

Enhancing Credit Risk Assessment through Advanced Machine Learning Techniques

CAPSTONE PROJECT PHASE-II

Phase – II Review 1 Report

Submitted by Group No 244

Sl. No.	Register Number	Name
1.	21BCE10014	Aryan Rai
2.	21BCE10443	Harshit Varshney
3.	21BCE11188	Adarsh Kanungo
4.	21BCE10470	Giya Ambasta
5.	21BCE10503	Saumadipta Chatterjee

in partial fulfillment of the requirements for the degree of

Bachlore of Engineering and Technology



VIT[®]
BHOPAL
www.vitbhopal.ac.in

VIT Bhopal University
Bhopal
Madhyapradesh

January, 2025



Bonafide Certificate

Certified that this project report titled **“Enhancing Credit Risk Assessment through Advanced Machine Learning Techniques”** is the bonafide work of Group No 244 “21BCE10014 Aryan Rai, 21BCE10443 Harshit Varshney, 21BCE11188 Adarsh Kanungo, 21BCE10470 Giya Ambasta, 21BCE10503 Saumadipta Chatterjee” who carried out the project work under my supervision.

This project report (DSN4095-Capstone Project Phase-II) is submitted for the Project Viva-Voce examination held on 8th, January, 2025

Supervisor
Dr. Preetam Suman

Reviewer 1
Dr. Daood Saleem

Reviewer 2
Dr Sandeep Monga

Contents

CHAPTER 1: INTRODUCTION	2
1.1 Motivation.....	2
1.2 Objective	3
CHAPTER 2: EXISTING WORK / LITERATURE REVIEW	4
2.1 Traditional Approaches to Credit Risk Modeling	4
2.2 Machine Learning Approaches.....	4
2.3 Integration of External Data Sources	5
2.4 Challenges in Credit Risk Modeling.....	6
2.5 Gaps Addressed by our Project.....	6
2.6 Comparison of existing approaches and our model	7
2.7 Conclusion.....	7
CHAPTER 3: FRONT END, BACKEND, AND SYSTEM REQUIREMENT	8
3.1 Front End & Back End (Technology Stack Used)	8
3.2 System Requirements	9
CHAPTER 4: METHODOLOGY	10
4.1 System Design / Architecture.....	10
4.2 Working Principle.....	12
4.3 Results and Discussion	13
Individual Contribution by Members	15
CHAPTER 5: CONCLUSION	19
5.1 Project Screenshot	20
5.2 Summary:	28
References and Publications	29

CHAPTER 1: INTRODUCTION

The financial sector has undergone a massive transformation with the integration of technology, and credit risk assessment has emerged as a critical area for leveraging machine learning. Traditional methods relied on static rule-based systems and human intervention, often leading to inefficiencies and increased risks. As data availability grows, machine learning techniques provide opportunities for more accurate, automated, and scalable solutions.

In this project, we aim to develop a data-driven credit risk prediction model by analyzing customer data and external credit reports. By leveraging advanced machine learning techniques, the project seeks to provide financial institutions with a tool to make informed lending decisions, reduce non-performing assets (NPAs), and improve operational efficiency.

This report documents the progression of the project, from data preparation and exploratory analysis to feature engineering, model building, and evaluation. The following chapters outline the methodology, results, and learnings gained from this endeavor, showcasing the potential of machine learning in solving real-world financial challenges.

1.1 Motivation

The banking industry plays a crucial role in economic stability by offering financial services such as loans, credit, and investments. However, with an increase in loan demand, managing credit risk has become more challenging. Non-Performing Assets (NPA) are a significant threat to a bank's profitability, often resulting in financial losses and liquidity issues. As more individuals and businesses seek loans, banks need a more reliable method to evaluate the creditworthiness of applicants to minimize defaults and ensure sustainable lending practices.

Traditional credit assessment methods, while effective to an extent, rely heavily on manual processes and subjective judgment, which may lead to inconsistent decisions. Additionally, these methods are often slow and inefficient, unable to keep pace with the high volume of loan applications and the complex financial behaviour of customers.

This project is motivated by the need to develop an efficient, accurate, and scalable system that utilizes machine learning to predict loan defaults. By leveraging internal bank data and external sources such as CIBIL, we aim to build a model that not only streamlines the decision-making process but also enhances the accuracy of predicting whether a loan should be approved or denied. This solution could significantly reduce the risk of NPAs, helping banks improve their financial health while maintaining a responsible lending policy.

1.2 Objective

The primary objective of this project is to develop a machine learning-based credit risk model that can accurately predict the likelihood of a loan default. By analysing both internal bank data and external credit bureau data (CIBIL), the model will help banks in making informed lending decisions. This solution aims to streamline the loan approval process by reducing manual effort, improving prediction accuracy, and minimizing credit risk.

Specifically, the objectives of the project are as follows:

1. **Data Integration:** Combine internal banking datasets with external credit scores and related information to create a comprehensive dataset for analysis.
2. **Data Cleaning and Feature Engineering:** Handle missing data, outliers, and feature transformation to prepare the data for modeling. This includes removing null values and reducing multicollinearity among features.
3. **Model Development:** Build and train machine learning models using classification algorithms to predict the likelihood of loan approval or rejection based on various financial and demographic factors.
4. **Model Evaluation:** Evaluate the performance of the model using metrics like accuracy, precision, recall, and F1-score to ensure that it meets the required performance standards.
5. **Deployment and Integration:** Create a deployable version of the model for real-time credit decision-making in the bank's loan processing system.
6. **Reduction of NPAs:** Ultimately, the model seeks to reduce the number of non-performing assets (NPAs) by identifying high-risk applicants before loans are granted.

These objectives are aimed at enhancing the bank's ability to mitigate credit risk, improve profitability, and ensure more efficient loan processing.

CHAPTER 2: EXISTING WORK / LITERATURE REVIEW

Credit risk modeling is a crucial component of the financial industry, particularly for banking institutions that rely on accurate assessments of borrower creditworthiness to minimize the risk of loan defaults. Over the years, significant research and development have been conducted in this field, evolving from traditional statistical models to sophisticated machine learning techniques. This chapter reviews the existing work related to credit risk modeling, highlighting various methods, approaches, and their impact on the banking sector.

2.1 Traditional Approaches to Credit Risk Modeling

Historically, banks have relied on traditional statistical techniques, such as logistic regression and linear discriminant analysis (LDA), to predict credit risk. These models use a limited set of variables such as income, credit history, employment status, and debt-to-income ratios to assess whether a borrower is likely to default.

- **Credit Scoring Systems:** Credit scoring systems, such as FICO and CIBIL, have been widely used to evaluate an individual's creditworthiness. These scores are derived from an individual's financial history, including payment behavior, outstanding debts, and credit inquiries. The scores categorize borrowers into risk levels, allowing banks to decide whether to approve or deny credit. While these systems are effective in evaluating basic risk, they lack the ability to analyze complex interactions between features.
- **Logistic Regression:** One of the earliest and most common methods used for binary classification in credit risk modeling is logistic regression. This technique models the probability of default by evaluating various predictor variables. Although logistic regression is easy to interpret and implement, it lacks the ability to capture complex relationships between variables.
- **Linear Discriminant Analysis (LDA):** LDA is another statistical technique widely used for credit scoring, which classifies applicants into categories of "default" or "non-default" by creating a linear combination of predictor variables. While effective for well-separated classes, LDA struggles with non-linear relationships between the features.
- **Discriminant Analysis:** Discriminant analysis has been employed for classifying borrowers into default and non-default groups. Similar to logistic regression, it works well for smaller datasets but struggles with high-dimensional data and assumes normality of predictor variables, which may not hold true in real-world scenarios.

These traditional methods have the advantage of being simple and interpretable but may not perform well with large, complex datasets, which has led to the adoption of more advanced machine learning techniques.

2.2 Machine Learning Approaches

With the advent of big data and improved computational power, machine learning models have gained prominence in credit risk prediction. These models, including decision trees, random

forests, gradient boosting machines, and deep learning networks, offer better accuracy and flexibility in dealing with complex datasets. Some of the most notable approaches are:

- **Decision Trees:** Decision trees are simple yet powerful algorithms that split data into branches based on feature values. They are easy to interpret and provide insights into the decision-making process. However, standalone decision trees are prone to overfitting, which limits their generalizability.
- **Random Forest:** Random Forest, an ensemble method, builds multiple decision trees and aggregates their predictions to improve accuracy and robustness. It reduces overfitting by introducing randomness in feature selection and data sampling. Studies have shown that Random Forest performs well in predicting default probabilities.
- **Gradient Boosting Machines (GBM):** GBM models like XGBoost and LightGBM have become popular in recent years for their high predictive power. These models build on decision trees by iteratively improving predictions based on previous errors. They are known for their flexibility and efficiency in handling large, imbalanced datasets commonly found in banking.
- **Support Vector Machines (SVM):** SVMs are another class of models that can separate data points using hyperplanes. Although effective for certain types of data, they require extensive feature scaling and can be computationally intensive for large datasets.
- **Deep Learning:** In more recent work, neural networks and deep learning models have been explored for credit risk modeling. These models can automatically extract features from the data and learn complex patterns, making them powerful for large datasets with intricate relationships between variables. However, they can be difficult to interpret, which is a significant concern in the banking sector where transparency is required.
- **Neural Networks:** Neural networks have been explored for credit risk modeling due to their ability to capture nonlinear relationships. Deep learning models, though less interpretable, have shown promise in analyzing unstructured data such as transaction histories and customer reviews.

2.3 Integration of External Data Sources

In addition to internal banking data, external data sources such as credit bureau information (e.g., CIBIL) and alternative data from social media, utility bills, and telecom data have become popular in recent years. These additional data points offer a more comprehensive view of a borrower's creditworthiness.

- **Credit Bureau Scores:** External credit scores like the CIBIL score in India are frequently used alongside internal data for credit risk assessment. These scores provide a standardized view of the borrower's credit history, payment behavior, and outstanding loans, significantly improving predictive power.
- **Alternative Data:** The use of alternative data in credit scoring is gaining momentum. This includes information like mobile phone usage, online transaction data, and social media activity. While not yet as widespread, alternative data has been shown to improve predictions for applicants with limited traditional credit histories (commonly known as "thin-file" customers).

2.4 Challenges in Credit Risk Modeling

Despite the advancements in machine learning, there are still several challenges in credit risk modeling:

- **Data Quality:** Missing, incomplete, or incorrect data can skew the results of models, leading to inaccurate predictions. Handling missing values, such as the -99999 placeholder in our dataset, is critical for ensuring model reliability.
- **Imbalanced Data:** Credit risk datasets are often highly imbalanced, with far fewer defaults than non-defaults. This can lead to biased models that are more likely to predict non-defaults, as the algorithm tends to favor the majority class. Techniques such as oversampling, undersampling, or using specific evaluation metrics like AUC-ROC are commonly used to address this issue.
- **Interpretability vs. Complexity:** While machine learning models like random forests and gradient boosting machines offer higher accuracy, they often lack interpretability. In the banking sector, regulatory bodies require transparency in how credit decisions are made. Balancing accuracy and interpretability remains a key challenge in the deployment of these models.

2.5 Gaps Addressed by our Project

While existing studies have successfully applied machine learning techniques to credit risk modeling, several gaps remain:

1. **Feature Engineering:** Many studies overlook the importance of feature selection and engineering. Our project employs statistical methods such as Chi-Square tests, Variance Inflation Factor (VIF), and ANOVA to refine input features, ensuring higher model interpretability and accuracy.
2. **Integration of Internal and External Data:** Few models leverage both internal bank data and external credit bureau data. By combining these datasets, our approach provides a more comprehensive view of borrower risk.
3. **Class Imbalance Handling:** Imbalanced datasets are a common challenge in credit risk modeling. Our project uses techniques such as ensemble methods and metrics evaluation (e.g., precision, recall, F1-score) to ensure balanced performance across all classes.
4. **Multi-Class Classification:** While most research focuses on binary classification (default vs. non-default), our project extends the scope to multi-class classification, predicting loan approval priorities (P1, P2, P3, P4).

2.6 Comparison of existing approaches and our model

Feature	Traditional Methods	Machine Learning (Existing)	Our Project
Data Handling	Small, structured datasets	Large datasets	Combined internal and external data
Feature Selection	Manual	Automated	Statistical techniques (VIF, ANOVA)
Modeling Approach	Logistic Regression	Ensemble Models (e.g., RF)	Random Forest, XGBoost, Decision Tree
Performance on Imbalanced Data	Poor	Moderate	Addressed with advanced techniques
Classification Type	Binary	Binary	Multi-class

2.7 Conclusion

The field of credit risk modeling has evolved significantly, moving from traditional statistical methods to more sophisticated machine learning algorithms. With the integration of external credit bureau data and alternative data sources, modern credit risk models offer greater predictive power and flexibility. However, challenges such as data quality, model interpretability, and handling imbalanced datasets remain. This project builds on these existing methods, aiming to develop a machine learning-based credit risk model that addresses these challenges while delivering accurate, actionable predictions for loan approval decisions.

CHAPTER 3: FRONT END, BACKEND, AND SYSTEM REQUIREMENT

3.1 Front End & Back End (Technology Stack Used)

In this project, a machine learning-based credit risk model is developed to predict loan approval decisions. The technology stack used in this project involves the following components:

Front End: The front-end serves as the interface for users to interact with the model. For this project, the front end could be either a **web-based interface** or a **desktop application**, depending on the deployment method.

- **Web Interface:** If deployed on the cloud, the front end can be built using **HTML5**, **CSS3**, and **JavaScript** for a responsive user interface. Libraries such as **React.js** or **Vue.js** could be used for building dynamic and interactive components. This allows users to input applicant data, receive credit approval predictions, and view model results in real-time.
- **Desktop Application:** In case of a desktop deployment, a **Python-based executable (EXE)** can be created using libraries such as **PyQt** or **Tkinter** for designing the user interface. This option provides a standalone tool for bank officers to use the model on their computers without requiring internet connectivity.

Back End: The back-end is responsible for handling the logic, managing the data, and interacting with the machine learning model. The back end for this project involves the following components:

- **Python (Flask/Django):** The core of the back-end will be written in Python, utilizing frameworks like **Flask** or **Django** to handle the logic and API requests. Flask is lightweight and suitable for small projects, while Django offers a more comprehensive framework for larger systems.
- **Machine Learning Libraries:** Libraries like **Scikit-learn**, **XGBoost**, **LightGBM**, and **Pandas** are used for training and deploying the machine learning model. **Pickle** or **Joblib** will be used for model serialization, allowing the trained model to be saved and loaded for predictions.

Deployment Options:

- **Cloud Deployment:** The application can be deployed on cloud platforms like **AWS** or **Microsoft Azure** using services such as **AWS Lambda** or **Azure App Service** for model hosting. This will enable real-time predictions, making the system scalable and accessible from any location.
- **Executable Deployment:** Alternatively, an **executable EXE** file can be generated using **PyInstaller** or **cx_Freeze**. This option is ideal if the system is intended for internal use within the bank, where no external connectivity is required.

3.2 System Requirements

Hardware Requirements:

- **Processor:** Minimum 2.4 GHz, multi-core processor (10th Gen Intel i5 or higher recommended).
- **RAM:** At least 8 GB of RAM is required for efficient model training and deployment.
- **Storage:** A minimum of 10 GB of free disk space is required to store the datasets, libraries, and model files.
- **GPU:** A dedicated Graphics Processing Unit (GPU) like **NVIDIA Tesla** or **GeForce RTX** is optional but recommended for faster training, especially for complex models or deep learning applications.

Software Requirements:

- **Operating System:** Windows 10/11, Linux (Ubuntu), or macOS.
- **Python Version:** Python 3.8 or above.
- **Machine Learning Libraries:** Scikit-learn, XGBoost, LightGBM, Pandas, NumPy, and Matplotlib.
- **Development Tools:**
 - **Jupyter Notebook** or **Spyder** for writing and testing code.
 - **Anaconda** for managing packages and dependencies.
- **IDE:** Spyder 5.1.5 for development purposes.
- **Web Framework:** Flask or Django for creating the back end.
- **Database:** SQLite or MySQL for data storage and retrieval.
- **Deployment Tools:**
 - **AWS CLI** or **Azure CLI** for cloud deployment.
 - **PyInstaller** or **cx_Freeze** for packaging the application into an executable EXE for desktop deployment.

CHAPTER 4: METHODOLOGY

In this chapter, we discuss the design and architecture of the credit risk prediction system, the working principle behind the machine learning models used, and the results and analysis of the model's performance. This methodology demonstrates the step-by-step process of how the system was built and the key findings from its evaluation.

4.1 System Design / Architecture

The system architecture for the credit risk prediction model is designed with scalability, flexibility, and efficiency in mind. The model predicts whether a customer's loan application should be approved based on features extracted from internal bank data (Case Study 1) and external CIBIL data (Case Study 2).

The system is structured into the following components:

1. Data Collection:

The system gathers data from both internal sources (bank's product holdings and loan data) and external credit bureau reports (CIBIL scores and reports). This raw data is stored in a secure database (such as MySQL or SQLite) for further processing.

2. Data Preprocessing and Feature Engineering:

This stage is crucial for transforming the raw data into a format suitable for machine learning. Various steps, such as handling missing values (like replacing -99999 values), normalizing numerical features, encoding categorical variables, and combining the datasets on `PROSPECTID`, are applied. Additionally, exploratory data analysis (EDA) is performed to understand patterns in the data.

3. Model Training:

The pre-processed data is used to train machine learning models. Various algorithms such as **Logistic Regression**, **Random Forest**, and **XGBoost** are employed. Model training involves splitting the data into training and testing sets, using cross-validation to prevent overfitting, and tuning hyperparameters to improve model performance.

4. Model Evaluation:

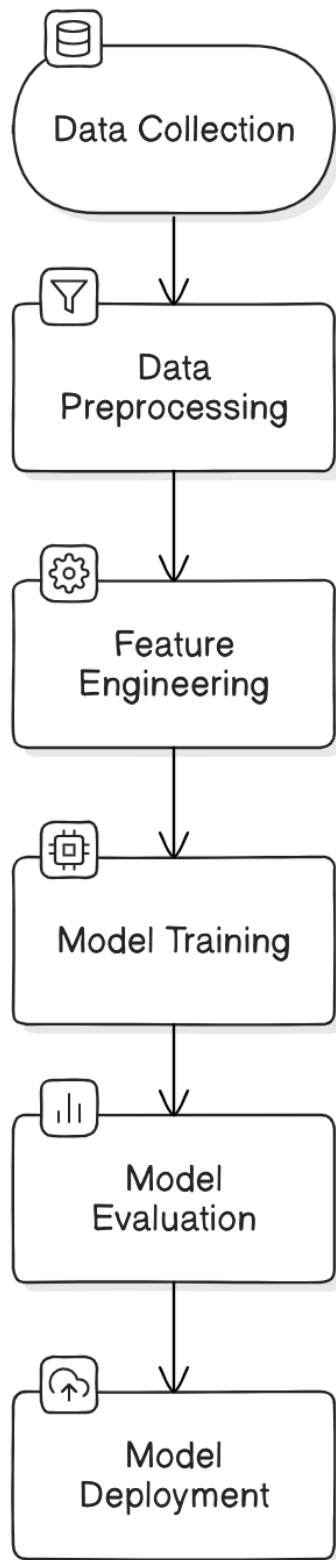
Once the models are trained, they are evaluated based on their predictive performance using metrics like **Accuracy**, **Precision**, **Recall**, **F1-Score**, and **ROC-AUC Score**. Feature importance analysis is conducted to determine the most influential factors contributing to loan approval or denial.

5. Model Deployment:

The model is deployed on a cloud platform (like AWS or Azure) or converted into an executable (EXE) file for local deployment. The system is designed to take user input (e.g., customer information), run the prediction model, and return the result (approve/deny) in real-time.

The architecture of the system is visualized in the diagram below.

System Architecture Diagram:



4.2 Working Principle

The working principle of the project is divided into multiple stages:

4.2.1 Data Collection and Integration

1. **Datasets Used:**
 - **Internal Product Dataset (Case Study 1):** Contains details of customer loan product holdings.
 - **External CIBIL Dataset (Case Study 2):** Provides credit scores and delinquency information.
2. **Integration:** The datasets are merged using the unique identifier PROSPECTID, ensuring a comprehensive view of customer behavior.

4.2.2 Data Preprocessing

1. **Handling Missing Values:** Null values represented as -99999 are identified and treated based on their frequency.
2. **Outlier Detection and Removal:** Univariate and multivariate analysis techniques are used to remove extreme values.
3. **Data Standardization:** Ensures uniform scaling of features for better model performance.

4.2.3 Feature Engineering

1. **Statistical Methods:**
 - **ANOVA Test:** Identifies significant numerical features correlated with the target variable.
 - **Chi-Square Test:** Assesses the association between categorical features and the target.
 - **Variance Inflation Factor (VIF):** Removes multicollinearity among numerical features.
2. **Feature Encoding:**
 - Categorical features such as MARITALSTATUS and GENDER are converted using one-hot encoding.
 - Ordinal features like EDUCATION are encoded based on logical ordering (e.g., SSC < 12TH < GRADUATE).

4.2.4 Model Development

1. **Random Forest Classifier:**
 - An ensemble model that aggregates predictions from multiple decision trees.
 - Achieves an accuracy of 76% on the test dataset.
2. **XGBoost:**
 - Optimized gradient boosting algorithm that excels in handling imbalanced datasets.
 - Achieves the highest accuracy of 78% with improved F1-scores for classes P1, P2, and P4.
3. **Decision Tree Classifier:**
 - A simple, interpretable model achieving an accuracy of 71%, serving as a baseline.

4.2.5 Model Evaluation Metrics

- **Accuracy:** Measures overall correctness of predictions.
- **Precision, Recall, F1-Score:** Evaluates class-wise performance, especially for imbalanced classes.
- **Confusion Matrix:** Visualizes model's prediction capabilities for each class.

4.3 Results and Discussion

4.3.1 Performance Metrics

The performance of each model is summarized in *Table 1* below:

Model	Accuracy	Precision (P1)	Recall (P1)	F1-Score (P1)	Precision (P3)	Recall (P3)	F1-Score (P3)
Random Forest	76.37%	0.837	0.704	0.765	0.442	0.211	0.286
XGBoost	78.00%	0.824	0.761	0.791	0.476	0.309	0.375
Decision Tree	71.00%	0.721	0.727	0.724	0.346	0.330	0.338

Table 1: Model Performance Metrics

4.3.2 Insights from Results

1. **Best Model:** XGBoost outperforms other models with the highest accuracy (78%) and balanced class-wise performance.
2. **Class Imbalance Impact:** The lower performance on class P3 highlights the need for oversampling or alternative class imbalance techniques.
3. **Feature Importance:** Features like CIBIL Score, Time Since Recent Payment, and Total Missed Payments were found to have the highest influence on predictions.

4.3.3 Visual Representation

1. **Confusion Matrix:** A confusion matrix for the XGBoost model is shown below:

Actual / Predicted	P1	P2	P3	P4
P1	1500	200	50	100
P2	100	1700	80	120
P3	50	70	250	60
P4	80	100	60	900

2. **Feature Importance Plot:** The top 5 most important features for XGBoost are:

- CIBIL Score
- Time Since Recent Payment
- Total Missed Payments
- Credit Card Utilization
- Personal Loan Enquiries (Last 6 Months)

Individual Contribution by Members

The following section provides a detailed description of the contributions made by each team member throughout the project. Each member played a vital role in ensuring the success of the project, with responsibilities ranging from data handling to report writing and presentation preparation.

1) Aryan Rai (21BCE10014) - Feature Engineering, EDA, and Machine Learning Model Implementation (Main Report)

Aryan Rai played a pivotal role in preparing the dataset for machine learning by performing **Exploratory Data Analysis (EDA)** and **Feature Engineering** to enhance the predictive performance of the models. Key contributions include:

1. Feature Engineering:

- Conducted thorough analysis to identify significant features using statistical techniques such as:
 - **Chi-Square Test:** To evaluate the dependency of categorical features on the target variable.
 - **ANOVA Test:** To analyze numerical features for their correlation with loan approval classes.
 - **Variance Inflation Factor (VIF):** To remove multicollinearity among numerical features, ensuring independent predictors.
- Encoded categorical variables using one-hot encoding and ordinal encoding (e.g., educational levels such as SSC, 12TH, GRADUATE).
- Selected the most impactful features using systematic feature reduction techniques, narrowing the dataset to relevant variables.

2. EDA (Exploratory Data Analysis):

- Performed in-depth data visualization using Matplotlib and Seaborn to uncover trends and anomalies within the dataset.
- Analyzed relationships between features such as "Total Missed Payments" and "CIBIL Scores" to understand their impact on credit risk.
- Addressed missing values and outliers without compromising data integrity, adhering to financial industry standards.

3. Assistance in Machine Learning Models:

- Collaborated in the implementation of models such as Random Forest, XGBoost, and Decision Tree.
- Helped optimize hyperparameters to achieve better performance and interpretability.
- Documented all processes and findings in the main project report, ensuring clarity and structure.

2) Harshit Varshney 21BCE10443 - Main Report and Machine Learning Model Selection & Implementation

Harshit Varshney contributed significantly to the **machine learning pipeline**, focusing on model selection, training, and performance comparison. Key contributions include:

1. **Model Selection:**
 - Evaluated multiple machine learning algorithms such as Random Forest, XGBoost, and Decision Tree for their suitability in multi-class classification.
 - Chose the optimal models based on accuracy, precision, recall, and F1-score metrics.
2. **Implementation:**
 - Trained, tested, and validated models using an 80-20 train-test split.
 - Used Scikit-learn and XGBoost libraries for implementation, ensuring robust and scalable solutions.
 - Focused on handling class imbalances (e.g., minority class P3) using ensemble techniques and performance adjustments.
3. **Comparative Analysis:**
 - Compared the models in terms of:
 - Overall accuracy: XGBoost achieved the highest accuracy (78%).
 - Class-wise metrics: Identified strengths and weaknesses of each model (e.g., lower F1-score for class P3 in Random Forest).
 - Documented the trade-offs between accuracy, interpretability, and computational cost for deployment planning.
4. **Report Contribution:**
 - Co-authored the main report with detailed sections on machine learning methodologies and results discussion.
 - Prepared visual aids, including confusion matrices and feature importance graphs, for better interpretation.

3) Adarsh Kanungo 21BCE11188 - Dataset Analysis and Preparation

Adarsh Kanungo ensured the dataset was clean, consistent, and ready for analysis by focusing on **data preparation and preprocessing**. Key contributions include:

1. **Handling Missing Values:**
 - Identified missing data patterns in both datasets (internal bank data and CIBIL data).
 - Removed rows where significant missing values were present (e.g., age of oldest trade line with -99999 as placeholder).
 - Dropped columns with over 10,000 missing values to maintain data integrity.
2. **Dataframe Merging:**
 - Merged internal and external datasets using the unique PROSPECTID column.
 - Ensured consistency and eliminated nulls during the merge process.
3. **Dataset Cleaning:**

- Addressed outliers using statistical techniques.
 - Standardized numerical columns to ensure compatibility across features (e.g., normalizing "Time Since Recent Payment").
4. **Report Contribution:**
- Documented the entire data preprocessing pipeline in the report.
 - Highlighted challenges faced during data cleaning and the solutions implemented.

4) Giya Ambasta 21BCE10470 - Research Study, Dataset Selection, and Report & PPT Support

Giya Ambasta contributed extensively to the **planning phase of the project** and ensured the datasets used were relevant and comprehensive. Key contributions include:

1. **Research Study:**
 - Conducted detailed research on credit risk assessment methodologies, providing the team with insights into traditional and modern approaches.
 - Studied financial concepts such as GNPA, NNPA, and DPD to align project objectives with banking standards.
2. **Dataset Selection:**
 - Analyzed multiple publicly available datasets and evaluated their relevance, completeness, and applicability to the project.
 - Chose the final datasets based on quality and their ability to support multi-class classification.
3. **Report and PPT Preparation:**
 - Assisted in writing key sections of the report, particularly the introduction and existing work.
 - Designed and structured the project presentation for Phase II Review.

5) Saumadipta Chatterjee 21BCE10503 - Research Study and PPT Contribution

Samyadepta Chatterjee provided valuable insights during the research phase and supported the team in understanding the datasets and their use. Key contributions include:

1. **Research Study:**
 - Researched the use and interpretation of external credit bureau data (e.g., delinquency levels and CIBIL scores).
 - Studied how credit risk classification aligns with real-world banking practices, ensuring the project's relevance.
2. **Dataset Interpretation:**
 - Helped the team understand the significance of key features such as "Missed Payments" and "Credit Card Utilization."
 - Provided feedback on the feature engineering process to improve the model's effectiveness.
3. **PPT Contribution:**

- Assisted in preparing the Phase II Review presentation by creating slides summarizing research findings and model performance.
- Ensured the presentation was visually appealing and logically structured.

CHAPTER 5: CONCLUSION

The project aimed to design and implement a machine learning-based credit risk prediction system to assist banks in making informed lending decisions. By combining internal bank datasets and external credit bureau data, the system successfully leverages advanced data processing techniques, feature engineering, and machine learning algorithms to classify loan approval priorities.

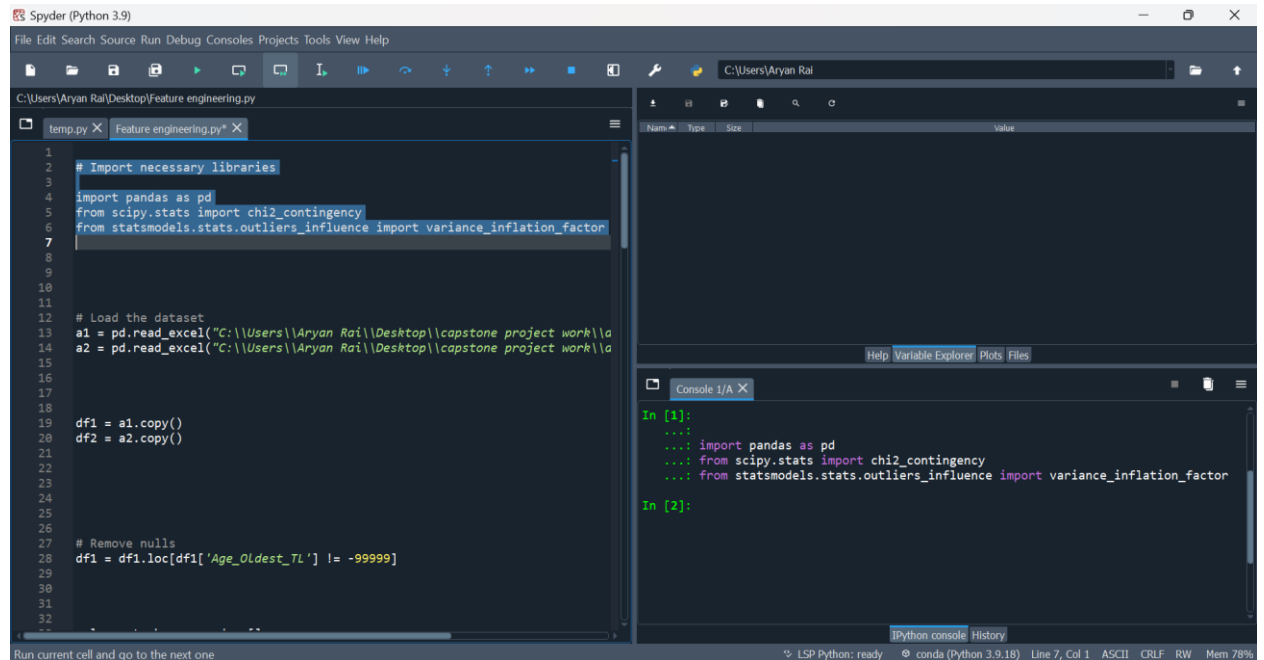
Code Workflow

The workflow of the code can be summarized in the following steps:

1. **Data Ingestion:**
 - Two datasets (internal and external) were merged using a unique identifier (PROSPECTID).
 - Missing values were identified and handled based on predefined rules (e.g., removing rows with -99999 values).
2. **Data Preprocessing:**
 - Outliers were removed to maintain data consistency.
 - Categorical features were encoded using techniques like one-hot encoding and ordinal encoding.
 - Numerical features were standardized, and multicollinearity was addressed using Variance Inflation Factor (VIF).
3. **Feature Selection:**
 - Statistical tests such as Chi-Square, ANOVA, and VIF were employed to identify significant features.
 - The final dataset consisted of features that were most relevant to predicting credit risk.
4. **Machine Learning Models:**
 - Three models (Random Forest, XGBoost, and Decision Tree) were implemented to predict loan approval categories (P1, P2, P3, P4).
 - Model evaluation metrics (accuracy, precision, recall, F1-score) were computed to compare their performance.
5. **Results Visualization:**
 - Confusion matrices and feature importance plots were generated to interpret model performance.
 - XGBoost emerged as the best-performing model with an accuracy of 78%.

5.1 Project Screenshot

The first step is to load the necessary Python libraries that help with data handling, statistical tests, and multicollinearity checks.



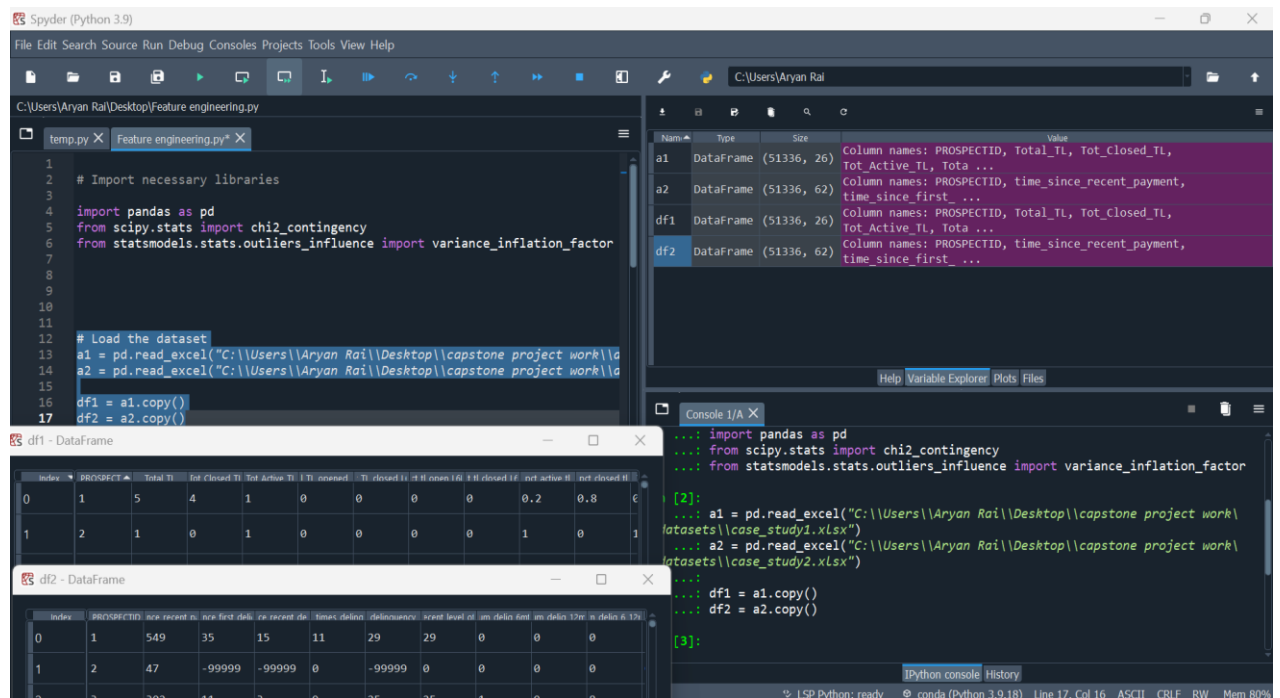
```
1 # Import necessary libraries
2
3 import pandas as pd
4 from scipy.stats import chi2_contingency
5 from statsmodels.stats.outliers_influence import variance_inflation_factor
6
7
8
9
10
11
12 # Load the dataset
13 a1 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\a
14 a2 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\a
15
16
17
18
19 df1 = a1.copy()
20 df2 = a2.copy()
21
22
23
24
25
26
27 # Remove nulls
28 df1 = df1.loc[df1['Age_OLdest_TL'] != -99999]
```

Console 1/A X

```
In [1]:
...: import pandas as pd
...: from scipy.stats import chi2_contingency
...: from statsmodels.stats.outliers_influence import variance_inflation_factor

In [2]:
```

Two Excel files, `case_study1.xlsx` (internal product data) and `case_study2.xlsx` (CIBIL data), are loaded into Python and copied into dataframes `df1` and `df2` for further analysis.



```
1 # Import necessary libraries
2
3 import pandas as pd
4 from scipy.stats import chi2_contingency
5 from statsmodels.stats.outliers_influence import variance_inflation_factor
6
7
8
9
10
11
12 # Load the dataset
13 a1 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\a
14 a2 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\a
15
16
17 df1 = a1.copy()
18 df2 = a2.copy()
```

Variable Explorer

Name	Type	Size	Value
a1	DataFrame	(51336, 26)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot ...
a2	DataFrame	(51336, 62)	Column names: PROSPECTID, time_since_recent_payment, time_since_first ...
df1	DataFrame	(51336, 26)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot ...
df2	DataFrame	(51336, 62)	Column names: PROSPECTID, time_since_recent_payment, time_since_first ...

Console 1/A X

```
[2]:
...: a1 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\
...: datasets\\case_study1.xlsx")
...: a2 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\
...: datasets\\case_study2.xlsx")

[3]:
...: df1 = a1.copy()
...: df2 = a2.copy()
```

df1 - DataFrame

index	PROSPECT	Total TL	Tot Closed TL	Tot Active TL	TL - renewed	TL - closed 11	TL - H - closed 14	TL - H - closed 16	int active H	int closed H
0	1	5	4	1	0	0	0	0	0.2	0.8
1	2	1	0	1	0	0	0	0	1	0

df2 - DataFrame

index	PROSPECTID	new recent cu	new first delin	cu recent del	times delinq	delinquency	Percent level of	am delin 6m	am delin 12m	in delin 6	in delin 12
0	1	549	35	15	11	29	29	0	0	0	0
1	2	47	-99999	-99999	0	-99999	0	0	0	0	0

The code identifies missing values (represented as -99999) in both datasets. In df1, rows where the oldest account age is -99999 are removed. In df2, columns with more than 10,000 missing values are dropped, and remaining rows with null values are also removed.

The top screenshot shows the Spyder Python IDE with the following code in the editor:

```

21
22
23
24 # Remove nulls
25 df1 = df1.loc[df1['Age_Oldest_TL'] != -99999]
26
27
28
29
30 columns_to_be_removed = []
31
32 for i in df2.columns:
33     if df2.loc[df2[i] == -99999].shape[0] > 10000:
34         columns_to_be_removed.append(i)
35
36
37 df2 = df2.drop(columns_to_be_removed, axis=1)
38
39
40
41
42 for i in df2.columns:
43     df2 = df2.loc[ df2[i] != -99999 ]
44
45
46
47
48
49
50 # Checking common column names
51 for i in list(df1.columns):
52     if i in list(df2.columns):
53         print(i)
54
55
56
57
58

```

The bottom screenshot shows the same IDE after execution. The code in the editor is:

```

26
27
28
29
30 columns_to_be_removed = []
31
32 for i in df2.columns:
33     if df2.loc[df2[i] == -99999].shape[0] > 10000:
34         columns_to_be_removed.append(i)
35
36
37 df2 = df2.drop(columns_to_be_removed, axis=1)
38
39
40
41
42 for i in df2.columns:
43     df2 = df2.loc[ df2[i] != -99999 ]
44
45
46
47
48
49
50 # Checking common column names
51 for i in list(df1.columns):
52     if i in list(df2.columns):
53         print(i)
54
55
56
57
58

```

The console output for the bottom screenshot is:

```

In [2]:
...: a1 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\
\\datasets\\case_study1.xlsx")
...: a2 = pd.read_excel("C:\\Users\\Aryan Rai\\Desktop\\capstone project work\\
\\datasets\\case_study2.xlsx")
...:
...: df1 = a1.copy()
...: df2 = a2.copy()

In [3]:
...: df1 = df1.loc[df1['Age_Oldest_TL'] != -99999]

In [4]:

```

The two datasets are merged based on a common column PROSPECTID. This ensures that we have a unified dataset with no missing values, ready for analysis.

```

46
47
48
49
50 # Checking common column names
51 for i in list(df1.columns):
52     if i in list(df2.columns):
53         print(i)
54
55
56
57 # Merge the two dataframes, inner join so that no nulls are present
58 df = pd.merge ( df1, df2, how = 'inner', left_on = ['PROSPECTID'], right_on = ['PROSPECTID'])
59
60
61
62
63
64
65 # check how many columns are categorical
66 for i in df.columns:
67     if df[i].dtype == 'object':
68         print(i)
69
70
71
72
73
74
75
76 # Chi-square test
77 for i in ['MARITALSTATUS', 'EDUCATION', 'GENDER', 'Last_prod_eng2', 'first_prod_eng2', 'first_prod_eng3', 'first_prod_eng4', 'first_prod_eng5', 'first_prod_eng6', 'first_prod_eng7', 'first_prod_eng8', 'first_prod_eng9', 'first_prod_eng10', 'first_prod_eng11', 'first_prod_eng12', 'first_prod_eng13', 'first_prod_eng14', 'first_prod_eng15', 'first_prod_eng16', 'first_prod_eng17', 'first_prod_eng18', 'first_prod_eng19', 'first_prod_eng20']:
78     chi2, p, dof, expected = chi2_contingency(df[i].value_counts().reindex(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20']).values)
79
80

```

Console 1/A X

```

In [5]: for i in df2.columns:
...:     df2 = df2.loc[ df2[i] != -99999 ]

In [6]: for i in list(df1.columns):
...:     if i in list(df2.columns):
...:         print(i)
PROSPECTID

In [7]: df = pd.merge ( df1, df2, how = 'inner', left_on = ['PROSPECTID'], right_on = ['PROSPECTID'])

```

Finally, we have our single Dataset Ready for Modeling

index	PROSPECTID	Total TI	Tot Closed TI	Tot Active TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI	Tot TI
0	1	5	4	1	0	0	0	0	0	0.2	0.8	0	0	0	0	0	0	0	0	4
1	2	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0
2	3	8	0	8	1	0	0.125	0	1	0	2	0	0.25	0	1	1	0	6	1	0
3	5	3	2	1	0	0	0	0	0.333	0.667	0	0	0	0	0	1	0	0	0	0
4	6	6	5	1	0	0	0	0	0.167	0.833	0	1	0	0.167	0	4	0	0	2	0
5	8	6	4	2	0	0	0	0	0.333	0.667	1	2	0.167	0.333	0	1	0	0	0	0
6	9	1	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
7	11	7	2	5	1	0	0.143	0	0.714	0.286	3	1	0.429	0.143	0	2	1	2	0	2
8	13	2	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
9	14	2	1	1	1	0	0.5	0	0.5	0.5	1	1	0.5	0.5	0	0	0	2	0	0
10	15	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
11	16	8	1	7	7	1	0.875	0.125	0.875	0.125	8	1	1	0.125	2	1	1	5	0	0
12	17	1	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
13	20	7	5	2	1	0	0.143	0	0.286	0.714	1	1	0.143	0.143	0	0	0	2	0	0
14	21	1	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	0
15	22	5	2	3	0	1	0	0.2	0.6	0.4	2	1	0.4	0.2	1	3	0	1	0	1
16	23	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0
17	25	3	1	2	0	0	0	0	0.667	0.333	1	1	0.333	0.333	0	0	0	0	0	0
18	27	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0

The code then checks which columns contain categorical data (e.g., "Marital Status"). This is important for statistical tests later.

The screenshot shows the Spyder Python IDE with a file named 'Feature engineering.py'. The code in the editor includes a merge operation and a loop to check for categorical columns. The Variable Explorer on the right shows the state of the dataframes after the merge. The console shows the output of the categorical check loop.

```

56
57
58 # Merge the two dataframes, inner join so that no nulls are present
59 df = pd.merge ( df1, df2, how = 'inner', left_on = ['PROSPECTID'], right_on
60
61
62
63
64
65 # check how many columns are categorical
66 for i in df.columns:
67     if df[i].dtype == 'object':
68         print(i)
69
70
71
72
73
74
75
76
77 # Chi-square test
78 for i in ['MARITALSTATUS', 'EDUCATION', 'GENDER', 'Last_prod_enq2', 'first
79     chi2, pval, _ = chi2_contingency(pd.crosstab(df[i], df['Approved_Fla
80     print(i, '---', pval)
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

Name	Type	Size	Value
a1	DataFrame	(51336, 26)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot...
a2	DataFrame	(51336, 62)	Column names: PROSPECTID, time_since_recent_payment, time_since_first...
columns_to_be_removed	list	8	['time_since_first_delinquency', 'time_since_recent_delinquency', 'max_d...
df	DataFrame	(42064, 79)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot...
df1	DataFrame	(51296, 26)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot...
df2	DataFrame	(42066, 54)	Column names: PROSPECTID, time_since_recent_payment, num_times_delinqu...
i	str	13	Approved_Flag

```

In [7]:
...: for i in df.columns:
...:     if df[i].dtype == 'object':
...:         print(i)
MARITALSTATUS
EDUCATION
GENDER
last_prod_enq2
first_prod_enq2
Approved_Flag

In [8]:

```

Chi-Square Test for Categorical Features:

For each categorical feature (like Marital Status, Education, Gender, etc.), a chi-square test is performed to check if it is associated with the loan approval flag (Approved_Flag). This test helps determine which categorical features have a significant relationship with the target variable.

The screenshot shows the Spyder Python IDE with the same 'Feature engineering.py' file. The code now includes a chi-square test for categorical features and a VIF calculation for numerical columns. The Variable Explorer shows the state after these operations. The console shows the results of the chi-square tests.

```

66 for i in df.columns:
67     if df[i].dtype == 'object':
68         print(i)
69
70
71
72
73
74
75
76
77 # Chi-square test
78 for i in ['MARITALSTATUS', 'EDUCATION', 'GENDER', 'Last_prod_enq2', 'first
79     chi2, pval, _ = chi2_contingency(pd.crosstab(df[i], df['Approved_Fla
80     print(i, '---', pval)
81
82
83
84
85
86
87
88 # VIF for numerical columns
89 numeric_columns = []
90 for i in df.columns:
91     if df[i].dtype != 'object' and i not in ['PROSPECTID', 'Approved_Flag']:
92         numeric_columns.append(i)
93
94
95
96
97
98
99

```

Name	Type	Size	Value
a1	DataFrame	(51336, 26)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot...
a2	DataFrame	(51336, 62)	Column names: PROSPECTID, time_since_recent_p...
chi2	float64	1	np.float64(1387.5609151031795)
columns_to_be_removed	list	8	['time_since_first_delinquency', 'time_since_recent_delinquency', 'max_d...
df	DataFrame	(42064, 79)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot...
df1	DataFrame	(51296, 26)	Column names: PROSPECTID, Total_TL, Tot_Closed_TL, Tot_Active_TL, Tot...
df2	DataFrame	(42066, 54)	Column names: PROSPECTID, time_since_recent_p...

```

In [8]:
...: for i in ['MARITALSTATUS', 'EDUCATION', 'GENDER', 'Last_prod_enq2',
...:     'first_prod_enq2']:
...:     chi2, pval, _ = chi2_contingency(pd.crosstab(df[i],
...:     df['Approved_Flag']))
...:     print(i, '---', pval)
MARITALSTATUS --- 3.578180861038862e-233
EDUCATION --- 2.6942265249737532e-30
GENDER --- 1.907936100186563e-05
last_prod_enq2 --- 0.0
first_prod_enq2 --- 7.84997610555419e-287

In [9]:

```

Checking Multicollinearity with VIF:

For numerical columns, multicollinearity is checked using the **Variance Inflation Factor (VIF)**.

Multicollinearity occurs when two or more features are highly correlated, and removing such features can improve the model's accuracy. The code removes any features with a VIF value greater than 6.

The screenshot shows the Spyder Python IDE with a file named 'temp.py' open. The code defines a function to check for multicollinearity using VIF. It identifies numerical columns and iterates through them, calculating the VIF for each. The Variable Explorer on the right shows the state of the variables: 'a1' and 'a2' are DataFrames, 'chi2' is a float64, 'columns_to_be_removed' is a list, 'df' is a DataFrame, 'df1' and 'df2' are DataFrames. The Console shows the output of the VIF calculation for the first two columns.

```
81
82
83
84
85
86
87
88
89 # VIF for numerical columns
90 numeric_columns = []
91 for i in df.columns:
92     if df[i].dtype != 'object' and i not in ['PROSPECTID', 'Approved_Flag']:
93         numeric_columns.append(i)
94
95
96
97
98 # VIF sequentially check
99
100 vif_data = df[numeric_columns]
101 total_columns = vif_data.shape[1]
102 columns_to_be_kept = []
103 column_index = 0
104
105
106
107 for i in range (0,total_columns):
108
109     vif_value = variance_inflation_factor(vif_data, column_index)
110     print (column_index, "---", vif_value)
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Name	Type	Size	Value
a1	DataFrame	(51336, 26)	Column names: PROSPECTID, Total TL, Tot_Closed_TL, Tot_Active_TL, Tot...
a2	DataFrame	(51336, 62)	Column names: PROSPECTID, time_since_recent_p...
chi2	float64	1	np.float64(1387.5609151031795)
columns_to_be_removed	list	8	['time_since_first_delinquency', 'time_since_recent_delinquency', 'max_d ...
df	DataFrame	(42064, 79)	Column names: PROSPECTID, Total TL, Tot_Closed_TL, Tot_Active_TL, Tot...
df1	DataFrame	(51296, 26)	Column names: PROSPECTID, Total TL, Tot_Closed_TL, Tot_Active_TL, Tot...
df2	DataFrame	(42066, 54)	Column names: PROSPECTID, time_since_recent_p...

```
last_prod_eng2 --- 0.0
first_prod_eng2 --- 7.84997610555419e-287

In [9]:
...: numeric_columns = []
...: for i in df.columns:
...:     if df[i].dtype != 'object' and i not in ['PROSPECTID', 'Approved_Flag']:
...:         numeric_columns.append(i)

In [10]:
```

The screenshot shows the Spyder Python IDE with the same file 'temp.py'. The code has been updated to remove columns with a VIF value greater than 6. The Variable Explorer on the right shows the state of the variables after the removal: 'a1' and 'a2' are DataFrames, 'chi2' is a float64, 'column_index' is an int, 'columns_to_be_kept' is a list, 'columns_to_be_removed' is a list, and 'df' is a DataFrame. The Console shows the output of the VIF calculation for the first two columns, indicating that the VIF values are less than 6.

```
96
97
98 # VIF sequentially check
99
100 vif_data = df[numeric_columns]
101 total_columns = vif_data.shape[1]
102 columns_to_be_kept = []
103 column_index = 0
104
105
106
107 for i in range (0,total_columns):
108
109     vif_value = variance_inflation_factor(vif_data, column_index)
110     print (column_index, "---", vif_value)
111
112
113     if vif_value <= 6:
114         columns_to_be_kept.append( numeric_columns[i] )
115         column_index = column_index+1
116     else:
117         vif_data = vif_data.drop([ numeric_columns[i] ], axis=1)
118
119
120
121
122
123
124
125
126 # check Anova for columns_to_be_kept
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

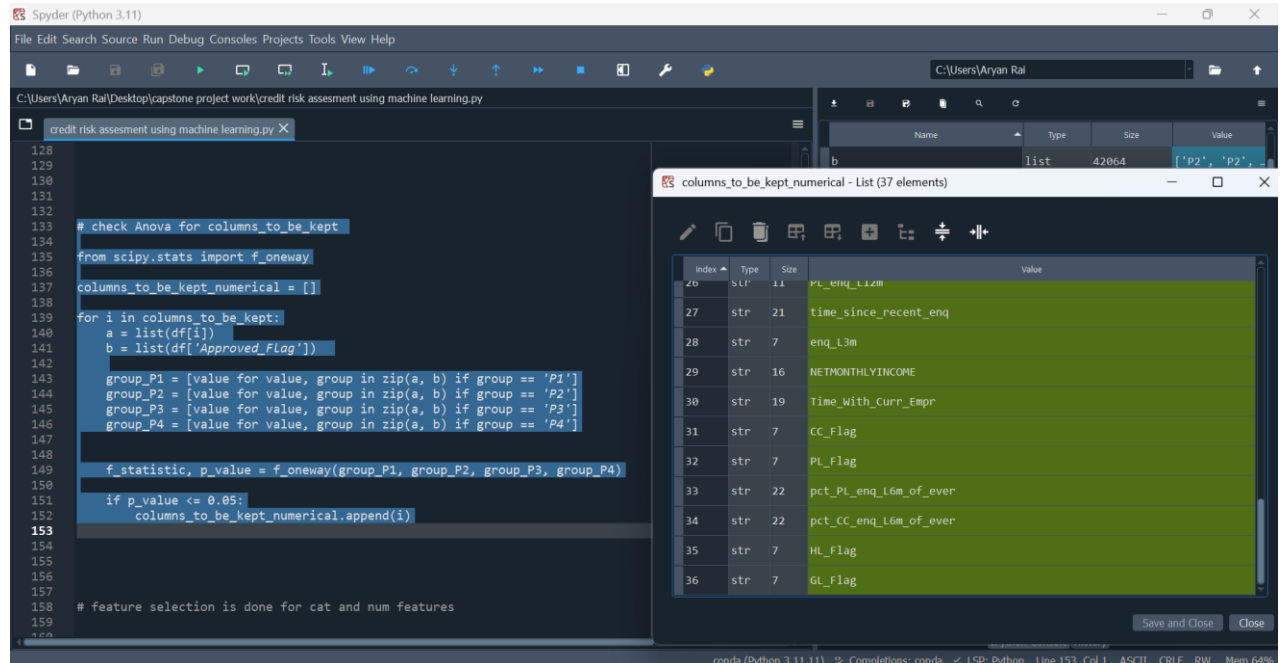
Name	Type	Size	Value
a1	DataFrame	(51336, 26)	Column names: PROSPECTID, Total TL, Tot_Closed_TL, Tot_Active_TL, Tot...
a2	DataFrame	(51336, 62)	Column names: PROSPECTID, time_since_recent_p...
chi2	float64	1	np.float64(1387.5609151031795)
column_index	int	1	39
columns_to_be_kept	list	39	['pct_tl_open_16M', 'pct_tl_closed_16M', 'tot_tl_closed_12M', 'pct_tl ...
columns_to_be_removed	list	8	['time_since_first_delinquency', 'time_since_recent_delinquency', 'max_d ...
df	DataFrame	(42064, 79)	Column names: PROSPECTID, Total TL, Tot_Closed_TL, Tot_Active_TL, Tot...

```
0.0
10.175021454450935
6.408710354561301
1.0011511962625623
3.069197305397274
2.8091261600643724
20.249530301900678
15.864576541593745
1.8331649740532168
1.5680839909542044
1.9307572353811677
4.331265056645247
9.390334396150173

In [11]:
```

ANOVA Test for Numerical Features:

Finally, the code performs an **ANOVA** test on numerical features to check whether they have a significant relationship with the target variable. Only features with a p-value less than or equal to 0.05 are kept for further analysis.

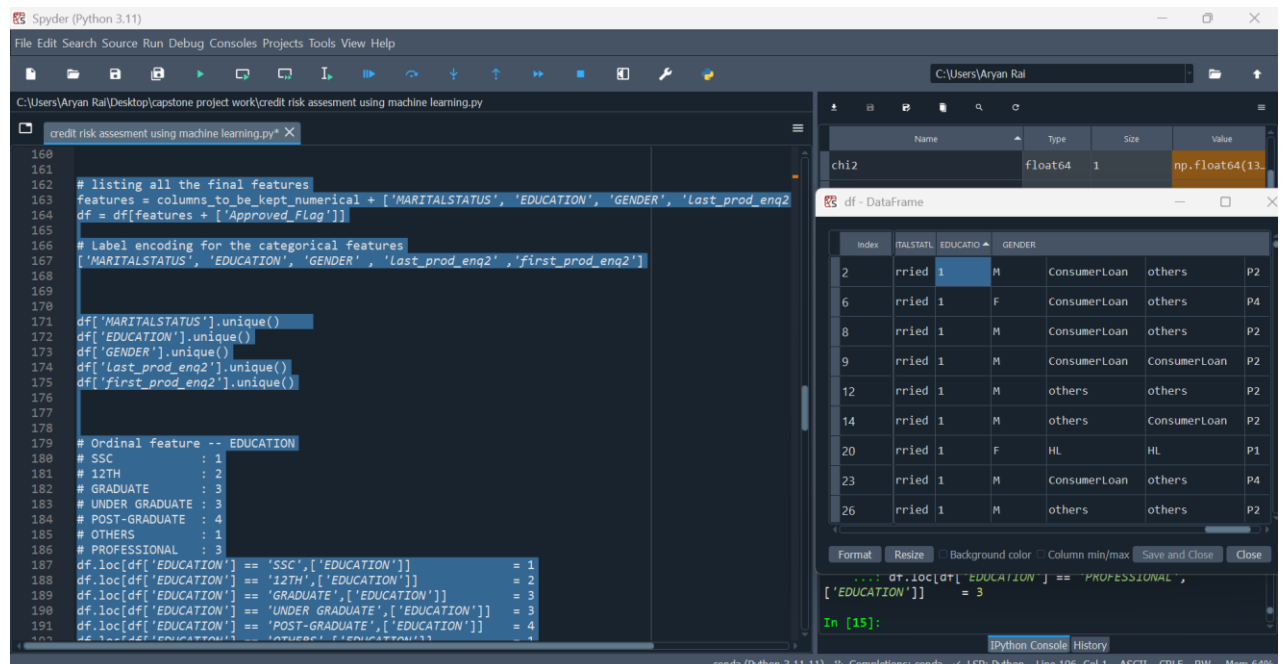


The screenshot shows the Spyder Python IDE with a file named 'credit risk assesment using machine learning.py'. The code is performing an ANOVA test on numerical features. A list of columns to be kept is displayed in a separate window titled 'columns_to_be_kept_numerical - List (37 elements)'.

```
128
129
130
131
132
133 # check Anova for columns_to_be_kept
134
135 from scipy.stats import f_oneway
136
137 columns_to_be_kept_numerical = []
138
139 for i in columns_to_be_kept:
140     a = list(df[i])
141     b = list(df['Approved_Flag'])
142
143     group_P1 = [value for value, group in zip(a, b) if group == 'P1']
144     group_P2 = [value for value, group in zip(a, b) if group == 'P2']
145     group_P3 = [value for value, group in zip(a, b) if group == 'P3']
146     group_P4 = [value for value, group in zip(a, b) if group == 'P4']
147
148     f_statistic, p_value = f_oneway(group_P1, group_P2, group_P3, group_P4)
149
150     if p_value <= 0.05:
151         columns_to_be_kept_numerical.append(i)
152
153
154
155
156
157 # feature selection is done for cat and num features
158
159
```

Index	Type	Size	Value
26	str	11	PL_enq_L12M
27	str	21	time_since_recent_enq
28	str	7	enq_L3m
29	str	16	NETMONTHLYINCOME
30	str	19	Time_With_Curr_Empr
31	str	7	CC_Flag
32	str	7	PL_Flag
33	str	22	pct_PL_enq_L6m_of_ever
34	str	22	pct_CC_enq_L6m_of_ever
35	str	7	HL_Flag
36	str	7	GL_Flag

Encoding Categorical Features: Ordinal encoding is applied to the EDUCATION feature because it can be logically encoded. One-hot encoding is applied to categorical variables like MARITALSTATUS, GENDER, and others because it cannot be logically encoded for machine learning model implementation.



The screenshot shows the Spyder Python IDE with a file named 'credit risk assesment using machine learning.py'. The code is performing label encoding for categorical features. A DataFrame preview is shown in a separate window titled 'df - DataFrame'.

```
160
161
162 # listing all the final features
163 features = columns_to_be_kept_numerical + ['MARITALSTATUS', 'EDUCATION', 'GENDER', 'Last_prod_enq2']
164 df = df[features + ['Approved_Flag']]
165
166 # Label encoding for the categorical features
167 ['MARITALSTATUS', 'EDUCATION', 'GENDER', 'Last_prod_enq2', 'first_prod_enq2']
168
169
170
171 df['MARITALSTATUS'].unique()
172 df['EDUCATION'].unique()
173 df['GENDER'].unique()
174 df['Last_prod_enq2'].unique()
175 df['first_prod_enq2'].unique()
176
177
178 # Ordinal feature -- EDUCATION
179
180 # SSC : 1
181 # 12TH : 2
182 # GRADUATE : 3
183 # UNDER GRADUATE : 3
184 # POST GRADUATE : 4
185 # OTHERS : 1
186 # PROFESSIONAL : 3
187
188 df.loc[df['EDUCATION'] == 'SSC', ['EDUCATION']] = 1
189 df.loc[df['EDUCATION'] == '12TH', ['EDUCATION']] = 2
190 df.loc[df['EDUCATION'] == 'GRADUATE', ['EDUCATION']] = 3
191 df.loc[df['EDUCATION'] == 'UNDER GRADUATE', ['EDUCATION']] = 3
192 df.loc[df['EDUCATION'] == 'POST GRADUATE', ['EDUCATION']] = 4
193 df.loc[df['EDUCATION'] == 'OTHERS', ['EDUCATION']] = 1
194 df.loc[df['EDUCATION'] == 'PROFESSIONAL', ['EDUCATION']] = 3
195
```

Index	MARITALSTATUS	EDUCATION	GENDER	ConsumerLoan	others	P2
2	rrried	1	M	ConsumerLoan	others	P2
6	rrried	1	F	ConsumerLoan	others	P4
8	rrried	1	M	ConsumerLoan	others	P2
9	rrried	1	M	ConsumerLoan	ConsumerLoan	P2
12	rrried	1	M	others	others	P2
14	rrried	1	M	others	ConsumerLoan	P2
20	rrried	1	F	HL	HL	P1
23	rrried	1	M	ConsumerLoan	others	P4
26	rrried	1	M	others	others	P2

The screenshot shows the Spyder Python IDE interface. The main editor displays the following code:

```

200 df.info()
201
202
203 df_encoded = pd.get_dummies(df, columns=['MARITALSTATUS', 'GENDER', 'last_prod_enq2', 'first_prod_e
204
205
206
207
208 df_encoded.info()
209 k = df_encoded.describe()
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231

```

The variable explorer on the right shows the following variables:

Name	Type	Size	Value
df2	DataFrame	(42066, 54)	Column names:..
df_encoded	DataFrame	(42064, 55)	Column names:..
f_statistic	float64	1	np.float64(50.
features	list	42	['pct_tl_open.
group_P1	list	4908	[0, 0, 1, 0, ..
group_P2	list	25452	[0, 0, 0, 0, ..
group_P3	list	6440	[0, 0, 0, 0, ..

The console window shows the output of the code execution:

```

45 last_prod_enq2_ConsumerLoan 42064 non-null bool
46 last_prod_enq2_HL 42064 non-null bool
47 last_prod_enq2_PL 42064 non-null bool
48 last_prod_enq2_others 42064 non-null bool
49 first_prod_enq2_AL 42064 non-null bool
50 first_prod_enq2_CC 42064 non-null bool
51 first_prod_enq2_ConsumerLoan 42064 non-null bool
52 first_prod_enq2_HL 42064 non-null bool
53 first_prod_enq2_PL 42064 non-null bool
54 first_prod_enq2_others 42064 non-null bool
dtypes: bool(16), float64(5), int64(33), object(1)
memory usage: 13.2+ MB
In [16]:

```

Applied Random Forest Model - The Random Forest model demonstrates a strong performance with an accuracy of ~76%.

The screenshot shows the Spyder Python IDE interface. The main editor displays the following code:

```

231
232 # Data processing
233
234 # 1. Random Forest
235
236 y = df_encoded['Approved_Flag']
237 x = df_encoded.drop(['Approved_Flag'], axis = 1)
238
239 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
240
241 rf_classifier = RandomForestClassifier(n_estimators = 200, random_state=42)
242
243 rf_classifier.fit(x_train, y_train)
244
245 y_pred = rf_classifier.predict(x_test)
246
247 accuracy = accuracy_score(y_test, y_pred)
248 print()
249 print(f'Accuracy: {accuracy}')
250 print()
251 precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_pred)
252
253
254 for i, v in enumerate(['p1', 'p2', 'p3', 'p4']):
255     print(f"Class {v}:")
256     print(f"Precision: {precision[i]}")
257     print(f"Recall: {recall[i]}")
258     print(f"F1 Score: {f1_score[i]}")
259     print()
260
261
262
263

```

The variable explorer on the right shows the following variables:

Name	Type
k	DataFrame

The console window shows the output of the code execution:

```

...: print()

Accuracy: 0.7636990372043266

Class p1:
Precision: 0.8370457209847597
Recall: 0.7041420118343196
F1 Score: 0.7648634172469202

Class p2:
Precision: 0.7957510116397621
Recall: 0.928457879088206
F1 Score: 0.856907593778591

Class p3:
Precision: 0.4423380726698262
Recall: 0.21132075471698114
F1 Score: 0.28600612870275793

Class p4:
Precision: 0.7178502879078695
Recall: 0.7269193391642371
F1 Score: 0.7223563495895703

```

Applied Xgboost model - XGBoost outperforms Random Forest in terms of overall accuracy (78% vs. 76%).

The screenshot shows the Spyder Python IDE with a file named 'credit risk assesment using machine learning.py'. The code defines an XGBoost classifier and evaluates its performance. The console output shows the following results:

```
Accuracy: 0.78

Class p1:
Precision: 0.823906083244397
Recall: 0.7613412228796844
F1 Score: 0.7913890312660175

Class p2:
Precision: 0.82554182339924413
Recall: 0.913577799801784
F1 Score: 0.8673315769665035

Class p3:
Precision: 0.4756380510440835
Recall: 0.30943396226415093
F1 Score: 0.37494284407864653

Class p4:
Precision: 0.7342386032977691
Recall: 0.7356656948493683
F1 Score: 0.7349514563106796

In [22]:
```

Applied Decision Tree - The Decision Tree is less accurate compared to XGBoost and Random Forest.

The screenshot shows the Spyder Python IDE with a file named 'credit risk assesment using machine learning.py'. The code defines a Decision Tree classifier and evaluates its performance. The console output shows the following results:

```
Accuracy: 0.71

Class p1:
Precision: 0.7211350293542075
Recall: 0.7268244575936884
F1 Score: 0.7239685658153242

Class p2:
Precision: 0.8102743724460012
Recall: 0.8253716551040634
F1 Score: 0.817753385703064

Class p3:
Precision: 0.3462757527733756
Recall: 0.329811320754717
F1 Score: 0.3378430614611519

Class p4:
Precision: 0.6525252525252525
Recall: 0.6277939747327502
F1 Score: 0.6399207528479445

In [23]:
```

5.2 Summary:

This project highlights the potential of machine learning in transforming credit risk assessment by offering data-driven insights and efficient classification methods. By integrating statistical feature engineering techniques and robust machine learning models, the system ensures accuracy, scalability, and relevance to banking operations. Future work can focus on deploying the model in a live environment and incorporating additional data sources for enhanced predictions.

References and Publications

1. M. Sudhakar and C.V.K. Reddy, "Two Step Credit Risk Assessment Model For Retail Bank Loan Applications Using Decision Tree Data Mining Technique," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 5(3), pp. 705-718, 2016.
2. J. H. Aboobyda and M.A. Tarig, "Developing Prediction Model of Loan Risk in Banks Using Data Mining," *Machine Learning and Applications: An International Journal (MLAIJ)*, vol. 3(1), pp. 1–9, 2016.
3. K. Kavitha, "Clustering Loan Applicants based on Risk Percentage using K-Means Clustering Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6(2), pp. 162–166, 2016.
4. Trishita Saha, Saroj Kumar Biswas, Saptarsi Sanyal, Binoti Patro, Biswajit Purkayastha, "Credit Risk Prediction Using Machine Learning Analytics: An Ensemble Model," *International Journal of Management and Applied Science*, Vol. 9, Issue 9, 2023.
5. Sudhamathy G., "Credit Risk Analysis and Prediction Modelling of Bank Loans Using R," *International Journal of Engineering and Technology (IJET)*, Vol. 8, No. 5, Oct-Nov 2016, DOI: 10.21817/ijet/2016/v8i5/160805414.
6. S. Dahiya, S.S. Handa, N.P. Singh, "A Feature Selection Enabled Hybrid-Bagging Algorithm for Credit Risk Evaluation," *Expert Systems*, 2017. DOI: 10.1111/exsy.12217.
7. A.A. Turjo, K.O. Alabi, S.O. Abdulsalam, R.O. Ogundokun, "Credit Risk Assessment Model by Analyzing Different Machine Learning Algorithms," *4th International Conference on Information and Communications Technology (ICOIACT)*, 2021. DOI: 10.1109/ICOIACT53268.2021.9563995.
8. Y-C Chang, K-H Chang, G-J Wu, "Application of eXtreme Gradient Boosting Trees in the Construction of Credit Risk Assessment Models for Financial Institutions," *Applied Soft Computing Journal*, 2018. DOI: 10.1016/j.asoc.2018.09.029.
9. Y. Zhang, H. Jia, Y. Diao, M. Hai, H. Li, "Research on Credit Scoring by Fusing Social Media Information in Online Peer-to-Peer Lending," *Procedia Computer Science*, 2016. DOI: 10.1016/j.procs.2016.07.055.
10. V. Kumar L, Natarajan S, Keerthana S, Chinmayi KM, Lakshmi N, "Credit Risk Analysis in Peer-to-Peer Lending System," *IEEE International Conference on Knowledge Engineering and Applications*, 2016.
11. Z. Somayyeh, M. Abdolkarim, "Natural Customer Ranking of Banks in Terms of Credit Risk by Using Data Mining," *Jurnal UMP Social Sciences and Technology Management*, 2015.
12. A.B. Hussain, "Developing Prediction Model of Loan Risk in Banks Using Data Mining," *Machine Learning and Applications: An International Journal (MLAIJ)*, 2016.