PREDICTIVE ANALYTICS SYSTEM FOR SMALL BUSINESS BANKRUPTCY RISKS

A PROJECT REPORT

Submitted by

PREETHAA U **311521104033**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2025

MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"PREDICTIVE ANALYTICS SYSTEM FOR SMALL BUSINESS BANKRUPTCY RISKS "** is the Bonafide work of " **PREETHAA U (311521104033)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S. Aarthi,M.E,Ph.D	Dr B.Monica Jenefer M.E, Ph.d
HEAD OF THE DEPARTMENT	**SUPERVISOR** **PROFESSOR**
Computer Science and Engineering	Computer Science and Engineering
Meenakshi Sundararajan Engineering College, (An Autonomous Institution)	Meenakshi Sundararajan Engineering College, (An Autonomous Institution)
No.363, Arcot Road, Kodambakkam, Chennai -600 024.	No.363, Arcot Road, Kodambakkam, Chennai -600 024.

Submitted for the project viva voce of Bachelor of Engineering in Computer Science and Engineering held on____________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we express our sincere gratitude to our Respected Correspondent **Dr. K. S. Lakshmi**, our beloved Secretary **Mr. N. Sreekanth**, Principal **Dr S.V. Saravanan** for their constant encouragement, which has been our motivation to strive towards excellence.

Our primary and sincere thanks go to **Dr S. Aarthi**, Head of the Department, Department of Computer Science and Engineering, for her profound inspiration, kind cooperation and guidance.

We are grateful to **Dr. B Monica Jenefer**, Internal Guide, Professor, Department of Computer Science and Engineering. We are extremely thankful and indebted for sharing expertise, sincere and valuable guidance and encouragement extended to us.

We would also like to thank all teaching and non-teaching staff of our department for sharing their knowledge and constant support during the course of the project.

We are grateful to our external guide **Mrs. S Jenita karthiga**, Senior Technical Associate at Skyware Solutions (p) ltd.,Chennai for giving us the opportunity to work on this project at their company. We also thank him for providing the necessary tools and guidance during the development of the project.

Above all, we extend our thanks to God Almighty without whose grace and blessings it wouldn't have been possible.

ABSTRACT

Small businesses face significant financial challenges, underscoring the need for early bankruptcy risk detection to ensure stability and informed decision-making. This paper introduces a predictive analytics system with a user-friendly interface (UI) that visualizes risk assessments and empowers businesses to take preventive actions. The proposed system leverages machine learning and artificial intelligence to predict bankruptcy risks using diverse financial indicators and historical data patterns. By analyzing a dataset containing over 20,000 entries, the system trains various classifiers to accurately forecast potential bankruptcies. The Random Forest Classifier (RFC) and feature selection techniques emerge as top-performing strategies after extensive evaluation, enabling businesses to intervene early or seek financial support. Additionally, the system provides dynamic risk scoring that updates with new financial inputs, offering continuous monitoring capabilities. The design focuses on simplicity, accessibility, and interpretability, ensuring that non-technical users can easily leverage insights for critical decisions. The model's effectiveness is evaluated using key performance indicators, including F1-score, accuracy, recall, and precision, enhanced by an interactive dashboard. This paper presents a detailed exploration of the system's methodology, real-world applicability, and performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND EXPANSIONS

ABBREVIATIONS	EXPANSIONS
AI	Generative adversarial networks
ML	Machine Learning
LR	Advanced Encryption Standard
RF	Random-access memory
TFIG	Open Source Computer Vision Library
UML	Unified Modeling Language
HDD	Hard Disk Drive
GB	Gigabyte
TB	Terabyte
IEEE	Institute of Electrical and Electronics Engineers

CHAPTER 1
INTRODUCTION

1.1 ABOUT THE PROJECT

Small business bankruptcy poses a significant challenge, often leading to economic instability. This project aims to develop a predictive analytics system to assess bankruptcy risks for small businesses by leveraging advanced machine learning techniques. The system employs algorithms such as Random Forest and Support Vector Machines (SVM) to analyze a diverse dataset of financial indicators, enabling the prediction of potential financial distress. By processing data from multiple sources, the system identifies patterns that are indicative of financial trouble, allowing for early detection. Once the risk of bankruptcy is identified, the system provides actionable insights and risk mitigation strategies, empowering business owners and stakeholders to make informed decisions and enhance financial stability.

This approach not only improves the accuracy of bankruptcy risk prediction but also ensures timely identification, helping businesses take proactive measures to avoid financial collapse. The integration of actionable insights further enhances the user experience, offering a user-friendly platform for stakeholders to assess risks, implement corrective actions, and safeguard financial stability, ultimately fostering a more secure business environment.

1.2 DOMAIN OVERVIEW

The domain of this project lies at the intersection of financial technology and risk management, specifically addressing the growing challenge of predicting bankruptcy in small businesses. As small businesses continue to play a vital role in the global economy, accurate prediction of financial distress becomes crucial for early intervention. This dual focus of financial stability and data-driven decision-making drives the need for advanced analytics to forecast bankruptcy risks and enable businesses to take timely actions to safeguard their future.

Machine learning (ML) plays a central role in this project, utilizing its capabilities to analyze and predict bankruptcy risks for small businesses. ML algorithms such as Random Forest and Support Vector Machines (SVM) are essential for identifying financial patterns and risk indicators that may signal potential business failure. This capability arises from ML's core ability to process large datasets and detect intricate trends—an ability that has advanced dramatically with the development of machine learning techniques over the years. Modern ML models harness powerful computational resources and sophisticated algorithms to carry out tasks ranging from financial analysis to optimizing business operations, forming the backbone of this bankruptcy prediction system.

Bankruptcy prediction represents a critical financial challenge for small businesses, with factors such as cash flow issues, poor management, and market conditions contributing to business failures. Business owners often face difficulty in identifying early signs of distress, making it hard to take corrective action. The dynamic and unpredictable nature of the business environment can worsen these issues, as businesses struggle to adapt to rapid changes in the market and economic conditions. This project acknowledges the complexity of financial distress and aims to address it using machine learning-driven tools. These tools are designed not only to predict the likelihood of bankruptcy but also to provide actionable insights and risk management strategies, enabling business owners to take proactive steps and ensure financial stability.

This domain requires a careful balance between financial analysis and business ethics. The integration of machine learning into bankruptcy prediction systems must respect business confidentiality and data security while effectively identifying risks. Additionally, the project aims to contribute to the broader conversation on responsible technology use, highlighting the need for tools that support financial stability without compromising sensitive information. By addressing both the technical challenges of predicting bankruptcy and the ethical implications of using such predictive models, this project aligns with ongoing efforts to enhance decision-making processes and safeguard businesses in the financial landscape.

1.3 EXISTING SYSTEM

The existing systems for predicting bankruptcy in small businesses primarily rely on financial statements, historical data, and basic statistical models. These traditional methods are largely reactive, identifying risk only after financial distress has begun to manifest, often too late to prevent business failure. This reliance on past performance data places a significant burden on businesses, requiring them to recognize signs of distress early enough to take action, which can be a complex and overwhelming process for owners already facing financial challenges.

Financial models, while useful for assessing historical data, often fail to account for the nuances of business operations, such as market shifts, management decisions, or unforeseen external factors. This limitation can lead to inaccurate predictions, either overestimating the risk of bankruptcy (false positives) or missing subtle warning signs of financial distress (false negatives), which may ultimately undermine the model’s effectiveness in providing early intervention insights.

Research in this field suggests that while machine learning models have shown promise in predicting bankruptcy, their application in small businesses is still limited. Simpler, rule-based approaches remain more common due to their lower implementation costs and ease of integration. Moreover, the challenges of predicting financial distress in real-time highlight the need for advanced analytics and algorithms, such as those used in machine learning, to better understand financial trends and improve the accuracy of bankruptcy prediction systems.

Studies have highlighted the effectiveness of using multiple machine learning models to improve the accuracy of bankruptcy prediction systems. When trained on diverse financial datasets, these models can minimize the likelihood of false positives while enhancing the system's ability to identify different types of financial distress. The potential of AI-driven systems to not only predict but also anticipate financial trouble before it culminates in bankruptcy opens new possibilities for proactive risk management and timely interventions.

1.4 PROBLEM STATEMENT

As the frequency and complexity of financial risks continue to grow, traditional bankruptcy prediction methods are struggling to keep up with the dynamic nature of small business environments. Current strategies often rely on static financial ratios and historical data, which fail to capture the evolving challenges and financial behaviors of modern businesses. This creates a significant gap in accurately forecasting bankruptcy risks. Addressing this issue is critical for enabling timely interventions and ensuring the stability of small businesses. This research aims to explore the integration of machine learning techniques into bankruptcy prediction, aiming to enhance the accuracy of financial distress forecasts and provide more reliable early warnings for businesses at risk.

1.5 CHAPTER OVERVIEW

The project report is organised with various chapters that denote the various functionalities and aspects of the system being developed.

Chapter 1 gives a general description of the project. It represents the basic idea of the project and introduces the topics of the existing system and the proposed system.

Chapter 2 deals with the related works of the project. A literature review for each related work is explained in detail.

Chapter 3 presents the system architecture and requirements. It specifies the hardware and software components that are required. It also lists the technologies used in the implementation of the project.

Chapter 4 explains the system design with the use of UML diagrams and data flow diagrams.

Chapter 5 contributes a detailed description of different modules that are there in the design and how they are implemented.

Chapter 6 gives a detailed description of the different test cases that were performed on the system.

Chapter 7 provides the conclusion. It also elucidates how the project can be further enhanced.

CHAPTER 2

LITERATURE SURVEY

2.1 A Bankruptcy Prediction Model using Random Forest

Shreya Joshi, Rachana Ramesh and Shagufta Tahsildar (2018, ICICCS)

This paper highlights the importance of early bankruptcy prediction models to support financial stability and informed decision-making. They propose using Random Forest algorithms to enhance prediction accuracy and reliability in identifying at-risk businesses. The authors underline the crucial impact of advanced predictive techniques in helping organizations take preventive measures and minimize the negative consequences of financial failure.

2.2 Bankruptcy Prediction for Small and Medium Enterprises Using Machine Learning Techniques

Joaquín García, Rubén Sánchez Prieto, and Marisa Gil (2019, IEEE Access)

This paper focuses on the application of machine learning methods to predict bankruptcy in small and medium enterprises (SMEs). They utilize algorithms like Random Forest and Support Vector Machines to improve prediction performance and reliability. The authors emphasize the importance of accurate risk assessment tools in supporting financial decision-making and reducing the impact of business failures on the economy.

2.3 Predicting Bankruptcy Using Machine Learning Techniques: An Empirical Study

Seyed Mohammadreza Davari, Abdul Mutalib Leman, Norhazilan Md Noor (2021, IEEE International Conference on Artificial Intelligence in Engineering and Technology)

This paper proposed a bankruptcy prediction method that uses multiple machine learning techniques, including Random Forest, SVM, and Gradient Boosting. Financial data from companies is classified as bankrupt or non-bankrupt. The authors combined different datasets and measured the performance of their proposed models. Seventy-five percent of the data is selected as training data and the rest is used for testing. The best-performing model achieves an accuracy of 92%. In addition to accuracy, the models are also evaluated using four other important performance metrics.

2.4 Bankruptcy Prediction Using Random Forests with Balanced Sampling

Cristian Bravo, David Maldonado, Sergio Weber (2013, Intelligent Systems Conference - IEEE)

This paper evaluates bankruptcy prediction techniques, focusing on the use of Random Forests with balanced sampling to address data imbalance issues. They compare traditional methods with ensemble classifiers, with Random Forest achieving strong prediction accuracy. Challenges include limited availability of bankruptcy cases and the need for better feature selection. The authors suggest future research directions, recommending advanced resampling techniques and feature optimization methods to further improve predictive performance.

2.5 Financial Crisis Prediction Using Support Vector Machines

Dinh Tran Ngoc Huy, D. D. Luu, Le Thi Thanh Thao (2020, WSEAS Transactions on Business and Economics)

In this paper, the authors explore how Support Vector Machines (SVM) can be applied to predict financial crises, focusing on analyzing financial indicators to assess risk levels. They discuss the use of SVM for classifying businesses at risk of failure and highlight strategies for selecting relevant financial features.

The authors suggest that SVM models can significantly improve early detection and decision-making processes. This study offers valuable insights for financial institutions seeking to mitigate risks and enhance predictive capabilities in crisis management.

2.6 Predicting Financial Distress in Small Businesses: The Role of Machine Learning Algorithms

Amine Tarazi, Sonia Ben Amor, Jihen Jouida (2020, International Journal of Financial Studies)

In this paper, the authors explore the role of machine learning algorithms in predicting financial distress in small businesses. They discuss how advancements in machine learning, particularly algorithms like Random Forest and SVM, have enhanced the accuracy of financial risk predictions. The review highlights the benefits of machine learning, including its ability to process large datasets, identify patterns, and provide early warning signals for potential financial trouble. The authors emphasize the potential of these techniques to support businesses in making data-driven decisions and improve their financial stability. These findings suggest that machine learning algorithms are a valuable tool for enhancing financial risk management in small businesses.

CHAPTER 3
SYSTEM ARCHITECTURE

3.1 PROJECT ARCHITECTURE

The architecture of the bankruptcy prediction system follows a structured workflow that begins with the Data Collection phase, where a comprehensive dataset of financial indicators from small businesses is compiled. During the Data Preprocessing step, this dataset undergoes cleaning, normalization, and feature extraction to ensure that the data is suitable for analysis. The cleaned data is then fed into the Model Training phase, where machine learning algorithms, specifically a Random Forest Classifier, are trained to identify patterns and trends that predict bankruptcy risks based on financial variables.

Following model training, the Real-Time Prediction phase begins. It involves continuously collecting real-time financial data from small businesses, which is then processed into structured formats during the Data Conversion step. This structured data undergoes the same preprocessing steps as the training data to maintain consistency. The processed real-time data is then passed through the Predictive Model phase, where the trained Random Forest Classifier is applied to predict the risk of bankruptcy based on the latest financial indicators.

If bankruptcy risk is predicted, the system triggers the Action Mechanism, allowing businesses to take immediate steps, such as consulting a financial advisor or adjusting their financial strategy based on the predictions. This mechanism is integrated into the User Interface (UI), which is designed to be intuitive and accessible, offering easy-to-follow guidance. The interface is further enhanced with a Financial Assistant, which helps users navigate the system, interpret the prediction results, and access support, ensuring that the system is both practical and user-friendly.

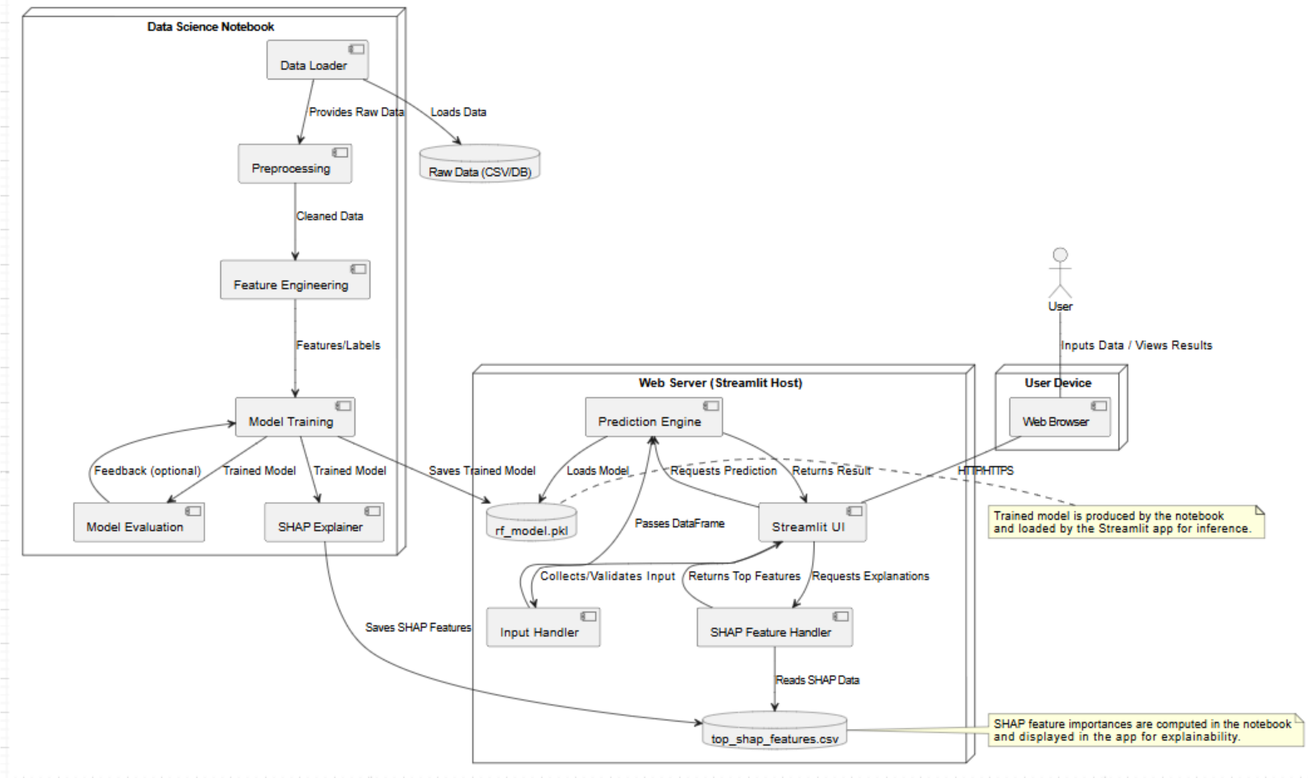


Figure 3.1 Project Architecture

3.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in the case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

S.No	**REQUIREMENTS**	**RECOMMENDED**	**MINIMUM REQUIREMENTS**
1	Operating System	Windows 11	Windows 10
2	RAM	10 GB	6 GB
3	HDD	1 TB	500 GB
4	Processor	Intel Quad Core	Intel Dual Core

TABLE 3.1 Hardware Requirements

3.3 SOFTWARE REQUIREMENTS

REQUIREMENTS	SPECIFICATION
FRONT END	STREAMLIT
TOOL	COLAB
CODING LANGUAGE	PYTHON
DATASET	.csv FILES
STORAGE	HDD/SSD

Table 3.2 Software Requirements

3.3.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy-to-learn syntax emphasises readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

3.3.2 COLAB

Colab is a cloud-based platform used to create and modify code, scripts, and documents. It provides an interactive environment for writing and

executing Python code, making it a popular tool for data analysis, machine learning, and research. Colab offers features like real-time collaboration, access to powerful computational resources, and the ability to run code on GPUs, making it a versatile choice for both beginners and advanced users.

3.3.3 WEB BROWSER

A web browser is a software application used to access information on the World Wide Web. It interprets and displays web pages, allowing users to navigate between them using hyperlinks. Common examples include Google Chrome, Mozilla Firefox, and Microsoft Edge. Browsers communicate with web servers via HTTP or HTTPS protocols to retrieve web pages and their associated resources such as images and scripts. They also support features like bookmarks, extensions, and private browsing modes. Modern browsers are highly customizable and often include tools for web development and debugging. Overall, web browsers play a crucial role in accessing and interacting with online content.

CHAPTER 4
SYSTEM MODELLING

4.1 UNIFIED MODELLING LANGUAGE(UML)

Unified Modeling Language is a standardised modelling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualising, constructing, and documenting the artefacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The primary goals in the design of the UML are as follows:

- Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.
- Provide extensibility and specialisation mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modelling language.
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.

4.2 USE CASE DIAGRAM

The use case diagram is used to define the core elements and processes that make up a system. The key elements are termed as “actors” and the processes are called “use cases”. The use case diagram shows which actors interact with each use case. This definition defines what a use case diagram is primarily made up of – actors and use cases.

In software and system engineering, a use case is a list of steps, typically defining interactions between a role (known in UML as an “actor”) and a system, to achieve a goal. The actor can be a human or an external system. In system engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals.

The purposes of use case diagrams can be as follows:

1. Used to gather requirements of a system.
2. Used to get an outside view of a system.
3. Identify external and internal factors influencing the system.
4. Showing the interaction among the requirements are actors.

Use cases help in identifying the operations that can be performed by an actor. It gives a list of the various applications that can be utilised by the system. The actor can be a real-time human or a system. It helps in identifying the various modules present in the system. A single-use case diagram captures a particular functionality of a system. Hence to model the entire system, a number of use case diagrams are used.

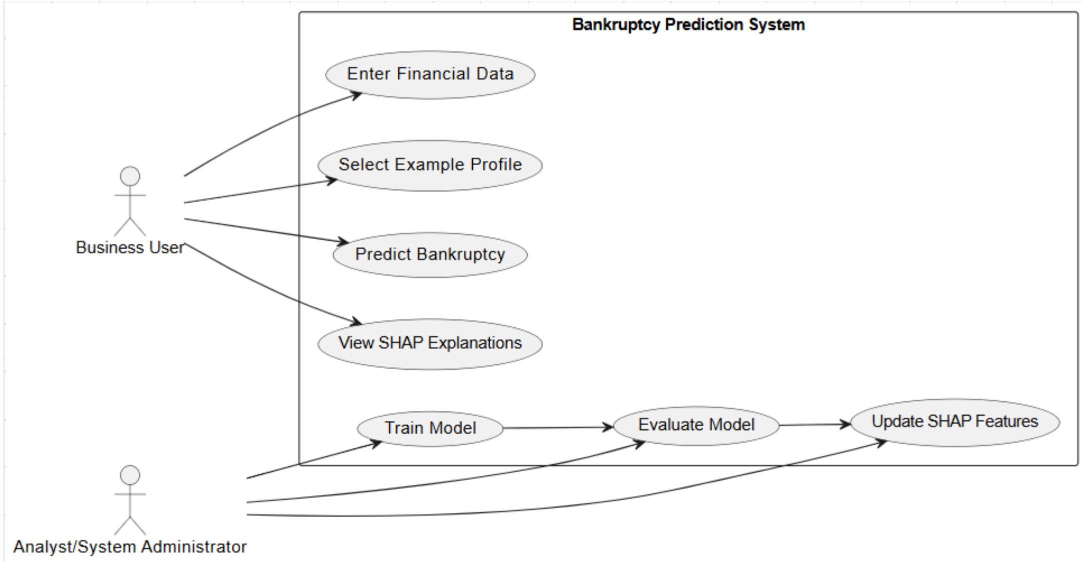


Figure 4.1 Use Case Diagram

A Use Case Diagram provides a graphical overview of the interactions between the users (actors) and the system. For the bankruptcy prediction system:

- **Actors**: Users, System Administrator, Prediction Model

  Use Cases:

- Users can submit financial data (e.g., balance sheets, ratios) for bankruptcy risk analysis.
- Users can view prediction results indicating the likelihood of business bankruptcy.
- Users can interpret model outputs through SHAP-based visual explanations.
- System Administrators can retrain and evaluate machine learning models with new data.
- Prediction Model processes historical and newly submitted financial data to generate bankruptcy predictions.
- Streamlit Dashboard presents interactive visualizations of model results and influential financial features.

4.3 CLASS DIAGRAM

The class diagram is a static diagram. It is the building block of every object-oriented system and helps in visualising and describing the system. A class diagram depicts the structure of the system through its classes, attributes, operations and relationships among the objects. A class is a blueprint that defines the variables and methods common to all objects of a certain kind. The class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints.

The characteristics of a Class Diagram are:

1. Each class is represented by a rectangle having a subdivision of three compartments - name, attributes and operations.
2. There are three types of modifiers which are used to decide the visibility of attributes and operations: + is used for public visibility, # is used for protected visibility, and – is used for private visibility.

In the diagram, classes are represented with boxes that contain three compartments. The top compartment contains the name of the class. It is printed in bold and centred, and the first letter is capitalised. The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase. The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

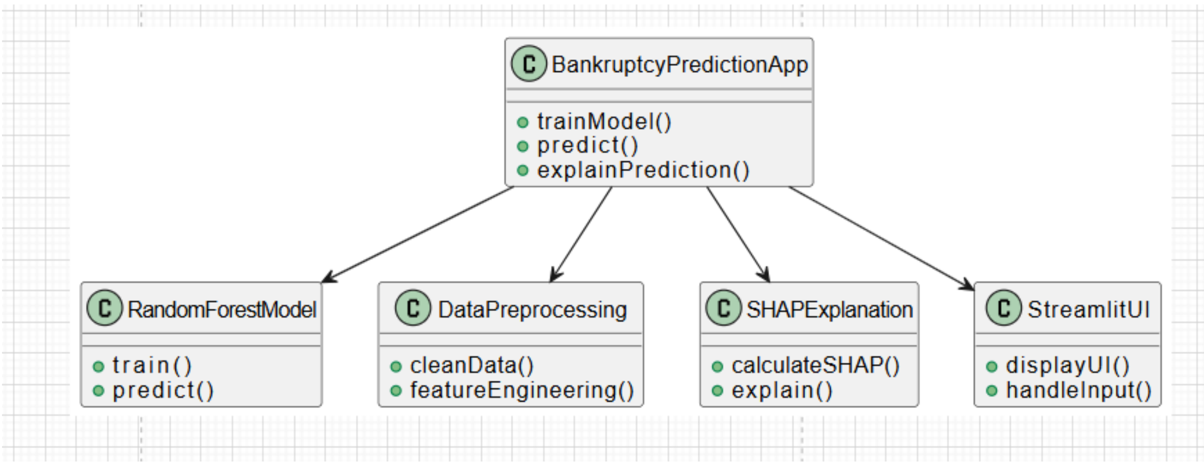


Figure 4.2 Class Diagram

The main classes involved in this diagram are the BankruptcyPredictionApp, RandomForestModel, DataPreprocessing, SHAPExplanation, StreamlitUI

- BankruptcyPredictionApp methods include: trainModel(), predict(), explainPrediction()
- RandomForestModel methods include: train(), predict()
- DataPreprocessing methods include: cleanData(), featureEngineering()
- SHAPExplanation methods include: calculateSHAP(), explain()
- StreamlitUI methods include: displayUI(), handleInput()

4.4 OBJECT DIAGRAM

An object diagram in the Unified Modeling Language (UML) models the static structure of a system at a particular point in time by showing instances of classes (objects), their attributes, and the links (relationships) between them. While class diagrams represent abstract design blueprints, object diagrams capture specific scenarios or snapshots during execution, illustrating how objects interact and relate under concrete conditions.

In an object diagram, objects are depicted as rectangles with their names underlined, along with their attribute values. Links between objects are shown as lines connecting them, representing actual associations in a specific instance of system operation. Object diagrams are useful in understanding real-time behavior, validating class diagrams, and visualizing test data or example configurations.

The purposes of object diagrams can be summarized as follows:

1. Represent a snapshot of system instances and their states at a specific moment.
2. Validate the structure and behavior of a class diagram through concrete examples.
3. Assist in debugging and test case generation by showing object configurations.
4. Clarify the interactions between specific object instances in real-world scenarios.

Object diagrams are typically used by software designers and testers to ensure the correctness and completeness of the system model by providing a tangible view of its runtime state.

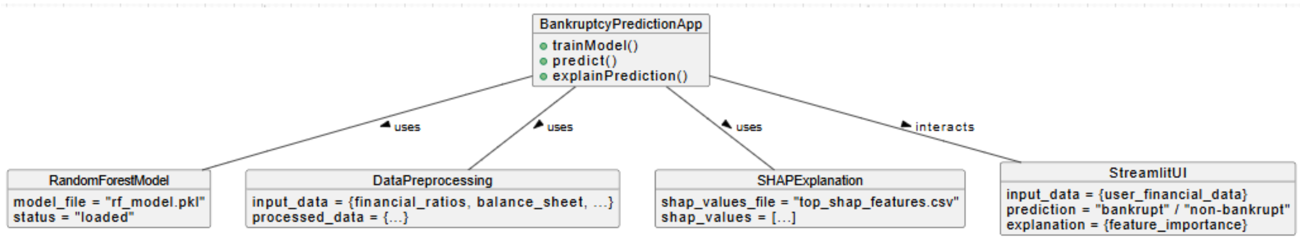


Figure 4.3 Object Diagram

In this diagram, the system is composed of several interconnected objects that represent the runtime state of the bankruptcy prediction application. At the core is the **BankruptcyPredictionApp**, which includes methods such as trainModel(), predict(), and explainPrediction()—responsible for orchestrating the entire prediction workflow. It utilizes the **RandomForestModel**, an object that encapsulates the trained model with attributes like model_file = "rf_model.pkl" and status = "loaded" to perform predictions.

The **DataPreprocessing** object manages the transformation of raw financial inputs, characterized by attributes like input_data and processed_data, ensuring the data is properly formatted before being passed to the model. For interpretability, the **SHAPExplanation** object generates explanations using attributes such as shap_values_file and shap_values, enabling transparency in predictions.

Lastly, the **StreamlitUI** object facilitates user interaction by collecting input_data, displaying the prediction result (either "bankrupt" or "non-bankrupt"), and presenting explanation details based on feature importance. These objects are linked through defined relationships: the **BankruptcyPredictionApp** utilizes the **RandomForestModel**, **DataPreprocessing**, and **SHAPExplanation**, while it interacts directly with the **StreamlitUI** to handle user input and present the output.

4.5 SEQUENCE DIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in which order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence.

Sequence diagrams are a popular dynamic modelling solution in UML because they specifically focus on lifelines or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends.

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. A sequence diagram shows different processes or objects that live simultaneously as parallel vertical lines (lifelines) and the messages exchanged between them and the order in which they occur as horizontal arrows.

The main purpose of the Sequence diagram is

- To capture the dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe the interaction among objects.

Sequence diagrams can be used

- To model the flow of control by time sequence.
- To model the flow of control by structural organisations.
- For reverse engineering

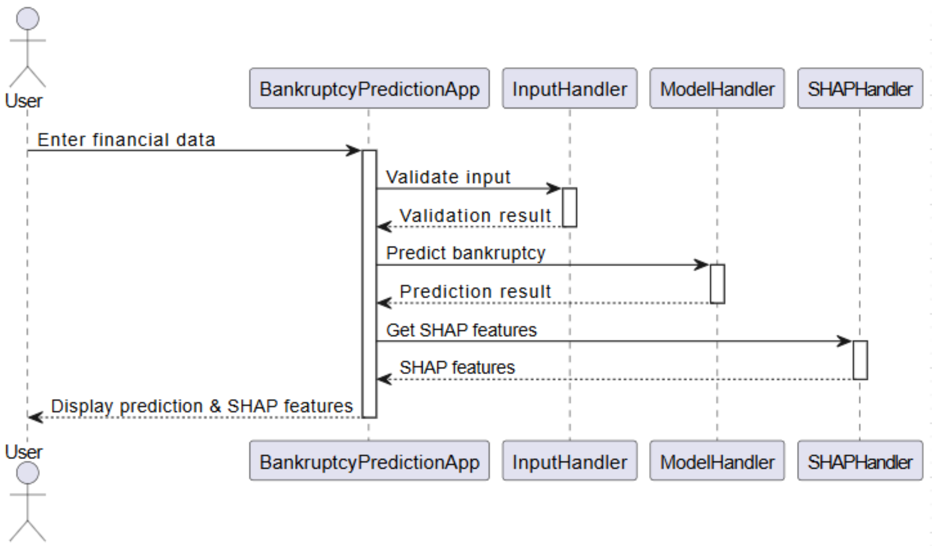


Figure 4.4 Sequence Diagram

This diagram illustrates the interaction flow among the core participants in the bankruptcy prediction system. The primary actors include the **User**, **BankruptcyPredictionApp**, **InputHandler**, **ModelHandler**, and **SHAPHandler**.

The process begins with the user entering financial data into the application. The **BankruptcyPredictionApp** then forwards this data to the **InputHandler** for validation. If the input is valid, it proceeds to the **ModelHandler**, which generates a bankruptcy prediction. To enhance interpretability, the app then interacts with the **SHAPHandler** to retrieve SHAP-based explanations. Finally, the application presents both the prediction result and its explanation back to the user, completing the interaction loop.

4.6 COLLABORATION DIAGRAM

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among objects in the Unified Modeling Language (UML).

Collaboration diagrams convey the same information as sequence diagrams but focus on object roles instead of the timings of messages. Collaboration diagrams represent a combination of information taken from class, sequence and use case diagrams describing both the static structure and dynamic behaviour of a system. The collaboration diagram describes the messages or roles sent between objects.

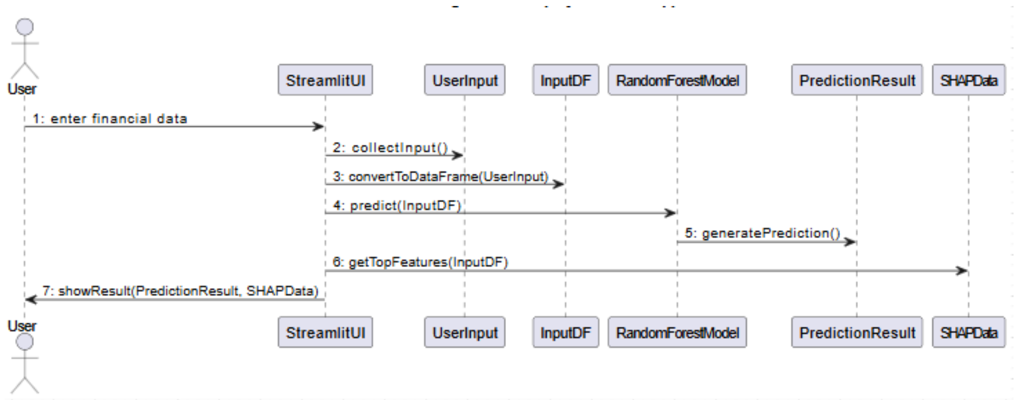


Figure 4.5 Collaboration Diagram

In this scenario, the user submits financial data through the system’s user interface. The system processes the input by preparing it for analysis and then passes it to the trained bankruptcy prediction model, which evaluates the data and generates a prediction result. Following this, the system retrieves SHAP-based explanations to provide interpretability, helping the user understand the key factors influencing the prediction outcome.

4.7 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organisational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

The activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc. Activity diagrams are constructed from a limited number of shapes, connected with arrows.

The most important shape types:

- rounded rectangles representations
- diamonds represent decisions
- bars represent the start (split) or end (join) of concurrent activities ● a black circle represents the start (initial node) of the workflow ● an encircled black circle represents the end (final node)

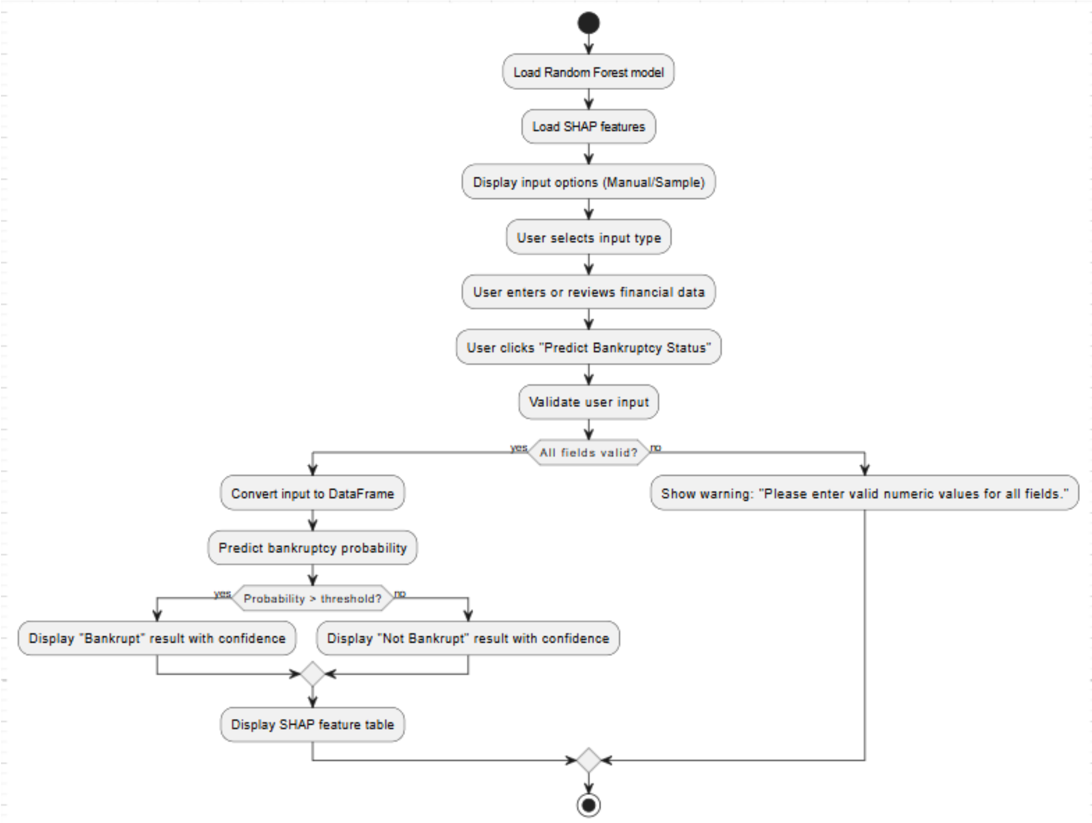


Figure 4.6 Activity Diagram

In this diagram, the user begins by opening the bankruptcy prediction application. The system initially loads the trained model and associated SHAP data, then presents the user with input options. The user either enters new financial data or reviews existing information. Upon clicking the "Predict" button, the system validates the input. If the input is valid, it is converted into a DataFrame and passed to the prediction model, which generates a result along with SHAP-based explanations. These outputs are then displayed to the user. If the input is invalid, the system instead shows an appropriate validation warning message.

4.8 STATE CHART DIAGRAM

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. It describes the different states of a component in a system. The states are specific to a component/object of a system. State chart diagrams are used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model reactive systems.

The state chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagrams is to model the lifetime of an object from creation to termination.

The main purpose of using State chart diagrams

- To model the dynamic aspect of a system.
- To model the lifetime of a reactive system.
- To describe different states of an object during its lifetime.
- An encircled black circle represents the end (final node)

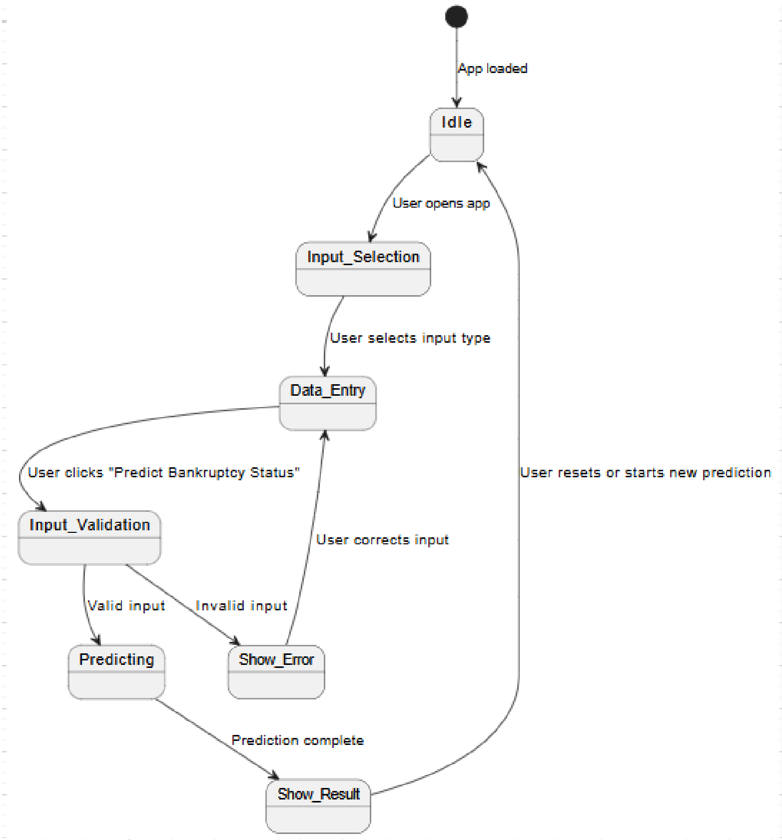


Figure 4.7 State Chart Diagram

This state chart diagram represents the user interaction flow within the bankruptcy prediction application. Initially, the system is in an idle state, waiting for user engagement. Once the user opens the app and selects their preferred input method, the system transitions to the data entry state, where the user provides the necessary financial information. After the data is submitted, the system enters the input validation state to verify its correctness. If the input is invalid, the system displays an error message and returns the user to the data entry state for correction. If valid, the system proceeds to the prediction state, where it processes the data to determine the bankruptcy status. The result is then displayed to the user, and the system returns to the idle state, ready for a new prediction cycle.

4.9 COMPONENT DIAGRAM

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems. They are used to model the physical aspects of a system. Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node. Component diagrams are used to visualise the organisation and relationships among components in a system. These diagrams are also used to make executable systems. The component diagram is a special kind of diagram in UML.

The purpose is also different from all other diagrams. It does not describe the functionality of the system but it describes the components used to make those functionalities. It can also be described as a static implementation view of a system. Static implementation represents the organisation of the components at a particular moment.

The purpose of the component diagram can be summarised as visualising the components of a system.

- Construct executables by using forward and reverse engineering.
- Describe the organisation and relationships of the components.

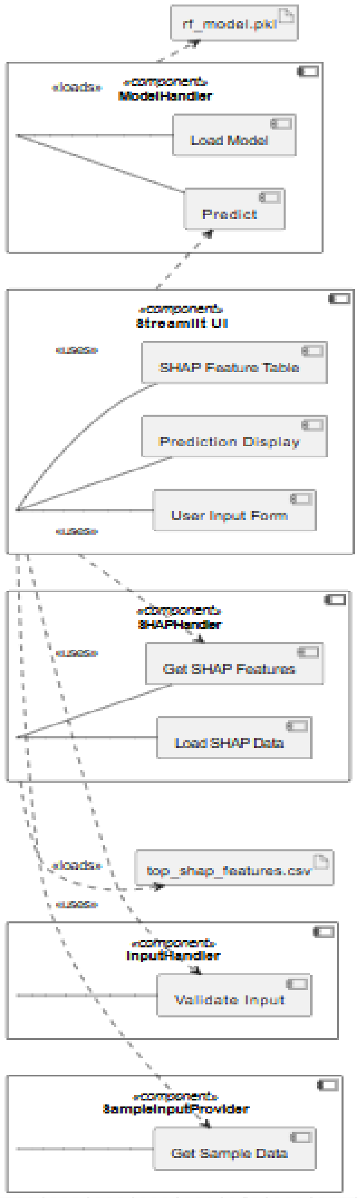


Figure 4.8 Component Diagram

This component diagram outlines the architecture of the bankruptcy prediction system, highlighting the main functional modules and their interactions. The system is composed of several key components: the User Interface Component (Streamlit UI) manages all user interactions; the Prediction Engine Component (ModelHandler) is responsible for generating bankruptcy predictions; the Explainability Component (SHAPHandler) provides model interpretability using SHAP values; the Input Handler Component (InputHandler) validates and processes user-provided data; and the Sample Data Provider Component

(SampleInputProvider) offers example financial data for user reference. The system relies on two primary artifacts—rf_model.pkl for the trained model and top_shap_features.csv for SHAP values.

Each component communicates through well-defined interfaces: the User Input Interface is provided by the UI and required by the InputHandler, the Prediction Interface is offered by the ModelHandler and consumed by the UI, the Explainability Interface is served by the SHAPHandler and utilized by the UI, and the Sample Data Interface is delivered by the SampleInputProvider and accessed by the UI. These interfaces ensure modularity and seamless integration across the system’s functionalities.

4.10 PACKAGE DIAGRAM

A package diagram is a UML structure diagram which shows packages and dependencies between the packages. The package diagram shows the arrangement and organisation of the model elements in a middle to large-scale project. The package diagram can show both the structure and dependencies between subsystems or modules. A package is rendered as a tabbed folder – a rectangle with a small tab attached to the left side of the top of the rectangle. If the members of the package are not shown inside the package rectangle, then the name of the package should be placed inside. The members of the package may be shown within the boundaries of the package. In this case, the name of the package should be placed on the tab. A diagram showing a package with content can show only a subset of the contained elements according to some criterion. Members of the package may be shown outside of the package by branching lines from the package to the members. The dotted arrows are dependencies. Packages can be built to represent either physical or logical relationships.

The entire system is divided logically into three packages – User Interface(UI), Technical and Domain. The package UI deals with objects and operations that are user interface-specific. The three sub-packages in UI pertain to information exchange between the user and the system. The package Technical deals with the storage of information in the datasets and the operations on the dataset and the model formed. The package Domain handles all important middleware operations that involve the core processing of the system.

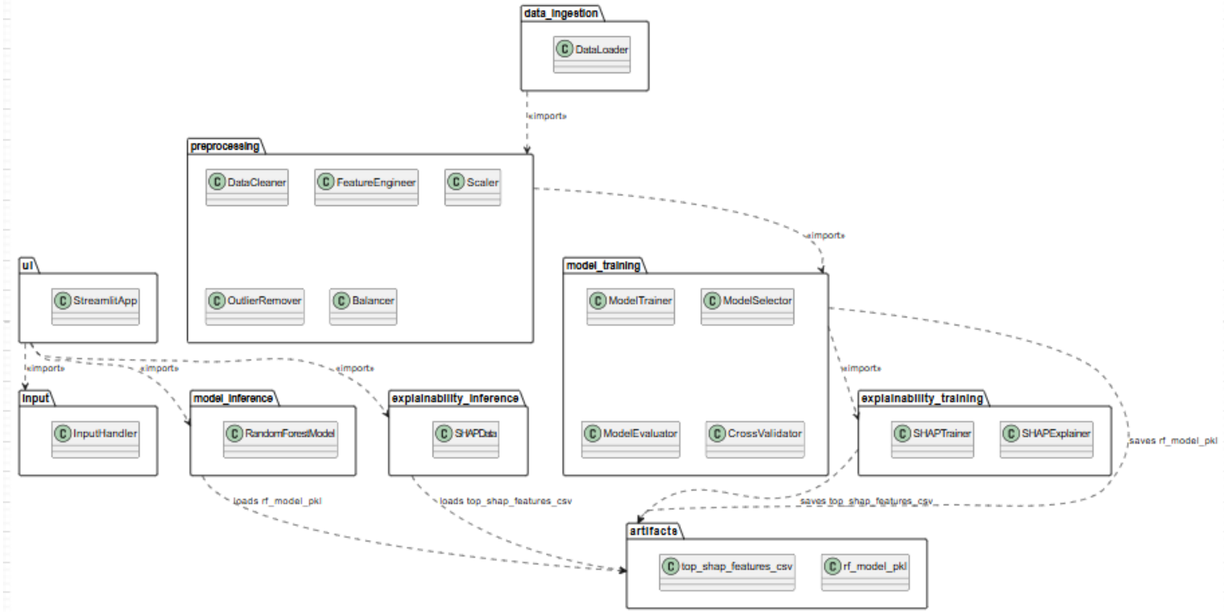


Figure 4.9 Package Diagram

In this diagram, the system is organized into several distinct packages that reflect its modular structure. The **UI package** is represented by the StreamlitApp, providing the user interface components. The **data_ingestion** and **preprocessing** packages, which include elements like DataLoader, DataCleaner, and FeatureEngineer, handle the initial stages of data preparation. The **model_training** and **explainability_training** packages, comprising components such as Trainer, Evaluator, and SHAP-related modules, represent the core processing logic of the system. For runtime operations, the **model_inference** and **explainability_inference** packages utilize components like RandomForestModel and SHAPData to deliver predictions and interpretability insights. The **input** package manages user data via the InputHandler, while the **artifacts** package stores the model and explanation outputs (rf_model.pkl, top_shap_features.csv). Dependencies across these packages illustrate the flow from raw data through preprocessing, training, and inference, ensuring a cohesive and traceable system architecture.

4.11 DEPLOYMENT DIAGRAM

A deployment diagram in the Unified Modeling Language models the physical deployment of artefacts on nodes. To describe a website, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artefacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g., JDBC, REST, RMI).

The nodes appear as boxes, and the artefacts allocated to each node appear as rectangles within the boxes. Nodes may have sub-nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers. Deployment diagrams are used by system engineers.

The purposes of deployment diagrams can be as follows:

1. Visualise the hardware topology of a system.
2. Describe the hardware components used to deploy software components.
3. Describe the runtime processing nodes.

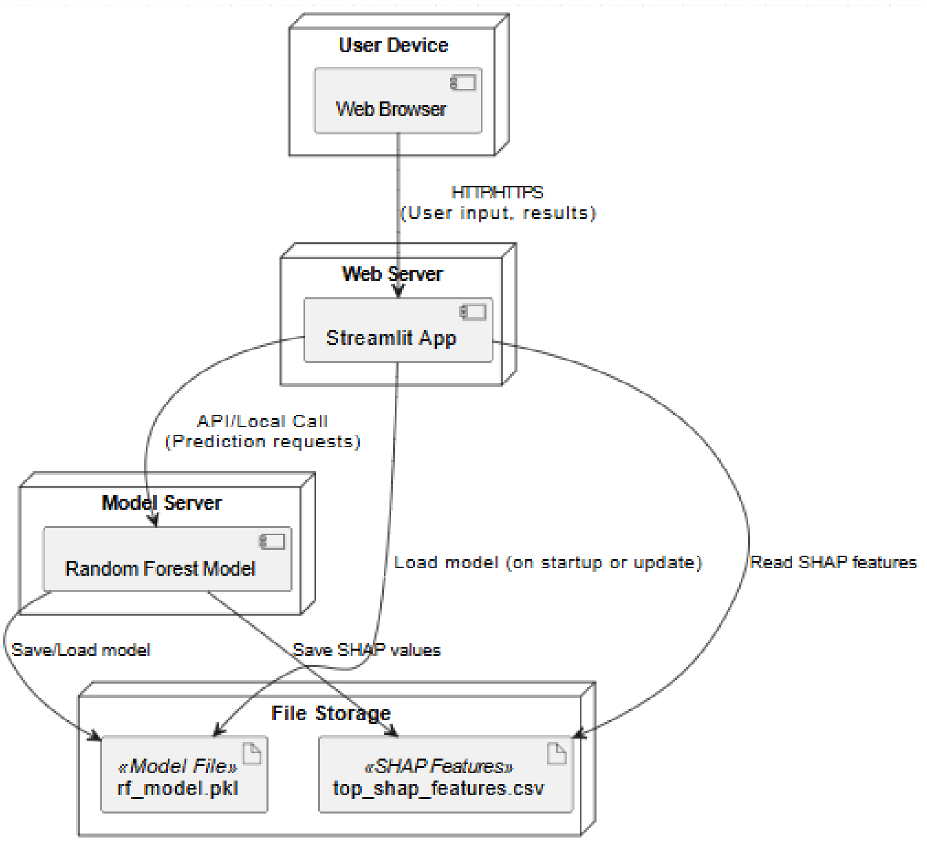


Figure 4.10 Deployment Diagram

In this diagram, the system's physical deployment is illustrated through various interconnected nodes. The **User Device** serves as the access point, where users interact with the system via a web browser. The **Web Server** hosts the **Streamlit application**, which facilitates the interface and business logic. The **Model Server** is responsible for managing the trained bankruptcy prediction model and handling prediction requests. Additionally, the **File Storage** node houses critical artifacts such as the trained model file and SHAP-based explanation data. Communication flows between these nodes: the web browser connects to the Streamlit app over HTTP/HTTPS; the app communicates with the model server either locally or via API; and it accesses the stored model and SHAP data files directly from the file storage system. This structure ensures efficient processing and delivery of predictions with explanatory insights.

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 PROPOSED SYSTEM

The proposed system integrates AI into the financial risk assessment domain, focusing on predicting bankruptcy risks for small businesses. By leveraging machine learning (ML) and data analytics, the system enhances detection and prediction capabilities, allowing for more accurate risk analysis. The system uses financial data from businesses, applying algorithms like Random Forest Classifier (RFC) for training, which is then used to assess the bankruptcy risk of new businesses. Real-time risk assessments and predictive analytics allow stakeholders to make informed decisions, proactively addressing potential financial issues before they escalate. Ethical considerations, transparency, and accuracy are prioritized to ensure fairness and reliability. The application presents users with bankruptcy risk predictions and provides actionable insights, allowing businesses to take necessary steps to mitigate financial distress. Additionally, the system offers resources and guidance for businesses facing financial challenges, providing support through recommendations for better financial planning and risk management strategies.

5.2 MODULE DESCRIPTION

5.2.1 Data collection and ingestion

Data collection is a critical module in the machine learning pipeline designed to predict bankruptcy risks for small businesses. It refers to the process of gathering all relevant financial and business data needed for accurate and reliable predictions. This step is essential because incomplete or inconsistent data can significantly degrade model performance, whereas comprehensive, high-quality data ensures more dependable outcomes.

The data is systematically sourced from internal databases, company financial statements such as balance sheets and profit and loss reports, and external providers that may include industry reports and economic indicators. This ensures that the dataset is both current and representative of the businesses under analysis. The collection process is designed to handle a variety of data formats and structures, enhancing the system's flexibility to accommodate different data sources.

To perform this process efficiently, several tools and techniques are used, including database connectors, CSV/Excel importers, and ETL (Extract, Transform, Load) frameworks. These tools enable streamlined ingestion and management of data, laying a strong foundation for the subsequent preprocessing and model training stages.

5.2.2 Data preprocessing and cleaning

Data preprocessing is a vital stage in the machine learning pipeline that prepares raw financial and business data for effective analysis and modeling. This phase is responsible for correcting errors, handling missing values, transforming features, and organizing the data into a clean and structured format suitable for training. Clean data is essential for producing reliable and accurate machine learning models, especially in real-world scenarios where inconsistencies and incomplete entries are common.

The preprocessing workflow includes several crucial steps. It begins with cleaning the data by addressing missing values, correcting inconsistencies, and removing duplicate entries. Numerical features are then normalized or standardized to ensure comparability across different scales. Categorical variables are encoded into machine-readable formats to make them compatible with modeling algorithms.

To address the common issue of class imbalance—where bankrupt companies are typically underrepresented—SMOTE (Synthetic Minority Over-sampling Technique) is applied, enhancing the model's ability to learn from minority class examples.

Additionally, the dataset is divided into training and testing subsets to prevent data leakage and allow for robust model evaluation. These preprocessing tasks are carried out using widely adopted Python libraries such as **pandas**, **scikit-learn**, and **imbalanced-learn**, ensuring that the data is accurately prepared for the subsequent stages of machine learning.

5.2.3 Feature Engineering and model selection

Feature engineering and model selection are integral steps in developing a robust bankruptcy prediction system. Feature engineering involves transforming raw financial data into meaningful inputs that can enhance model performance. This includes generating new features such as financial ratios and rolling averages, which capture essential financial patterns and trends. Additionally, feature selection techniques are applied to identify the most predictive indicators of bankruptcy. These methods range from statistical tests to model-based approaches like feature importance scores derived from Random Forest. The goal is to reduce dimensionality by eliminating irrelevant or redundant features, thereby improving both model accuracy and interpretability. Tools such as **domain knowledge**, **scikit-learn feature selection**, and **financial ratio formulas** support this process.

Following feature engineering, a variety of machine learning algorithms were evaluated to determine the best-performing model for predicting bankruptcy. The classification models assessed included **Naive Bayes**, **Logistic Regression**, **Random Forest Classifier**, **K-Nearest Neighbors (KNN)**, and **Stochastic Gradient Descent (SGD)**. Each model presents unique strengths—for example, Logistic Regression offers simplicity and interpretability, while

Random Forest enhances prediction accuracy through an ensemble of decision trees.

The selection process was guided by key validation metrics such as **accuracy**, **precision**, **recall**, and **F1-score**, ensuring the chosen model delivers reliable performance in identifying at-risk businesses.

5.2.4 Model Training and Evaluation

The Once the most suitable model was identified through a comprehensive evaluation process, the training phase began using a preprocessed financial dataset. This dataset comprised features derived from company financial records and was labeled to indicate whether a business was bankrupt or solvent. The model was trained on this labeled data to recognize patterns and signals associated with financial distress. The goal of the training process was to ensure that the model could generalize effectively to new, unseen data, enabling accurate and explainable predictions when deployed in real-world scenarios.

Throughout the model development phase, the performance of each algorithm was assessed using standard classification metrics, including accuracy, precision, recall, and F1-score. These metrics provided a comprehensive measure of each model's predictive capability and guided the selection of the final model.

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

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

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

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

where,

FP - The number of False Positives

FN - The number of False Negatives

TP - The number of True Positives

TN - The number of True Negatives

	Accuracy	Precision	Recall	F1-score
Logistic Regression	76.52	71.64	63.40	67.27
Naive Bayes	77.02	85.56	47.65	61.21
Random Forest	93.63	91.91	91.31	91.61
K Nearest Neighbors	65.13	73.53	13.07	22.20
SGD Classifier	78.39	72.50	69.61	71.02

Figure 5.1 Performance Metrics Results

Ultimately, the Random Forest Classifier was chosen as the optimal model. While neural networks offer advantages such as unsupervised learning capabilities and flexible architectures for complex data patterns, the Random Forest model aligned more closely with the project’s goals. It offers a reduced risk of overfitting, ease of interpretation, and cost-effective training. Furthermore, Random Forests are well-suited to the dataset's characteristics, particularly in handling missing values without introducing bias. They also require minimal preprocessing, eliminating the need for feature scaling or the conversion of categorical variables, thus streamlining the training process.

Risk Scoring & Prediction module assigns a bankruptcy risk score to each company, helping stakeholders identify high-risk firms early and take preventive actions. It works by applying a trained model to new or existing business data, outputting a probability or risk score that indicates the likelihood of bankruptcy. High-risk cases are flagged for further review or intervention. The tools and techniques used for this module include selected machine learning models and probability scoring functions.

5.2.5 Model Interpretation and Explainability

The Model interpretation and explainability play a crucial role in ensuring the transparency and trustworthiness of the bankruptcy prediction system, particularly in the financial domain where stakeholders must fully understand and trust the model's decisions. This module explains how and why the model arrives at its predictions, helping users interpret both individual and overall outcomes. The system employs SHAP (SHapley Additive exPlanations) to break down each prediction, identifying the contribution of each feature to the model's output. This includes generating feature importance plots and local explanations to clarify why a particular business is classified as high-risk or low-risk. Such transparency not only fosters trust among users but also supports compliance with regulatory requirements by making the decision-making process interpretable. Tools used for this purpose include the SHAP library and various visualization techniques for feature importance.

In conjunction with explainability, the system also performs risk scoring and prediction to quantify the likelihood of bankruptcy for each company. The trained machine learning model is applied to incoming or existing business data to produce a probability score that reflects the risk level. High-risk cases are flagged for further analysis or intervention, allowing stakeholders to take preventive action where necessary. This functionality helps organizations identify vulnerabilities early and make data-driven decisions. The risk scoring process is powered by the selected machine learning model and appropriate probability scoring functions to ensure precision and reliability.

5.2.6 Risk Scoring and Prediction

The Risk Scoring & Prediction module assigns a bankruptcy risk score to each company, helping stakeholders identify high-risk firms early and take preventive actions. It works by applying a trained model to new or existing business data, outputting a probability or risk score that indicates the likelihood of bankruptcy. High-risk cases are flagged for further review or intervention. The tools and techniques used for this module include selected machine learning models and probability scoring functions.

The Visualization & Dashboard Module presents results and insights in an interactive, user-friendly format, enabling non-technical stakeholders to easily interact with the bankruptcy prediction system. It provides clear, actionable insights and explanations for model predictions. This module works by using Streamlit to build an interactive web dashboard, where users can select between manual data entry or sample business profiles, input or review financial indicators via web forms, and generate a bankruptcy risk prediction with just a click of a button. The prediction result, along with its confidence score, is instantly displayed, and a table of top influential features (SHAP) is provided to explain the prediction. Visualizations, such as SHAP feature importance, are directly displayed in the app, which has an intuitive interface requiring no coding knowledge. Tools and techniques used in this module include Streamlit, Matplotlib, SHAP, Pandas, and Google Colab.

5.2.7 Visualization and Dashboard Module

This Module is designed to present results and insights in an interactive, user-friendly format. It is needed because it enables non-technical stakeholders to easily interact with the bankruptcy prediction system and provides clear, actionable insights and explanations for the model's predictions.

The module works by using Streamlit to build an interactive web dashboard. Users have the option to select between manual data entry or sample business profiles, input or review financial indicators via web forms, and generate a bankruptcy risk prediction with just a click. The prediction result, along with its confidence score, is displayed instantly. Additionally, users can view a table of the top influential features (SHAP) that explain the prediction. Visualizations, such as SHAP feature importance, are shown directly in the app, offering an intuitive interface that requires no coding knowledge. Tools and techniques used in this module include Streamlit, Matplotlib, SHAP, Pandas, and Google Colab.

5.2.8 Integration and Deployment

The Integration & Deployment module makes the bankruptcy prediction system accessible and shareable via executable notebooks and Python scripts. It offers multiple access points: an interactive web app for business users and notebooks/scripts for data scientists, with no complex setup required. This enables reproducible results, easy sharing of outputs, and collaboration.

The workflow—data preprocessing, model training, prediction, and report generation—is implemented in Google Colab and/or Python scripts. Users can input data step-by-step and generate outputs. For business users, a Streamlit app (app_py.py) loads the trained model and SHAP feature data, allowing instant predictions via a browser.

The app can be hosted locally or on cloud platforms like Streamlit Cloud or Heroku, with results downloadable or shareable. Code and outputs are versioned and shared via GitHub or Google Drive.

The best-performing algorithm, Random Forest Classifier (RFC), is deployed in the app. Users input financial data, and the app processes it to predict bankruptcy risk. Results include predictions, risk factors, and recommendations. The app can also offer personalized financial guidance, such as budgeting tips and debt management advice, helping users understand their financial risk and avoid bankruptcy.Tools used include Google Colab, Python scripts, Streamlit, and GitHub/Google Drive.

CHAPTER 6
SYSTEM TESTING

6.1 INTRODUCTION

Software testing is a process of evaluating software comprehensively to verify whether it is running in the desired fashion and to identify any bugs present. This is done to keep the quality of the deliverable high.

By the standard of ANSI, the definition of Software Testing is - A process of analysing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

6.2 TESTING APPROACHES

There are two main kinds of testing approaches under software testing, which are:

i. White box testing

ii. Black box testing

6.2.1 WHITE BOX TESTING

White box testing involves analysing the internal structures of the code and not merely the functionality. It is used to identify any errors present in the code structure. It is usually done for unit testing although it can be done for integration and system testing too. White box testing is used for debugging code within a subsystem, between subsystems and so on. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. It is also called Glass Box, Clear Box and Structural Testing.

Advantages of White Box Testing:

● White Box Testing has simple and clear rules to let a tester know when the testing is done.

● White Box Testing Techniques are easy to automate, this results in a developer having to hire fewer testers and smaller expenses.

● It shows bottlenecks which makes the optimization quite easy for the programmers.

6.2.2 BLACK BOX TESTING

As opposed to white box testing, black box testing is used to analyse the functionality of the software. It can be done in all levels of testing from unit testing to system testing. It is generally done for testing higher levels of code. It is also called Behavioral/Specification-Based/Input-Output Testing.

Advantages of Black Box Testing

● Black box tests are always executed from a user's point of view since it would help in exposing discrepancies significantly.

● Black box testers also do not need to know any programming languages. **6.3**

TESTING LEVELS

Tests are grouped together based on the level of detail they contain. The purpose of levels of testing is to make software testing systematic and easily identify all possible test cases at a particular level. In general, there are four levels of testing:

6.3.1 Unit Testing

6.3.2 Integration Testing

6.3.3 System Testing

6.3.4 Acceptance Testing.

6.3.1 UNIT TESTING

A Unit is a smallest testable portion of a system or application which can be compiled, linked, loaded, and executed. This kind of testing helps to test each module separately.

6.3.2 INTEGRATION TESTING

Integration means combining. In this testing phase, different software modules are combined and tested as a group. Integrating testing checks the data flow from one module to another module.

6.3.3 SYSTEM TESTING

System testing is performed on a complete, integrated system. It does checking of the system's compliance with the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing. System testing is most often the final test to verify that the system meets the specification. It evaluates both functional and non-functional needs.

6.3.4 ACCEPTANCE TESTING

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.

6.4 TESTING TYPES

There are two types of testing which are classified as such based on the manner in which the testing is done. They are:

6.4.1 Manual Testing

6.4.2 Automation Testing

6.4.1 MANUAL TESTING

Manual testing is the process of testing software by hand. This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end users in mind. Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results to high-level guides that steer exploratory testing sessions.

Figure 6.1 Manual Testing

6.4.2 AUTOMATION TESTING

Automation testing is the process of testing the software using an automation tool to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools.

6.5 ENCRYPTION AND MASKING USING TEXTPAD TOOL

Testpad is a test plan tool that helps you find the bugs that matter. Less time messing around with spreadsheets or old-school test case management means more time actually testing. A simple test planning tool that reinvents test case management as checklists that anyone can use. It takes a refreshingly simple approach that makes it really fast to write, run and maintain tests.

6.6 TEST RESULTS:

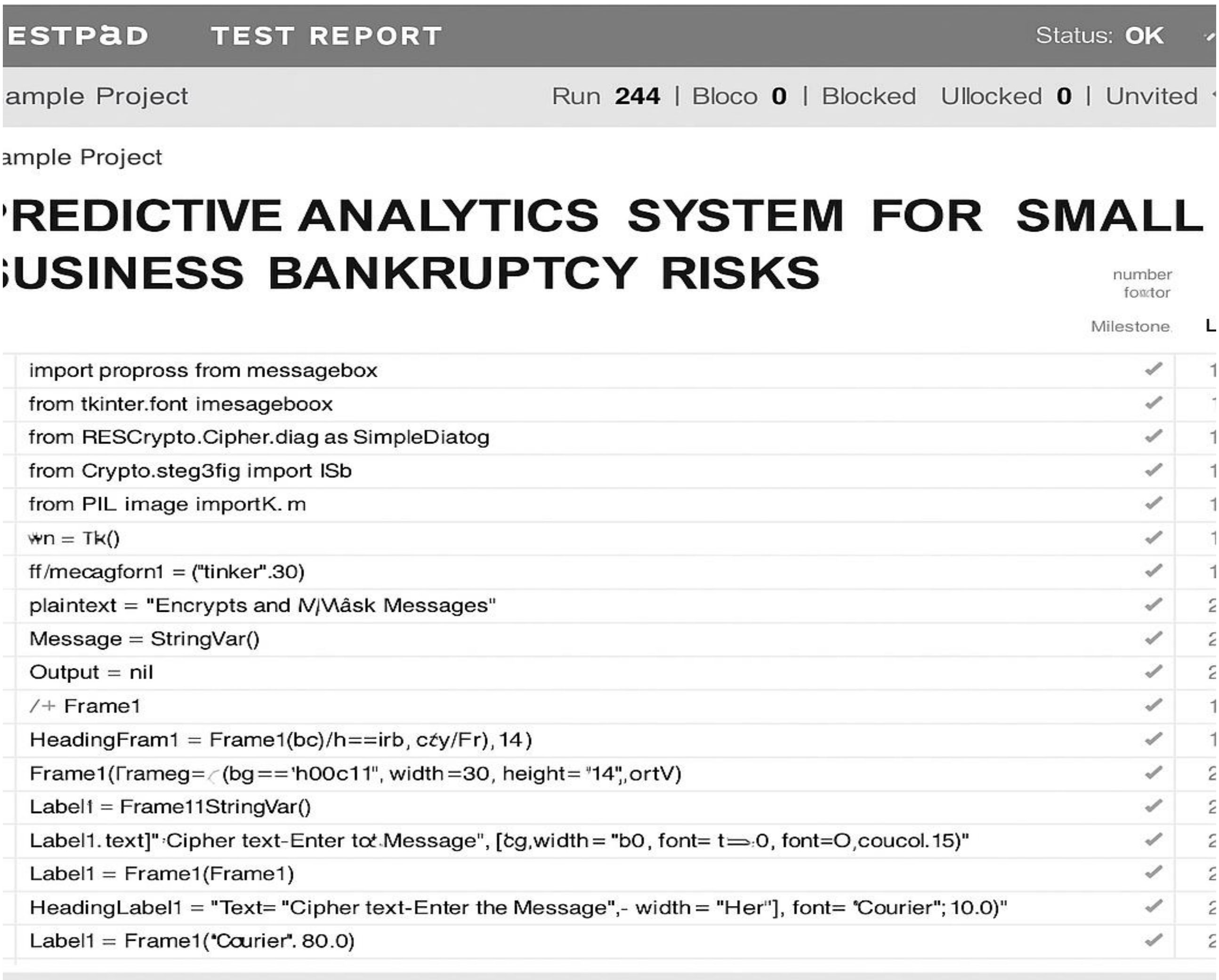


Figure 6.2 Manual Test Result

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

Bankruptcy prediction continues to be a critical challenge for businesses and financial institutions, highlighting the need for innovative solutions to identify and mitigate financial distress. This study demonstrates the potential of leveraging machine learning and artificial intelligence in predicting bankruptcy risks. By utilizing the Random Forest Classifier, the proposed bankruptcy prediction system allows users to assess the financial health of businesses and proactively address potential risks in real-time. Additionally, incorporating predictive analytics and financial sentiment analysis enhances the system's accuracy and reliability. Moving forward, further research and development in this domain are essential to refine prediction models, improve decision-making processes, and ultimately support businesses in preventing financial distress and ensuring long-term sustainability.

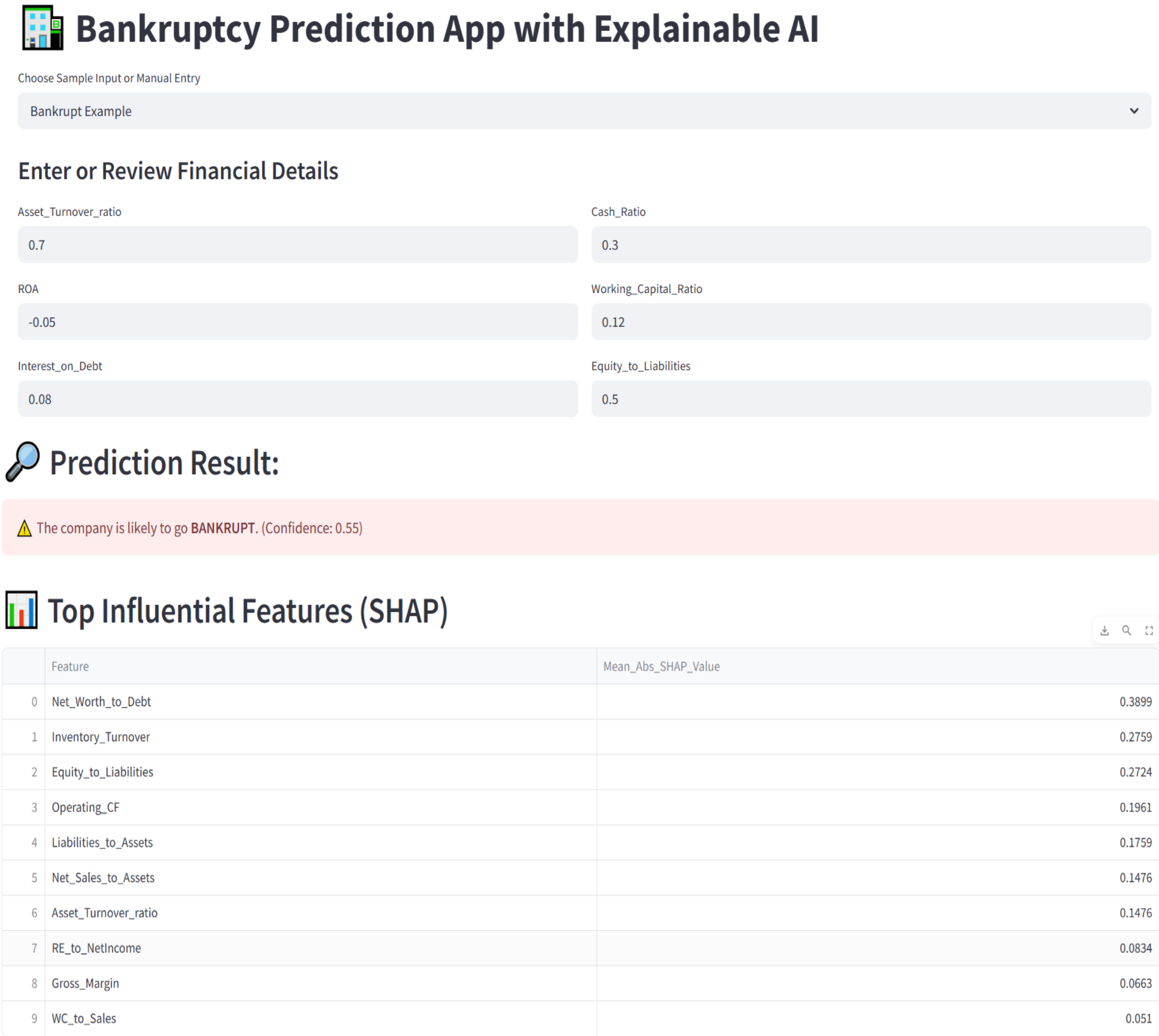
7.2 FUTURE ENHANCEMENT

Future enhancements of the bankruptcy prediction system may focus on integrating advanced machine learning techniques and real-time financial monitoring to improve prediction accuracy. Prioritizing ethical considerations, such as data privacy and transparency, will be crucial in the system's development and deployment. Incorporating advanced predictive analytics and financial sentiment analysis could provide deeper insights into a business's financial health, refining the accuracy of bankruptcy risk assessments.

APPENDIX

SCREENSHOTS

BANKRUPTCY



	Feature	Mean_Abs_SHAP_Value
0	Net_Worth_to_Debt	0.3899
1	Inventory_Turnover	0.2759
2	Equity_to_Liabilities	0.2724
3	Operating_CF	0.1961
4	Liabilities_to_Assets	0.1759
5	Net_Sales_to_Assets	0.1476
6	Asset_Turnover_ratio	0.1476
7	RE_to_NetIncome	0.0834
8	Gross_Margin	0.0663
9	WC_to_Sales	0.051

NON-BANKRUPT

Return_on_Assets	RE_to_Sales
0.18	0.26
NetWC_to_Assets	RE_to_Assets
0.24	0.22
EBIT_to_Assets	RE_to_NetIncome
0.19	1.2
EBIT_to_Sales	Inventory_Turnover
0.25	7.5
Current_Ratio	Inventory_to_Sales
3.0	0.09

🔍 Predict Bankruptcy Status

🔎 Prediction Result:

✅ The company is **NOT likely to go bankrupt**. (Confidence: 0.01)

📊 Top Influential Features (SHAP)

	Feature	Mean_Abs_SHAP_Value
0	Net_Worth_to_Debt	0.3899
1	Inventory_Turnover	0.2759
2	Equity_to_Liabilities	0.2724
3	Operating_CF	0.1961
4	Liabilities_to_Assets	0.1759
5	Net_Sales_to_Assets	0.1476
6	Asset_Turnover_ratio	0.1476
7	RE_to_NetIncome	0.0834
8	Gross_Margin	0.0663
9	WC_to_Sales	0.051

🏢 Bankruptcy Prediction App with Explainable AI

Choose Sample Input or Manual Entry

Non-Bankrupt Example

Enter or Review Financial Details

Asset_Turnover_ratio	Cash_Ratio
2.0	1.5
ROA	Working_Capital_Ratio
0.18	2.8
Interest_on_Debt	Equity_to_Liabilities
0.01	2.2
Debt_to_Equity	Net_Sales_to_Assets
0.25	2.1

REFERENCES

1. Zhang, Yao, and Huan Liu. "Bankruptcy prediction using machine learning algorithms: A comparative study." In 2020 International Conference on Data Mining and Big Data (DMBD), pp. 10-15. IEEE, 2020.

2.Nguyen, Cuong, and Dat Nguyen. "Bankruptcy prediction using a deep learning approach." In 2019 International Conference on Data Mining and Big Data (DMBD), pp. 90-95. IEEE, 2019.

3.Yang, Bo, and He Liu. "Predicting bankruptcy using artificial intelligence: A case study of financial companies." In 2021 International Conference on Artificial Intelligence and Financial Engineering (AI-Fin), pp. 101-107. IEEE, 2021.

4.Iqbal, Muhammad, and R. H. Khan. "Machine learning for bankruptcy prediction: A deep learning approach." In 2019 International Conference on Big Data and Cloud Computing (BDCC), pp. 202-207. IEEE, 2019.

5.Li, Xin, and Zhao Li. "An enhanced prediction model for financial distress using machine learning techniques." In 2020 International Conference on Data Science and Artificial Intelligence (ICDSAI), pp. 139-144. IEEE, 2020.

6.Salehahmadi, Zahra, and Ali Farhadi. "Financial distress prediction using ensemble learning algorithms." In 2020 International Conference on Machine Learning and Computing (ICMLC), pp. 181-186. IEEE, 2020.

7.Ahmad, Waseem, and Muhammad Sajjad. "Financial distress prediction using deep neural networks: A case study of Pakistan." In 2018 International Conference on Big Data and Artificial Intelligence (BDAI), pp. 165-170. IEEE, 2018.

8. Altman, Edward I., and V. Narayanan. "Predicting bankruptcy with the Altman Z-score model." In 2018 6th International Conference on Financial Analytics and Engineering (ICFAE), pp. 1-7. IEEE, 2018.

9. Kumar, Vikas, and Ravi Shankar. "Bankruptcy prediction using ensemble learning and hybrid model." In 2019 IEEE International Conference on Data Science and Engineering (ICDSE), pp. 221-226. IEEE, 2019.

10. Chen, Hao, and Ying Zhang. "Application of machine learning algorithms in bankruptcy prediction models." In 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE), pp. 74-79. IEEE, 2020.

11. Patel, Narendra, and Ruchir Shah. "Predictive modeling for financial distress using machine learning techniques." In 2019 IEEE International Conference on Machine Learning and Computing (ICMLC), pp. 345-350. IEEE, 2019.

12. Liu, Fang, and Shun Liu. "A novel hybrid approach for bankruptcy prediction using machine learning." In 2018 International Conference on Business Intelligence and Big Data (BID), pp. 102-107. IEEE, 2018.

13. Yang, Jie, and Li Li. "Financial distress prediction using support vector machines and machine learning algorithms." In 2020 International Conference on Business and Industrial Applications (ICIBA), pp. 125-130. IEEE, 2020.

14. Raza, Syed Waqas, and Amjad Umar. "Financial distress prediction using random forest and support vector machines." In 2017 International Conference on Computing, Networking and Communication (ICNC), pp. 512-517. IEEE, 2017.

15. Jang, Dae Seung, and Su Mi Han. "A comparative study of financial distress prediction models based on machine learning techniques." In 2021 International Conference on Computer Science and Artificial Intelligence (ICCSAI), pp. 87-93. IEEE, 2021.