Insolvency Predictor Using Time Series

Mini Project submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the esteemed guidance of

S.Radha
Sr. Assistant Professor

By

A. SHRIYA	**(21R11A05A7)**
A. SAI SRUJANA	**(21R11A05A8)**
SHYAMALA CHANDU	**(21R11A05E6)**

Department of Computer Science and Engineering
Accredited by NBA

Geethanjali College of Engineering and Technology
(UGC Autonomous)
(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

August-2024

Geethanjali College of Engineering & Technology

(UGC Autonomous)
(Affiliated to JNTUH, Approved by AICTE, New Delhi)
Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Accredited by NBA

CERTIFICATE

This is to certify that the B.Tech Mini Project report entitled **“Insolvency Predictor Using Time Series”** is a bonafide work done by **A. Shriya(21R11A05A7),A. Sai Srujana (21R11A05A8), Shyamala Chandu (21R11A05E6)**, in partial fulfillment of the requirement of the award for the degree of Bachelor of Technology in “**Computer Science and Engineering**” from Jawaharlal Nehru Technological University, Hyderabad during the year 2023-2024.

Internal Guide	**HOD – CSE**
S. Radha	**Dr A. SreeLakshmi**
Sr. Assistant Professor	Professor

External Examiner

Geethanjali College of Engineering & Technology

(UGC Autonomous)

(Affiliated to JNTUH Approved by AICTE, New Delhi)
Cheeryal (V), Keesara(M), Medchal Dist.-501 301.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Accredited by NBA

DECLARATION BY THE CANDIDATE

We, **A. Shriya, A. Sai Srujana, Shyamala Chandu**, bearing Roll Nos. **21R11A05A7, 21R11A05A8, 21R11A05E6**, hereby declare that the project report entitled **"Insolvency Predictor Using Time Series"** is done under the guidance of **Mrs.S.Radha**,**Sr**.**Assistant Professor**, Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**.

This is a record of bonafide work carried out by me/us in **Geethanjali College of Engineering and Technology** and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

A.Shriya(21R11A05A7)
A.Sai Srujana(21R11A05A8)
Shyamala Chandu(21R11A05E6)

Department of CSE,
Geethanjali College of Engineering and Technology,
Cheeryal.

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. A. Sree Lakshmi**, **Professor, Headof Department** of Computer Science, Geethanjali College of Engineering and Technology, Cheeryal, whose motivation in the field of software development has made us to overcome all hardships during the course of study and successful completion of project.

We would like to express our profound sense of gratitude to all for having helped us in completing this dissertation. We would like to express our deep-felt gratitude and sincere thanks to our guide **S.Radha, Sr.Assistant Professor**, Department of Computer Science, Geethanjali College of Engineering and Technology, Cheeryal, for his skillful guidance, timely suggestions and encouragement in completing this project sucessfully.

We would like to express our sincere gratitude to our **Principal Prof. Dr. S. Udaya Kumar** for providing the necessary infrastructure to complete our project. We are also thankful to our Secretary **Mr.G.R. Ravinder Reddy** for providing an interdisciplinary & progressive environment.

Finally, we would like to express our heartfelt thanks to our parents who were very supportive both financially and mentally and for their encouragement to achieve our setgoals.

A.Shriya (21R11A05A7)

A.Sai Srujana(21R11A05A8)

Shyamala Chandu(21R11A05E6)

ABSTRACT

Bankruptcy prediction is an important problem in finance, since successful predictions would allow stakeholders to take early actions to limit their economic losses. This is often the first step used by ratings agencies to detect financial distress in firms. Based on the predictions of bankruptcy models, ratings agencies investigate and assess credit risk.
Predicting bankruptcy involves forecasting the likelihood of a firm experiencing financial distress or facing insolvency. This predictive capability is invaluable to creditors and investors, enabling them to assess the risk associated with lending or investing in a particular entity. Additionally, early warnings of potential bankruptcies are crucial for public policymakers, allowing them to implement proactive measures to minimize the broader economic impact of such occurrences.In essence, refining these models not only safeguards the interests of investors and creditors but also facilitates informed decision-making by policymakers, contributing to the stability and resilience of modern economies.

LIST OF FIGURES

TABLE OF CONTENTS

1. INTRODUCTION

1.1. ABOUT THE PROJECT

The project titled "Insolvency Predictor using Time Series" aims to develop an advanced predictive model that forecasts bankruptcy risks for companies, playing a pivotal role in the financial sector. Early detection of potential financial distress is essential for stakeholders such as creditors, investors, and policymakers. By predicting the likelihood of insolvency, the model aids in mitigating financial losses, providing insights into credit risks, and contributing to overall economic stability. With the use of cutting-edge data analytics and machine learning techniques, the project ensures that stakeholders can act proactively, taking necessary measures before companies face irreparable financial challenges. This initiative is particularly vital in today's volatile economic environment, where unforeseen market fluctuations can severely impact financial health.

At the core of the project is the integration of tools like Power BI for powerful data visualization, enhancing the decision-making process. The project leverages Exploratory Data Analysis (EDA) to understand key trends in historical financial data, while undersampling and oversampling techniques are used to address class imbalances commonly found in insolvency datasets. Logistic Regression is employed to build a robust predictive model that can accurately distinguish between solvent and insolvent companies. These methodologies allow the model to provide reliable forecasts, empowering stakeholders to make informed, data-driven decisions. By harnessing the power of machine learning, this initiative supports efforts to reduce financial risks and ensure greater stability within the corporate landscape.

1.2. OBJECTIVES

The objectives of the "Insolvency Predictor using Time Series" project are:

1. Develop a Predictive Model: Create a model to forecast bankruptcy risk for companies, enabling stakeholders to identify early signs of financial distress.

2. Enhance Decision-Making: Improve stakeholders' ability to make informed decisions by providing clear and actionable insights from predictive data.

3. Utilize Data Visualization Tools: Maximize the use of tools like Power BI and Tableau to effectively visualize data and improve analytical capabilities.

4. Risk Assessment: Enable creditors and investors to assess the financial risks associated with companies more accurately.

5. Support Proactive Financial Measures: Provide early warnings for policymakers to implement proactive strategies to minimize broader economic impacts.

6. Balance Data through Sampling: Use Undersampling and Oversampling techniques to handle imbalanced datasets for more accurate predictions.

7. Leverage Logistic Regression: Employ Logistic Regression for building the core prediction model.

2. SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

In the context of **bankruptcy prediction**, the **existing systems** typically rely on traditional financial analysis methods, such as ratio analysis and statistical techniques. These systems often use financial metrics like liquidity ratios, profitability ratios, and leverage ratios to assess a firm's financial health. Key existing methods include:

1. **Altman's Z-Score Model**: A statistical model that uses financial ratios to predict the likelihood of bankruptcy.
2. **Logistic Regression Models**: Simple models that utilize historical financial data to estimate the probability of a firm's default.
3. **Machine Learning Models**: Some modern systems use basic machine learning algorithms like decision trees or support vector machines.

Limitations of the Existing System:

1. **Limited Use of Time Series Data**: Many existing systems do not adequately incorporate time series data to capture trends over time, leading to less accurate predictions.
2. **Static Models**: Traditional models rely heavily on static financial ratios, which may not reflect sudden market changes or a firm's operational dynamics.
3. **Imbalanced Data Handling**: Many models struggle with handling imbalanced datasets where the number of bankrupt companies is far smaller than non-bankrupt ones, leading to biased predictions.
4. **Lack of Visualization Tools**: Existing systems often lack powerful visualization tools like **Power BI** or **Tableau**, making it difficult for stakeholders to easily interpret data and predictions.
5. **Limited Feature Engineering**: The models may not capture all relevant features, leading to oversimplified predictions that miss important signals of financial distress.

6. **Overfitting**: Some machine learning models tend to overfit the data, making predictions less generalizable to new cases.

7. **Ineffective for Policy Makers**: Existing systems may not provide timely insights for policymakers to take early action, limiting their ability to prevent broader economic impacts.

2.2. PROPOSED SYSTEM

The "Insolvency Predictor using Time Series" project proposes a more advanced and dynamic bankruptcy prediction model that leverages time series data, modern data analytics tools, and machine learning techniques. This system is designed to provide accurate and timely predictions of a company's likelihood of facing financial distress, offering stakeholders early warnings and more comprehensive risk assessments.

Key Features of the Proposed System:

1. **Time Series Analysis**: Unlike traditional models, this system uses time series data to capture trends, patterns, and changes over time, improving the accuracy of bankruptcy predictions.

2. **Data Visualization Tools**: The system integrates advanced visualization platforms like Power BI and Tableau to create intuitive, interactive dashboards, enabling users to visualize trends and interpret results easily.

3. **Machine Learning Algorithm**s: The system employs machine learning models, including Logistic Regression, to make more sophisticated predictions based on historical financial data and market trends.

4. **Handling Imbalanced Datasets**: The proposed system incorporates techniques like Undersampling and Oversampling to address the common issue of imbalanced datasets, ensuring more balanced and reliable predictions.

5. **Predictive Accuracy**: By leveraging advanced data analysis techniques, the system aims to enhance predictive accuracy, offering more precise forecasts of insolvency risk.

6. **Early Warning Mechanism**: The model provides early detection of financial distress, allowing stakeholders such as creditors, investors, and policymakers to take proactive measures to prevent or minimize financial losses.
7. **Automated Data Processing**: The system is designed to process and analyze large datasets efficiently, offering real-time or near-real-time predictions for timely decision-making.
8. **Scalability and Flexibility**: The proposed system can be easily scaled to analyze multiple companies or sectors and adapt to new financial indicators or models as needed.
9. **Improved Decision-Making Support**: With data-driven insights and interactive visualizations, the system enhances decision-making processes for various stakeholders involved in managing financial risks.

2.3. FEASIBILITY STUDY

2.3.1.. Details

The "Insolvency Predictor using Time Series" project focuses on predicting the likelihood of a company facing financial distress by analyzing historical financial data using time series techniques. The model integrates machine learning algorithms, such as Logistic Regression, and data visualization tools like Power BI and Tableau. It handles large datasets and uses sampling techniques to overcome imbalanced data, aiming for accurate and timely predictions that help stakeholders make informed decisions to minimize financial risk.

2.3.2. Impact on Environment

The project is digital, so its direct environmental impact is minimal. However, by promoting financial stability and preventing bankruptcies, it may indirectly support sustainable business practices. Companies in distress may resort to unsustainable cost-cutting measures, including environmental degradation. Early insolvency prediction can prevent such situations, encouraging more responsible financial management, which may ultimately benefit environmental sustainability in the long run.

2.3.3. Safety

The project primarily deals with financial data and does not have a direct physical safety component. However, by predicting financial distress, it can indirectly contribute to societal safety by reducing economic instability, which often leads to job losses, social unrest, and financial crises. Implementing secure data handling practices, especially for sensitive financial data, is essential to ensure digital security and privacy.

2.3.4. Ethics

The project must adhere to ethical standards, particularly in terms of data privacy and confidentiality. Since it deals with sensitive financial information, it is crucial to protect this data and avoid any misuse. Additionally, the model should be unbiased, providing fair predictions without discriminating against smaller firms or certain industries. Ethical considerations also include transparency in the model’s decision-making process to ensure that stakeholders trust the results.

2.3.5. Cost

The primary costs of the project involve developing and implementing the predictive model, processing the financial data, and utilizing tools like **Power BI**. Additional costs may include cloud storage for data and computational resources for machine learning model training. These costs are relatively low compared to the potential financial benefits of preventing bankruptcies and managing economic risks effectively.

2.3..6 Type

This is a **data-driven predictive modeling** project focused on **financial risk management**. It combines elements of **machine learning**, **data analytics**, and **time series forecasting**. The project is technical in nature, requiring expertise in data science, financial analysis, and software development to build and deploy a functional prediction model for real-world applications.

2.4. SCOPE OF THE PROJECT

The scope of the **"Insolvency Predictor using Time Series"** project is broad and impactful, aiming to provide a sophisticated solution for bankruptcy prediction. The model can be applied across various industries, helping stakeholders such as creditors, investors, policymakers, and business owners assess financial risks effectively. It leverages **time series data**, **machine learning algorithms**, and **data visualization tools** like **Power BI** and **Tableau** to provide accurate, real-time forecasts of insolvency risk.The project's scope includes the integration of advanced techniques like **Undersampling**, **Oversampling**, and **Logistic Regression** to handle imbalanced datasets and improve the reliability of predictions. It also involves creating user-friendly dashboards for stakeholders, enhancing decision-making capabilities. The model can be scaled to analyze data from multiple companies or sectors and adapted to incorporate new financial indicators.In the long term, the project could be extended to provide broader economic insights, allowing policymakers to detect systemic financial risks and take preventive measures, contributing to the stability of the financial system. Additionally, it could be customized for specific industries or regions, making it highly adaptable and valuable.

2.5. SYSTEM CONFIGURATION

1. **Hardware Requirements**:
    - **Processor**: Intel Core i5 or higher (or equivalent AMD processor)
    - **RAM**: Minimum 8 GB (16 GB or higher recommended for large datasets)
    - **Storage**: At least 500 GB HDD or 256 GB SSD (SSD recommended for faster data processing)
    - **GPU**: Optional but recommended for machine learning model training (NVIDIA GTX 1650 or higher)
2. **Software Requirements**:
    - **Operating System**: Windows 10 or higher, macOS, or Linux (Ubuntu 18.04 or higher)
    - **Programming Language**: Python (version 3.8 or higher)

   - **Libraries/Packages**:
     - Pandas (for data manipulation)
     - NumPy (for numerical computations)
     - Scikit-learn (for machine learning algorithms like Logistic Regression)
     - Matplotlib/Seaborn (for data visualization)
     - Statsmodels (for time series analysis)
     - TensorFlow/PyTorch (optional, for more advanced machine learning)
   - **IDE**: Jupyter Notebook or PyCharm for code development
   - **Data Visualization Tools**:
     - **Power BI** (for dashboard creation and reporting)
     - **Tableau** (for advanced interactive visualizations)

3. **Cloud/Database Requirements**:
   - **Cloud Platform (Optional)**: AWS, Google Cloud, or Microsoft Azure for handling large datasets and scaling the model
   - **Database**: MySQL, PostgreSQL, or NoSQL databases like MongoDB for storing historical financial data.
4. **Other Tools**:
   - **Version Control**: Git/GitHub for collaboration and version control.
   - **Virtual Environment**: Conda or Virtualenv for managing Python libraries and dependencies.

3. LITERATURE OVERVIEW

3.1. TITLE – "IMPACT OF POWER BI ON BUSINESS"

The research paper highlights the transformative impact of technology on the modern business landscape, emphasizing innovations such as AI, big data analytics, and the internet in reshaping organizational dynamics. It discusses how automation enhances efficiency and adaptability, enabling companies to allocate resources strategically. Power BI is identified as a key player in this transformation, providing valuable insights into customer preferences and behaviors. However, the paper underscores the critical challenge of balancing powerful data insights with robust cybersecurity measures to safeguard sensitive information, emphasizing the necessity for businesses to integrate innovative solutions while maintaining a competitive edge in a rapidly evolving environment.

Furthermore, the study explores how Power BI revolutionizes data utilization by converting raw information into actionable insights through visualizations and predictive analytics. Despite its benefits, the paper identifies several challenges, including usability concerns for newcomers, data quality issues, and potential compatibility problems across various versions. It emphasizes the importance of proactively addressing these challenges to maximize Power BI's effectiveness in business analytics. Ultimately, the research concludes that Power BI not only empowers organizations to make data-driven decisions but also fosters a culture of collaboration and agility, essential for thriving in an increasingly competitive market.

3.2. TITLE – "Bankruptcy Prediction Using Machine Learning Techniques"

The research highlights the challenges faced by the European Central Bank (ECB) in managing crisis intervention and recovery planning since the 2008 financial crisis, particularly regarding the assessment of banks in distress. By employing XGBoost, the study presents a robust classification model capable of handling multicollinearity among variables without necessitating their removal. The findings suggest that specific financial ratios can serve as critical indicators for anticipating bank failures, ultimately providing valuable insights for regulators and bank management to implement timely preventive measures.

The study titled "Bankruptcy Prediction Using Machine Learning Techniques" explores various advanced machine learning methods, including Extreme Gradient Boosting (XGBoost), Support Vector Machine (SVM), and a deep neural network, to predict bankruptcy among 3,728 Belgian Small and Medium Enterprises (SMEs) from 2002 to 2012. The research focuses on utilizing three easily obtainable financial ratios—return on assets, current ratio, and solvency ratio—to achieve a bankruptcy prediction accuracy of 82-83%. Although this accuracy aligns with several existing models, the simplicity of the approach makes it accessible and user-friendly for stakeholders such as shareholders, managers, and banks.

The authors compare their results to previous studies, particularly referencing Brédart's earlier work, which also reported similar accuracy using simpler models. Despite employing more sophisticated machine learning techniques, the study finds that these do not significantly enhance prediction accuracy beyond that achieved by a basic neural network with a single hidden layer. The overlap in the feature distributions between bankrupt and solvent firms limits the effectiveness of advanced models. Ultimately, the research underscores the value of simple, easily implementable models in bankruptcy prediction, providing reliable tools for decision-makers in the financial sector.

3.3. TITLE – "Bankruptcy prediction using machine learning and an application to the case of the COVID-19 recession"

This study investigates the application of machine learning techniques to bankruptcy prediction, focusing on a comprehensive quarterly dataset of financial ratios from public U.S. firms spanning 1970 to 2019. The research highlights the effectiveness of tree-based ensemble methods, particularly XGBoost, in achieving high accuracy for out-of-sample bankruptcy predictions. The model is subsequently employed to forecast the overall bankruptcy rate in the U.S. for the second half of 2020, following the economic downturn caused by the COVID-19 pandemic. The findings align with economists’ predictions of a significant rise in bankruptcies, while suggesting that the levels may not exceed those observed in 2010.

The introduction emphasizes the evolution of bankruptcy prediction from early univariate models to more sophisticated multivariate statistical models, such as the Altman Z-score. It notes that while previous studies have often concentrated on the predictive models themselves, this research balances the importance of both data and models. Utilizing the extensive Compustat database, the authors aim to adopt a more data-centric approach to enhance the robustness of their bankruptcy prediction models, reflecting a shift in methodology within the field.

3.4. TITLE – “An introductory study on time series modeling and forecasting”

This study highlights the critical role of time series modeling and forecasting across various practical domains, showcasing ongoing research aimed at enhancing the accuracy and efficiency of these methods. It provides a concise overview of popular time series forecasting models, categorizing them into three primary classes: stochastic models, neural networks, and support vector machine (SVM)-based models. The discussion encompasses fundamental issues related to time series modeling, such as stationarity, parsimony, and overfitting. Experimental results from six real-time datasets support the analysis, demonstrating the importance of model selection and evaluation using various performance measures, including MSE, MAD, RMSE, MAPE, and Theil's U-statistics.

In time series analysis, researchers fit suitable models to datasets, estimating parameters from known values to understand the nature of the series and facilitate future forecasting. This method is particularly valuable in scenarios where the underlying statistical patterns are unclear or when no satisfactory explanatory model exists. Given its significant applications in strategic decision-making and precautionary measures, the pursuit of effective forecasting models remains a critical area of research, with ongoing efforts aimed at refining these methodologies for improved predictive accuracy.

4. SYSTEM DESIGN

4.1. SYSTEM ARCHITECTURE

4.1.1. Data Source Layer:

- **Data Collection:** Financial datasets from multiple sources such as CSV files, databases (SQL, NoSQL), or external APIs.
- **ETL Process:** Data cleaning, preprocessing, and transformation to handle missing values and format the data for analysis.

4.1.2. Backend Layer (Flask Application):

- **Flask Framework:** Acts as the server handling user requests and interacting with the model.
- **Logistic Regression Model:** Implemented using Python libraries such as scikit-learn. This model is used for predicting the probability of bankruptcy based on historical financial data.
- **API Services:** RESTful APIs to handle:
  - Upload of new datasets for prediction.
  - Providing prediction results to the frontend or reporting tools (e.g., Power BI).
- **Data Pipeline:** Handles training, testing, and storing model outputs in a database (PostgreSQL/MySQL).

4.1.3. Data Storage Layer:

- **Database (SQL/NoSQL):** Stores historical financial data, training data, and results of predictions.
- **Model Storage:** If needed, you can store trained models in a serialized format (e.g., .pkl files).

4.1.4. Visualization and Reporting Layer:

- **Power BI/Tableau:** Fetches prediction results via API or database to visualize company financial health, predictive risk analysis, and trend reports.
- **Integration:** Power BI/Tableau reports can be embedded into the Flask application for real-time analytics.

4.1.5. Frontend Layer (Flask-Rendered Pages or React/Vue.js):

- **User Interface:** Allows users to upload datasets, view predictive insights, and download reports. Flask can render these pages with Jinja2 templates or integrate with frontend frameworks.
- **Interaction with Backend:** Communicates with Flask APIs to display the predictions and financial health of companies.

4.2. UML DIAGRAMS

Use Case Diagram

The use case diagram for Insolvency Predictor using Time Series illustrates the interactions between the system and its users. In this diagram, the primary actors include User and Software. For a user, use cases involve uploading the values into the form.The next steps involve building and testing of model by the developer. The model then predicts the status of bankruptcy and displays the graphs denoting the relation of attributes with bankruptcy over a period of time using time series and powerbi.

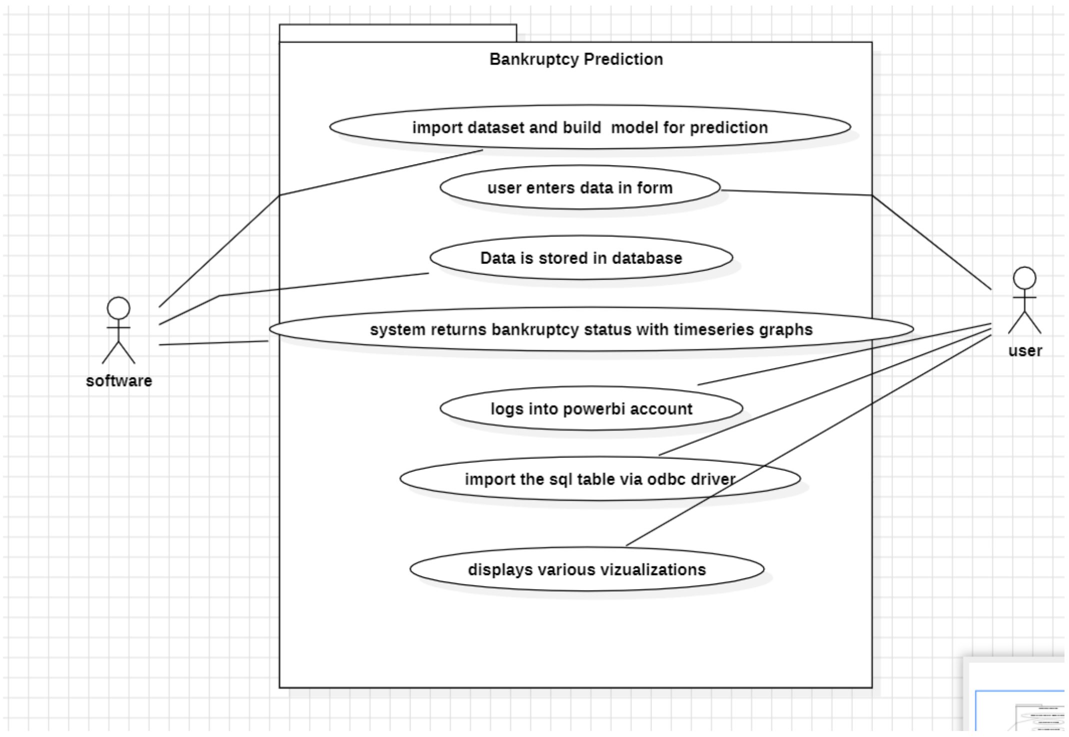

Bankruptcy Prediction
import dataset and build model for prediction
user enters data in form
Data is stored in database
system returns bankruptcy status with timeseries graphs
logs into powerbi account
import the sql table via odbc driver
displays various vizualizations
software
user

Fig 4.2.1 Usecase Diagram

Sequence Diagram

The sequence diagram for Insolvency Predictor Using Time Series outlines the sequence of interactions between users and software. For instance, the process starts by building a model to predict bankruptcy. The user then enters the data into the form. The system then returns the if it is bankrupt or not along with the time series graph of how the attributes are related with bankruptcy. The user is also provided with powerbi visuals to see which attributes contributes more towards bankruptcy.

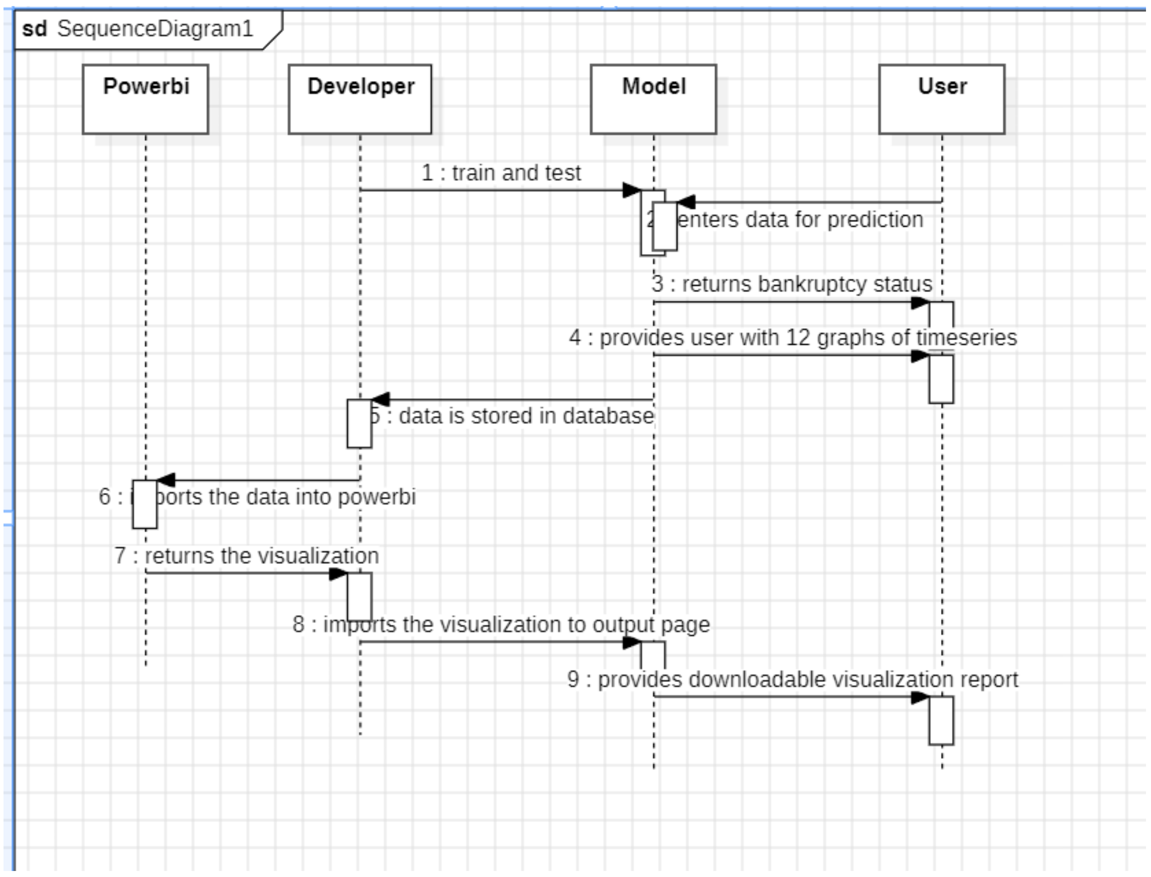
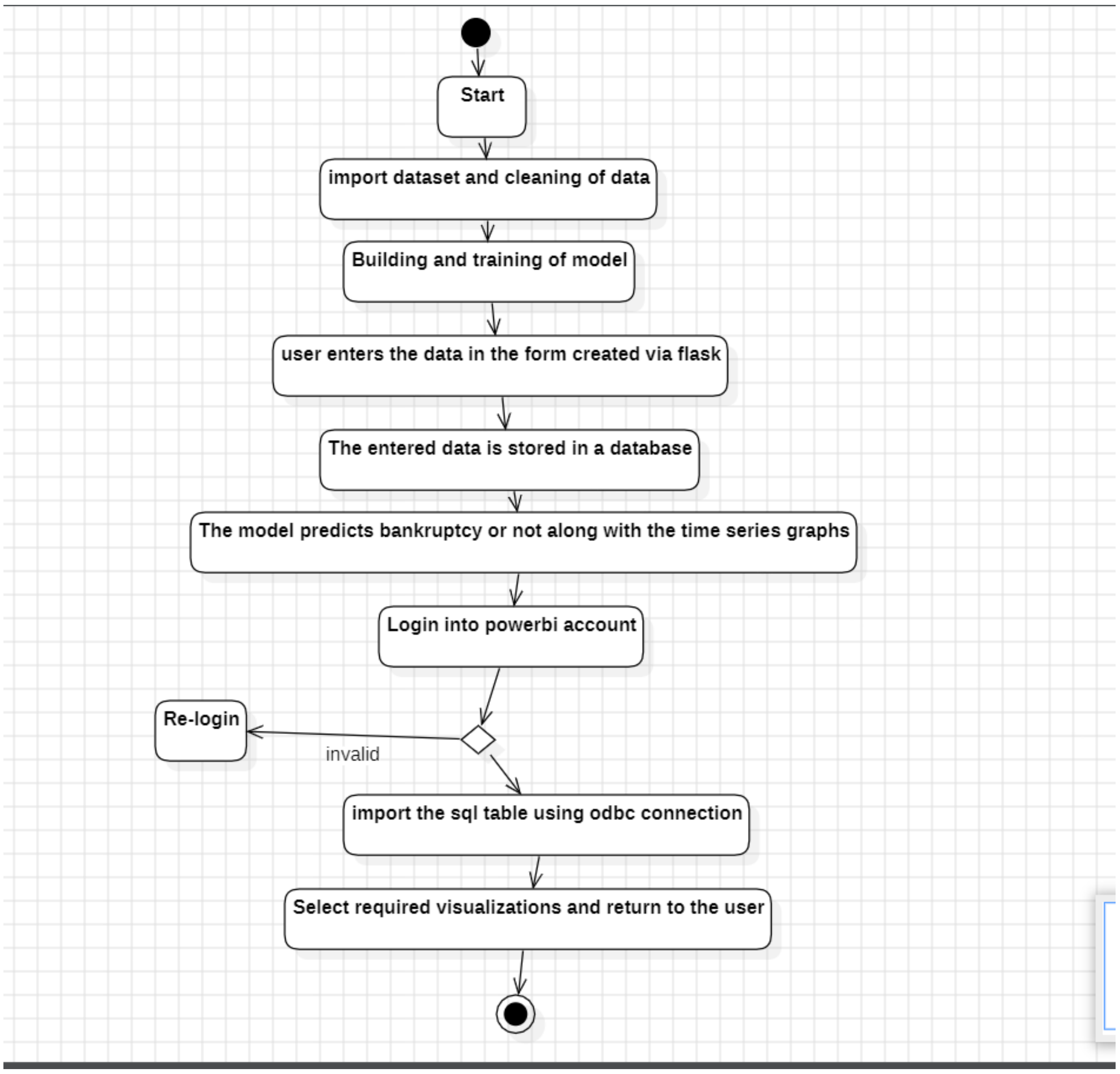

sd SequenceDiagram1
Powerbi
Developer
Model
User
1 : train and test
2 : enters data for prediction
3 : returns bankruptcy status
4 : provides user with 12 graphs of timeseries
5 : data is stored in database
6 : imports the data into powerbi
7 : returns the visualization
8 : imports the visualization to output page
9 : provides downloadable visualization report

Fig 4.2.2 Sequence Diagram

Activity Diagram

The activity diagram for Insolvency Predictor Using Time Series represents the workflow of specific processes within the system. For example, the process of predicting bankruptcy involves building and training of a machine learning model to predict bankruptcy. The user then enters the data into the form.The system then uses the model to predict the bankruptcy along with timeseries graphs showing the relation of the user entered attributes with bankruptcy over a period of time.



Start
import dataset and cleaning of data
Building and training of model
user enters the data in the form created via flask
The entered data is stored in a database
The model predicts bankruptcy or not along with the time series graphs
Login into powerbi account
Re-login
invalid
import the sql table using odbc connection
Select required visualizations and return to the user

Fig 4.2.3 Activity Diagram

Class Diagram

User class:

The User class manages the interaction with the prediction model. The user starts by entering the values or the attributes into the flask form. The model then returns the result of bankruptcy along with the time series graphs.

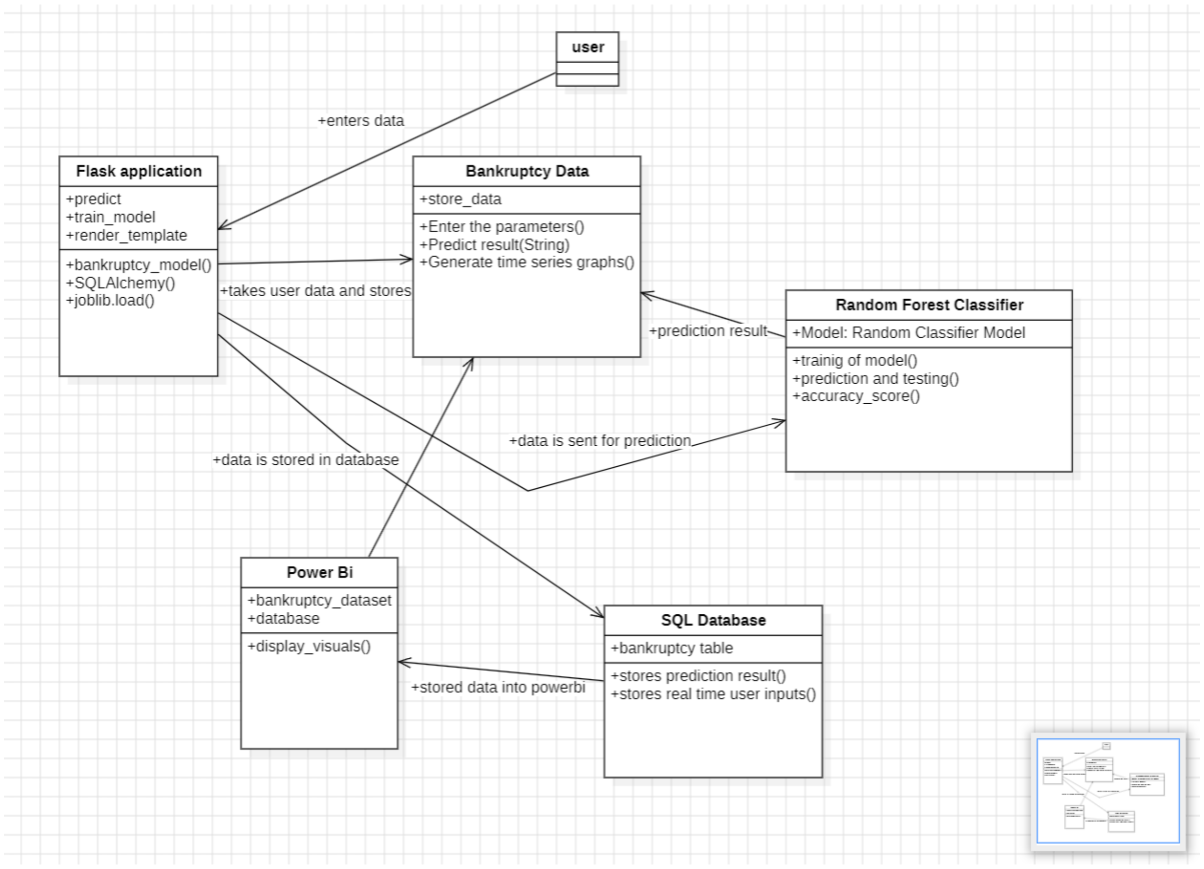


Fig 4.2.4

Flask Application Class:

The flask Application class stores the data into sql for further predictions. The data is then passed on to the machine learning model where the model analyses the data and predicts the status of bankruptcy.

Random Forest Classifier Class:

The data comes as a input from the flask Application class. The data is then analysed here based on the trained model and result is predicted with atmost accuracy. The prediction result is also stored in the sql database along with the user inputs.

SQL Database Class:

The SQL Database class is used to store the input data of the user coming from the flask application along with prediction result. The data stored in the sql is then served as a input for the power bi for visualizations.

Power Bi Class:

The data stored in the sql table is connected/imported into the powerbi. Then the selected visualizations are generated such as bar graphs are generated on the data stored in the database. The graphs helps in easy analyzing of data and understand how each attribured is related with bankruptcy.

Data Flow Diagram

The Data Flow Diagram (DFD) for the "Insolvency Predictor Using Time Series" illustrates the flow of data between the user, system components, and visualization tools. The system begins with the User inputting financial data (e.g., liabilities, assets, expenses) into the Flask Application through a web interface. The Flask application processes the input and stores it in an SQL Database.Simultaneously, the financial data is passed to the RandomForestClassifier, a machine learning model, which predicts whether the company will face bankruptcy. The prediction result (e.g., "Bankrupt" or "Not Bankrupt") is returned to the Flask application and stored back into the SQL Database along with the user’s financial data for future use.Power BI/Tableau then connects to the SQL database to retrieve the financial data and prediction results, visualizing them in dashboards and charts. These visualizations provide the user with insights into the company’s financial health and the likelihood of bankruptcy, helping stakeholders make informed decisions. This data flow ensures that the entire system, from user input to prediction and visualization, works cohesively to provide accurate, real-time financial insights.

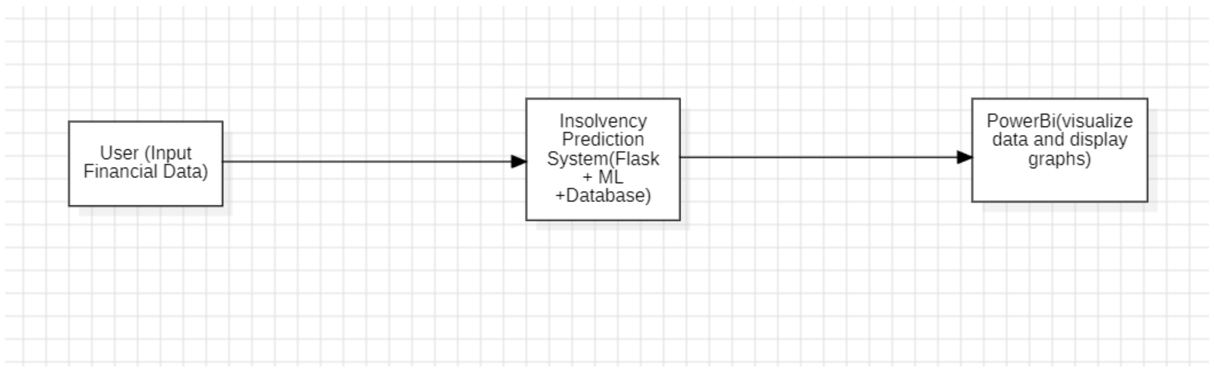

User (Input Financial Data)
Insolvency Prediction System(Flask + ML +Database)
PowerBi(visualize data and display graphs)

Fig 4.2.5.1. level-0 DFD

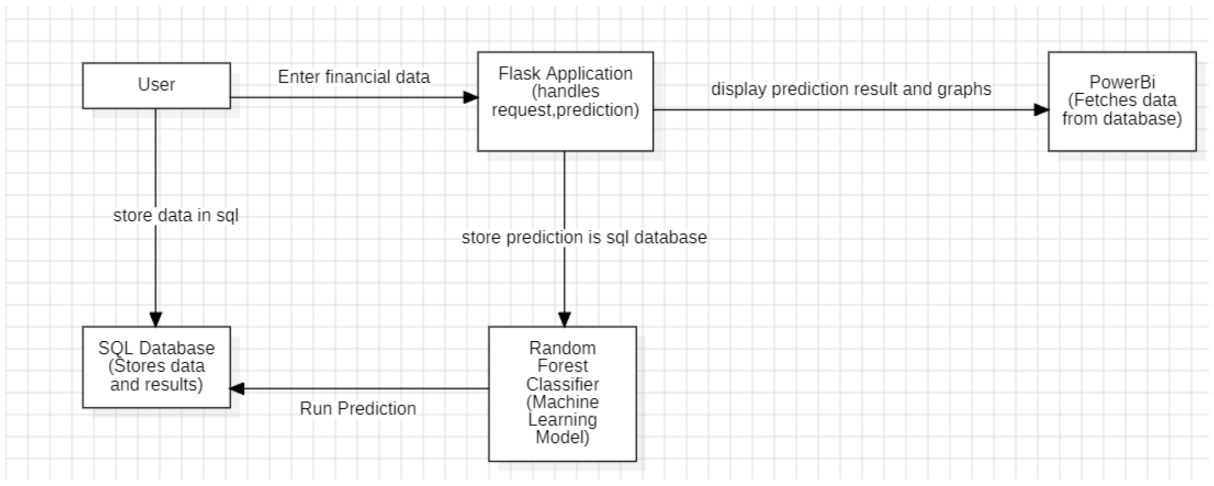

User
Enter financial data
Flask Application (handles request,prediction)
display prediction result and graphs
PowerBi (Fetches data from database)
store data in sql
store prediction is sql database
SQL Database (Stores data and results)
Run Prediction
Random Forest Classifier (Machine Learning Model)

Fig 4.2.5.2. level-1

5. IMPLEMENTATION

5.1. Machine Learning Model Development

The first step is to develop and train a machine learning model that will predict whether a company will go bankrupt based on historical financial data.

5.1.1. Data Collection and Preprocessing

Dataset Selection:

- Obtain historical financial data from various companies. The dataset should include financial indicators such as revenue, expenses, assets, liabilities, and an outcome variable indicating whether the company went bankrupt (binary classification).

Data Cleaning:

- Handle missing data by filling or removing null values.

- Standardize or normalize the financial indicators to ensure consistency in model training**.**

-Approximately 13 key columns are taken by feature selection.

- Feature Selection:

- Key features might include:

- total_expense_assets
- tax_rate
- cash_current_liability
- liability_assets_flag
- quick_asset_turnover_rate
- net_value_growth_rate

- The target variable is the “bankrupt?” (1 for bankrupt, 0 for non-bankrupt).

5.1.2. Model Training and Validation:

- Model Selection:

- Choose a suitable machine learning algorithm for binary classification. Common algorithms for this task include:

- Logistic Regression
- Random Forest
- Gradient Boosting (XGBoost)
- Support Vector Machine (SVM)

-Here, we used "random forest" algorithm as it gave the most accurate results.

- Split the dataset into training and testing sets (e.g., 80% training, 20% testing).

- Train the model using the training data and evaluate its performance on the testing set using metrics like accuracy, precision, recall, and F1-score.

5.1.2.1. Load the dataset

```
data = pd.read_csv('bank.csv')

X = data[['revenue', 'expenses', 'assets', 'liabilities']]

y = data['bankrupt']
```

5.1.2.2. Split the data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

-Here, we are dividing the data into 80 to 20 percent.

5.1.2.3. Train the model

```
model = RandomForestClassifier()

model.fit(X_train, y_train)
```

5.1.2.4. Make predictions and evaluate

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy}"
```

5.1.3. Time Series Analysis for Financial Trend Visualization:

After training the machine learning model, we apply time series analysis on financial data to provide users with trend visualizations based on historical performance.

-We applied synthetic data generation.

5.1.3.1. Graph Generation

- Generate time series graphs based on the financial indicators, displaying trends such as revenue growth or changes in liabilities over time.

- Use these graphs to give users a visual representation of their company's financial performance**.**

5.1.4. User Input and Real-Time Prediction:

The next step is to build an interface where users can input their financial data, which is then passed to the machine learning model for bankruptcy prediction**.**

5.1.4.1. User Input Interface

-Design the Interface:

- Build a form that allows users to input their current financial indicators (e.g., revenue, expenses, liabilities, assets).

- Input Validation:

- Ensure that the user inputs are validated for correctness (e.g., positive numbers for revenue and assets).

5.1.4.2. Real-Time Bankruptcy Prediction

- Once the user enters their data, the system runs the trained machine learning model to predict bankruptcy.

- The user’s financial data is passed to the model, and a prediction is returned (either “bankrupt” or “not bankrupt”)

5.1.5. Data Storage in SQL Database

After making predictions, the user's financial data and the prediction results are stored in an SQL database for future access and visualization in Power BI.

5.1.5.1. Database Schema

- The SQL database stores all relevant information such as user input, predictions, and time series data.

5.1.5.2. Inserting User Data into SQL

- Store the user’s input and the bankruptcy prediction in the database using SQL insert commands.

5.1.5.3. Integration with Power BI for Visualization

After storing the data, the final step is to visualize the financial trends and prediction results in Power BI using direct connections to the SQL database.

5.1.5.4. Connecting SQL to Power BI

- Use Power BI’s SQL Server connector to establish a connection to the SQL database or use ODBC drivers.

- Import the necessary tables (financial data, predictions) from the SQL database into Power BI.

5.1.5.5. Creating Visuals

-Graphs:

- Use “line charts” to display time series data such as revenue trends.

- “Pie Charts”:

- Display the percentage of users predicted to go bankrupt vs. not bankrupt using a pie chart.

5.1.5.6. Dynamic Visualization

- Power BI will automatically update the visualizations as new data is entered, providing real-time insights.

5.2. SAMPLE CODE

5.2.1. app.py

```
from flask import Flask, render_template, request, redirect
from flask_sqlalchemy import SQLAlchemy
import joblib
import numpy as np
import pandas as pd
import plotly.graph_objs as go
from plotly.subplots import make_subplots

app = Flask(__name__)

# Configure the SQLite database (or replace with your database connection
    string)
app.config['SQLALCHEMY_DATABASE_URI']                                    =
    'sqlite:///bankruptcy_data.db'
db = SQLAlchemy(app)

# Load the trained model
model = joblib.load('bankruptcy_model.pkl')

# Define a model for the user input data and prediction result
class BankruptcyData(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    current_liability_to_current_assets = db.Column(db.Float, nullable=False)
    liability_assets_flag = db.Column(db.Integer, nullable=False)
    total_expense_assets = db.Column(db.Float, nullable=False)
    cash_current_liability = db.Column(db.Float, nullable=False)
    fixed_assets_turnover_frequency = db.Column(db.Float, nullable=False)
    fixed_assets_to_assets = db.Column(db.Float, nullable=False)
    net_value_growth_rate = db.Column(db.Float, nullable=False)
    revenue_per_person = db.Column(db.Float, nullable=False)
    total_assets_to_gnp_price = db.Column(db.Float, nullable=False)
    quick_asset_turnover_rate = db.Column(db.Float, nullable=False)
    tax_rate = db.Column(db.Float, nullable=False)
    cash_total_assets = db.Column(db.Float, nullable=False)
    prediction_result = db.Column(db.String(50), nullable=False)

# Create the database tables inside the application context
with app.app_context():
    db.create_all()

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # Get form data
    form_data = request.form

    # Prepare the input data for the model
    input_data = [[
        float(form_data['current_liability_to_current_assets']),
        int(form_data['liability_assets_flag']),
        float(form_data['total_expense_assets']),
        float(form_data['cash_current_liability']),
        float(form_data['fixed_assets_turnover_frequency']),
        float(form_data['fixed_assets_to_assets']),
        float(form_data['net_value_growth_rate']),
        float(form_data['revenue_per_person']),
        float(form_data['total_assets_to_gnp_price']),
        float(form_data['quick_asset_turnover_rate']),
        float(form_data['tax_rate']),
        float(form_data['cash_total_assets'])
    ]]

    # Make the prediction using the loaded model
    prediction = model.predict(input_data)[0]

    # Interpret the prediction result
    prediction_result = 'Bankrupt' if prediction == 1 else 'Non-bankrupt'

    # Save the data along with the prediction result to the database
    new_entry = BankruptcyData(
      current_liability_to_current_assets=float(form_data['current_liability_to_c
      urrent_assets']),
        liability_assets_flag=int(form_data['liability_assets_flag']),
        total_expense_assets=float(form_data['total_expense_assets']),
        cash_current_liability=float(form_data['cash_current_liability']),

      fixed_assets_turnover_frequency=float(form_data['fixed_assets_turnover_
      frequency']),
        fixed_assets_to_assets=float(form_data['fixed_assets_to_assets']),
        net_value_growth_rate=float(form_data['net_value_growth_rate']),
        revenue_per_person=float(form_data['revenue_per_person']),
        total_assets_to_gnp_price=float(form_data['total_assets_to_gnp_price']),
        quick_asset_turnover_rate=float(form_data['quick_asset_turnover_rate']),
        tax_rate=float(form_data['tax_rate']),
        cash_total_assets=float(form_data['cash_total_assets']),
        prediction_result=prediction_result
    )

    # Add to the session and commit
    db.session.add(new_entry)
    db.session.commit()

    # Create a timeseries for 12 months for each attribute and plot them
    months = pd.date_range(start='2024-01-01', periods=12, freq='M')
    attributes = {
        'Current Liability to Current Assets': input_data[0][0],
        'Liability-Assets Flag': input_data[0][1],
        'Total Expense / Assets': input_data[0][2],
        'Cash / Current Liability': input_data[0][3],
        'Fixed Assets Turnover Frequency': input_data[0][4],
        'Fixed Assets to Assets': input_data[0][5],
        'Net Value Growth Rate': input_data[0][6],
        'Revenue per Person': input_data[0][7],
        'Total Assets to GNP Price': input_data[0][8],
        'Quick Asset Turnover Rate': input_data[0][9],
        'Tax Rate': input_data[0][10],
        'Cash / Total Assets': input_data[0][11]
    }

    # Plot 12 graphs in subplots
    fig = make_subplots(rows=4, cols=3, subplot_titles=list(attributes.keys()))

    row = 1
    col = 1

    for i, (attribute, value) in enumerate(attributes.items()):
        # Generate synthetic data over 12 months (you can replace this with actual
    timeseries forecasting if available)
      data = np.random.normal(value, 0.1 * value, size=12)
     fig.add_trace(go.Scatter(x=months, y=data, mode='lines', name=attribute),
    row=row, col=col)

      col += 1
      if col > 3:
          col = 1
          row += 1

    fig.update_layout(height=900, width=1200, title_text="12 Attribute Time-
     Series")

    # Convert Plotly figure to JSON to be passed to the HTML template
    graphJSON = fig.to_json()

    return render_template('result.html', prediction=prediction_result,
      graphJSON=graphJSON)

    if __name__ == '__main__':
        app.run(debug=True)
```

5.2.2. index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bankruptcy Prediction Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 600px;
            margin: 50px auto;
            background-color: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            text-align: center;
        }
        label {
            font-weight: bold;
            margin-top: 10px;
        }
        input[type="text"] {
            width: 100%;
            padding: 8px;
            margin: 5px 0 20px 0;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        input[type="submit"] {
            width: 100%;
            background-color: #4CAF50;
            color: white;
            padding: 10px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Bankruptcy Prediction Form</h1>
        <form action="/predict" method="POST">
            <label for="current_liability_to_current_assets">Current Liability to Current
Assets</label>
            <input type="text" name="current_liability_to_current_assets" required><br>

            <label for="liability_assets_flag">Liability-Assets Flag</label>
            <input type="text" name="liability_assets_flag" required><br>

            <label for="total_expense_assets">Total Expense / Assets</label>
            <input type="text" name="total_expense_assets" required><br>

            <label for="cash_current_liability">Cash / Current Liability</label>
            <input type="text" name="cash_current_liability" required><br>

            <label for="fixed_assets_turnover_frequency">Fixed Assets Turnover
Frequency</label>
            <input type="text" name="fixed_assets_turnover_frequency" required><br>

            <label for="fixed_assets_to_assets">Fixed Assets to Assets</label>
            <input type="text" name="fixed_assets_to_assets" required><br>

            <label for="net_value_growth_rate">Net Value Growth Rate</label>
            <input type="text" name="net_value_growth_rate" required><br>

            <label for="revenue_per_person">Revenue per Person</label>
            <input type="text" name="revenue_per_person" required><br>

            <label for="total_assets_to_gnp_price">Total Assets to GNP Price</label>
            <input type="text" name="total_assets_to_gnp_price" required><br>

            <label for="quick_asset_turnover_rate">Quick Asset Turnover Rate</label>
            <input type="text" name="quick_asset_turnover_rate" required><br>

            <label for="tax_rate">Tax Rate</label>
            <input type="text" name="tax_rate" required><br>

            <label for="cash_total_assets">Cash / Total Assets</label>
            <input type="text" name="cash_total_assets" required><br>

            <input type="submit" value="Submit">
        </form>
    </div>
</body>
</html>
```

5.2.3. <u>result.html</u>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Result</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
    <div class="container">
     <center><h1>Prediction Result</h1></center>
      <h2 style="color: rgb(0, 217, 255);"><p>The company is predicted to be: <strong>{{
prediction }}</strong></p></h2>
     <button><a href="/">Go Back</a></button>

      <!-- Render the graphs -->
      <div id="timeseries-graph" style="width:100%; height:900px;"></div>

      <script>
        var graphs = {{ graphJSON|safe }};
        Plotly.newPlot('timeseries-graph', graphs.data, graphs.layout);
      </script>
    </div>
</body>
</html>
```

5.2.4. load and test model.py

```
import pandas as pd
import joblib

# Load the saved model
loaded_model = joblib.load('bankruptcy_model.pkl')

# Create a new data sample for prediction (replace with appropriate feature names)
new_data = pd.DataFrame({
    ' Current Liability to Current Assets': [0.118250477],
    ' Liability-Assets Flag': [0],
    ' Total expense/Assets': [0.064855708],
    ' Cash/Current Liability': [0.000147336],
    ' Fixed Assets Turnover Frequency': [0.000116501],
    ' Fixed Assets to Assets': [0.424205762],
    ' Net Value Growth Rate': [0.000326977],
    ' Revenue per person': [0.034164182],
    ' Total assets to GNP price': [0.00921944],
    ' Quick Asset Turnover Rate': [65500000000],
    ' Tax rate (A)': [0],
    ' Cash/Total Assets': [0.004094406],
})

# Predict using the loaded model
new_prediction = loaded_model.predict(new_data)

# Output the prediction result (0 = Non-bankrupt, 1 = Bankrupt)
print(f"Predicted Bankruptcy: {new_prediction}")
```

5.2.5 train and save model.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib

# Load the dataset
df = pd.read_csv('bank1.csv')
```

```
# Select features and target (replace with actual feature names and target column)
X = df[[' Current Liability to Current Assets', ' Liability-Assets Flag', ' Total expense/Assets',' Cash/Current Liability',' Fixed Assets Turnover Frequency',' Fixed Assets to Assets',' Net Value Growth Rate',' Revenue per person',' Total assets to GNP price',' Quick Asset Turnover Rate',' Tax rate (A)',' Cash/Total Assets']]
y = df['Bankrupt?']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy}")
print(classification_report(y_test, y_pred))

# Save the trained model
joblib.dump(model,'bankruptcy_model.pkl')
```

6. TESTING

6.1. Unit Testing:

- **Objective**: Test individual components (functions, APIs, and models) for correctness.
- **Data Preprocessing**: Ensure proper data cleaning, handling missing values, and feature scaling.
    - Example: Test a function that removes NaN values or scales the financial ratios.
- **Model Training & Prediction**: Validate the logistic regression model with a small test dataset.
    - Example: Test whether the model can train without errors and produce predictions within a valid range (0-1).
- **API Testing**: Test API endpoints for correct responses (status codes, data formats).
    - Example: Ensure that the /predict API returns a valid prediction when provided with input data.

6.2. Integration Testing:

- **Objective**: Ensure that different components (Flask API, model, Power BI integration) work together.
- **Data Flow**: Test the end-to-end flow of data from the frontend (dataset upload) through the Flask API, model processing, and visualization tools.
    - Example: Upload a dataset, run predictions via the Flask API, and verify the output is stored or visualized in Power BI/Tableau.
- **Model & API**: Test that the logistic regression model is correctly invoked by the API and returns expected outputs for various data inputs.

6.3. End-to-End Testing:

- **Objective**: Simulate real-world usage and test the system as a whole.
- **User Scenario**: Simulate a user uploading financial data, triggering predictions, and viewing results on the frontend or in Power BI.
    - Example: Upload a test CSV file and verify that the prediction results are accurate, correctly displayed, and visualized.

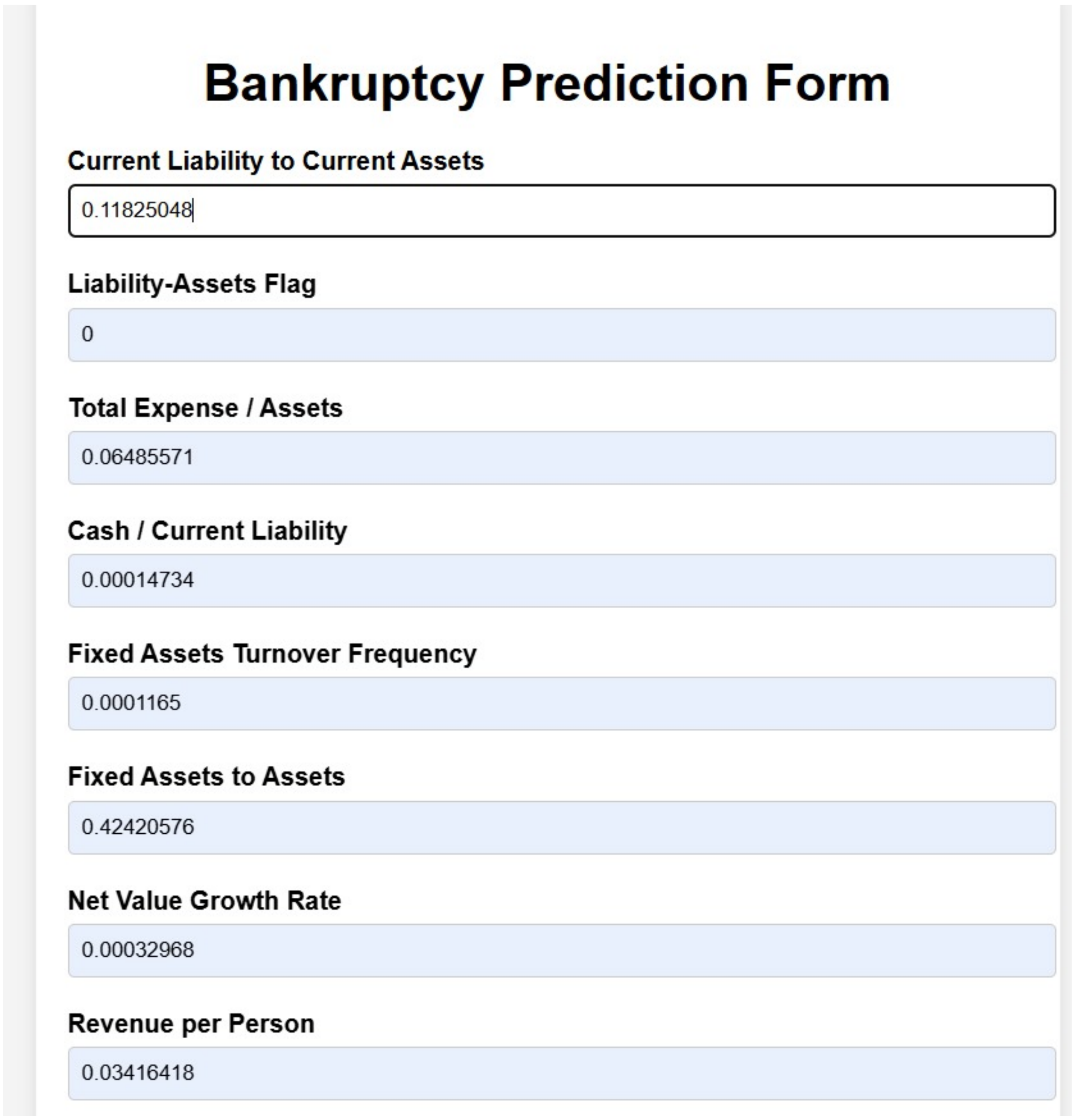
6.4. Performance Testing:

- **Objective**: Evaluate the system's responsiveness and stability under load.
- **Model Prediction Speed**: Test how quickly the logistic regression model can process different dataset sizes and return predictions.
    - Example: Simulate predicting bankruptcy for 100, 1,000, and 10,000 companies and measure response time.
- **API Load Testing**: Measure how well the API handles multiple simultaneous requests.
    - Example: Use tools like Apache JMeter or Locust to simulate multiple users uploading data or requesting predictions at the same time, and ensure the API does not crash or slow down significantly.

6.5. Security Testing:

- **Objective**: Ensure the system is protected from security vulnerabilities, especially since it handles financial data.
- **Data Validation**: Test that the API and Flask application validate inputs to prevent SQL injection, cross-site scripting (XSS), and other attacks.
    - Example: Attempt to upload maliciously formatted financial data or perform an injection attack and ensure the system rejects it.

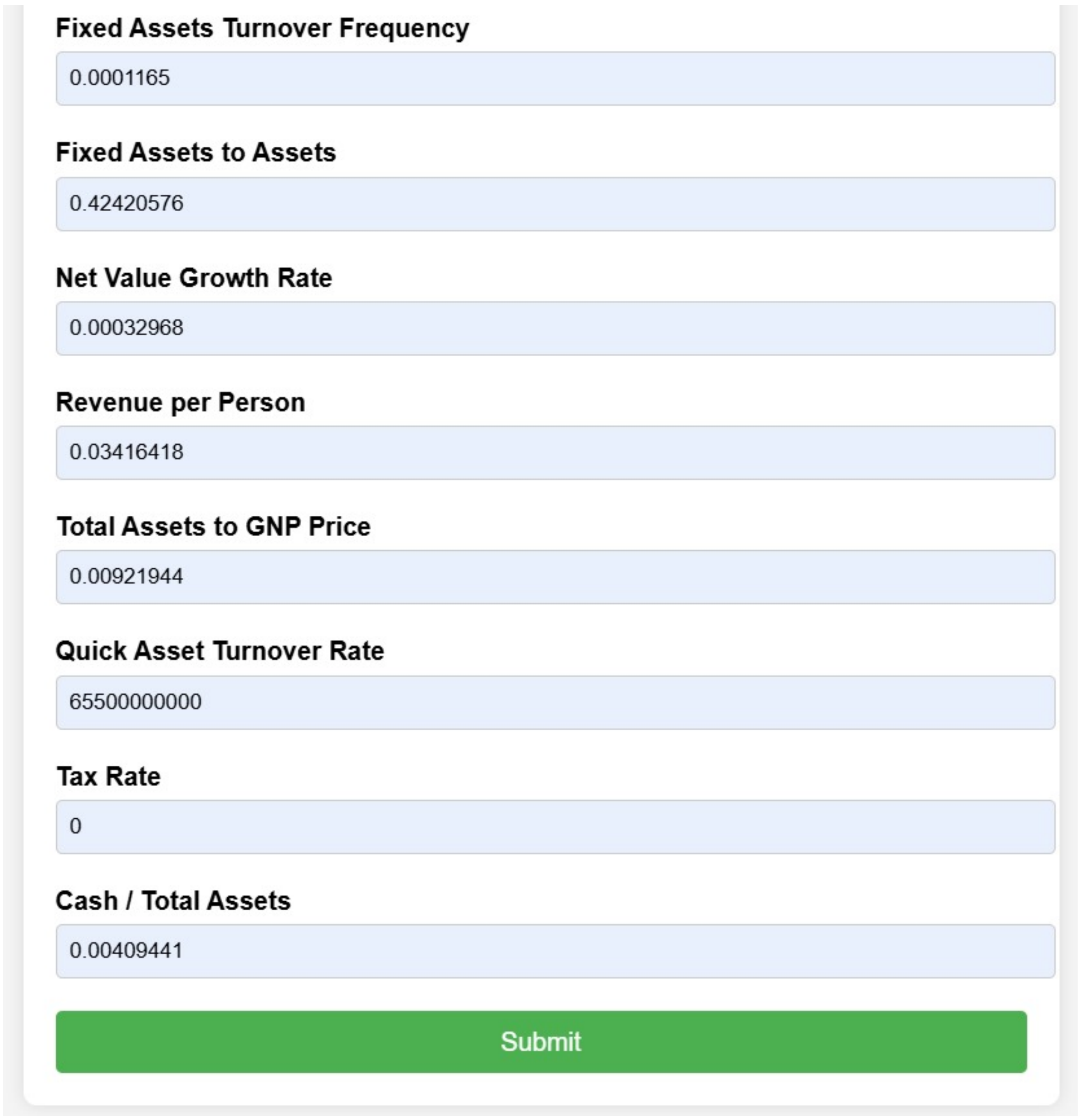
7.OUTPUT SCREENS



Bankruptcy Prediction Form

Current Liability to Current Assets	0.11825048
Liability-Assets Flag	0
Total Expense / Assets	0.06485571
Cash / Current Liability	0.00014734
Fixed Assets Turnover Frequency	0.0001165
Fixed Assets to Assets	0.42420576
Net Value Growth Rate	0.00032968
Revenue per Person	0.03416418

Fig 7.1 User input



Fixed Assets Turnover Frequency

0.0001165

Fixed Assets to Assets

0.42420576

Net Value Growth Rate

0.00032968

Revenue per Person

0.03416418

Total Assets to GNP Price

0.00921944

Quick Asset Turnover Rate

65500000000

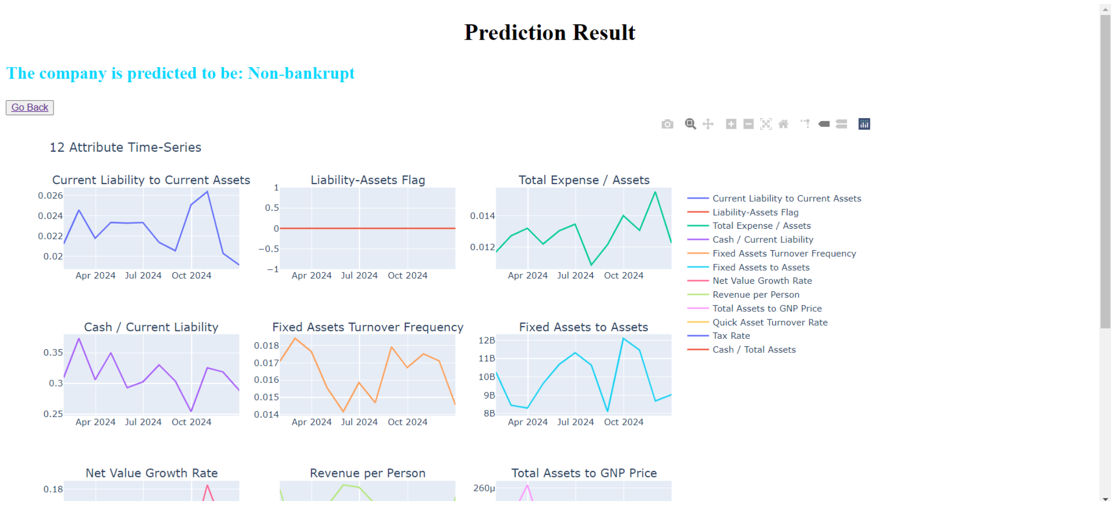
Tax Rate

0

Cash / Total Assets

0.00409441

Fig 7.2 User input


Prediction Result

The company is predicted to be: Non-bankrupt

Go Back

Fig 7.3 Result page

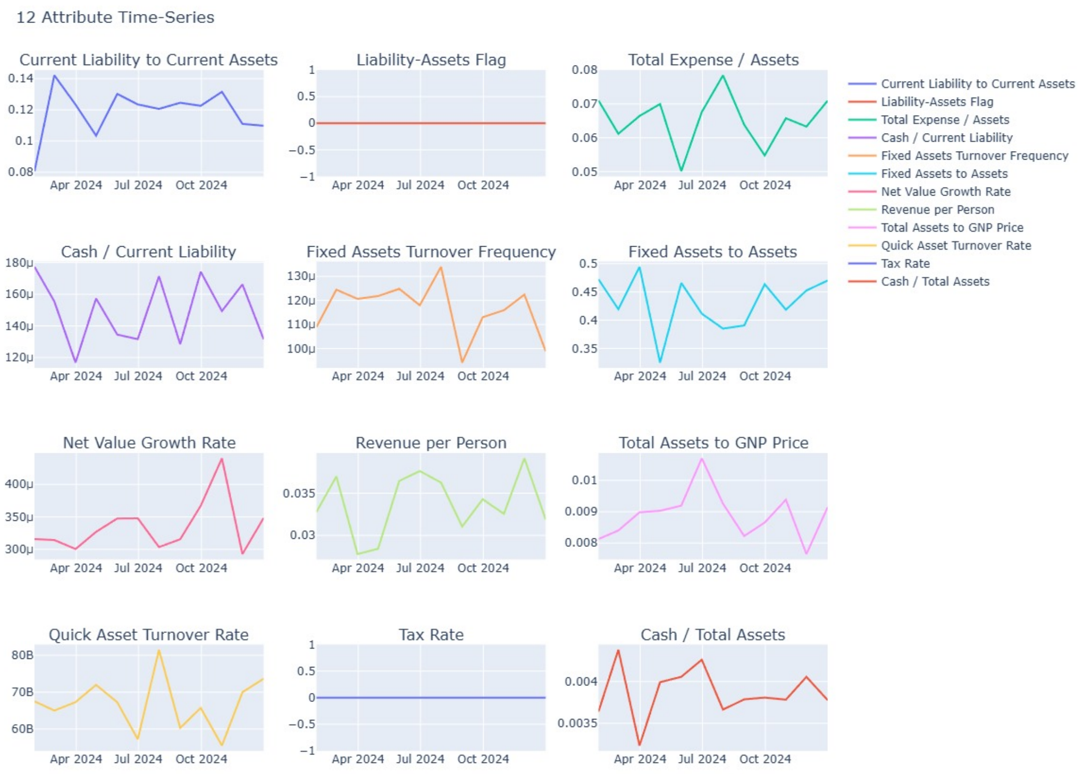
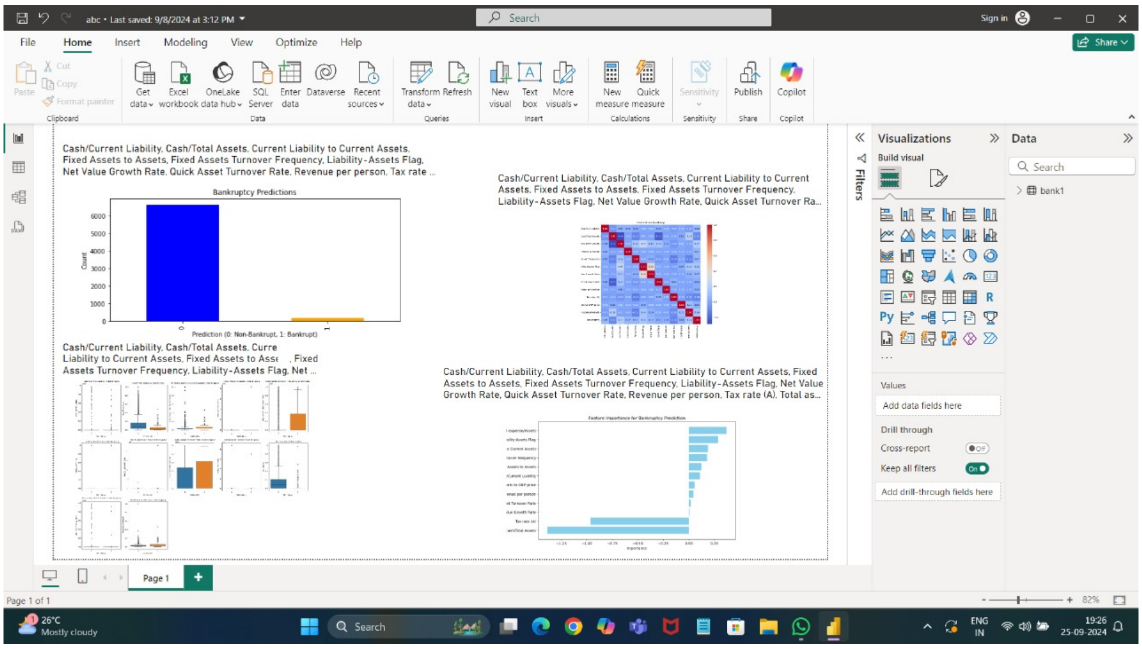


Fig 7.4 Time Series



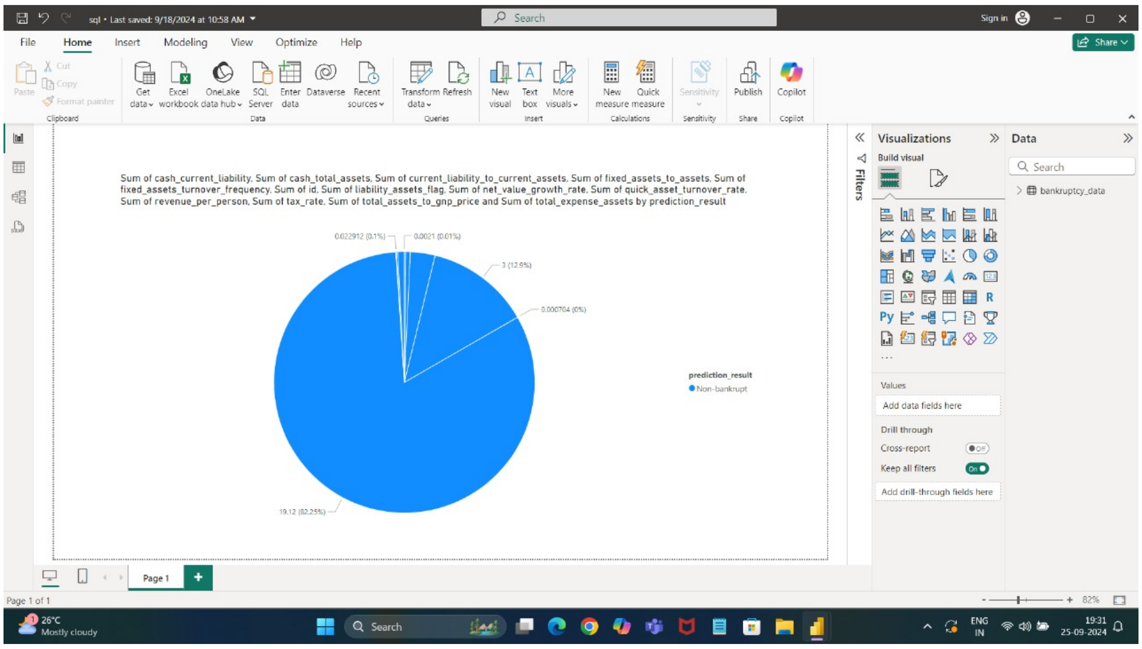
Cash/Current Liability, Cash/Total Assets, Current Liability to Current Assets, Fixed Assets to Assets, Fixed Assets Turnover Frequency, Liability-Assets Flag, Net Value Growth Rate, Quick Asset Turnover Rate, Revenue per person, Tax rate ...

Cash/Current Liability, Cash/Total Assets, Current Liability to Current Assets, Fixed Assets to Assets, Fixed Assets Turnover Frequency, Liability-Assets Flag, Net Value Growth Rate, Quick Asset Turnover Ra...

Cash/Current Liability, Cash/Total Assets, Curre Liability to Current Assets, Fixed Assets to Asse , Fixed Assets Turnover Frequency, Liability-Assets Flag, Net ...

Cash/Current Liability, Cash/Total Assets, Current Liability to Current Assets, Fixed Assets to Assets, Fixed Assets Turnover Frequency, Liability-Assets Flag, Net Value Growth Rate, Quick Asset Turnover Rate, Revenue per person, Tax rate (A), Total as...

Fig 7.5 Power bi



Sum of cash_current_liability, Sum of cash_total_assets, Sum of current_liability_to_current_assets, Sum of fixed_assets_to_assets, Sum of fixed_assets_turnover_frequency, Sum of id, Sum of liability_assets_flag, Sum of net_value_growth_rate, Sum of quick_asset_turnover_rate, Sum of revenue_per_person, Sum of tax_rate, Sum of total_assets_to_gnp_price and Sum of total_expense_assets by prediction_result

Fig 7.6 Power bi sql

8. CONCLUSION

8.1. CONCLUSION

The bankruptcy prediction project provides an effective solution for predicting financial distress using a logistic regression model. The integration with Flask allows for seamless data processing and model prediction, while Power BI and Tableau enhance data visualization and reporting capabilities. This project aids stakeholders like investors and policymakers by providing early warning signs of potential bankruptcies, enabling proactive decision-making and mitigating financial risk .Throughout the project, testing—ranging from unit and integration to performance and security—ensures the system's robustness, scalability, and security. By leveraging data-driven insights, the project demonstrates how predictive analytics can improve the stability of financial systems and reduce economic losses.

8.2. FURTHER ENHANCEMENTS

1. **Model Improvement**:

- **Incorporate More Advanced Models**: Explore more sophisticated algorithms like Random Forest, XGBoost, or neural networks to improve prediction accuracy.
- **Hyperparameter Tuning**: Use techniques like grid search or random search to fine-tune model parameters for better performance.

2. Real-Time Data Integration:

- **Streaming Data**: Incorporate real-time financial data using tools like Apache Kafka to enable dynamic and up-to-date bankruptcy predictions.
- **Automated Model Retraining**: Implement an automatic retraining pipeline that updates the model as new financial data becomes available.

3. **Enhanced Visualization**:

- **Interactive Dashboards**: Build more interactive and user-friendly dashboards with Power BI/Tableau, enabling stakeholders to drill down into specific data points or trends.
- **Predictive Reports**: Allow automatic generation and scheduling of predictive reports for different stakeholders.

4. **Scalability and Deployment**:

- **Cloud Deployment**: Scale the application using cloud platforms (AWS, Azure) to handle larger datasets and more users.
- **Microservices Architecture**: Break down the system into microservices to improve scalability and maintainability.

5. **Advanced Security Measures**:

- **Encryption**: Implement data encryption to protect sensitive financial data.
- **Role-Based Access Control**: Introduce more granular role-based access for different user groups to ensure secure data management.

9. BIBLIOGRAPHY

9.1. REFERENCES

1. **Plotly Documentation**. (n.d.). *Plotly: A High-Level API for Data Visualization*. Retrieved from https://plotly.com/python/

-The official documentation for Plotly, which can help you understand how to create interactive visualizations for your project.

2. **Flask Documentation**. (n.d.). *Flask: The Python micro framework for building web applications*. Retrieved from https://flask.palletsprojects.com/

-This documentation provides comprehensive guides on how to implement Flask applications.

3. **Scikit-learn Documentation**. (n.d.). *Machine Learning in Python*. Retrieved from https://scikit-learn.org/stable/

   -The official documentation for Scikit-learn, which includes details on various machine learning algorithms and their implementations

10. APPENDICES

Appendix A: Project Environment Setup

1. Software Requirements:

- Operating System: Windows/Linux/MacOS
- Programming Language: Python 3.x
- Framework: Flask
- Libraries:
    - scikit-learn: Logistic regression, data preprocessing
    - pandas: Data manipulation
    - numpy: Numerical operations
    - matplotlib and seaborn: Data visualization
    - Flask: Web framework
    - Power BI / Tableau: Data visualization tools

2. Installation Instructions:

- Install Python: https://www.python.org/downloads/
- Install required libraries:

3. System Configuration:

- Minimum hardware requirements: 4GB RAM, 2GHz Processor
- Recommended environment: Conda or Virtualenv for package management.

Appendix B: Dataset Description

1. Data Source:

- Source of financial data used (e.g., public datasets, proprietary datasets, or collected from specific organizations).

2. Dataset Features:

- Financial Ratios:
    - Liquidity ratios, debt-to-equity ratios, profitability ratios, etc.
    - Detailed descriptions of each feature and its relevance to bankruptcy prediction.
- Target Variable:
    - Binary classification target (1 = Bankruptcy, 0 = Non-bankruptcy).
- Time Period:
    - Years covered by the dataset (e.g., 2010-2020).

3. Preprocessing Steps:

- Handling missing values.
- Feature scaling (standardization or normalization).
- Undersampling/oversampling methods for handling class imbalance.

Appendix C: Model Details

1. Evaluation Metrics:

- Accuracy: Overall correctness of the model.
- Precision, Recall, F1-Score: Performance metrics in handling imbalanced datasets.
- Confusion Matrix: Example output showing true positives, false positives, true negatives, and false negatives.
- ROC Curve & AUC Score: Visualization of the model's ability to distinguish between classes.

Appendix D: API Endpoints

1. API Documentation:

- POST /upload: Upload financial data for prediction.
    - Input: CSV file containing financial ratios.
    - Response: Success message and stored file path.
- POST /predict: Predict bankruptcy based on uploaded data.
    - Input: JSON format with financial ratios.
    - Response: Prediction result (e.g., probability of bankruptcy).

2. Sample API Calls:

- Example for testing API endpoints using curl or Postman:

Appendix E: Testing Details

1. Unit Test Example:

- Code snippet for testing the logistic regression model:

2. Integration Test Example:

- Example test for ensuring the data flows correctly from frontend to API and back:

Appendix F: Sample Outputs

1. Confusion Matrix:

- Visualization of the confusion matrix from model evaluation.

2. ROC Curve:

- Receiver Operating Characteristic curve demonstrating model performance.

3. Power BI / Tableau Visualization:

- Screenshot or description of how predictions are visualized using Power BI/Tableau.
- Example graphs showing bankruptcy risk for a sample set of companies.

Appendix G: References

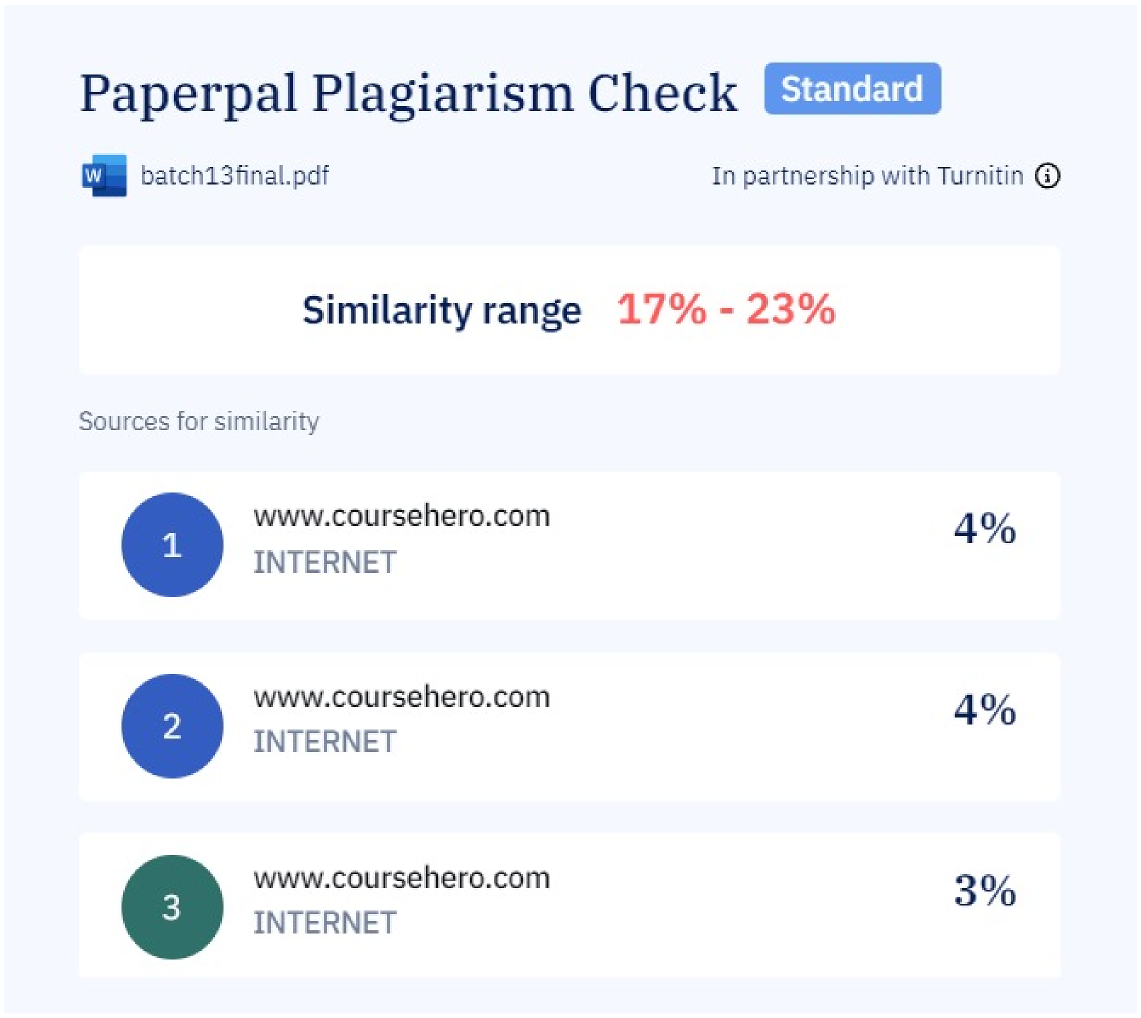
1. Research Papers:

- Bankruptcy prediction methodologies.
- Logistic regression use cases in finance.

2. Online Resources:

- Links to tutorials or resources on logistic regression, Flask, or Power BI integration.

11.PLAGARISM REPORT



Paperpal Plagiarism Check Standard

batch13final.pdf

In partnership with Turnitin

Similarity range 17% - 23%

Sources for similarity

1	www.coursehero.com INTERNET	4%
2	www.coursehero.com INTERNET	4%
3	www.coursehero.com INTERNET	3%