A Project Report on

HYBRID MACHINE LEARNING FOR BANKRUPTCY PREDICTION POLISH CASE STUDY

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE&ENGINEERING

Submitted by

Nallamilli Lakshmi M Vigneswara Reddy	**21MH1A0542**
Damalanka Pujitha Jagadeeswari	**22MH5A0502**
Vudathu Aishwarya	**21MH1A0565**
Boda Nikhil	**22MH5A0501**

Under the esteemed supervision of

Mr. T.Veerraju M.Tech(Ph.D)

Associate Professor

Department of Computer Science and Engineering

Aditya College of Engineering & Technology

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC with A+

Grade Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956

(2021-2025)

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC with A+ Grade
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem, Kakinada District - 533 437, A.P.
Ph: 99591 76665, Email: office@acet.ac.in, www.acet.ac.in

VISION

To induce higher planes of learning by imparting technical education with

- ✓ International standards
- ✓ Applied research
- ✓ Creative Ability
- ✓ Value based instruction and to emerge as a premiere institute

MISSION

Achieving academic excellence by providing globally acceptable technicaleducation by forecasting technology through

- ✓ Innovative Research And development
- ✓ Industry Institute Interaction
- ✓ Empowered Manpower

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY * SURAMPALEM *

Principal

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM- 533 437

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC with A+ Grade
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem, Kakinada District - 533 437, A.P.
Ph: 99591 76665, Email: office@acet.ac.in, www.acet.ac.in

Department of Computer Science and Engineering

Vision

- ✓ To become a center for excellence in Computer Science and Engineering education and innovation.

Mission

- ✓ Provide state of art infrastructure.
- ✓ Adapt Skill based learner centric teaching methodology.
- ✓ Organize socio-cultural events for better society.
- ✓ Undertake collaborative works with academia and industry.
- ✓ Encourage students and staff self-motivated, problem solving individuals using Artificial Intelligence.
- ✓ Encourage entrepreneurship in young minds.

Head of the Department

Head of the Department
Computer Science and Engineering
Aditya College of Engineering & Technology (A)
SURAMPALEM-533 437

Aditya College of Engineering & Technology

An AUTONOMOUS Institution

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC with A+ Grade Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956

Department of Computer Science&Engineering

CERTIFICATE

This is to certify the project report entitled "Hybrid Machine Learning for Bankruptcy Prediction: Polish Case" Study is a bonified work carried out by **Nallamilli Lakshmi M Vigneswara Reddy** bearing with reg No: **21MH1A0542**, **Damalanka Pujitha Jagadeeswari** bearing with reg No: **22MH5A0502**, **Vudathu Aishwarya** bearing with reg No: **21MH1A0565**, **Boda Nikhil** bearing with reg No: **22MH5A0501** at the college for the award of Bachelor of Technology in COMPUTER SCIENCE & ENGINEERING from ADITYA COLLEGE OF ENGINEERING& TECHNOLOGY(A) during the academic year of 2021-2025.

Project Guide	Head of the Department
Mr.T.Veerraju	Dr.G.S.N.Murthy .
Dept. Of CSE	Dept. Of CSE
Associate Professor	**Professor**

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled "**Hybrid Machine Learning for Bankruptcy Prediction: Polish Case Study**" has been undertaken by us and this work has been submitted to **ADITYA COLLEGE OF ENGINEERING&TECHNOLOGY(A)**, Surampalem affiliated to JNTUK, Kakinada, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING.** We further declare that this project work has not been submitted in full or part to any other University or educational institute for the award of any degree or diploma.

PROJECT ASSOCIATES

Nallamilli Lakshmi M Vigneswara Reddy	**21MH1A0542**
Damalanka Pujitha Jagadeeswari	**22MH5A0502**
Vudathu Aishwarya	**21MH1A0565**
Boda Nikhil	**22MH5A0501**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to my **project supervisor**, **Mr. T.Veerraju,** Associate Professor, who has guided us a lot and encouraged us in every step of project work, his valuable moral support and guidance has been helpful in successful completion of this Project.

We wish to express our sincere thanks to **Dr. G. S. N. Murthy, Head of the Department of CSE**, for his valuable guidance given to us throughout the period of the project work.

We feel elated to thank **Dr A. Ramesh, Principal** of Aditya College of Engineering and Technology for his cooperation in completion of our project and throughout our course.

We feel elated to thank **Dr. P.S.V.V.S. Ravi Kumar**, **Dean (Academics)** of Aditya College of Engineering and Techonology for his cooperation in completion of our project work.

We avail this opportunity to express our deep sense and heart full thanks to the **Management** of **Aditya College of Engineering and Technology(A)** for providing a great support for us by arranging the trainees, and facilities needed to complete our project and for giving us the opportunity for doing this work.

Above all, we extend our heartfelt gratitude to our beloved parents, whose unwavering support, encouragement, and sacrifices have been our greatest source of strength throughout this journey.Their guidance and belief in us have been invaluable in achieving this milestone.

PROJECT ASSOCIATES

Nallamilli Lakshmi M Vigneswara Reddy	**21MH1A0542**
Damalanka Pujitha Jagadeeswari	**22MH5A0502**
Vudathu Aishwarya	**21MH1A0565**
Boda Nikhil	**22MH5A0501**

ABSTRACT

Bankruptcy prediction is a critical aspect of financial risk management, helping businesses, investors, and financial institutions make informed decisions to mitigate potential losses. This project introduces a robust machine learning-based bankruptcy prediction system that analyzes key financial indicators to determine a company's financial health. Our approach integrates advanced feature selection and optimization techniques, such as Genetic Algorithms (GA) for identifying the most relevant financial attributes and Particle Swarm Optimization (PSO) for fine-tuning model parameters, ensuring enhanced prediction accuracy.The system utilizes multiple machine learning algorithms, including Logistic Regression, Random Forest, and Decision Trees, to classify companies as "Bankrupt" or "Non- Bankrupt." It processes critical financial metrics, such as cash flow, profit margins, liabilities, assets, and debt-to-equity ratios, to derive meaningful insights. By efficiently handling large datasets, managing missing values, and addressing class imbalances, our model ensures high reliability in predicting financial distress.In addition to classification, the system generates risk scores, visual analytics, and detailed financial reports to assist stakeholders in proactive decision-making. The implementation of this hybrid machine learning framework enhances early warning systems, enabling businesses to anticipate financial difficulties, restructure operations, and prevent potential insolvency. The system is designed with a user-friendly interface, making it accessible to financial analysts, investors, and corporate decision-makers.By leveraging the power of data-driven prediction models, this project contributes to strengthening financial stability, improving investment strategies, and optimizing risk management practices. The integration of AI-driven insights into bankruptcy prediction enhances the accuracy and efficiency of financial forecasting, making it a valuable tool for organizations striving for long-term sustainability.

Keywords: Random Forest, Logistic Regression, Genetic Algorithm, Particle Swarm Optimization, predictive analytics,optimization techniques,corporate insolvency detection, financial health evaluation, business failure forecasting, risk mitigation strategies.

INDEX

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

1.1 INTRODUCTION TO PROJECT:

Bankruptcy is a serious financial condition that affects businesses, individuals, and economies. It occurs when an entity is unable to pay its debts and is forced to either restructure its financial obligations or liquidate assets to settle outstanding payments. As financial markets grow more complex, accurately predicting bankruptcy has become a major challenge. Traditional methods, such as financial ratio analysis and historical data evaluation, often fail to identify early warning signs of financial distress. These limitations have led to the adoption of advanced machine learning techniques for more accurate and timely bankruptcy prediction.Our project introduces a Hybrid Machine Learning Model for predicting bankruptcy, incorporating Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and machine learning models such as Logistic Regression, Random Forest, and Decision Trees. These models analyze multiple financial indicators, including cash flow, liabilities, profit margins, and debt-to-equity ratios, to identify patterns and assess the likelihood of bankruptcy. By integrating deep learning algorithms, our system enhances prediction accuracy, allowing businesses and financial institutions to take proactive measures before financial failure occurs.This document provides a detailed overview of our bankruptcy prediction model, including its methodology, implementation, and evaluation. It discusses the motivation behind the work, existing challenges, the proposed solution, and the architectural framework used to develop the system. Our goal is to create a reliable and efficient bankruptcy prediction tool that helps stakeholders make informed financial decisions, minimize risks, and improve overall financial stability.

Introduction to Machine Learning:

Machine Learning (ML) is widely used in financial forecasting, including bankruptcy prediction. ML models analyze historical financial data to identify patterns and detect early signs of financial distress. Unlike traditional statistical

methods, ML algorithms continuously learn from new data, improving their predictive accuracy over time.In bankruptcy prediction, ML techniques follow a structured approach:

- Feature Extraction and Selection: Identifying key financial indicators, such as cash flow, debt-to- equity ratio, and profit margins, that influence bankruptcy risks.
- Optimization Techniques: Fine-tuning machine learning models using methods like Genetic Algorithms (GA) for selecting important financial features and Particle Swarm Optimization (PSO) for improving model accuracy.
- Several ML techniques are used for bankruptcy prediction, including:Logistic Regression (LR): A simple yet effective model that estimates bankruptcy probability based on financial ratios.
- Decision Trees (DT): A model that classifies companies based on financial attributes, making it easy to interpret.
- Random Forest (RF): An ensemble method that combines multiple decision trees to improve prediction accuracy.

Introduction to Machine Learning in Bankruptcy Prediction

Machine learning plays a vital role in predicting bankruptcy by analyzing financial data and identifying key risk factors. Unlike traditional statistical methods, machine learning models can process large datasets, handle imbalanced financial records, and improve prediction accuracy over time.

A typical bankruptcy prediction system consists of:

Feature Selection: Identifies the most relevant financial indicators using Genetic Algorithms (GA).

Data Preprocessing: Handles missing values, normalizes financial ratios, and removes redundant data.

Model Training: Uses machine learning models such as Logistic Regression, Decision Trees, and Random Forest to classify companies as "Bankrupt" or "Non-Bankrupt."

Optimization: Fine-tunes model parameters using Particle Swarm Optimization (PSO) to enhance accuracy.

Key techniques used in bankruptcy prediction:

Genetic Algorithms (GA): Selects the most significant financial features for model training.

Particle Swarm Optimization (PSO): Optimizes hyperparameters to improve prediction performance.

Logistic Regression (LR): Estimates bankruptcy probability based on financial trends.

Decision Trees (DT) and Random Forest (RF): Classify financial data and enhance prediction reliability.

Key Concepts of Machine Learning in Bankruptcy Prediction:

Machine learning is a powerful tool in bankruptcy prediction, allowing financial analysts to identify early warning signs of financial distress. It enables datadriven decision-making by analyzing historical financial data and predicting the likelihood of bankruptcy. Key concepts involved in machine learning-based bankruptcy prediction

Feature Selection:

Genetic Algorithms (GA): Helps in selecting the most significant financial indicators, such as debt-to-equity ratio, cash flow, and profitability, to improve model performance.

Model Training and Classification:

Logistic Regression (LR): Estimates the probability of bankruptcy based on financial trends.

Decision Trees (DT): Uses a tree-like structure to classify companies as "Bankrupt" or "Non-Bankrupt."

Random Forest (RF): An ensemble learning method that improves prediction accuracy by combining multiple decision trees.

Optimization Techniques:

Particle Swarm Optimization (PSO): Adjusts model parameters to enhance accuracy and reduce classification errors.

Evaluation Metrics:

Precision, Recall, and F1-score: Used to assess the model's performance in distinguishing bankrupt from non-bankrupt companies.

Confusion Matrix: Helps in understanding false positives and false negatives in bankruptcy predictions.

Data Preprocessing:

Handling Missing Values: Ensures financial records are complete for accurate predictions.

Data Normalization: Converts financial data into a standard format to improve model efficiency.

Predictive Insights and Decision-Making:

Machine learning models provide bankruptcy probability scores, helping stakeholders take preventive actions before financial collapse.These insights assist financial institutions, investors, and business owners in making well-informed financial decisions.By integrating feature selection, machine learning classifiers, optimization techniques, and evaluation metrics, the system ensures a robust and reliable bankruptcy prediction framework, enabling businesses to proactively mitigate financial risks.

Advantages:

High Accuracy: Deep learning models outperform traditional statistical methods in bankruptcy prediction. Automated Feature Selection: Unlike traditional ML, deep learning automatically extracts relevant financial features.

Scalability: Deep learning models improve as the dataset size increases.

Robustness: Handles large-scale financial data and detects hidden patterns.

Versatility: Used in various financial applications, including risk assessment and fraud detection.

Disadvantages

High Data Requirements: Deep learning models require extensive labeled financial data.

Computational Complexity: Training deep learning models requires significant computing power (e.g., GPUs).

Interpretability Issues: Deep learning models act as "black boxes," making it hard to interpret their decisions.

Overfitting Risks: Complex models may perform well on training data but fail on new financial data.

Vulnerability to Data Imbalance: Most bankruptcy datasets contain fewer bankrupt cases, leading to biased predictions.

How Machine Learning Works in Bankruptcy Prediction:

Machine learning plays a crucial role in predicting bankruptcy by analyzing vast amounts of financial data to identify patterns and assess the likelihood of financial distress. The process begins with data collection and preprocessing, where financial indicators such as cash flow, liabilities, profit margins, and debt-to-equity ratios are gathered from historical records. Missing values are handled, and data is normalized to ensure consistency. Next, feature selection techniques, including Genetic Algorithms (GA), are applied to identify the most relevant financial variables that contribute to bankruptcy prediction.Once the data is prepared, various machine learning models such as Logistic Regression, Random Forest, and Decision Trees are trained on labeled datasets containing bankrupt and non-bankrupt companies. Particle Swarm Optimization (PSO) is used to fine- tune model parameters for improved accuracy. The models learn complex relationships within financial data and detect early warning signs of financial distress. After training, the models undergo evaluation and validation using performance metrics like accuracy, precision, recall, and F1-score to ensure reliable predictions.When deployed, the system receives new financial data and classifies companies as "Bankrupt" or "Non-Bankrupt", providing risk scores and

insights to decision-makers. By leveraging machine learning, businesses, investors, and financial institutions can make data-driven decisions, anticipate potential financial failures, and implement proactive strategies to prevent bankruptcy, ensuring financial stability and long-term sustainability.

1.2 LITEARTURE SURVEY:

This chapter reviews past research on bankruptcy prediction methods, including traditional statistical techniques, machine learning algorithms, and deep learning advancements. Studies highlight the importance of feature selection, data preprocessing, and model evaluation for improved prediction performance.

References:

1. Shetty, Musa, & Brédart (2022) :

Shetty, Musa, and Brédart (2022) investigate the effectiveness of machine learning models in predicting corporate bankruptcies. Their study builds on previous research by Brédart (2014) that used a simple neural network with Belgian bankruptcy data. The authors explore advanced techniques, including deep neural networks, Random Forests, Support Vector Machines (SVM), and Extreme Gradient Boosting (XGBoost). Despite utilizing only three financial ratios, these methods show promising accuracy in predicting bankruptcy, demonstrating that even limited data can yield reliable forecasts. The study underscores the potential of sophisticated machine learning models in enhancing financial risk prediction, emphasizing that advanced algorithms can lead to more accurate forecasts than traditional methods. This research illustrates the growing role of artificial intelligence in corporate finance, where machine learning has the power to reshape financial stability assessment practices.

2.Shi & Li (2019) :

Shi and Li's (2019) systematic literature review explores corporate bankruptcy prediction models from 1968 to 2017. Using Scopus data, the study identifies core academic contributions in this area, examining the key models and methodologies used for

forecasting bankruptcy. The authors highlight how interest in bankruptcy prediction grew post-2008, following the global financial crisis, which underscored the importance of accurate financial forecasting. Their review also finds that there is limited co-authorship among researchers, indicating that the field has been more individually driven than collaborative. The paper provides an essential overview of bankruptcy prediction research, pointing out the evolution of predictive techniques and areas where future research collaboration is necessary to improve predictive accuracy.

3.Alam et al. (2019):

In their 2019 paper, Alam, Shaukat, Mushtaq, Ali, Khushi, Luo, and Wahab examine methods for predicting corporate bankruptcy to foster a healthier corporate environment. The authors aim to develop predictive models that assess the likelihood of bankruptcy using financial indicators and company health metrics. Their study underscores the necessity of accurate bankruptcy prediction, not only for financial stability but also to ensure better corporate governance and decision-making. By leveraging advanced machine learning techniques, the authors emphasize the importance of improving bankruptcy forecasting models to help businesses avoid financial distress.

4.Perboli & Arabnezhad (2021) :

Perboli and Arabnezhad (2021) present a machine learning-based Decision Support System (DSS) designed for mid- and long-term company crisis prediction. Their system analyzes various financial and operational indicators to assess a company's risk of entering a financial crisis. The study's key focus is to provide a proactive forecasting tool for stakeholders, enabling them to take preventive measures before a crisis occurs. By incorporating machine learning, the DSS enhances traditional financial risk assessment methods, providing more accurate and timely predictions. The authors highlight the value of integrating machine learning into corporate decision- making, as it offers more dynamic, adaptable, and reliable insights compared to traditional forecasting techniques.

5.Boughaci & Alkhawaldeh (2020):

Boughaci and Alkhawaldeh (2020) compare various machine learning techniques for credit scoring and bankruptcy prediction within the banking and finance sectors. Their study evaluates models like Decision Trees, Random Forests, Neural Networks, and Support Vector Machines (SVM), assessing each technique's predictive accuracy for both credit risk and bankruptcy forecasting. By comparing different machine learning classifiers, the authors aim to identify the most effective methods for these critical financial assessments. Their research highlights the importance of selecting appropriate models to enhance decision- making processes in banking and finance, providing insights that could lead to more reliable credit evaluation and bankruptcy predictions. This comparative study supports the integration of machine learning into financial risk management practices, offering improved tools for financial institutions.

1.3 Existing System:

Bankruptcy prediction systems are essential for assessing financial stability and minimizing risks for businesses and investors. These systems analyze financial data to predict potential insolvency and assist in proactive decision-making.

A key component of these systems is the creation and utilization of financial datasets, which are crucial for training machine learning models. Traditional bankruptcy prediction methods rely on Logistic Regression (LR), Decision Trees (DT), and Random Forest (RF) to classify companies as "Bankrupt" or "Non-Bankrupt." Some systems also use Genetic Algorithms (GA) for feature selection and Particle Swarm Optimization (PSO) for hyperparameter tuning to improve model performance.

However, existing systems face several challenges, including imbalanced financial data, difficulty in capturing complex financial patterns, and sensitivity to economic fluctuations. Additionally, many conventional approaches fail to provide high accuracy in predicting bankruptcy risks.

While some advanced models incorporate external financial indicators such as market trends and economic conditions, there is still a need for a more efficient and accurate hybrid machine learning model to improve bankruptcy prediction and reduce financial risks.

DISADVANTAGES

1.**Model Complexity:**Integration of multiple machine learning algorithms can increase model complexity, making it harder to interpret and maintain.

2.Computational Intensity: Running multiple algorithms simultaneously can be computationally intensive, requiring more time and resources for training and inference.

3.Overfitting Risk: Hybrid models may be prone to overfitting, especially if not carefully tuned or if the dataset is small or highly imbalanced.

1.Complexity: Bankruptcy prediction is influenced by numerous financial and non-financial factors, including market conditions, regulatory changes, industry trends, and company-specific variables. Traditional models often struggle to capture the dynamic nature of financial distress, making prediction highly complex.

2.Dependency on High-Quality Data: Accurate bankruptcy prediction relies on well- maintained financial records, including balance sheets, income statements, and cash flow reports. However, financial misreporting, incomplete disclosures, and accounting manipulations can reduce the reliability of the model's predictions.

3.Computational Resources: Machine learning and deep learning models, such as neural networks and LSTMs, require substantial computational power for training and real-time analysis. Small businesses and startups may lack the necessary infrastructure, making large-scale adoption challenging.

4.**Accessibility and Inclusiveness:** Many small businesses and financially struggling firms may not have access to sophisticated bankruptcy prediction tools due to high costs or lack of technical expertise. This creates an imbalance where only well-funded organizations can leverage advanced predictive analytics.

5.User Experience and Dependence: While automated models can improve decision- making, over-reliance on AI-based predictions without human expertise can be risky. Poorly interpreted results may lead to incorrect financial decisions, such as unnecessary restructuring or liquidation.

6.Privacy Concerns: Bankruptcy prediction models require access to sensitive financial data, which poses significant privacy risks. Unauthorized access or data breaches could expose businesses or individuals to financial fraud, identity theft, or reputational damage. Ensuring data security and regulatory compliance is critical.

7.Interoperability and Standardization: Different financial institutions, credit agencies, and governments use varied financial reporting standards and bankruptcy laws, making it difficult to create a universally applicable bankruptcy prediction model. The lack of standardized datasets and regulations adds to the complexity.

8.Market Volatility and External Factors: Economic downturns, global crises, political instability, and unforeseen financial shocks can suddenly alter a company's financial health. Bankruptcy prediction models may struggle to incorporate such unpredictable external factors, leading to inaccurate forecasts.

1.4 Scope of the Existing System:

The scope of the existing bankruptcy prediction systems involves analyzing financial stability and minimizing risks for businesses, investors, and financial institutions. These systems assess financial data to predict potential insolvency, allowing stakeholders to take proactive measures.

Current systems rely on machine learning models such as Logistic Regression (LR), Decision Trees (DT), and Random Forest (RF) for classification. Feature selection techniques like Genetic Algorithms (GA) and hyperparameter tuning using Particle Swarm Optimization (PSO) are used to enhance prediction accuracy. Some systems also integrate external financial indicators, such as stock market trends and economic conditions, to refine their models.

However, the effectiveness of these systems is limited by various challenges, including data imbalance, difficulties in capturing complex financial patterns, and sensitivity to market fluctuations. Many existing solutions lack a holistic approach that incorporates multiple financial and non- financial factors, which affects their accuracy and reliability.

Limitations of the Existing System:

1. **Model Complexity** – The integration of multiple machine learning techniques increases system complexity, making it harder to interpret and maintain.
2. **Computational Intensity** – Running advanced models requires high computational power, leading to increased training time and resource consumption.
3. **Overfitting Risk** – Hybrid models may overfit on limited or imbalanced datasets, reducing their generalization ability.
4. **Complexity of Financial Predictions** – Traditional models struggle to capture the dynamic nature of financial distress, as they do not account for evolving market conditions and industry-specific trends.

5. **Dependence on High-Quality Data** – Bankruptcy prediction relies on accurate financial records, but misreporting and incomplete disclosures reduce the reliability of predictions.

6. **Computational Resource Constraints** – Many startups and small businesses lack the infrastructure to run advanced predictive models effectively.

7. **Accessibility and Inclusiveness** – High implementation costs and technical expertise requirements limit access to predictive systems, disadvantaging smaller enterprises.

8. **Privacy and Security Concerns** – Bankruptcy models require sensitive financial data, raising concerns over unauthorized access, data breaches, and regulatory compliance.

9. **Lack of Standardization** – Different financial reporting standards and bankruptcy laws across countries make it difficult to develop a universally applicable prediction model.

1.5 Proposed System:

The proposed system aims to develop an intelligent and efficient machine learning-based solution for predicting corporate bankruptcy. By leveraging historical financial data, the system classifies companies as either "bankrupt" or "non-bankrupt" using advanced classification and optimization algorithms. The main

objective is to assist stakeholders such as investors, auditors, and financial institutions in making informed decisions and mitigating financial risks.

To achieve this, the system incorporates multiple machine learning techniques:

Random Forest (RF) is used for its ensemble learning capabilities that improve prediction accuracy and handle overfitting efficiently.

Logistic Regression (LR) is applied as a baseline statistical model due to its simplicity, interpretability, and effectiveness in binary classification.

Genetic Algorithm (GA) is used to select optimal features and enhance model performance by identifying the most relevant financial indicators.

Particle Swarm Optimization (PSO) is used for fine-tuning hyperparameters and further boosting the model's prediction accuracy.

1.6 Novelty of Proposed System:

1.Hybrid Machine Learning Model for Enhanced Accuracy:

The system integrates Decision Trees (DT), Random Forest (RF), and Logistic Regression (LR) to leverage the strengths of multiple models, ensuring higher accuracy in bankruptcy prediction.

2.Optimization Through Genetic Algorithms (GA) and Particle Swarm Optimization (PSO):

The model employs GA for feature selection and PSO for hyperparameter tuning, improving performance and achieving higher precision, recall, and F1-score compared to traditional models.

3.Addressing Imbalanced Datasets with Custom Techniques:

The system applies feature balancing techniques and optimization methods to improve classification performance, ensuring accurate predictions for both bankrupt and non-

bankrupt companies.

4.Feature Selection for Improved Efficiency:

Genetic Algorithm-based feature selection helps identify the most relevant financial indicators, reducing dimensionality, enhancing efficiency, and preventing overfitting.

5.Scalability and Adaptability to Different Financial Markets:

The model is adaptable across various financial markets, allowing customization based on different economic conditions and company data.

6.Practical Applications for Financial Risk Management:

The system can be deployed in financial institutions,businesses, and investment firms for bankruptcy risk prediction, credit risk valuation, and financial decision-making.

7.User-Friendly Dashboard with Real-Time Risk Assessment :

An interactive web-based interface allows financial professionals to upload financial data, visualize risk levels, and generate predictive insights, categorizing companies into High, Medium, or Low bankruptcy risk levels.

2.System Study

2.1 Feasibility Study:

A successful feasibility study is crucial for any project, ensuring that it is viable, cost-effective, and efficient before implementation. Many projects fail due to incorrect assumptions or inaccurate data. Therefore, a proper feasibility study ensures that the project is backed by factual, updated, and reliable information.

The **main objective** of this feasibility study is to evaluate three key aspects:

Technical Feasibility:
Assessing whether the project can be developed using the available technologies, tools, and expertise.

Operational Feasibility: Ensuring that the system can function effectively in real-world scenarios and be used by businesses or financial institutions.

Economic Feasibility: Analyzing the development, implementation, and maintenance costs to determine its financial viability.

Importance of Conducting a Feasibility Study

Conducting a feasibility study is essential for:

- Understanding the scope and impact of the bankruptcy prediction system.
- Identifying the potential risks and challenges before implementation.
- Making data-driven decisions to enhance the accuracy and efficiency of the system.
- Estimating costs, required resources, and expected returns.

General Criteria for Feasibility Study

1.User Requirements – The system should efficiently process financial data, analyze trends, and predict bankruptcy risk with high accuracy.

2.Resource Availability – The project will use **Flask** as the backend framework, along with **machine learning models**, a **database for financial records**, and web-based visualization tools.

3.Impact on Organizations – The system aims to assist businesses, financial analysts, and stakeholders in making informed financial decisions and mitigating risks.

Investigation and Evaluation Procedure

The feasibility study evaluates the project by:

Identifying key challenges such as **data availability, model accuracy, and integration with financial databases**.

Defining **expected performance**, including **prediction accuracy and response time**.

Estimating the **cost of development**, including hosting, model training, and database management.

Selecting the best **machine learning algorithms** and **efficient data processing techniques**.

Outcome of the Study

1.Brief Statement of the Proposed Project – The Bankruptcy Prediction System is designed to analyze financial data and predict whether a company is at risk of bankruptcy using machine learning models deployed via Flask.

2.Details of Findings –

The system will be built using **Python (Flask, Scikit-Learn, Pandas, NumPy)**.

Financial data will be processed and stored in a **relational database** such as **SQLite**.
Machine learning algorithms like **Logistic Regression, Random Forest, or Neural** Networks will be used to enhance accuracy.
The system will feature a **web-based UI** for user interaction and result visualization.

3.Recommendations and Final Conclusions –

Implementation Study: The project is technically feasible with the available resources and expertise.

Operational Study: The system will be easy to use for financial analysts and businesses.

Economic Study: Development costs will mainly involve **data acquisition, cloud hosting, and model training**, making it a **cost-effective** solution.

2.1.1 Economic Feasibility:

Economic feasibility assesses whether the **Bankruptcy Prediction System** is financially viable by evaluating costs, benefits, and overall return on investment. This analysis ensures that the project can be executed within budget constraints while delivering value to the stakeholders

2.1.2 Technical Feasibility:

Technical feasibility focuses on evaluating whether the project can be successfully implemented using the available technologies, tools, and expertise. This assessment is crucial in determining whether the **Bankruptcy Prediction System** can be developed efficiently and function effectively in

real-world applications.

2.1.3 Social Feasibility:

Social feasibility assesses the **acceptance and adaptability** of the **Bankruptcy Prediction System** by its users, including financial analysts, business owners, and investors. The system must be user- friendly, instill confidence, and be perceived as an essential tool rather than a threat.

3.REQUIREMENT ANALYSIS

System Requirements:

A requirement is a feature that the system must have or a constraint that it must to be accepted by the client. Requirement Engineering aims at defining the wants of the system under construction. Requirement Engineering include two main activities requirement elicitation which results in the specification of the system that the client understands and analysis which in analysis model that the developer can unambiguously interpret. A requirement may be a statement about what the proposed system will do. Requirements can be divide into two major categories:

Functional Requirements.

Non-Functional Requirements

3.1 Functional Requirements:

A Functional Requirement describes a service that the software must offer. It details a software system or its component, including inputs, behaviors, and outputs. Functional Requirements describe interactions between the system and its environment independent of its implementation.

- Upload financial datasets for training and testing.
- Apply machine learning models to predict bankruptcy.
- Display prediction results with risk assessment labels.
- Allow for real-time data input and analysis.

Data Collection:

- Collect a comprehensive dataset containing financial records

of various businesses, ensuring diversity in industry types and company sizes.

- Gather data attributes such as assets, liabilities, revenue, expenses, and cash flow metrics.

Data Preporcessing:

- Standardize input financial records to ensure consistency.
- Apply data cleaning techniques to handle missing values and anomalies.
- Perform feature engineering to improve predictive performance.

Train the Machine Learning Model:

- Define an appropriate machine learning architecture for bankruptcy prediction.
- Choose suitable loss functions and optimizers for accurate prediction.

Bankruptcy Prediction:

- Implement algorithms to classify businesses as high-risk or low-risk based on financial health.
- Generate detailed reports explaining the prediction results.

3.2 Non-Functional Requirements:

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional

requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement. Non-Functional Requirements specifies the standard attribute of a software . They judge the software supported Responsiveness, Usability, Security, Portability, and other non-functional standards that are critical to the success of the software.An example of a nonfunctional requirement, “how fast does the website load?” Failing to satisfy non-functional requirements may result in systems that fail to satisfy user needs.Non-functional Requirements allow you to impose constraints or restrictions on the planning of the system across the varied agile backlogs.

- User Interface and Human Factors
- Software Requirements
- Hardware Requirements
- Usability
- Reliability
- Performance
- Supportability
- Physical Environment
- Security Requirment

3.2.1 User Interface and Human Factors:

Users must be able to input financial data, train models, and review predictions through an interactive dashboard.

3.2.2 Requirment Specifications:

Operating System	: Windows 7/8/10/11
Programming Language	: Python
Libraries	: TensorFlow, Keras, Pandas, NumPy, Scikit-learn
IDE/Workbench	: VisualStudio Code
Technology	: Python 3.6+
Database	: SQLLite

3.2.3 Hardware Requirements

Processor	: I3/Intel Processor
RAM	:4GB (min)
Hard Disk :	:128 GB

3.2.4 Usability

The bankruptcy prediction system is designed to be user-friendly, ensuring ease of access and interaction for both technical and non-technical users. The system features a well-structured graphical user interface (GUI) with intuitive navigation, enabling users to upload financial datasets, initiate predictions, and interpret results effectively. Clear visualizations and reports enhance understanding, making it easier for businesses, financial analysts, and investors to make informed decisions.

3.2.5 Reliability

The system ensures high reliability by implementing robust error-handling mechanisms and data validation techniques. It is designed to provide consistent and accurate bankruptcy predictions with minimal downtime. Backup and recovery strategies are integrated to prevent data loss and ensure seamless operation even in case of unexpected failures. The system undergoes rigorous testing to ensure stability across different financial datasets and market conditions.

3.2.6 Performance

Performance optimization is a key aspect of the system, ensuring fast and efficient processing of large financial datasets. Advanced machine learning algorithms are optimized for speed, and the system is capable of handling real-time financial data with minimal latency. Parallel processing and caching techniques are used to enhance computational efficiency and improve response times.

3.2.7 Supportability

The system is designed to be scalable and maintainable, allowing for future enhancements and integration with additional financial indicators. It supports modular updates, enabling improvements in machine learning models, feature selection methods, and optimization techniques. Comprehensive documentation and user guides are provided to assist users in system operation, troubleshooting, and maintenance.

3.2.8 Physical Environment

The bankruptcy prediction system is a software-based solution that can be deployed on cloud platforms or onpremises servers. It does not require specialized hardware and is accessible via web-based or desktop interfaces.

The system is designed to function efficiently in various environments, including corporate offices, financial institutions, and research organizations, ensuring flexibility in deployment.

3.2.9 Security Requirements

Security is a critical aspect of the system, as it handles sensitive financial data. The system incorporates strong authentication mechanisms, role-based access control (RBAC), and encryption protocols to protect data integrity and confidentiality. Secure data storage and transmission methods are implemented to prevent unauthorized access and cyber threats. Compliance with financial regulations and data protection laws ensures that user and company information remains protected at all times.

4.System Analysis

4.1 Introduction:

System analysis is the process of examining and understanding a system's structure, functionality, and requirements to improve its efficiency and effectiveness. It involves studying the existing system, identifying limitations, and proposing enhancements to address challenges.In the context of bankruptcy prediction, system analysis plays a crucial role in designing a robust financial risk assessment model. It helps in understanding financial data patterns, selecting appropriate machine learning techniques, and ensuring seamless integration of predictive analytics into decision-making processes. By conducting a thorough analysis, the system can be optimized to improve accuracy, reliability, and usability, ultimately aiding businesses, financial institutions, and investors in mitigating financial risks and making informed decisions.This phase involves requirement gathering, feasibility analysis, and identifying functional and non-functional aspects to develop a well-structured bankruptcy prediction system that meets the stakeholders' needs.

4.2 Usecases:

4.2.1 Actors:

Actors represent roles which may include human users, external hardware, or other systems. Our proposed system is quite simple in terms of user interface. It is only having three actors: User, System, and Admin.

Following are the actors in our project:

1.User:

User 

The user is the actor who interacts with the system by submitting financial data to predict bankruptcy.

2.System:

System 

The system is responsible for processing user data, running predictive models, and displaying results.

3.Admin:

Admin 

The admin is responsible for managing the system, training machine learning models, and analyzing datasets.

4.2.2 List of use cases:

Use Case Name	Actors
Login	User and Admin
Upload Financial Dataset	Admin
Preprocess Dataset	Admin
Run Bankruptcy Prediction Model	Admin
Submit Financial Data for Prediction	User
Display Prediction Result	System

Use Case Name	Description
Login	The user can log into the system.
Upload Financial Dataset	The dataset is uploaded by the Admin.
Preprocess Dataset	Financial data is cleaned and prepared for analysis.

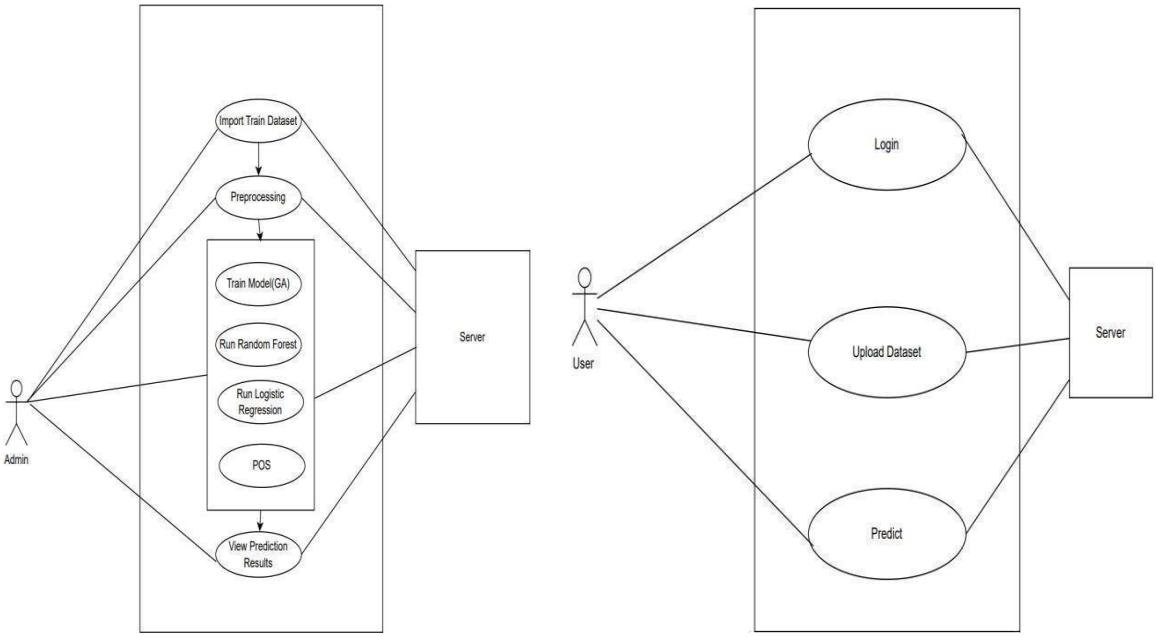
Run Bankruptcy Prediction Model	After preprocessing, the system runs machine learning algorithms.
Submit Financial Data for Prediction	User submits company financial data for prediction.
Display Prediction Result	The system displays whether the company is predicted to go bankrupt or not.

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

4.2.3 Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, a use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors. Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration.They provide a good high-level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.A use case diagram is a graphical depiction of a user's possible interactions with

a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined.The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous. The main purpose of a use case diagram is to show what system functions are performed for which actor.

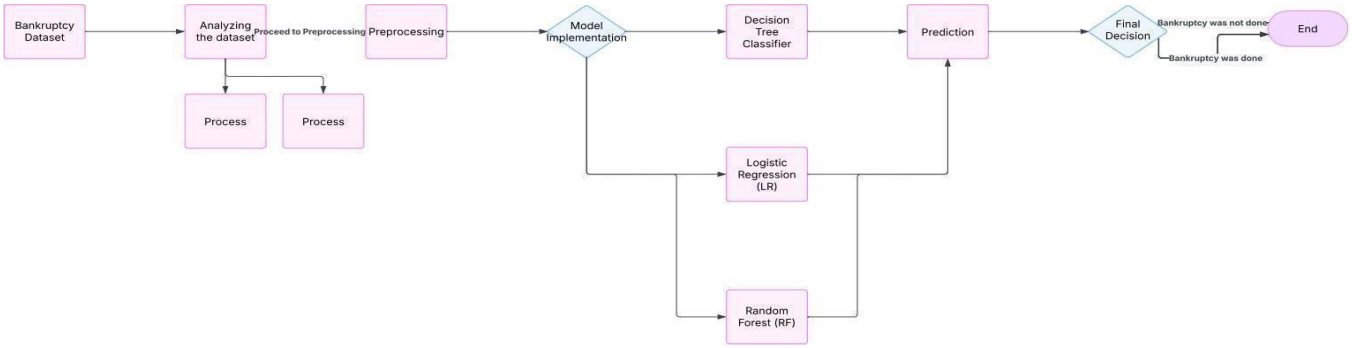


5.System Design

5.1 Introduction:

System design is the process of defining the architecture, components, modules, and data flow of a system to meet specified requirements. It acts as a bridge between system analysis and implementation, ensuring that all functional and non-functional requirements are effectively addressed. In a bankruptcy prediction system, system design focuses on structuring the machine learning models, data preprocessing pipelines, and user interface components. It involves designing the overall architecture, selecting suitable algorithms, and ensuring efficient data processing to enhance accuracy and usability. The system is structured to handle financial data efficiently, integrate predictive analytics, and provide meaningful insights to users. This phase includes designing UML diagrams such as use case diagrams, class diagrams, sequence diagrams, and data flow diagrams to visualize system interactions. A well-designed system ensures scalability, reliability, and security, making it an essential step toward successful implementation.

5.2 System Architecture:



System Design for Bankruptcy Prediction System

The system design for the **Bankruptcy Prediction System** is categorized into three main types: **GUI Design, UML Design, and Database Design**. Each component ensures smooth project development and implementation.

5.3 System Model:

5.3.1 Introduction:

The system model defines the structure and interactions of various components involved in the bankruptcy prediction system. It provides a high-level overview of how data flows from input to output, ensuring seamless data processing, analysis, and prediction generation. The model is designed to be scalable, efficient, and capable of handling large financial datasets. It integrates multiple subsystems to manage data preprocessing, feature selection, machine learning, result visualization, and reporting.

5.3.2 Subsystems:

The bankruptcy prediction system consists of several key subsystems that work together to process financial data and generate bankruptcy risk assessments.

1.Data Acquisition Subsystem:

Function: Collects financial data from various sources such as financial reports, company databases, or user uploads.

Input: Raw financial data, including balance sheets, income statements, and cash flow reports.

Process: Validate and clean financial data.
Identify missing values and handle outliers. Output: Structured and clean financial dataset.

2.Data Preprocessing & Transformation Subsystem:

Function: Ensures data quality before feeding it into machine learning models. Process:

Normalize numerical values (e.g., revenue, profit, debt-to-equity ratio).

Convert categorical values into numerical representations (e.g., company size: Small = 0, Medium = 1, Large = 2).

Handle missing or incomplete data using imputation techniques. Output: Processed dataset ready for feature selection.

3.Feature Selection & Dimensionality Reduction Subsystem:

Function: Selects the most relevant financial features that impact bankruptcy prediction. Process:

Use Genetic Algorithms (GA) to eliminate irrelevant or redundant financial indicators. Perform Principal Component Analysis (PCA) for dimensionality reduction.

Rank features based on their importance in predicting bankruptcy. Output: Optimized dataset with selected financial attributes.

4.Machine Learning Model Training & Prediction Subsystem:

Function: Applies machine learning algorithms to classify companies as Bankrupt or Non-Bankrupt. Process:Train models such as Logistic Regression, Decision Trees, Random Forest, and Naïve Bayes on historical financial data.

Use Particle Swarm Optimization (PSO) for hyperparameter tuning. Validate model accuracy using cross-validation techniques.

Predict bankruptcy status for new financial data. Output: Bankruptcy classification results.

5.4 Object Description:

5.4.1 Objects:

The bankruptcy prediction system consists of the following key objects:

1.User:

Description: Represents an individual using the system, such as investors, financial analysts, or business owners. Attributes:

- User_ID: Unique identifier for each user. □ Name: Name of the user. □ Email: User's email for communication.
- Password: Encrypted password for authentication.
- User_Role: Defines user type (Admin, Analyst, General User).

Methods:

- Login(): Allows the user to log in.
- Logout(): Ends the user session.
- UploadFinancialData(): Enables the user to upload financial data. □ ViewReports(): Allows users to access prediction reports.

2.Financial Data:

Description: Represents the financial records used for bankruptcy prediction. Attributes:

- Company_ID: Unique identifier for the company.
- Balance_Sheet: Stores key financial indicators.
- Income_Statement: Stores revenue, expenses, and profits.
- Cash_Flow_Statement: Captures cash inflows and outflows.
- Debt_to_Equity_Ratio: Represents financial leverage.
- Return_on_Assets (ROA): Measures profitability.
- Net_Profit_Margin: Indicates overall profitability.
- Total_Assets: Represents the total assets owned by the company.
- Total_Liabilities: Represents the total debt obligations. □ Timestamp: Stores the last update time. Methods:
- ValidateData(): Ensures financial records are accurate.
- NormalizeData(): Converts raw financial values into standardized formats.
- FilterData(): Removes missing or irrelevant values.

3.Feature Selection

Description: Represents the process of selecting key financial indicators. Attributes:

- Selected_Features: List of the most relevant features.
- Algorithm_Used: Specifies the method used (Genetic Algorithm, PCA, etc.). Feature_Importance_Score: Ranks features based on importance.

Methods:

- SelectFeatures():Chooses the best attributes for model training.

RemoveRedundantFeatures(): Eliminates unnecessary data.

4.Machine Learning Model

Description: Represents the predictive model used to classify companies. Attributes:

- Model_ID: Unique identifier for the trained model.
- Algorithm_Type: Specifies the model type (Logistic Regression, Decision Tree, Random Forest, Naïve Bayes).
- Hyperparameters: Stores model configurations.
- Training_Accuracy: Measures model performance.
- Validation_Accuracy: Evaluates model generalization. Date_Trained: Timestamp of the last training.

Methods:

- TrainModel(): Trains the model using historical financial data.
- PredictBankruptcy(): Classifies a company as "Bankrupt" or "Non-Bankrupt."
- EvaluatePerformance(): Measures model accuracy, precision, recall, and F1-score.

5.Prediction Result

Description: Stores the output of
the bankruptcy prediction model.
Attributes:

- Prediction_ID: Unique identifier for each prediction.
- Company_ID: Links to the respective company.
- Risk_Score: Probability of bankruptcy.
- Prediction_Label: "Bankrupt" or "Non-Bankrupt."
- Gener

ated_On:
Timestamp of
prediction.
Methods:

- GenerateReport(): Creates a detailed financial risk report.
- ExportData(): Saves results in CSV, or Excel.

6.Report & Visualization

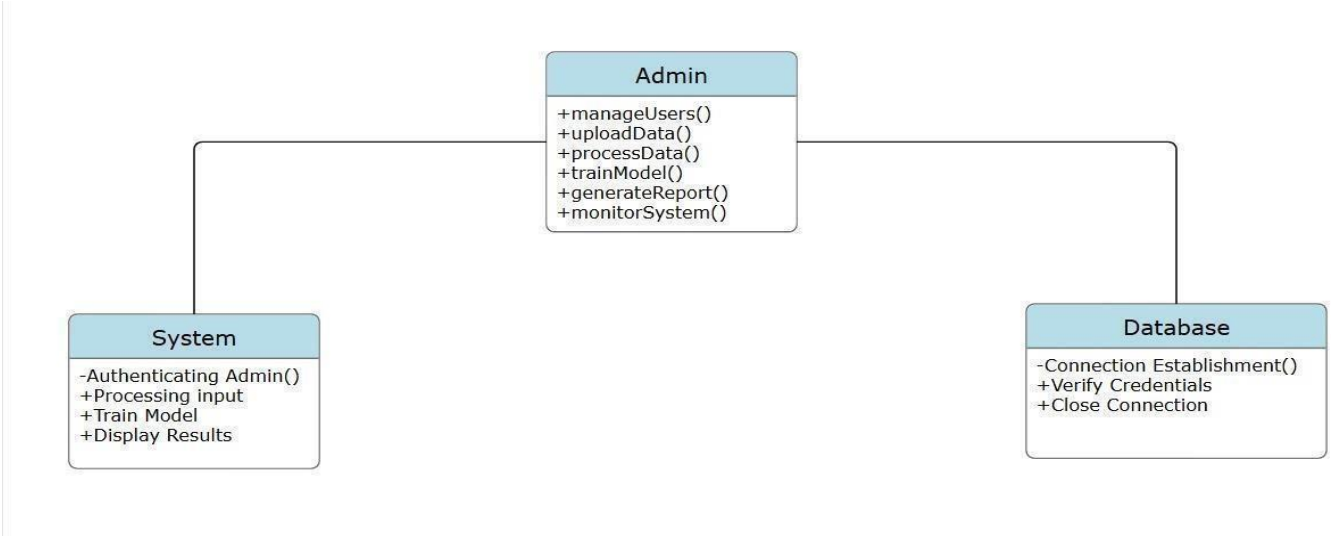
Description: Represents the graphical and textual
representation of results. Attributes:

- Report_ID: Unique identifier.
- User_ID: Links the report to a user.
- Visualization_Type: Specifies report format (Table, Graph, Pie Chart).
- File_Format: PDF, Excel, CSV.
- Date_Created:

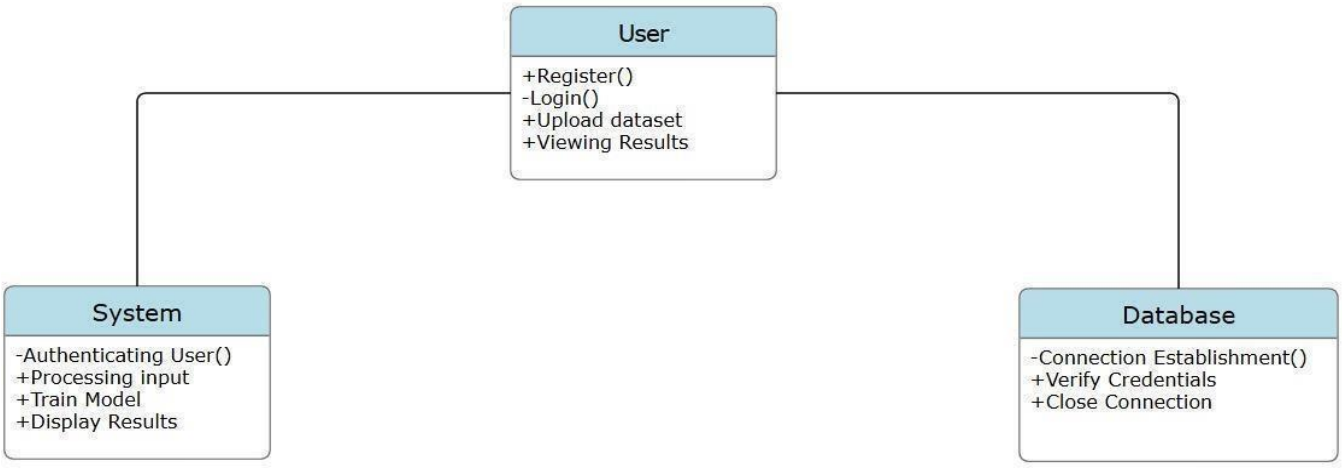
Timestamp when the report was
generated. Methods:

- DisplayGraphs(): Generates risk analysis visualizations.
- DownloadReport(): Allows users to save reports.

5.4.2 Class Diagram:



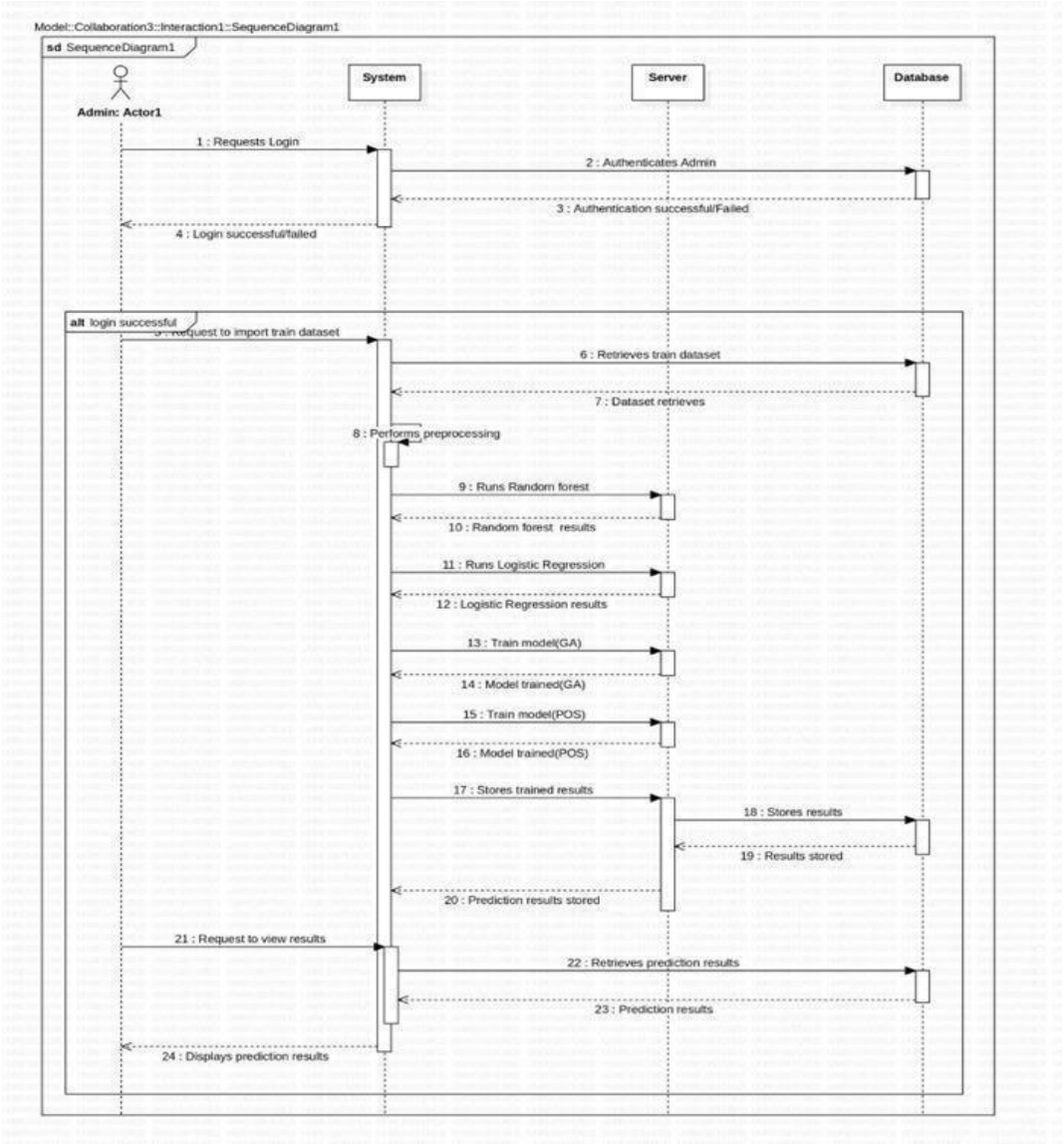
Admin Class Diagram



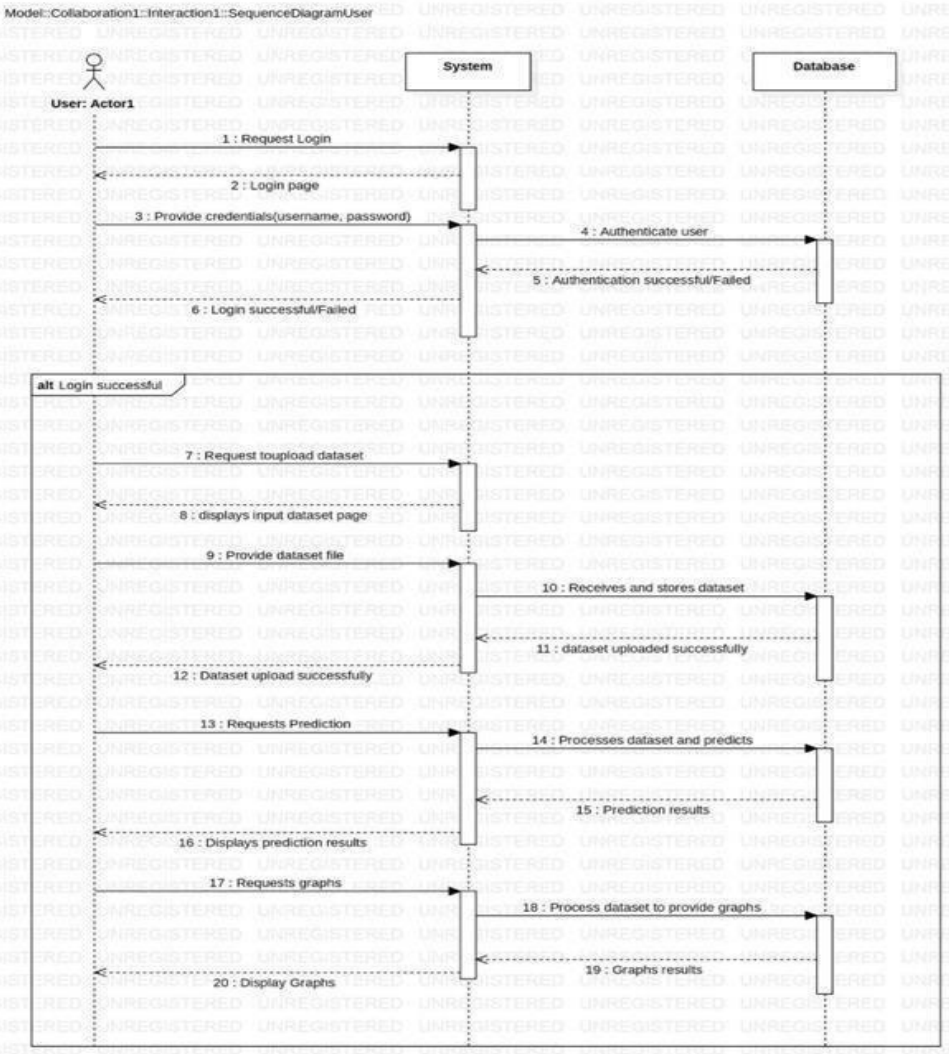
User Class Diagram

5.5 Dynamic Model:

5.5.1 Sequence Diagram:

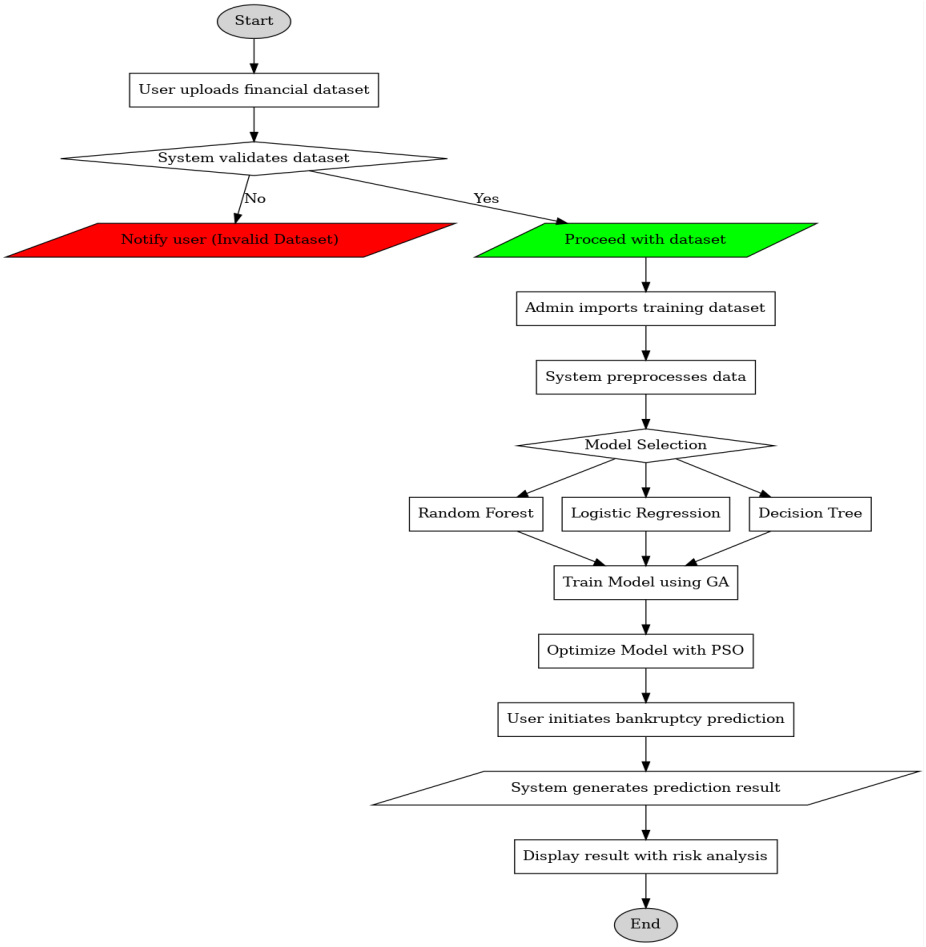


Admin Sequence Diagram



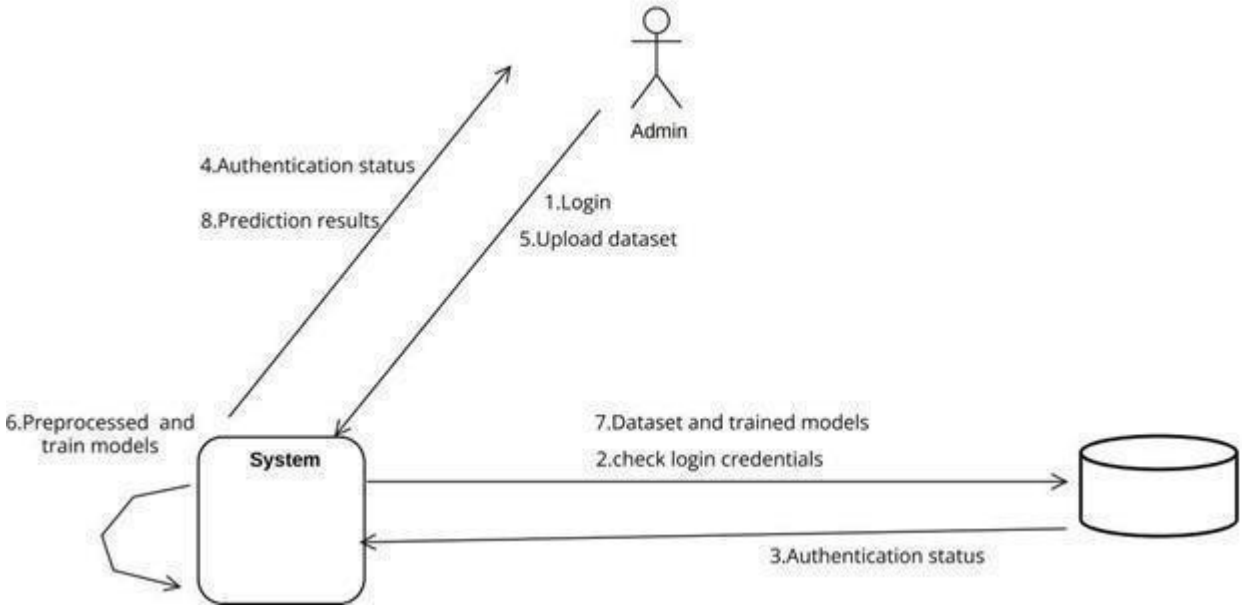
User Sequence Diagram

5.5.2 Activity Diagram:

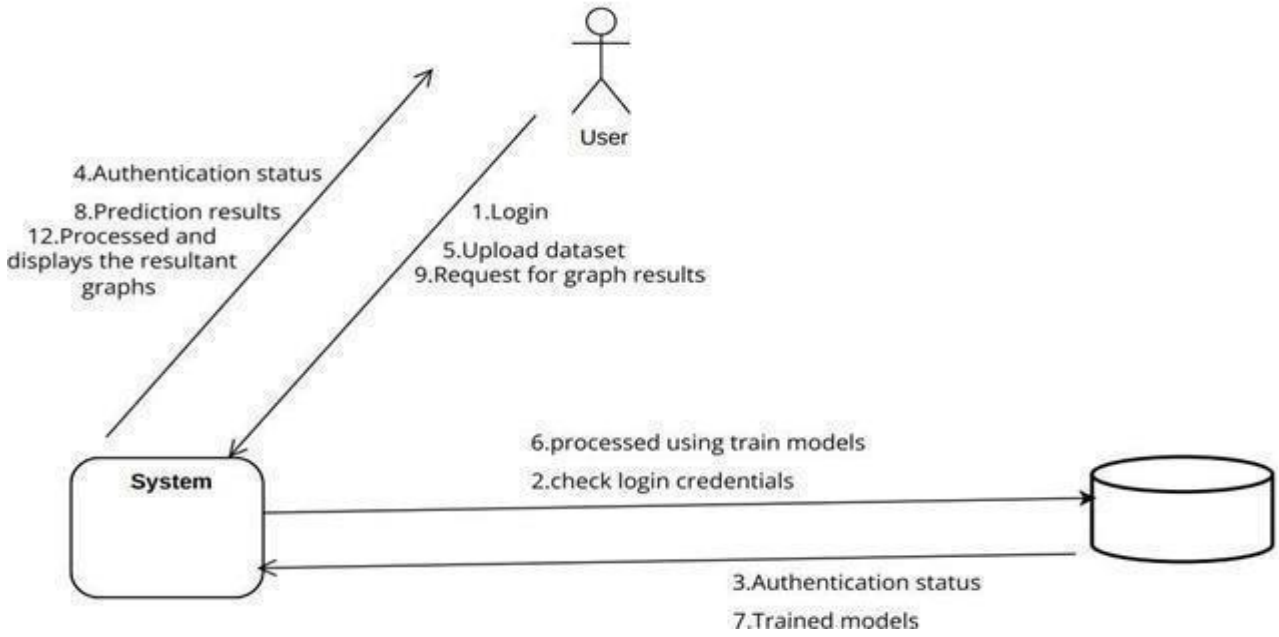


5.6 Object Collobration:

5.6.1 Object Collobration Diagram:



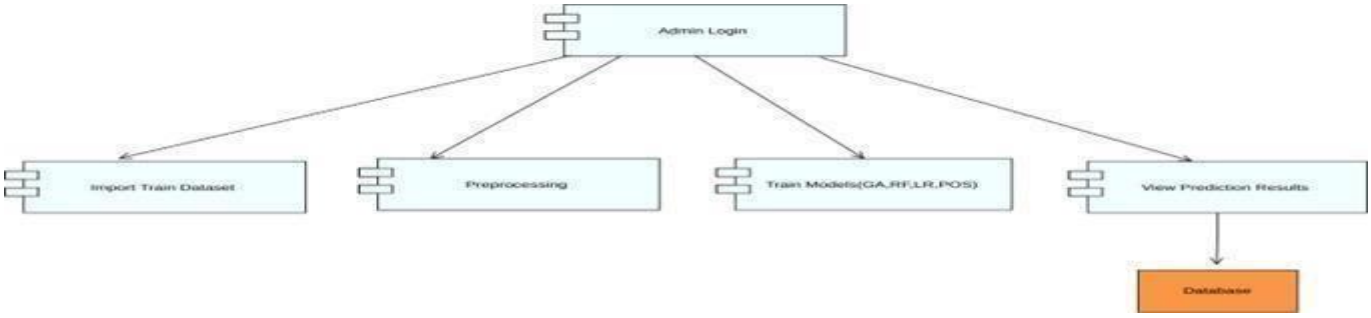
Admin Collaboration Diagram



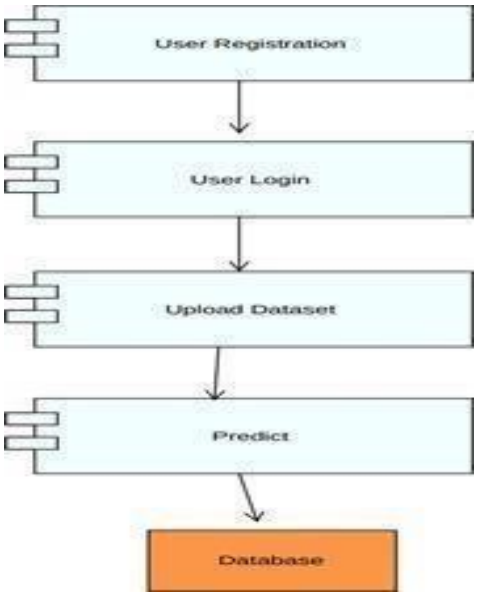
User Collaboration Diagram

5.7 Static Model:

5.7.1 Component Diagrams:



Admin Component Diagram



User Component Diagram

5.7.2 Deployment Diagrams:

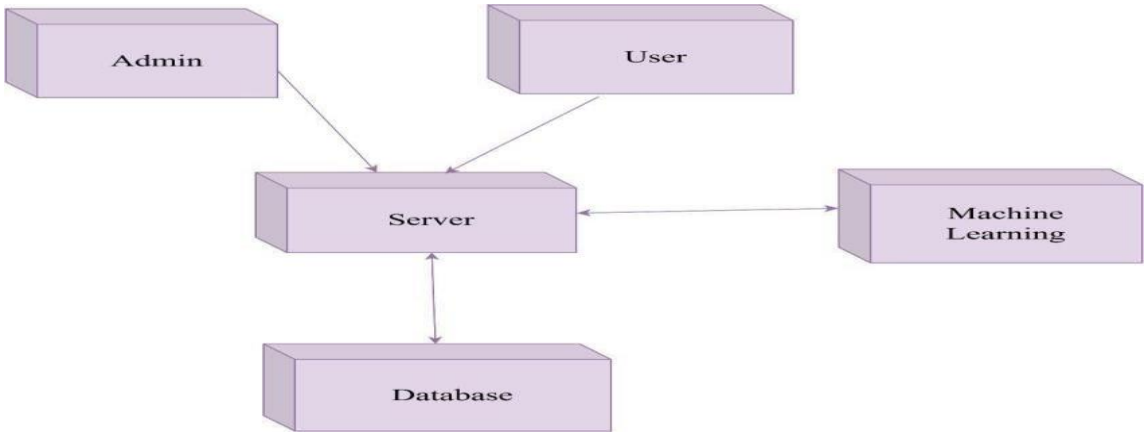


Figure: Deployment Diagram

Data Flow Diagram:

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Game and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for

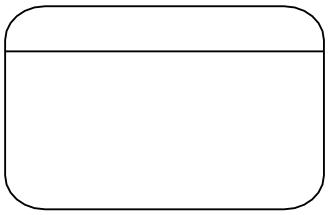
analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

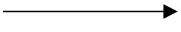
DFD Symbols:

In the DFD, there are four symbols

- A square defines a source(originator) or destination of system data.
- An arrow identifies data flow. It is the pipeline through which the information flows.
- A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
- An open rectangle is a data store, data at rest or a temporary repository of data.

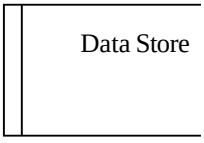


Process that transforms data flow.



Data flow

Source or Destination of data



Constructing a DFD

Several rules of thumb are used in drawing DFD’S:

Process should be named and numbered for an easy reference. Each name should be representative of the process. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source.One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal. When a process is exploded into lower level details, they are numbered. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized. A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out. Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

Silent Feature of DFD's

1.The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.

2.The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.

3.The sequence of events is not brought out on the DFD.

Data Flow:

1)A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.

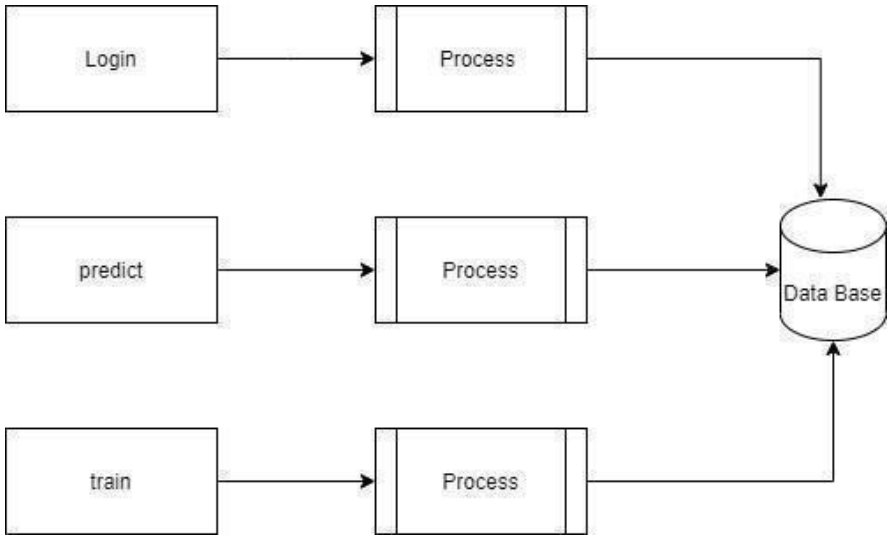
2)A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.

3)A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.

4)A Data flow to a data store means update (delete or change).

5)A data Flow from a data store means retrieve or use. A data flow has a noun phrase label on the same arrow move together as one package.more than one data flow noun phrase can

appear on a single arrow as long as all of the flows



6.Modules

6.1Module Description:

6.1.1 Admin Module: This module provides administrator access and control over the system.

Admin Login:

Purpose: Allows administrators to log into the system.

Key Features: Username and password verification.

Outcome: Access to manage datasets, train models, and oversee system operations.

Manage Datasets:

Purpose: Enables the admin to upload and preprocess financial datasets.

Key Features: Interface to upload financial data, clean missing values, and apply normalization techniques.

Outcome: Prepared dataset ready for model training.

6.1.1.1 Train Dataset Model

Initiates training of machine learning models using historical financial data to enhance bankruptcy prediction accuracy.

6.1.1.2 Preprocessing&Algorithm Training:

Allows selection of algorithms such as Decision Trees (DT), Random Forest (RF), Logistic Regression (LR), and hybrid models. Enables real-time monitoring of training progress and performance evaluation using metrics like accuracy, precision, recall, and AUC score.

6.1.1.3 View Results:

A well-trained predictive model capable of accurately assessing bankruptcy risks and aiding in financial decision-making.

6.1.2 User Module: This module manages user interactions with the system.

6.1.2.1 Login And Registration:

6.1.2.1.1 User Registration:

6.1.2.1.2 Purpose: Allows financial analysts and decision-makers to create accounts.

6.1.2.1.3 Key Features: Form to enter user details (e.g., name, email, password) with data validation.

6.1.2.1.4 Outcome: User profile created and stored, enabling secure access.User Login:

6.1.2.1.5 Purpose: Authentication for users to access bankruptcy prediction tools.

6.1.2.1.6 Key Features: Username and password input, verification against stored credentials.

6.1.2.2 Upload Dataset:

User Can Upload Financial dataset and can predict the financial data is bankrupted or non bankrupted

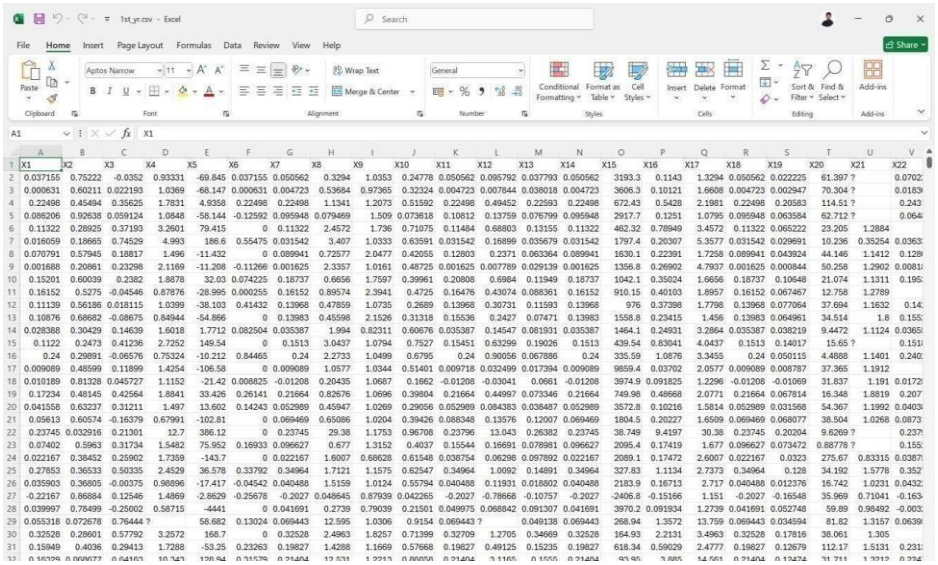
6.1.2.3 Getting Results:

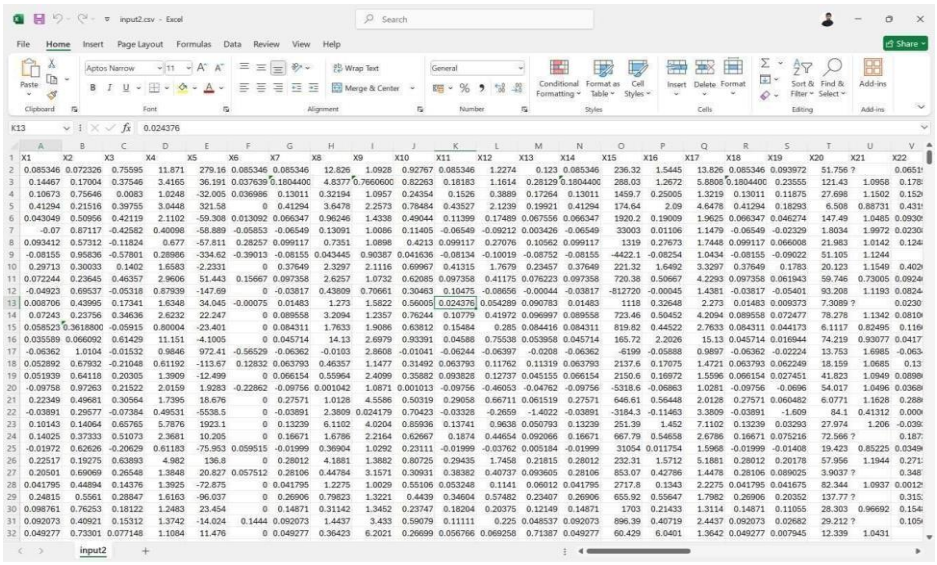
Users gain access to input financial data and generate predictions.

6.1.3 System:

6.1.3.1 Gathering datasets:

The dataset used for bankruptcy prediction consists of financial records from multiple companies, categorized based on their financial health. It includes key financial ratios, profitability metrics, cash flow indicators, and market-based factors that influence bankruptcy risk.The dataset is structured into two subsets: training data and testing data to ensure accurate model performance.





6.1.3.2 Splitting Datasets:

Splitting a dataset is a crucial step in training and evaluating machine learning models. It ensures that models generalize well to new, unseen data rather than just memorizing the training examples. Typically, datasets are divided into three main parts: training set, validation set, and test set.

1. **Training Set**

The training set is the largest portion of the dataset, used to train the machine learning model. The model learns patterns, relationships, and features from this data. A common split ratio is 70-80% of the total dataset.

2. Validation Set

The validation set helps fine-tune model parameters and hyperparameters. It acts as an intermediate check, ensuring that the model does not overfit to the training data. It typically consists of 10-15% of the dataset.

3. Test Set

The test set is a completely unseen portion of data used to evaluate the final model's performance. It represents real-world scenarios and assesses generalization. It usually accounts for 10-20% of the total dataset.

Different Splitting Strategies

1. **Holdout Method (Train-Test Split)**
    - The dataset is divided into two or three parts: training, validation, and test sets.
    - A common approach is an **80-20** or **70-30** split.
    - This method is simple and works well for large datasets.
2. **k-Fold Cross-Validation**
    - The dataset is divided into **k** equal parts (folds). ○ The model is trained on **k-1 folds** and tested on the remaining fold. ○ The process repeats **k times**, ensuring every instance is used for both training and validation.
    - This method is useful for small datasets where data scarcity is an issue.
3. **Stratified Splitting**

- Ensures that both training and test sets maintain the same class distribution.
- This is particularly useful for **imbalanced datasets**, such as bankruptcy prediction, where one class (e.g., bankrupt companies) is much smaller than the other.

Choosing the Right Splitting Strategy

Scenario	**Recommended Splitting Method**
Large datasets (≥ 10,000 samples)	Train-test split (80-20 or 70-30)
Small datasets (< 1,000 samples)	k-Fold Cross-Validation (k=5 or 10)
Imbalanced datasets	Stratified

Importance of Proper Dataset Splitting A well-structured dataset split helps in:

- **Avoiding Overfitting:** Prevents the model from memorizing patterns that do not generalize.
- **Ensuring Generalization:** Helps the model perform well on new, unseen data.
- **Enhancing Model Evaluation:** Provides a realistic measure of model performance before deployment.

6.1.3.3 Algorithm Choosing and validation:

For this bankruptcy prediction system, the selected algorithms are **Random Forest (RF), Logistic Regression (LR), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO)**. These algorithms were chosen based

on their ability to handle financial data, interpretability, and prediction accuracy.

1. Choosing the Right Algorithm 1.1 Logistic Regression (LR)

- A simple and interpretable model used for binary classification (bankrupt or non-bankrupt).
- Works well when financial ratios and indicators have a linear relationship with bankruptcy risk.
- Provides probability-based predictions, making it useful for financial decision-making.

1.2 Random Forest (RF)

- An ensemble learning method that builds multiple decision trees to enhance accuracy.
- Reduces overfitting by averaging multiple trees, making it more robust for bankruptcy prediction.
- Handles missing values and noisy financial data effectively.

1.3 Genetic Algorithm (GA)

- A heuristic search and optimization technique based on natural selection principles.
- Used for feature selection and model optimization in bankruptcy prediction.
- Helps in selecting the most relevant financial indicators to improve model accuracy.

1.4 Particle Swarm Optimization (PSO)

- A population-based optimization algorithm inspired by the social

behavior of birds and fish.

- Used to optimize hyperparameters of machine learning models.
- Enhances the predictive capability of RF and LR by fine-tuning their parameters for better accuracy.

2. Model Validation Techniques

After selecting the algorithms, validation ensures that they perform well on unseen financial data. The following validation methods were used:

2.1 Holdout Validation (Train-Test Split)

- The dataset is divided into **training (80%)** and **testing (20%)** sets.
- The model is trained on the training data and evaluated on the test data.
- Suitable for ensuring the model generalizes well to new bankruptcy cases.

2.3 k-Fold Cross-Validation

- The dataset is split into **k subsets** (e.g., k=5 or k=10).
- The model is trained and tested on different subsets to ensure performance consistency.
- Helps in reducing bias and variance in bankruptcy prediction.

2.4 Stratified Cross-Validation

- Ensures that both bankrupt and non-bankrupt cases are equally represented in each fold.
- Useful for imbalanced datasets where bankrupt companies are fewer.

3. Performance Evaluation Metrics

To assess the accuracy and reliability of the selected algorithms, the following metrics were used:

- **Accuracy:** Measures the overall correctness of bankruptcy predictions.
- **Precision & Recall:** Ensures that the model correctly identifies bankrupt companies.
- **F1-Score:** A balanced metric that considers both precision and recall.
- **ROC-AUC Score:** Evaluates the model's ability to differentiate between bankrupt and non-bankrupt firms.

6.1.3.4 Examine the accuracy:

Evaluating the accuracy of bankruptcy prediction models is essential to ensure their reliability in assessing financial stability. The models used—**Random Forest (RF), Logistic Regression (LR), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO)**—are examined using various performance metrics.

1. Methods to Evaluate Accuracy

1.1 Confusion Matrix

The confusion matrix helps analyze model performance by comparing actual and predicted classifications. It highlights correct and incorrect predictions, allowing insight into misclassification rates.

1.2 Accuracy Score

Accuracy measures the proportion of correct predictions among total

predictions. A higher accuracy indicates better model performance, but for imbalanced datasets, additional metrics are needed to avoid misleading interpretations.

1.3 Precision, Recall, and F1-Score

- **Precision** determines how many of the predicted bankrupt firms were actually bankrupt.
- **Recall** evaluates the model's ability to detect bankrupt firms correctly.
- **F1-Score** balances precision and recall, ensuring the model does not overly favor one at the expense of the other.
-

1.4 ROC-AUC Score

This metric assesses how well the model distinguishes between bankrupt and non-bankrupt firms. A higher score indicates better predictive capability, making it useful for imbalanced datasets.

2. Comparative Accuracy of Algorithms

- **Logistic Regression (LR):** Works well for simple and linear financial data but may struggle with complex patterns.
- **Random Forest (RF):** Provides higher accuracy by combining multiple decision trees, making it effective for handling non-linear relationships.
- **Genetic Algorithm (GA):** Optimizes feature selection, improving the model's ability to focus on relevant financial indicators.

- **Particle Swarm Optimization (PSO):** Enhances model performance by fine-tuning parameters, leading to better accuracy and stability.

3. Factors Influencing Accuracy

- **Quality of Financial Data:** Inaccurate or incomplete financial records can negatively impact predictions.
- **Data Imbalance:** If the dataset has significantly fewer bankrupt firms than non-bankrupt ones, accuracy alone is not a reliable measure.
- **Hyperparameter Optimization:** Techniques like PSO fine-tune model parameters, improving accuracy and generalization.

6.1.3.5 Using the best model for prediction:

Selecting the best model for bankruptcy prediction is critical to ensuring high accuracy and reliable financial risk assessments. Among the models used—**Random Forest (RF), Logistic Regression (LR), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO)**—the most suitable one is identified based on

various performance metrics and real-world applicability.

1. Selecting the Best Model

1.1 Performance-Based Selection

- **Logistic Regression (LR)**: Works well for binary classification but struggles with complex financial data and non-linear relationships.
- **Random Forest (RF)**: Provides high accuracy by leveraging multiple decision trees and reducing overfitting. It is well-suited for financial prediction due to its ability to capture intricate patterns.
- **Genetic Algorithm (GA)**: Enhances feature selection, removing irrelevant financial indicators and improving model efficiency.

- **Particle Swarm Optimization (PSO)**: Fine-tunes hyperparameters, optimizing the model's decision boundaries and boosting predictive accuracy.

Since **Random Forest (RF) consistently outperforms the other models in handling financial complexity and improving prediction stability**, it is chosen as the primary bankruptcy prediction model.

2. Optimizing the Best Model for Prediction

2.1 Feature Selection with GA

To improve RF's performance, **Genetic Algorithm (GA) is used for feature selection.** GA eliminates irrelevant or redundant financial attributes, ensuring that the model focuses on the most critical indicators of bankruptcy risk.

2.2 Hyperparameter Optimization with PSO

- **Particle Swarm Optimization (PSO)** is applied to fine-tune Random Forest parameters such as the number of trees, maximum depth, and splitting criteria.
- This ensures that RF operates at peak efficiency, reducing overfitting and increasing accuracy.

3. Model Evaluation for Prediction

After optimization, the final RF model is validated using various metrics:

- **Accuracy:** Measures the proportion of correct predictions.
- **Precision and Recall:** Ensures that bankruptcy cases are correctly classified while minimizing false positives.

- **ROC-AUC Score:** Evaluates the model's ability to differentiate between bankrupt and non-bankrupt companies.

6.2 Algorithm Used:

6.2.1 Architecture of Random Forest (RF):

Random Forest (RF) is an ensemble learning method that operates by constructing multiple decision trees and combining their predictions to improve accuracy and reduce overfitting. It works by:

Selecting random subsets of features for each decision tree.

Training multiple decision trees independently on different random samples of the dataset.

Aggregating the predictions of all trees (majority voting for classification, averaging for regression).

Reducing variance and improving generalization performance.

RF is highly effective in handling missing values and noisy data, making it suitable for bankruptcy prediction by analyzing financial attributes.

6.2.2 Working of Logistic Regression (LR)

Logistic Regression (LR) is a statistical model used for binary classification tasks. It models the probability of an event occurring by fitting data to a logistic function. The key steps in LR are:

Identifying relevant financial features influencing bankruptcy.

Applying a weighted sum of input features and passing it through a logistic function to produce probabilities.

Classifying companies as either "bankrupt" or "non-bankrupt" based on a decision threshold.

Training the model using optimization techniques such as Gradient Descent to minimize errors. LR is widely used due to its interpretability and efficiency in predicting financial distress in companies.

6.2.3 Optimization using Genetic Algorithm (GA) and Particle Swarm Optimization (PSO)

To enhance model performance, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) were applied as optimization techniques:

Genetic Algorithm (GA): GA is inspired by the process of natural selection, where potential solutions evolve over generations. It works by:

Encoding potential model parameters as chromosomes.

Using selection, crossover, and mutation operations to evolve better solutions.

Optimizing hyperparameters and feature selection to enhance prediction accuracy.

Particle Swarm Optimization (PSO): PSO is a population-based optimization technique inspired by swarm intelligence, where particles (solutions) adjust their positions based on:

- Their own experience (personal best).The experience of the best-performing particle in the swarm (global best).Gradually converging toward the optimal hyperparameters to improve model performance.Both GA and PSO are effective in fine-tuning machine learning models for bankruptcy prediction, ensuring better accuracy and robustness.

Relevant Hyperparameters to Tune

For optimal performance, various hyperparameters were fine-tuned for each algorithm:

Random Forest (RF):

Number of decision trees (n_estimators)

Maximum tree depth (max_depth)

Minimum samples per split (min_samples_split)

Logistic Regression (LR):

Regularization strength (C)

Solver type (e.g., ‘liblinear’, ‘saga’)

Genetic Algorithm (GA) & Particle Swarm Optimization (PSO):

Population size
Number of generations (GA)

Inertia weight and acceleration coefficients (PSO)

Tuning these hyperparameters ensures that the bankruptcy prediction model achieves high accuracy while avoiding overfitting or underfitting.

7.System Implementation

7.1 Selected Software:

7.1.1 Visual Studio IDE:

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It is available for Windows, macOS, and Linux. VS Code supports various programming languages and includes features like debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and built-in Git support. It can be customized with themes, extensions, and plugins from the Visual Studio Marketplace.

VS Code provides a rich development experience with features such as:

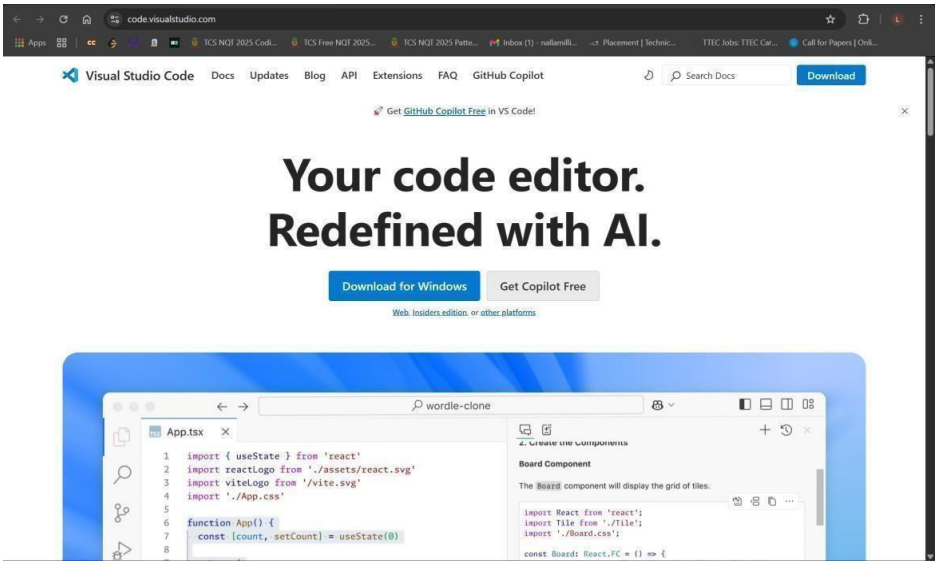
- Built-in terminal for running commands
- IntelliSense for code completion and suggestions
- Debugging tools for various programming languages
- Integration with Git and version control
- A large library of extensions to enhance development

Installation Stepsfor Visual Studio Code

Step 1: Download the Installer:

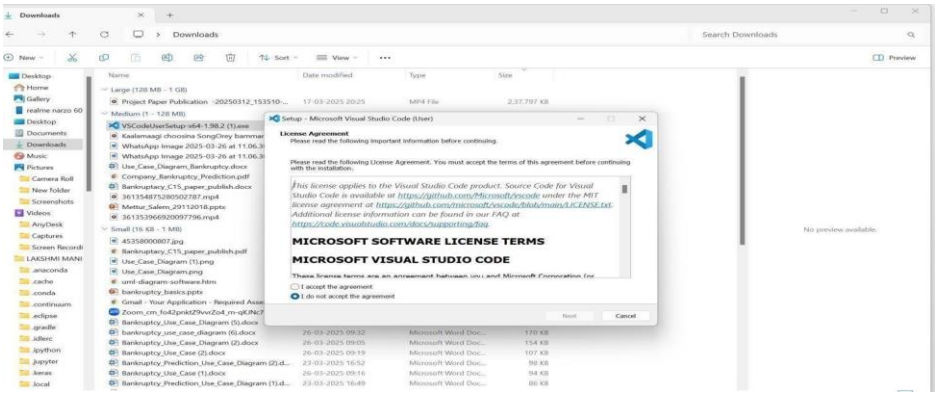
Visit the official website: https://code.visualstudio.com/

Download the appropriate installer for your operating system (Windows, macOS)



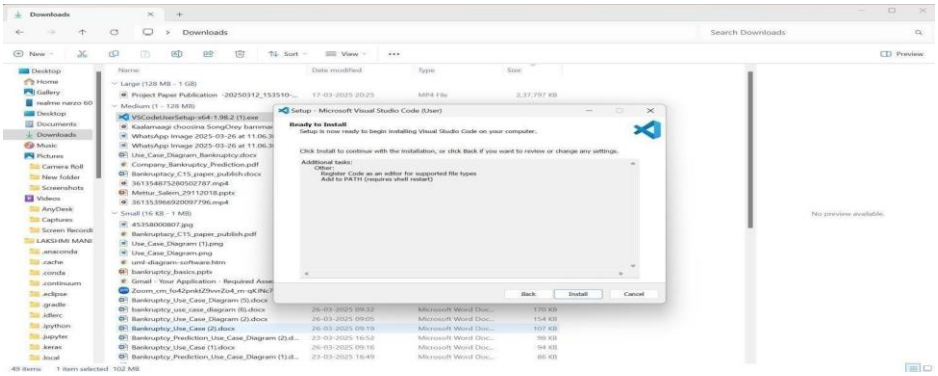
Step 2: Run the Installer

Open the downloaded executable file and click **Next** to start the installation.



Step 3: Choose Installation Location

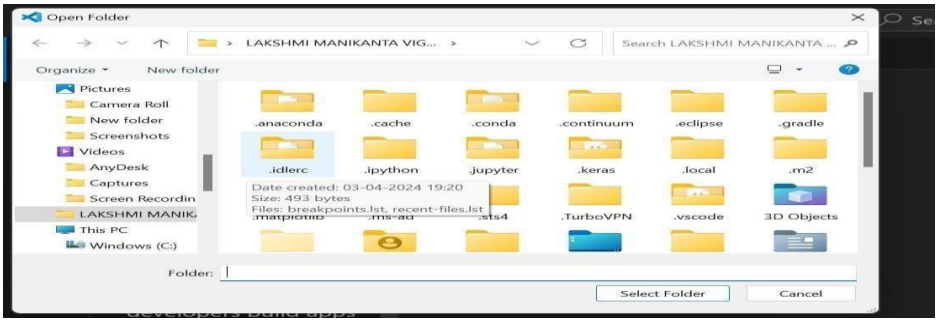
Select the directory where you want to install VS Code and click **Next**.



Step 4: Select Additional Tasks Enable options such as: o "Add 'Open with Code' action to Windows Explorer"

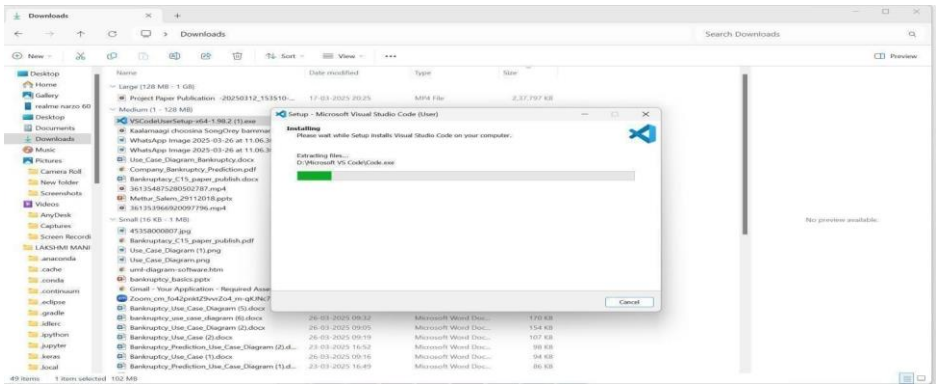
"Add to PATH" (for command-line usage)

Click Next to proceed.



Step 5: Install VS Code

Click **Install** to begin the installation process.



Step 6: Complete the Installation

- Click **Finish** to exit the setup wizard and launch **Visual Studio Code**.

7.1.2 Python

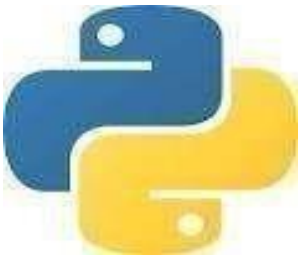


Fig:Logo

Python is a high-level, interpreted, interactive and object-oriented scripting language

Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

•Python is Interpreted-Python is processed at runtime by the interpreter. You need to compile your program before executing it. This is similar to PERL and PHP.

•Python is Interactive - You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

•Python is Object-Oriented - Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

•Python is a Beginner's Language - Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula3, C, C++, Algol-68,Small Talk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL) Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress. Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating. system. Its implementation began in December 1989.

Python FeaturesPython's features include-

•Easy-to-learn-Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

•Easy-to-read-Python code is more clearly defined and visible to the eyes.

•Easy-to-maintain-Python's source code is fairly easy-to-maintain

•A broad standard library - Python's bulk of the library is very portable and cross platform

compatible on UNIX, Windows, and Macintosh.

•Interactive Mode - Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

•Portable - Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

•Extendable- You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

•Databases-Python provides interfaces to all major commercial databases.

•Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

•It provides very high-level dynamic data types and supports dynamic type checking.

•It supports automatic garbage collection.

•It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Install Python Step-by-Step in Windows

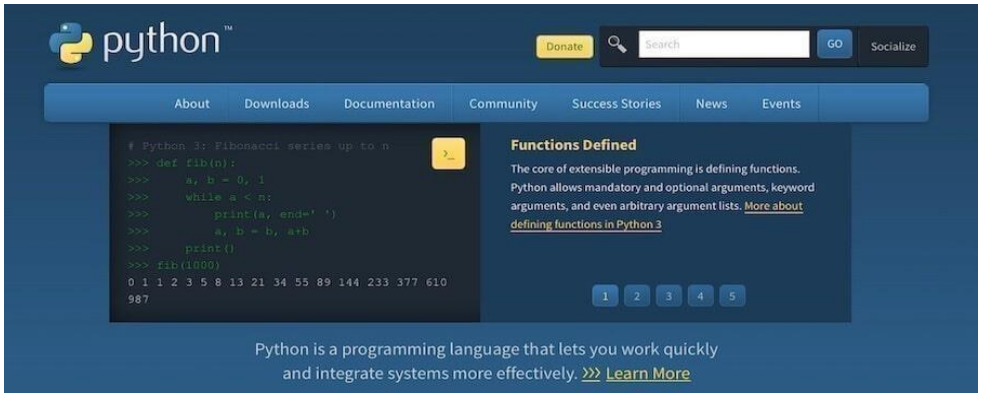
How to Install Python on Windows and Mac:

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

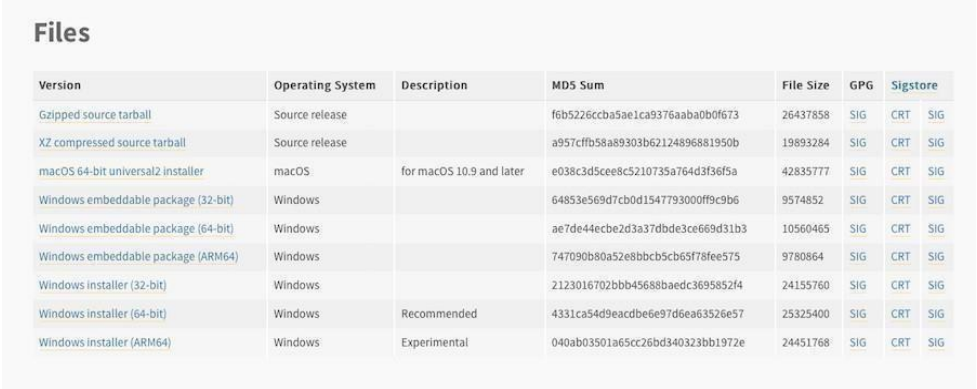
Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices. Before you start with the installation process of Python. First, you need to know about your System requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system:

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.



Step 2: Download the Python installer.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore	
Gzipped source tarball	Source release		f6b5226ccba5ae1ca9376aaba0b0f673	26437858	SIG	CRT	SIG
XZ compressed source tarball	Source release		a957cffb58a89303b62124896881950b	19893284	SIG	CRT	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	e038c3d5cee8c5210735a764d3f36f5a	42835777	SIG	CRT	SIG
Windows embeddable package (32-bit)	Windows		64853e569d7cb0d1547793000ff9c9b6	9574852	SIG	CRT	SIG
Windows embeddable package (64-bit)	Windows		ae7de44ecbe2d3a37dbde3ce669d31b3	10560465	SIG	CRT	SIG
Windows embeddable package (ARM64)	Windows		747090b80a52e8bbcb5cb65f78fee575	9780864	SIG	CRT	SIG
Windows installer (32-bit)	Windows		2123016702bbb45688baedc3695852f4	24155760	SIG	CRT	SIG
Windows installer (64-bit)	Windows	Recommended	4331ca54d9eacdbe6e97d6ea63526e57	25325400	SIG	CRT	SIG
Windows installer (ARM64)	Windows	Experimental	040ab03501a65cc26bd340323bb1972e	24451768	SIG	CRT	SIG

Visit the official Python website and download the latest version of Python 3.x for Windows. The website will 32-bit or 64-bit).Automatically detect your operating system and offer the appropriate installer for your system

Step 3: Run the Installer

Locate the downloaded installer file (usually in your Downloads folder) and double-click on it to run the installation process. You may be prompted by the User Account Control (UAC) to allow the installation. Click Yes to proceed.

Step 4: Customize the Installation (Optional) On the installer's welcome screen, you'll see two options: Install Now and Customize installation. If you want to install Python with the default settings, simply click Install Now.

If you want to customize the installation (changing the installation directory or selecting specific components, for instance), click Customize installation. You should see the following:

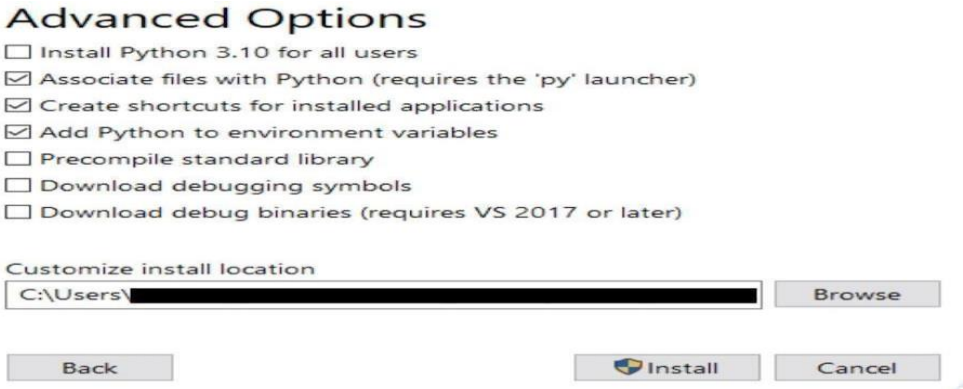
Pick and choose what you'd like to be installed alongside the base installation. Your options include: •

Documentation: This includes the Python documentation file with the installation.

pip: This option installs pip, which allows you to install other Python packages as you'd like. tcl/tk and IDLE: This option installs tkinter and IDLE.

Python test suite: Selecting this option installs the standard library test suite, which is useful for testing your output. py launcher; for all users: These two options make it so you can launch Python from the command line When you're done making your selections, click

Next.You'll be then taken to a new dialog box that offers advanced options:Again, you're



presented with a number of options to choose from, including:

- Install Python 3.11 for all users
- Associate files with Python (requires the 'py' launcher)
- Create shortcuts for installed applications
- Add Python to environment variables
- Precompile standard library
- Download debugging symbols
- Download debug binaries (requires VS 2017 or later)

Verify that the installation directory chosen is correct and then you're reading to install.

Step 5: Install Python

After selecting your desired installation settings, click Install to begin the installation process. The installer will copy the necessary files to your computer and set up Python. This process may take a few minutes.

Step 6: Verify the installation

Once the installation is complete, you can verify that Python has been installed correctly by opening the Command Prompt (search for "cmd" in the Start menu) and typing the following command:

python –version

Press Enter, and you should see the version of Python you installed displayed in the output. This confirms that Python has been successfully installed on your computer.

Advantages of Python

Extensive Libraries

Python provides a wide range of libraries for various tasks like web development, data science, machine learning, and more.

Extensible

Python can be extended to other languages such as C and C++. This is useful for performance-critical applications.

Embeddable

You can embed Python code into code written in other languages like C++. This helps in adding scripting capabilities to other applications.

Improved Productivity

Python's simplicity and powerful libraries increase developer productivity. Less code is needed to perform complex tasks compared to Java or C++.

IoT Opportunities

Python supports platforms like Raspberry Pi, making it ideal for Internet of Things (IoT) projects.

Simple and Easy

Python has a simple syntax. For example, printing "Hello World" just needs a single print statement, unlike Java which requires a class structure.

Readable

Python code is clean and readable, often resembling plain English. It doesn't use curly braces for blocks; instead, it relies on indentation, making code easier to understand.

Object-Oriented

Python supports both procedural and object-oriented programming. It allows you to

encapsulate data and functions inside classes and objects.

Free and Open-Source

Python is completely free to use and distribute. You can even access and modify the source code.

Portable

Python code is platform-independent. Write once, run anywhere (WORA), as long as you avoid system-dependent features.

Disadvantages of Python

Speed Limitations:

Python is an interpreted language and executes code line by line, which makes it slower than compiled languages like C++ or Java. This is not a problem unless your project is speed-sensitive.

Weak in Mobile Computing and Browsers

Python is rarely used for mobile app development or client-side browser scripting. Although options like Kivy and Brython exist, they are not as widely adopted due to performance and security concerns.

7.2 Introduction to Keras Framework

Keras is a powerful and easy-to-use open-source software library for deep learning. It provides a high-level interface for building and training neural networks, and it is designed to enable fast experimentation with deep neural networks. Keras is written in Python and can run on top of popular deep learning frameworks like TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). TensorFlow, in particular, is the most commonly used backend for Keras, and since the release of TensorFlow 2.0, Keras has been integrated directly into TensorFlow as tf.keras.

Key Features of Keras:

- **User-Friendly API:**

  Keras is known for its user-friendly API which is intuitive and easy to learn. This makes it accessible for beginners, while still being flexible enough for research.
- **Modularity:**

  Keras models are assembled by connecting configurable building blocks (modules) together, with few restrictions.
- **Easy Extensibility:**

  It is easy to add new components as new classes and functions, making Keras customizable and open to innovation.
- **Multi-Backend, Multi-Platform:**

  As mentioned earlier, Keras can run on top of different backends and can be executed on different platforms, including CPUs, GPUs, and even TPUs when using TensorFlow as a backend.

7.2 Sample Code:

App2.py

```
import os
import numpy as np import pandas as pd import joblib
# To load the saved models from flask_sqlalchemy
import SQLAlchemy from flask import Flask,
render_template, request, redirect, url_for,
flash,session
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user from werkzeug.utils import
secure_filename
from werkzeug.security import generate_password_hash,
check_password_hash from sklearn.preprocessing import StandardScaler
import io import base64 from flask_session import Session import
matplotlib.pyplot as plt from io import StringIO import matplotlib.pyplot as plt
import seaborn as sns import io
import tkinter
as tk import
base64 from
sklearn.tree
import
plot_tree
from sklearn.ensemble import
RandomForestClassifier import
tempfile import matplotlib
matplotlib.use('Agg') # Initialize
Flask App
app = Flask( name , template_folder='templates')
app.config['SESSION_TYPE'] = 'filesystem'
app.config['SESSION_PERMANENT'] = False
```

```
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['UPLOAD_FOLDER'] = 'uploads'
app.config['ALLOWED_EXTENSIONS'] = {'csv', 'xlsx'}
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///C:/Users/nikhi/OneDrive/Desktop/UI/instance/site.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS']=False
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True) Session(app) # Initialize Database
db = SQLAlchemy(app) # Initialize Flask-Login
login_manager = LoginManager() login_manager.init_app(app)
login_manager.login_view = 'login' def get_user_by_id(user_id): if user_id:
# Instead of User.query.get(), use db.session.get() for SQLAlchemy 2.0 return db.session.get(User, user_id) # Get user from database return None
def generate_plot_url(fig): img = io.BytesIO()
fig.savefig(img, format='png', bbox_inches="tight")
img.seek(0) return base64.b64encode(img.getvalue()).decode() def plot_bankruptcy_pie_chart(df):
bankruptcy_counts = df['Bankruptcy Prediction GA'].value_counts()
fig, ax = plt.subplots(figsize=(6, 4))
bankruptcy_counts.plot(kind='pie', labels=bankruptcy_counts.index, autopct='%1.1f%%', colors=['green', 'red'], ax=ax)
plt.title("Bankruptcy Prediction Distribution") plt.ylabel('') # Hide y-
```

```
label plot_pie_url =
generate_plot_url(fig) plt.close(fig)
return plot_pie_url def plot_pairplot(df):
fig = sns.pairplot(selected_features, diag_kind='kde', corner=True)
fig.fig.suptitle("Pairplot of Financial Features", y=1.02)
plot_pairplot_url =
generate_plot_url(fig.fig)
plt.close(fig.fig) return
plot_pairplot_url def
plot_bankruptcy_prediction(
df):
# Count occurrences of each bankruptcy status
bankruptcy_counts = df['Bankruptcy Prediction GA'].value_counts()
# Create the bar chart
fig, ax = plt.subplots(figsize=(6, 4)) bankruptcy_counts.plot(kind='bar',
color=['green', 'red'], ax=ax) plt.title("Bankruptcy Prediction Results")
plt.xlabel("Prediction") plt.ylabel("Count") plt.xticks(rotation=0) #
Convert plot to URL plot_bankruptcy_url = generate_plot_url(fig)
plt.close(fig)
return plot_bankruptcy_url # 1. Box Plot def plot_boxplot(df):
fig, ax = plt.subplots(figsize=(8, 6)) boxprops = dict(linestyle='-', linewidth=1.5,
color='black') sns.boxplot(data=df.iloc[:, -3:], orient="h", boxprops=boxprops)
plt.title("Box Plot of Bankruptcy Predictions") plt.xlabel("Prediction Categories")
plt.ylabel("Values") plt.grid(True)
plot_box_url =
generate_plot_url
(fig) plt.close(fig)
return
plot_box_url # 2.
Histogram def
plot_histogram(df
```

```
): fig, ax = plt.subplots(figsize=(6, 4))
df.iloc[:, -3].hist(bins=20, ax=ax, color='skyblue', edgecolor='black')
plt.title("Histogram of Last Column") plot_hist_url =
generate_plot_url(fig) plt.close(fig) return plot_hist_url
def generate_plot_url(fig): img
= io.BytesIO() fig.savefig(img,
format='png',
bbox_inches="tight")
img.seek(0) return
base64.b64encode(img.getvalu
e()).decode() try:
X = df.iloc[:, :-2] #
Select all columns
except last two y =
df.iloc[:, -1] #
Select last column
as target
# Train model
rf_model = RandomForestClassifier(n_estimators=10,
random_state=42) rf_model.fit(X, y) # Create feature
importance plot fig, ax = plt.subplots(figsize=(8, 6))
feature_importances =
pd.Series(rf_m
odel.feature_importances_,
index=X.columns).sort_values(ascen
ding=False)
feature_importances.plot(kind='bar', ax=ax, color='teal') plt.title("Feature
Importance (Multi-Class)")
plot_feature_importance_url =
generate_plot_url(fig) plt.close(fig)
return plot_feature_importance_url
```

except Exception as e: print(f"Error in plot_feature_importance: {e}") # Debugging return None

Decision Tree Visualization for Multi-Class from sklearn.tree import DecisionTreeClassifier def plot_decision_tree(df): try:

X = df.iloc[:, :-2] # Select all columns except last two y = df.iloc[:, -1] # Select last column as target

Train a single decision tree

dt_model = DecisionTreeClassifier(max_depth=3, random_state=42) dt_model.fit(X, y)

Create decision tree plot fig, ax = plt.subplots(figsize=(12, 8))

class_labels = list(map(str, y.unique())) # Ensure correct class names

plot_tree(dt_model, feature_names=X.columns, class_names=class_labels, filled=True, ax=ax) plt.title("Decision Tree (Multi-Class)")

plot_tree_url = generate_plot_url(fig)

plt.close(fig) return plot_tree_url

except Exception as e:

print(f"Error in plot_decision_tree: {e}") # Debugging return None

class User(UserMixin, db.Model):

tablename = 'user' id = db.Column(db.Integer, primary_key=True) full_name = db.Column(db.String(100), nullable=False) email = db.Column(db.String(120), unique=True, nullable=False) password = db.Column(db.String(256), nullable=False)

@login_manager.user_loader def load_user(user_id):

return User.query.get(int(user_id))

File Upload

Helper def

allowed_file(

filename):

return '.' in filename and filename.rsplit('.', 1)[1].lower() in app.config['ALLOWED_EXTENSIONS']

def generate_plot_url(fig): img = io.BytesIO()

fig.savefig(img, format='png', bbox_inches="tight") img.see k(0) return

base64.b64encode(img.getvalue()).decode() def

determine_bankruptcy_status(df, file_name): ga_high_risk_count = (df['Bankruptcy Prediction GA'] == "High Risk").sum() pso_high_risk_count = (df['Bankruptcy Prediction PSO'] == "High Risk").sum()

total_records = len(df)

Calculate risk percentages

ga_risk_percentage = (ga_high_risk_count / total_records) * 100 pso_risk_percentage = (pso_high_risk_count

/ total_records) * 100

Average risk percentage from both models avg_risk_percentage = (ga_risk_percentage + pso_risk_percentage)

/ 2 if avg_risk_percentage > 70: return f"{file_name} is **BANKRUPT** (Both models strongly predict bankruptcy)" elif avg_risk_percentage > 50:

return f"{file_name} is at HIGH RISK (Both models indicate significant bankruptcy risk)" elif avg_risk_percentage > 30:

return f"{file_name} is at MODERATE RISK (At least one model suggests a noticeable risk)" elif avg_risk_percentage > 10: return f"{file_name} is at LOW RISK (Minimal bankruptcy indications)" else: return f"{file_name} is NOT BANKRUPT (Both models indicate very low risk)"

Load Models and Scalers def load_models():

Load the GA-optimized model and scaler

model_ga = joblib.load("model/model_ga.sav", mmap_mode=None) model_pso = joblib.load("model/model_pso.sav")

scaler_ga = joblib.load("model/scaler_ga.sav") # Assuming scaler for 64 features scaler_pso = joblib.load("model/scaler_pso.sav") # Assuming scaler for 24 features num_features1 = model_ga.coef_.shape[1] # Number of features in the model num_features2 = model_pso.coef_.shape[1] # Number of features in the model print(f"Number of features used by model_ga: {num_features1}") print(f"Number of features used by model_pso: {num_features2}") return model_ga, model_pso, scaler_ga, scaler_pso

def calculate_bankruptcy(row, model, scaler): # Use 24 features for this model (model_pso)

X_values = np.array([row[f'X{i}'] for i in range(1, 25)]).reshape(1, -1) # First 24 features X_scaled = scaler.transform(X_values) # Apply scaling Z = model.predict_proba(X_scaled)[0][1]

print(f"X_values: {X_values}") # Debugging line print(f"Predicted Probability: {Z}") # Debugging line

if Z > 0.8:

return "High Risk" elif Z > 0.5:

return "Medium Risk" else:

return "Low Risk"

def calculate_bankruptcy_standardscaler(row, model, scaler): # Use 64 features for this model (model_ga) X_values = np.array([row[f'X{i}'] for i in range(1, 65)]).reshape(1, -1) # First 64 features X_scaled = scaler.transform(X_values) # Apply scaling

Z = model.predict_proba(X_scaled)[0][1]

print(f"X_values: {X_values}") # Debugging line print(f"Predicted Probability: {Z}") # Debugging line if Z > 0.8: return "High Risk"

return render_template('index.html')

@app.route('/login',

methods=['GET', 'POST'])

def login(): if request.method

== 'POST':

password = request.form.get('password') print("Request form:", request.form) print("Request args:", request.args)

user = User.query.filter_by(email=email).first()

if user and check_password_hash(user.password, password): login_user(user) return redirect(url_for('predict')) flash('Invalid email or password.', 'danger')

return render_template('login.html')

@app.route('/register', methods=['GET', 'POST']) def register(): if request.method == 'POST': full_name = request.form.get('name') email = request.form.get('email') password =

request.form.get('password') confirm_password = request.form.get('confirm-password')

if not all([full_name, email, password, confirm_password]): flash('All fields are required!', 'danger') return redirect(url_for('register'))

if password != confirm_password: flash('Passwords do not match.', 'danger') return redirect(url_for('register'))

if User.query.filter_by(email=email).first(): flash('Email already exists.', 'warning') return redirect(url_for('register'))

hashed_password = generate_password_hash(password, method='pbkdf2:sha256') new_user = User(full_name=full_name, email=email, password=hashed_password) try: db.session.add(new_user) db.session.commit() print("User successfully added!") except Exception as e: db.session.rollback() print("Error adding user:", e)

flash('Account created successfully!', 'success') return redirect(url_for('login')) return render_template('register.html')
@app.route('/preview', methods=['GET', 'POST'])
@login_required def preview():

flash("Please upload a file first to view the graphs.", "warning") return redirect(url_for('predict')) file_name = session.get('filename', 'Uploaded File')

```
# Assuming session['df'] is a JSON string, we use
pd.read_json() try: df = pd.read_json(session['df']) except
ValueError as e: flash(f"Error loading data:
{str(e)}", "danger") return redirect(url_for('predict'))
final_bankruptcy_status = determine_bankruptcy_status(df, file_name)
plot_box_url = plot_boxplot(df)
plot_hist_url = plot_histogram(df) plot_feature_importance_url =
plot_feature_importance(df) plot_tree_url = plot_decision_tree(df)
                                plot_bankruptcy_url =
                                plot_bankruptcy_prediction(df)
                                plot_pie_url
= plot_bankruptcy_pie_chart(df) plot_pairplot_url = plot_pairplot(df)

return render_template('preview.html',
tables=df.to_html(classes='table table-bordered').strip(),
                                final_bankruptcy_status=final_bankruptcy
_status, plot_box_url=plot_box_url,
                                plot_hist_url=plot_hist_url,
plot_feature_importance_url=plot_feature_importance_url,
plot_tree_url=plot_tree_url, plot_pie_url=plot_pie_url,
plot_bankruptcy_url=plot_bankruptcy_url,
plot_pairplot_url=plot_pairplot_url) @app.route('/logout')
@login_required def logout(): logout_user() flash('You have been logged
out.', 'info') return redirect(url_for('login'))
@app.route('/forgot-password', methods=['GET',
'POST']) def forgot_password(): if
request.method == 'POST': email =
request.form.get('email') new_password =
request.form.get('password') user =
User.query.filter_by(email=email).first() if not
user: flash('Email not found.', 'danger')
```

return redirect(url_for('forgot_password'))

hashed_password = generate_password_hash(new_password, method='pbkdf2:sha256') user.password = hashed_password db.session.commit()

flash('Password reset successfully! You can now log in.', 'success') return redirect(url_for('login')) return render_template('forgot_password.html') @app.route('/predict', methods=['GET', 'POST']) @login_required def predict(): model_ga, model_pso, scaler_ga, scaler_pso = load_models() # Load models and scalers

if request.method == 'POST': session.pop('df', None) # Clear previous session data session.pop('file_name', None)

file = request.files.get('file')

Check if a file is uploaded and it's of the allowed type if not file or not allowed_file(file.filename): flash("Invalid file type! Upload CSV or Excel.", "danger") return redirect(url_for('predict'))

Secure the filename and save the file to the specified folder filename = secure_filename(file.filename) filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename) file.save(filepath) file_name_without_extension = os.path.splitext(filename)[0] # Read the file into a DataFrame (CSV or Excel) try: if filename.endswith('.csv'): df = pd.read_csv(filepath) else: df = pd.read_excel(filepath) except Exception as e: flash(f"Error reading the file: {str(e)}", "danger") return redirect(url_for('predict'))

Replace '?' with NaN and handle non-numeric values df.replace('?', np.nan, inplace=True) df
= df.apply(pd.to_numeric, errors='coerce')

# Fill NaN values with column means
df = df.fillna(df.mean()).reset_index(drop=True)

Apply prediction using both models try: df['Bankruptcy Prediction PSO'] =df.apply(lambda row: calculate_bankruptcy_standardscaler(row, model_pso, scaler_pso), axis=1)
df['Bankruptcy Prediction GA'] = df.apply(lambda row: calculate_bankruptcy(row, model_ga, scaler_ga), axis=1)

flash(f"Error during prediction: {str(e)}", "danger") return redirect(url_for('predict'))

--- Ensure Data is in Float Format and Handle Missing Data --- numeric_predictions = df[['Bankruptcy Prediction PSO', 'Bankruptcy Prediction
GA']].apply(pd.to_numeric, errors='coerce')

Fill NaN values with 0 or another appropriate placeholder numeric_predictions.fillna(0, inplace=True)

Check if all values in prediction columns are NaN if numeric_predictions.isnull().all().any(): flash('No valid data for plotting!', 'warning') return redirect(url_for('predict'))

final_bankruptcy_status = determine_bankruptcy_status(df, file_name_without_extension) session['df'] = df.to_json()
session['file_name'] = file_name_without_extension

```
return render_template('predict.html', final_bankruptcy_status=final_bankruptcy_status, file_uploaded=True) # If GET request, render the upload form
return render_template('predict.html', file_uploaded=False)

if __name__ == '__main__': with app.app_context():
db.create_all() app.run(debug=True) root = tk.Tk() root.mainloop()
```

8.System Testing

8.1 Software Testing:

Software testing plays a crucial role in ensuring the accuracy, reliability, and efficiency of the bankruptcy prediction system. Various testing methodologies were used to validate the functionality, performance, and robustness of the system.

8.2 Goals Of Testing:

Software testing ensures that the bankruptcy prediction system functions correctly, efficiently, and securely. The primary goals of testing include:

1. Ensuring Accuracy and Reliability

• Verify that the system produces correct bankruptcy predictions based on historical financial data.

• Ensure consistent performance across different datasets and scenarios.

2. Detecting and Fixing Bugs

• Identify and eliminate errors in data preprocessing, model training, and prediction processes.

• Ensure that edge cases (e.g., missing or incorrect data) are handled correctly.

3. Improving Performance

• Optimize model execution speed and efficiency to provide quick bankruptcy predictions.

• Ensure that the system can handle large datasets without significant delays.

4. Validating Model Predictions

• Evaluate the effectiveness of the selected machine learning models (RF, LR, GA, and PSO).

• Compare predictions with actual bankruptcy cases using performance metrics.

5. Ensuring Security and Data Integrity

• Protect sensitive financial data from unauthorized access.

• Ensure that data stored in the SQLite database is accurate and consistent.

6. Enhancing User Experience

• Verify that the system provides clear, interpretable predictions and insights.

• Ensure a user-friendly interface for stakeholders, such as investors and financial analysts.

8.3 Testing Methodology:

8.3.1 BLACK BOX TESTING:

Black Box Testing, also known as Behavioural Testing, is a software testing method in which the internal structure, design, or implementation of the system being tested is unknown to the tester. This method focuses on the functionality of the system and ensures that the application meets its specified requirements.

Black Box Testing is classified into four types:

a) Boundary Value Analysis

b) Equivalence Partitioning

c) Cause-Effect Graph-Based Testing

d) Error Guessing

a) Boundary Value Analysis:

This technique is used when the module being tested depends on multiple independent variables. It is particularly important in cases where boundary conditions are crucial, such as numerical inputs in financial systems.

b) Equivalence Partitioning:

Equivalence Partitioning divides input data into valid and invalid partitions to minimize the number of test cases while ensuring maximum coverage. This technique helps identify potential errors without excessive testing.

c) Cause-Effect Graph-Based Testing:

Cause-Effect Graphing is a systematic method that helps in testing combinations of input conditions efficiently.

d) Error Guessing:

Error Guessing is an experience-based testing technique where testers predict potential errors based on intuition and past experiences. Common scenarios include:

- Divide by zero errors
- Handling of empty input lists
- Negative values in input fields

Fig: Black Box Testing

Black Box Testing in Bankruptcy Prediction Project:

In our Bankruptcy Prediction Project, Black Box Testing is applied to ensure the accuracy and reliability of predictions without examining the underlying code. The test cases are designed based on expected outputs for given inputs.

Black Box Testing steps for this project:

1. Understanding Requirements – Analyze the prediction model's expected behavior.
2. Selecting Test Inputs – Provide valid financial data as well as invalid or extreme values.
3. Determining Expected Outputs – Define expected bankruptcy predictions for test cases.
4. Executing Test Cases – Run the bankruptcy prediction model using the selected inputs.
5. Comparing Results – Validate the actual outputs against expected results.
6. Reporting Defects – Any discrepancies are reported and fixed, followed by re-testing.

Types of Black Box Testing Used in the Project:

- Functional Testing – Validates if the bankruptcy prediction system correctly processes financial data.
- Non-Functional Testing – Checks performance, scalability, and usability aspects of the system.
- Regression Testing – Ensures updates or bug fixes do not impact previously working functionalities.

Test Objectives:

- All input fields must accept valid data and reject invalid entries.
- The bankruptcy prediction model should process inputs without delays.
- The system must provide accurate classification (bankrupt or non-bankrupt).
- Error messages should be user-friendly and meaningful.

8.3.2 White Box Testing:

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases.

Types of White Box Testing:

- Logic Coverage Criteria
- Basis Path Testing
- Data Flow Testing
- Mutation Testing

Basis Path Testing:

Basis path testing is the oldest structural testing technique. This testing is based on the control structure of the program. A flow graph is prepared based on the control structure, and all possible paths are covered and tested. Path coverage is useful for detecting more errors, but the challenge with this technique is that programs containing loops may have an infinite number of possible paths, making it impractical to test all of them.

Data Flow Testing:

Data flow testing is a family of test strategies based on selecting paths through the

program's control flow in order to explore sequences of events related to the status of variables or data objects.Data flow testing focuses on the points at which variables receive values and the points at which these values are used.

Mutation Testing:

Mutation Testing is a type of software testing where certain statements in the source code are mutated to check if test cases can detect the errors.The changes in the mutant program are kept extremely small to ensure that they do not affect the overall objective of the program.

8.3.3 Gray Box Testing:

Gray-box testing is a software testing technique that combines elements of both white-box and black-box testing.Testers have partial knowledge of the internal structure of the system while focusing on testing functionalities and behaviors.This approach helps identify defects related to both system logic and user interactions.

Gray-box testing includes the following techniques:

1. **Regression Testing:**

   Regression testing ensures that new updates or modifications to the bankruptcy prediction system do not introduce new defects.It verifies that the core functionalities, such as financial data processing, model training, and predictions, remain stable after changes.

2. **Pattern Testing:**

   Pattern testing involves analyzing historical data patterns to identify inconsistencies or anomalies in predictions.This is useful in the bankruptcy prediction system to detect trends in financial distress and validate the effectiveness of prediction models.

3. **Orthogonal Array Testing:**

   Orthogonal array testing is a systematic testing method that reduces the number of test cases while maximizing coverage.This technique is used to test different combinations of financial attributes and ensure that the machine learning models perform consistently across various financial scenarios.

4. **Matrix Testing:**

   Matrix testing helps assess the relationship between system components, such as financial data inputs, model selection, and prediction outputs.
   It ensures that all modules work together as expected and that dependencies between different features are handled correctly.

By utilizing gray-box testing techniques, the bankruptcy prediction system can achieve a balance between internal logic validation and functional verification, improving accuracy, reliability, and overall system quality.

8.4 Levels of Testing and PyTest

8.4.1 Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs.All decision branches and internal code flow should be validated.It is the testing of individual software units of the application.It is done after the completion of an individual unit before integration.

8.4.2 System Testing:

System testing ensures that the entire integrated software system meets requirements.
It tests a configuration to ensure known and predictable results.An example of system testing is the configuration-oriented system integration test.System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.4.3 Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program.Testing is event driven and is more concerned with the basic outcome of screens or fields.Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent.Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.4.4 Acceptance Testing:

Acceptance testing is conducted to determine whether the bankruptcy prediction system meets the business and user requirements.It ensures that the software functions as expected in real-world scenarios and satisfies the needs of stakeholders, including financial analysts, investors, and decision-makers.

8.4.5 Regression Testing:

Regression testing is performed to ensure that any modifications, updates, or enhancements to the bankruptcy prediction system do not negatively impact its existing functionality.This testing verifies that new changes, such as algorithm improvements, feature updates, or bug fixes, do not introduce new defects or disrupt the system’s overall performance.

8.5 PyTest:

8.5.1 Pytest Report:

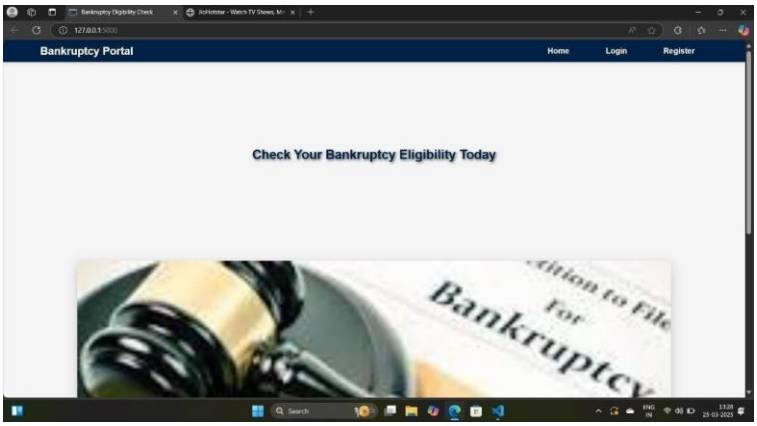
Pytest is a Python testing framework that originated from the PyPy project.It can be used to write various types of software tests, including unit tests, integration tests, end-to-end tests, and functional tests.Its features include parametrized testing, fixtures, and assert rewriting.Pytest fixtures provide the contexts for tests by passing in parameter names in test cases.Its parametrization eliminates duplicate code for testing multiple sets of input and output.Its rewritten assert statements provide detailed output for causes of failures.
Pytest was developed as part of an effort by third-party packages to address Python's built-in module unittest's shortcomings.It originated as part of PyPy, an alternative implementation of Python to the standard CPython.Since its creation in early 2003, PyPy has had a heavy emphasis on testing.

8.2 Test Case Scenarios:

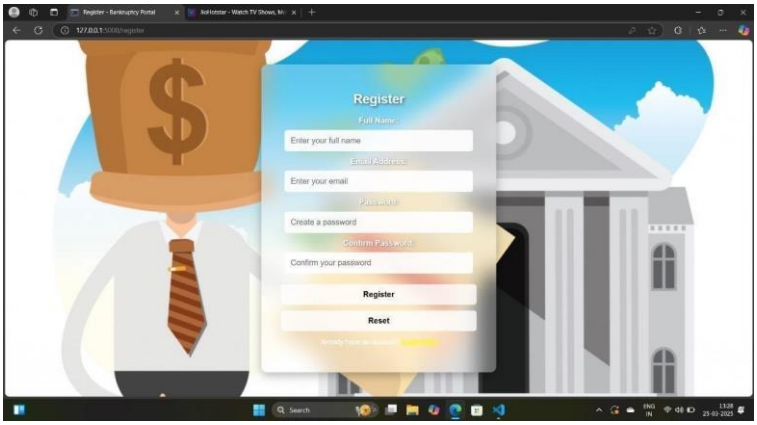
Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC_01	User uploads a valid financial dataset	User uploads a properly formatted dataset	System accepts the dataset and proceeds to validation	Valid dataset should be processed successfully	Pass/Fail
TC_02	User uploads an invalid dataset	User uploads a corrupted or incorrectly formatted file	System rejects the dataset and shows an error message	System should notify the user of invalid data	Pass/Fail
TC_03	Admin uploads training dataset	Admin provides a dataset for model training	System accepts and preprocesses the dataset	Training dataset should be processed correctly	Pass/Fail
TC_04	Feature selection using Genetic Algorithm	System applies GA to select important features	GA optimizes feature selection	Only relevant financial indicators are selected	Pass/Fail
TC_05	Model training with Decision Tree	System trains Decision Tree on dataset	Model is trained successfully	Model should learn from training data	Pass/Fail
TC_06	Model optimization using PSO	System applies Particle Swarm Optimization	PSO optimizes model parameters	Prediction accuracy improves	Pass/Fail
TC_07	User initiates bankruptcy prediction	User selects company data for prediction	System processes data and provides result	Prediction result should be displayed	Pass/Fail
TC_08	System handles class imbalance	System applies costsensitive learning techniques	System adjusts classification thresholds	System correctly classifies bankrupt and non-bankrupt cases	Pass/Fail
TC_09	Security - Unauthorized access attempt	Unauthorized user tries to access admin panel	System blocks access and logs event	Only authorized users should access admin features	Pass/Fail

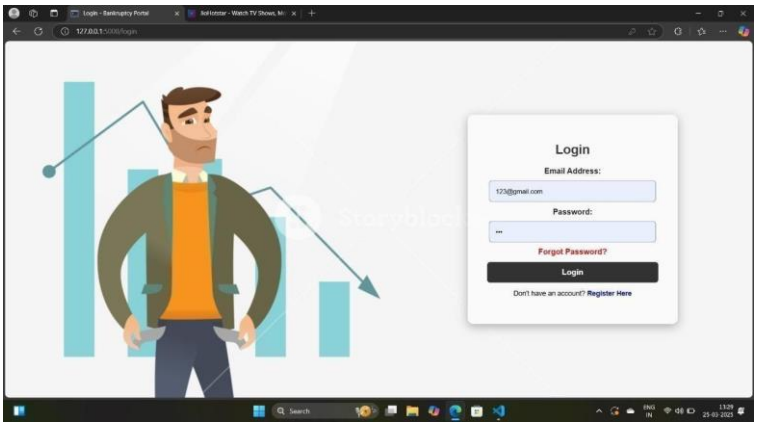
9.SCREENSHOTS

HOME PAGE : This is the home page which consists of dual logins called admin login and the user login.



Registration Page: This is the Registration Page where user can Register.

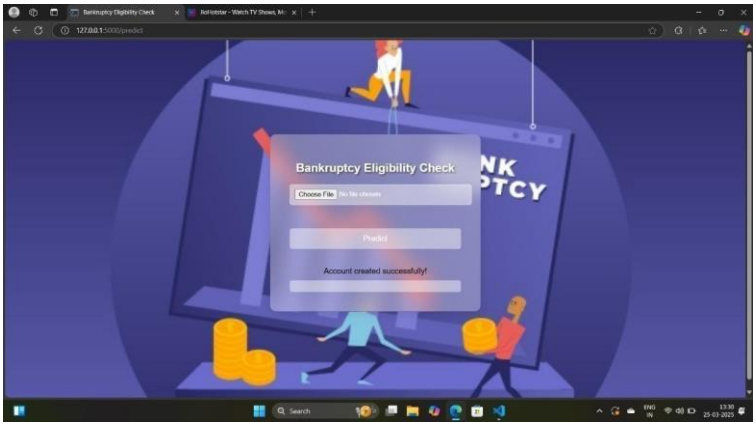




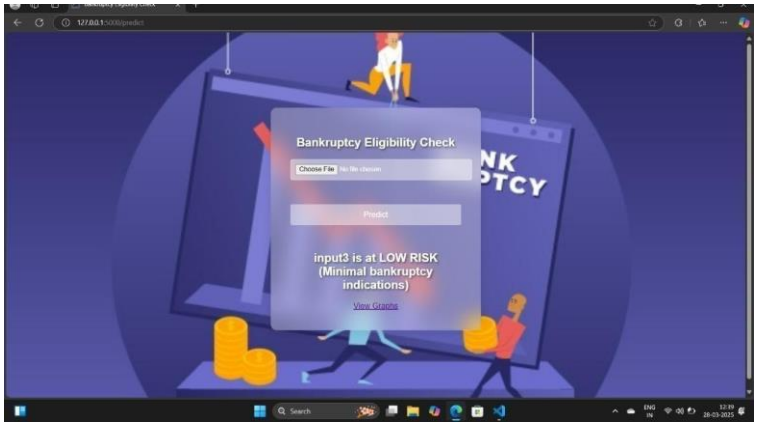
USER LOGIN PAGE:

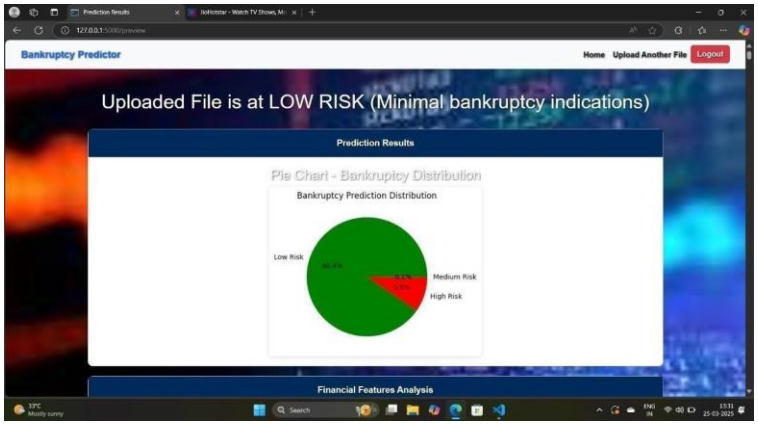
Dataset Upload Page:

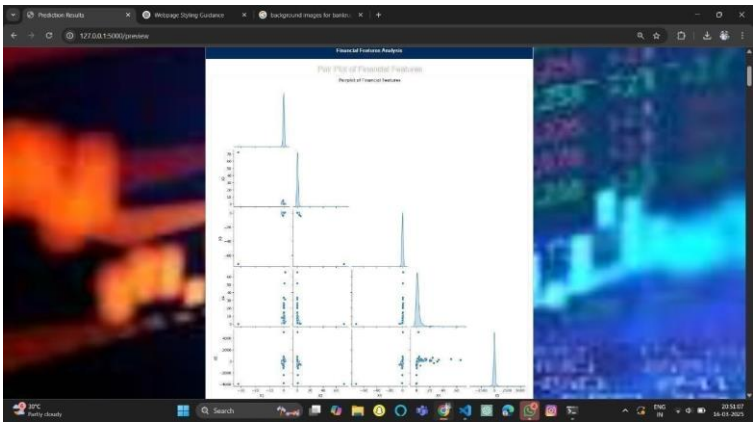
OUTPUTS:

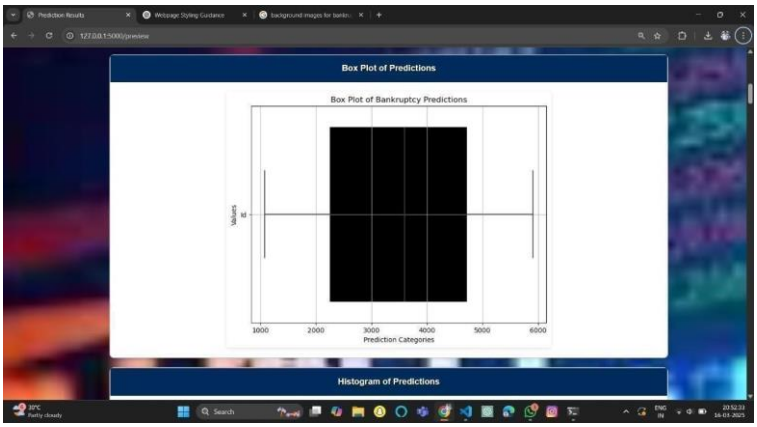


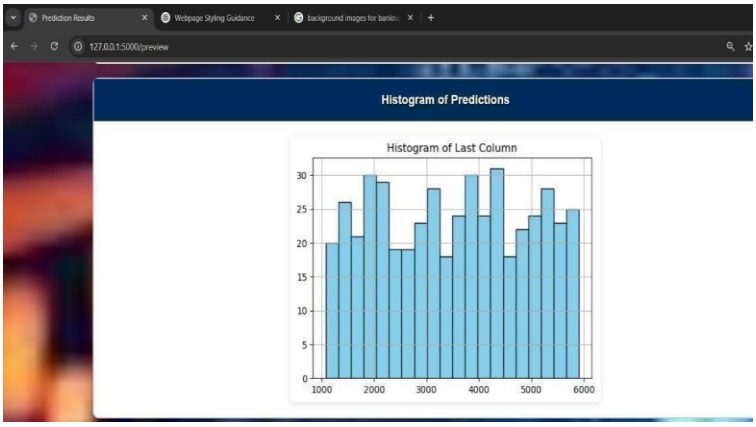
Dataset I uploaded is Lowrisk








Box Plot of Predictions
Box Plot of Bankruptcy Predictions
Values
1000
2000
3000
4000
5000
6000
Prediction Categories
Histogram of Predictions


Histogram of Predictions
Histogram of Last Column
30
25
20
15
10
5
0
1000
2000
3000
4000
5000
6000

10.CONCLUSION & FUTURE ENHANCEMENTS

This project successfully develops a bankruptcy prediction system using machine learning techniques, specifically Random Forest (RF) and Logistic Regression (LR), to evaluate the financial stability of companies.By leveraging historical financial data, key financial ratios, and other relevant indicators, the system provides a data-driven approach to predicting the likelihood of bankruptcy.The project involves extensive data preprocessing, including handling missing values, feature selection, and normalization to ensure the models perform optimally.The Random Forest algorithm is particularly beneficial due to its ability to handle large datasets and reduce overfitting by aggregating multiple decision trees, leading to higher predictiveaccuracy.Logistic Regression, on the other hand, provides a more interpretable model by estimating the probability of bankruptcy based on selected financial features.

Future Enhancements for Bankruptcy Prediction System:

Integration of Deep Learning Models:

• Implement Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks for improved time-series analysis of financial data.

• Utilize Convolutional Neural Networks (CNNs) for automated feature extraction from financial documents to enhance accuracy.

Enhanced Data Security and Privacy:

• Implement blockchain technology to secure financial transactions and ensure data integrity.

• Use differential privacy techniques to protect sensitive financial data from unauthorized access while maintaining analytical accuracy.

Real-Time Financial Data Analysis:

• Enable real-time data collection from stock markets, financial statements, and economic indicators to improve prediction accuracy.

• Integrate APIs to fetch financial reports and automate bankruptcy risk assessments.

11.BIBLIOGRAPHY

Research Papers and Articles:

1. S. Shetty, M. Musa, and X. Brédart, "Bankruptcy prediction using machine learning techniques," *Journal of Risk and Financial Management*, vol. 15, no. 1, p. 35, Jan. 2022, doi:10.3390/jrfm15010035.

2. Y. Shi and X. Li, "An overview of bankruptcy prediction models for corporate firms: A systematic literature review," *Intangible Capital*, vol. 15, no. 2, p. 114, Oct. 2019, doi:10.3926/ic.1354.

3. T. M. Alam, K. Shaukat, M. Mushtaq, Y. Ali, M. Khushi, S. Luo, and A. Wahab, "Corporate bankruptcy prediction: An approach towards better corporate world," *The Computer Journal*, vol. 64, no. 11, pp. 1731–1746, Nov. 2019, doi:10.1093/comjnl/bxaa056.

4. G. Perboli and E. Arabnezhad, "A machine learning-based DSS for mid and long-term company crisis prediction," *Expert Systems with Applications*, vol. 174, Jul. 2021, Art. no. 114758, doi:10.1016/j.eswa.2021.114758.

5. D. Boughaci and A. A. K. Alkhawaldeh, "Appropriate machine learning techniques for credit scoring and bankruptcy prediction in banking and finance: A comparative study," *Risk and Decision Analysis*, vol. 8, nos. 1–2, pp. 15–24, May 2020, doi:10.3233/RDA-180051.

Reference Textbooks:

Software Engineering & Software Testing:

6. Ian Sommerville, *Software Engineering*, Pearson, 10th ed., 2015. Publisher's Page

7. Roger S. Pressman and Bruce R. Maxim, *Software Engineering: A Practitioner's*

8. *Approach*, McGraw Hill, 9th ed., 2019. Publisher's Page

9. William E. Perry, *Effective Methods for Software Testing*, Wiley, 3rd ed., 2006. Publisher's Page

10. Glenford J. Myers, Corey Sandler, and Tom Badgett, *The Art of Software Testing*, Wiley, 3rd ed., 2011. Publisher's Page

11. William E. Perry, *Effective Methods for Software Testing*, Wiley, 3rd ed., 2006. 📖 Amazon Link

Python and Machine Learning:

10. Mark Lutz, *Learning Python*, O'Reilly, 5th ed., 2013. Available Online

11. Jake VanderPlas, *Python Data Science Handbook*, O'Reilly, 2016. Available Online

12. Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly, 2nd ed., 2019. Available Online

Software and Tools Used:

Development Environment:

13. Visual Studio Code (VS Code) - https://code.visualstudio.com/

Python Libraries for Machine Learning:

14. Scikit-Learn - https://scikit-learn.org/
15. Pandas - https://pandas.pydata.org/
16. NumPy - https://numpy.org/
17. Matplotlib - https://matplotlib.org/

Database Management and Data Sources:

18. SQLite (for storing financial data) - https://www.sqlite.org/