390/electronics14081650.

[16] D. Shukla and S. Gupta, “Feature Engineering Techniques to Enhance Credit Scoring Models,” in *2024 6th International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*, IEEE, 2024, pp. 1–5.

[17] H.-W. Teng, J.-Y. Lin, and K.-W. Lu, “Enhancing Credit Score Predictions with Dynamic Feature Engineering using Deep Learning,” *SSRN Electronic Journal*, 2023, doi: 10.2139/ssrn.4375313.

[18] A. E. Khandani, A. J. Kim, and A. W. Lo, “Consumer Credit Risk Models via Machine-Learning Algorithms *.” [Online]. Available: http://www.federalreserve.gov/releases/chargeoff/.

[19] X. Dastile and T. Celik, “Making Deep Learning-Based Predictions for Credit Scoring Explainable,” *IEEE Access*, vol. 9, pp. 50426–50440, 2021, doi: 10.1109/ACCESS.2021.3068854.

[20] A. Nerella, N. Kolli, B. S. Sahu AI Engineer, G. Gladson Battu, and T. Preethi Priyanka, “AI-Driven Risk Assessment Models for Personalized Credit Scoring in Emerging FinTech Ecosystems,” in *The 16th International IEEE Conference on Computing, Communication and Networking Technology (ICCCNT)*, Jul. 2025, pp. 1–5.

[21] Y. Song, Y. Wang, X. Ye, R. Zaretzki, and C. Liu, “Loan default prediction using a credit rating-specific and multi-Loan default prediction using a credit rating-specific and multi-objective ensemble learning schemebjective ensemble learning scheme,” *Inf. Sci. (N. Y).*, vol. 629, pp. 599–617, Jun. 2023.

[22] A. Umagba, B. Abara, Z. Isa, E. Okoro, and M. Yahaya, “A multi class machine learning model for predicting credit default in credit risk management.” [Online]. Available: https://ssrn.com/abstract=4105836

[23] M. Anis and M. Ali, “Investigating the Performance of Smote for Class Imbalanced Learning: A Case Study of Credit Scoring Datasets,” *European Scientific*, vol. 13, no. 33, pp. 340–353, Nov. 2017, doi: 10.19044/esj.2017.v13n33p340.

[24] M. Schmitt, “Explainable Automated Machine Learning for Credit Decisions: Enhancing Human Artificial Intelligence Collaboration in Financial Engineering.”

[25] Bank of Zambia, “Zambia FinScope 2025 Survey Topline Findings,” Lusaka, 2025. Accessed: Jun. 21, 2026. [Online]. Available: https://www.boz.zm/sites/default/files/2026-04/Zambia%20Finscope%202025%20survey%20topline%20findings.pdf

[26] M. Mwela, “Building capacity to mobilize & align finance for development Financial Technology and Financial Inclusion in Zambia,” University of Cape Town, Cape Town, 2024.

[27] R. Hirk, K. Hornik, and L. Vana, “Multivariate ordinal regression models: an analysis of corporate credit ratings,” *Stat. Methods Appt.*, vol. 28, no. 3, pp. 507–539, Sep. 2019, doi: 10.1007/s10260-018-00437-7.

[28] M. Desprez, K. Zawada, and D. Ramp, “Overcoming the ordinal imbalanced data problem by combining data processing and stacked generalizations,” *Machine Learning with Applications*, vol. 7, p. 100241, Mar. 2022, doi: 10.1016/j.mlwa.2021.100241.

[29] Á. Paz *et al.*, “Machine Learning and Metaheuristics Approach for Individual Credit Risk Assessment: A Systematic Literature Review,” *Biomimetics*, vol. 10, no. 5, p. 326, May 2025, doi: 10.3390/biomimetics10050326.

[30] S. Osei, N. B. Mpinda, K. J. Sadefo, and J. Fadugba, “Accuracies of some Learning or Scoring Models for Credit Risk Measurement,” hal-03194081, Apr. 2021. Accessed: Jun. 21, 2026. [Online]. Available: https://hal.science/hal-03194081

[31] E. Amigo, J. Gonzalo, S. Mizzaro, and J. Carrillo-de-Albornoz, “An Effectiveness Metric for Ordinal Classification: Formal Properties and Experimental Results,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 3938–3949. doi: 10.18653/v1/2020.acl-main.363.

[32] A. Galdran, “Performance Metrics for Probabilistic Ordinal Classifiers,” in *Medical image computing and computer assisted intervention -- MICCAI 2023*, 3rd ed., vol. 14222, 2023, pp. 357–366. doi: 10.1007/978-3-031-43898-1_35.

[33] W. Silva, R. J. Pinto, and J. S. Cardoso, “A Uniform Performance Index for Ordinal Classification with Imbalanced Classes,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2018, pp. 1–8.

[34] I. Brown and C. Mues, “An experimental comparison of classification algorithms for imbalanced credit scoring data sets,” *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3446–3453, Feb. 2012, doi: 10.1016/j.eswa.2011.09.033.

[35] “Cracking the Credit Code: Alternative Data and AI for Financial Inclusion Report About IFC.” [Online]. Available: www.ifc.org.

[36] J. Chamangwa, “An analysis of the implementation of automated credit risk management strategy on loan recovery rate among Standard bank Zambia clients in Lusaka district, Zambia,” The University of Zambia, 2022.

[37] B. I. Adekunle, S. Owoade, E. Ogbuefi, T. O. Odofin, A. O. Agboola, and S. O. Abanigbo, “Using Python and microservices for real-time credit risk assessment in

embedded lending systems," *Risk and Financial Management*, vol. 4, no. 8, pp. 206–218, 2021.

[38] G. A. Gajwani, "Microservices Architecture for Loan Trading Platforms: A Digital Transformation Approach," *World Journal of Advanced Research and Reviews*, vol. 26, no. 2, pp. 3629–3637, May 2025, doi: 10.30574/wjarr.2025.26.2.2014.

[39] M. Abdulsatar, H. Ahmad, D. Goel, and F. Ullah, "Towards deep learning enabled cybersecurity risk assessment for microservice architectures," *Cluster Comput.*, vol. 28, no. 6, Oct. 2025, doi: 10.1007/s10586-024-05092-0.

[40] M. R. Machado, D. T. Chen, and J. R. Osterrieder, "An analytical approach to credit risk assessment using machine learning models," *Decision Analytics Journal*, vol. 16, p. 100605, Sep. 2025, doi: 10.1016/j.dajour.2025.100605.

[41] F. Mwape and E. K. Mukuka, "Enhancing radiology workflows: Semi-automated cervical cancer reporting at the Cancer Diseases Hospital in Zambia," The University of Zambia, Lusaka, 2025.

[42] Government of Zambia, "Data Protection Act No. 3 of 2021," Lusaka, 2021. Accessed: Jun. 21, 2026. [Online]. Available: https://www.parliament.gov.zm/sites/default/files/documents/acts/Act%20No.%203%20The%20Data%20Protection%20Act%202021_0.pdf

[43] Y. Zhou, S. Aryal, and M. R. Bouadjenek, "Review for Handling Missing Data with special missing mechanism," Apr. 2024.

[44] J. Xu *et al.*, "Adaptive Feature Projection with Distribution Alignment for Deep Incomplete Multi-View Clustering," *IEEE Transactions on Image Processing*, vol. 32, pp. 1354–1366, 2023, doi: 10.1109/TIP.2023.3243521.

[45] R. Modak and V. G. Avula, "Efficient Feature Store Architectures for Real-time Machine Learning Model Deployment in High-Throughput Systems," *SSRN Electronic Journal*, 2025, doi: 10.2139/ssrn.5317167.

[46] B. J. Hutagaol, R. S. Sitorus, and D. M. Simanjuntak, "Machine Learning for Securing API Gateways: A Systematic Literature Review," *Journal of Computer Networks, Architecture and High-Performance Computing*, vol. 7, no. 3, pp. 982–993, Aug. 2025, doi: 10.47709/cnahpc.v7i3.6788.

[47] S. Horchidan, E. Kritharakis, V. Kalavri, and P. Carbone, “Evaluating model serving strategies over streaming data,” in *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, New York, NY, USA: ACM, Jun. 2022, pp. 1–5. doi: 10.1145/3533028.3533308.

[48] B. Derakhshan, R. A. Mahdiraji, T. Rabl, and V. Markl, “Continuous Deployment of Machine Learning Pipelines,” in *e 22nd International Conference on Extending Database Technology (EDBT)*, OpenProceedings, Apr. 2019, pp. 397–408.

[49] Object Management Group, “OMG Unified Modeling Language (OMG UML), Version 2.5.1,” Standard, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/

[50] Roy Thomas Fielding, “Architectural styles and the design of network-based software architectures,” University of California, Irvine, California, 2000.

[51] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” pp. 785–794, Jun. 2016, doi: 10.1145/2939672.2939785..

[52] F. Assunção, N. Lourenço, B. Ribeiro, and P. Machado, “Evolution of Scikit-Learn Pipelines with Dynamic Structured Grammatical Evolution,” Apr. 2020.

[53] Pallets Projects, “Flask documentation.” Accessed Oct. 2026. [Online]. Available: https://flask.palletsprojects.com

[54] Render, “Deploy for free — Render documentation.” Accessed Oct. 2026. [Online]. Available: https://render.com/docs/free

[55] InfinityFree, “Free web hosting.” Accessed Oct. 2026. [Online]. Available: https://www.infinityfree.com

APPENDICES

Appendix A: API Specification and Sample Payloads

Request to POST /api/score (gateway) or POST /v1/predict (Flask):

```
{
  "avg_loan_amount": 1500,
  "avg_installments": 1,
  "loan_count": 3,
  "avg_days_late": 1,
  "max_days_late": 4,
  "overdue_loan_count": 0
}
```

Flask response (200):

```
{
  "risk": "medium",
  "probabilities": {"low": 0.159, "medium": 0.722, "high": 0.071, "very high":
0.047},
  "model_version": "v1"
}
```

Gateway response (200):

```
{
  "success": true,
  "prediction": { "risk": "medium", "probabilities": {...}, "model_version":
"v1" }
}
```

Error responses:

```
400  {"error": "Missing required fields: [...]"}
400  {"error": "Request body must be non-empty valid JSON"}        (gateway)
502  {"error": "Could not reach the credit scoring service", "detail": "..."}
```

Appendix B: Postman Test Collection

Table B.1: Requests in the Postman collection

Folder	Request	Purpose
Python Model Service	Health Check	Confirm the service and model version
Python Model Service	Predict - Valid Applicant	Normal prediction
Python Model Service	Predict - Missing Fields	Demonstrate 400 validation
PHP Gateway	Welcome	Confirm the gateway is reachable

Folder	Request	Purpose
PHP Gateway	Score - Valid Applicant	Prediction through the gateway
PHP Gateway	Score - Invalid JSON	Gateway rejects bad input
PHP Gateway	Score - Missing Fields	Flask validation relayed unchanged
PHP Gateway	Score - Upstream Unreachable	Demonstrate 502 handling

The collection uses two environment variables, flask_base and gateway_base, so the same requests run against local or hosted services.

Appendix C: Deployment Configuration

requirements.txt (scoring service):

```
flask
pandas
scikit-learn
xgboost
joblib
gunicorn
```

Render settings: build command pip install -r requirements.txt; start command gunicorn app:app; instance type Free.

Configuration constants:

```
// credit-gateway/config.php
define('PYTHON_SERVICE_URL', 'https://credit-rating-engine.onrender.com');
// credit-frontend/config.php
define('GATEWAY_URL', 'https://credit-gateway.infinityfreeapp.com');
```

Appendix D: Feature Engineering Source Code

```
def build_member_features(bnpl):
    agg = bnpl.groupby("Id").agg(
        TotalRepaid=("RepaymentAmount", "sum"),
        NumberOfInstallments=("RepaymentAmount", "count"),
        MostRecentRepaymentDate=("RepaymentDate", "max")).reset_index()
    summary = (bnpl.drop(["RepaymentAmount", "RepaymentDate", "RepaymentType"],
axis=1)
                   .drop_duplicates(subset="Id").merge(agg, on="Id",
how="left"))
    summary["Loan_count"] = bnpl.groupby("MemberId")["Id"].transform("count")
    rep = summary[summary["MostRecentRepaymentDate"].notnull()].copy()
    r = pd.to_datetime(rep["MostRecentRepaymentDate"], format="ISO8601",
utc=True)
```

```
    m = pd.to_datetime(rep["MaturityDate"], format="ISO8601", utc=True)
    rep["DaysLate"] = (r - m).dt.total_seconds() / 86400
    return rep.groupby("MemberId").agg(
        avg_loan_amount=("LoanAmount", "mean"),
        avg_installments=("NumberOfInstallments", "mean"),
        loan_count=("Loan_count", "max"),
        avg_days_late=("DaysLate", "mean"),
        max_days_late=("DaysLate", "max"),
        overdue_loan_count=("LoanStatus", lambda s: (s == "overdue").sum()),
    ).reset_index()
```