EXPLAINABLE AI AND AUDITABILITY ON CREDIT RISK ANALYSIS

A PROJECT REPORT

Submitted by

MOHAMED ASIK. N(812620243012)

VAISHNAVI. S (812620243022)

AKASH KUMAR . P (812620243301)

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

M.A.M. COLLEGE OF ENGINEERING

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**EXPLAINABLE AI AND AUDITABILITY ON CREDIT RISK ANALYSIS"**is the Bonafide work of **MOHAMED ASIK N, VAISHNAVI S , AKASH KUMAR P** who carried outthe project work under my supervision.

SIGNATURE

Mr. K.ILANGO

HEAD OF THE DEPARTMENT

Artificial Intelligence And Data Scince

M.A.M College of Engineering.

Trichy – 621 105

SIGNATURE

Mrs .D.POORANI

SUPERVISOR

Artificial Intelligence And Data Scince

M.A.M College of Engineering.

Trichy – 621 105

Certified that the candidates are examined in the project viva-voce Examination held on_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The completion of the project has been made possible because of the involvement of many individuals at various stages, I would like to thank them in this record.

I would like to express my sincere thanks to my principal **Dr. M. SHANMUGA PRIYA** who accepted my request and granted permission to carry out this project.

I would like to take this opportunity to express my sincere thanks to my Head of the Department of Artificial Intelligence and Data Science, **Prof. K. ILANGO** for his immense knowledge and professional expertise.

I would also like to express my deep gratitude to **Prof. D. POORANI** , as she was my project guide and provided immense support and valuable guidance during the planning and creation of this project, without whom this project would not have been possible.

I would like to express my most sincere thanks to my parents for their endless motivation and support for the creation of the project.

Finally, I express my humble thanks to each one of my friends who have assisted me in the implementation of this project.

ABSTRACT

The predictive accuracy of machine learning models has evolved remarkably, yet the interpretability of these models remains a significant concern in critical decision-making processes. This project delivers into enhancing model interpretability using LIME (Local Interpretable Model-agnostic Explanations) techniques .The study begins with data preprocessing, including handling missing values, encoding categorical variables, and scaling features. A comprehensive exploratory data analysis (EDA) and visualization help comprehend the dataset's characteristics, followed by outlier detection using the Elliptical Envelope and Tukey's method .Several classification algorithms, including Logistic Regression, XGBoost, LightGBM, and Random Forest, are employed to predict loan statuses, achieving commendable accuracy rates. However, the emphasis shifts to understanding these models' decisions .LIME technique applied to interpret and explain model predictions. LIME generates locally faithful explanations by approximating complex model behaviour around individual instances. Visualizations, such as force plots and feature importance graphs, offer insight into the factors influencing predictions .The insights gleaned from LIME ,shed light on the models decision-making processes, empowering stakeholders to comprehend and trust the models predictions. This project demonstrates the significance of model interpretability in ensuring transparency and reliability in machine learning applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

LIME	Local Interpretable Model - agnostic Explanation
EDA	Exploratory Data Analysis
XAI	Explainable AI
GDPR	General Data Protection Regulation
ECOA	Equal Credit Opportunity Act
HELOC	Home Equity Line of Credit
LC	Lending Club
ML	Machine Learning
KAIRI	Key AI Risk Indicators
GBDT	Gradient Boosting Decision Tree
DNNs	Deep Neural Networks
NLP	Natural Language Processing
GUI	Graphical User Interface
CLI	Command Line Interface
XG BOOST	Extreme Gradient Boosting
LIGHT GBM	Light Gradient Bosting Machine
AI	Artificial Intelligence

CHAPTER 1

INTRODUCTION

1.1 PROJECT OBJECTIVE

In recent years, machine learning models have achieved remarkable predictive accuracy across various domains. However, the interpretability of these models remains a critical concern, especially in scenarios where decision-making impacts individuals' lives, such as loan approval systems. The opaque nature of many advanced models often leads to a lack oftrust and understanding, hindering their widespread adoption.

1.11 PROJECT OUTLINE

This project aims to delve into the interpretability aspect of machine learning models, particularly focusing on enhancing transparency and trustworthiness in predicting loan statuses. The primary objective is to elucidate and interpret complex model predictions using state-of-the-art interpretability technique: LIME (Local Interpretable Model-agnostic Explanations).The project follows a structured approach, beginning with data preprocessingto handle missing values, encode categorical variables, and scale features. Subsequently, exploratory data analysis (EDA) and visualization techniques are employed to gain insights into the dataset's characteristics. The identification and handling of outliers using robust methods are pivotal in ensuring model robustness .Several classification algorithms, including Logistic Regression, XGBoost, Light GBM, and Random Forest, are applied to predict loan statuses. While achieving commendable accuracy rates, the core focus shifts tounderstanding the rationale behind these model predictions .The introduction of LIME enables the interpretation of machine learning models' decisions. LIME generates localized explanations, providing insights into individual predictions. Visualizations derived from these techniques facilitate a clear understanding of the models' decision-making processes.

1.2 PROBLEM STATEMENT

The current landscape of credit risk analysis relies heavily on AI models, which often operate as black boxes, making it challenging for stakeholders to understand how decisions are made. Lack of explainability hinders trust, increases regulatory scrutiny, and limits the ability to identify and mitigate biases. Additionally, the absence of auditability raises concerns about model reliability, robustness, and compliance with regulatory standards. Therefore, there is a pressing need to develop transparent, interpretable, and auditable AI solutions for credit risk analysis.

1.2.1 OBJECTIVES

- To improve the transparency and interpretability of AI models used in credit riskassessment.
- To establish mechanisms for auditing AI-driven credit risk analysis processes toensure compliance and reliability.
- To mitigate the impact of biases in AI models and enhance fairness andaccountability in credit risk decision-making.
- To foster trust and confidence among stakeholders by providing clear explanationsof credit risk assessment outcomes.

1.2.2 METHODOLOGY

- Conduct an extensive review of existing literature on explainable AI (XAI) techniques, interpretable machine learning methods, and their applicability to credit risk analysis, considering various financial products and customer segments.
- Evaluate current industry practices and challenges in auditing AI models used for credit risk assessment across diverse financial institutions and regulatory environments, including global perspectives and best practices.
- Identify common biases and ethical considerations inherent in AI-driven credit riskanalysis and explore effective mitigation strategies.

1.3 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is the ability of a computer program or a machine to think andlearn. It is also a field of study which tries to make computers "smart". As machines become increasingly capable, mental facilities once thought to require intelligence are removed from the definition. AI is an area of computer sciences that emphasizes the creation of intelligentmachines that work and reacts like humans. Some of the activities computers with artificial intelligence are designed for include: Face recognition, Learning, Planning, Decision makingetc.,

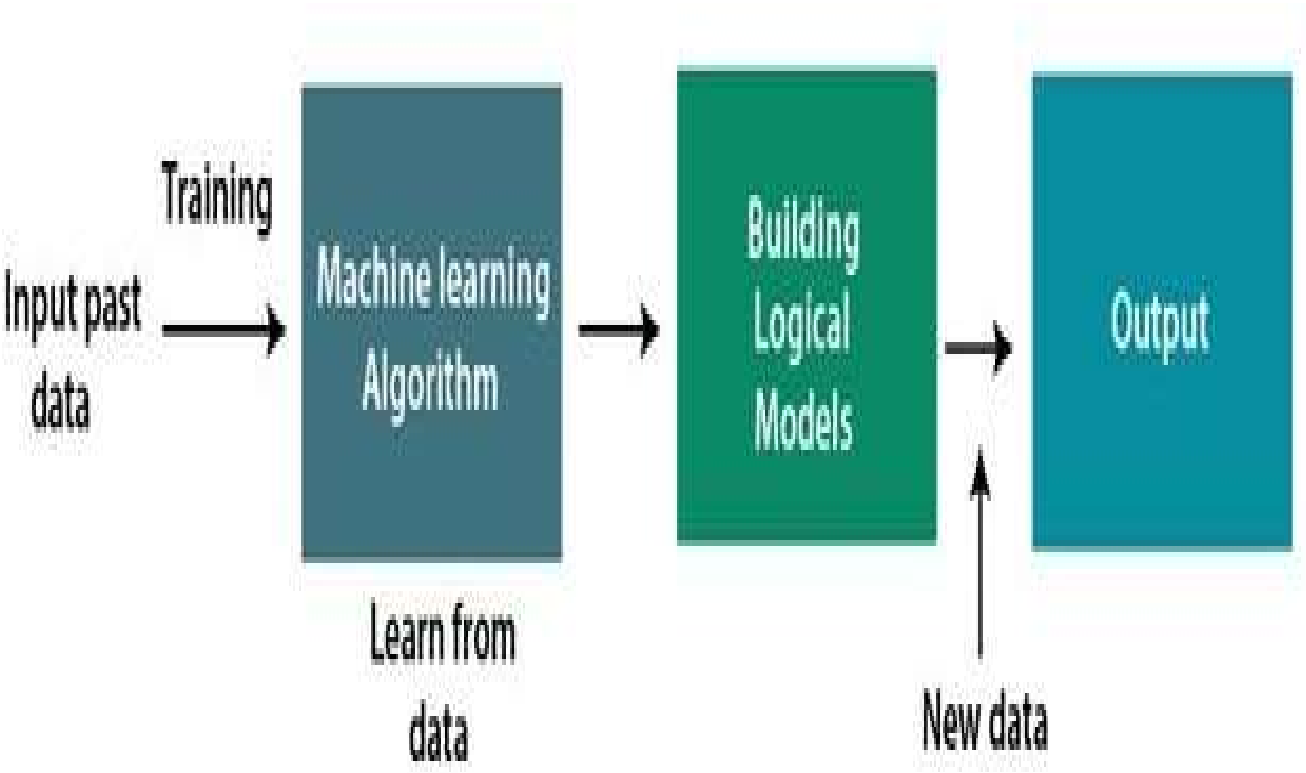
Artificial intelligence is the use of computer science programming to imitate human thought and action by analysing data and surroundings, solving or anticipating problems andlearning or self-teaching to adapt to a variety of tasks.

1.4 MACHINE LEARNING

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information**.** Currently, it is being used for various tasks such as image recognition**,** speech recognition**,** email filtering**,** Facebook auto-tagging**,** recommender system**,** and many more. Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959**.** We can define it in a summarized way as: “Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed”.A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it**.** The accuracy of predicted output

depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



Figur1.4

WORKING OF MACHINE LEARNING ALGORITHM

1.4.1 FEATURES OF MACHINE LEARNING:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amountof the data.

1.4.2 CLASSIFICATION OF MACHINE LEARNING

At a broad level, machine learning can be classified into three types:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

1) SUPERVISED LEARNING

Supervised learning is a type of machine learning method in which we provide samplelabeled data to the machine learning system in order to train it, and on that basis, it predicts the output.The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not.The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is spam filtering.

Supervised learning can be grouped further in two categories of algorithms:

- Classification

- Regression

2) UNSUPERVISED LEARNING

Unsupervised learning is a learning method in which a machine learns without any supervision. The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data.

It can be further classifieds into two categories of algorithms:

- Clustering
- Association

3) REINFORCEMENT LEARNING

Reinforcement Learning: Reinforcement learning is a learning method in which an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties based on its actions, allowing it to learn which

actions lead to desirable outcomes. Unlike supervised learning, where the correct actions are provided, and unsupervised learning, where no guidance is given, reinforcement learning operates based on a system of trial and error.

The goal of reinforcement learning is to maximize the cumulative reward obtained over time. It's like training a pet: rewarding good behaviour and discouraging bad behaviour to teach it the desired actions. Reinforcement learning is used in various applications, including gameplaying, robotics, and autonomous driving, where agents must learn to navigate and make decisions in complex environments.

AI vs XAI

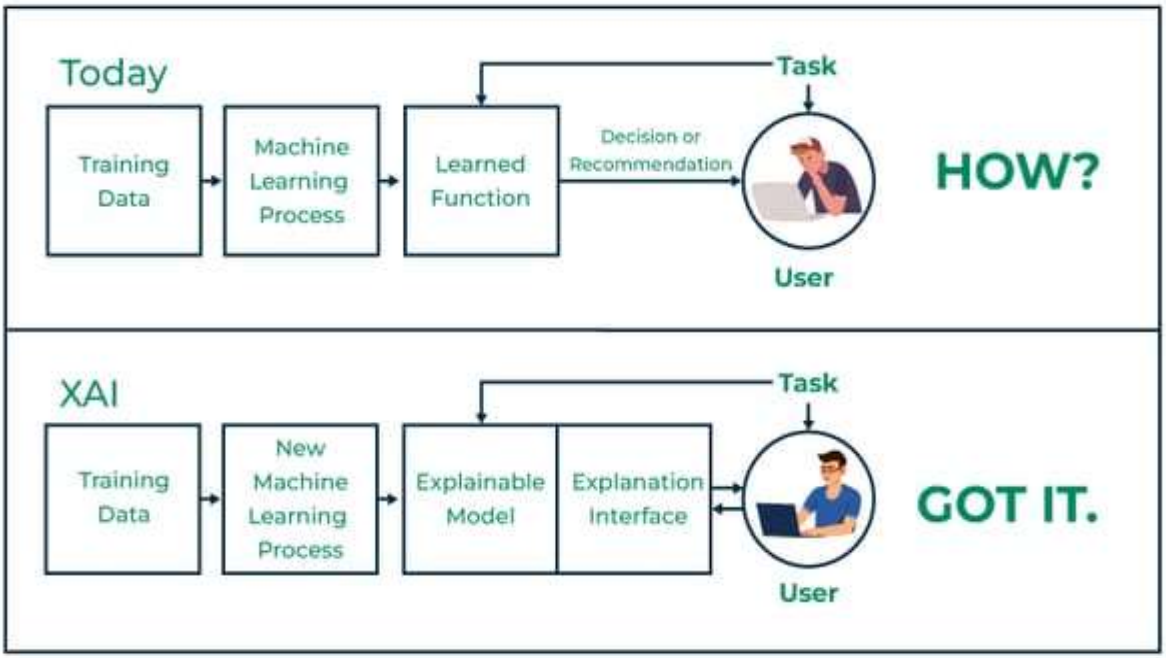


FIGURE 1.5

1.5 EXPLAINABLE AI

LIME (Local Interpretable Model-agnostic Explanations)

1. LIME provides local model interpretability by approximating a complex model'sbehaviour around specific instances.
2. It builds simpler, interpretable models (such as linear models) to approximate thecomplex model's predictions for individual instances.
3. Visualize and interpret the explanations provided by LIME to understand how yourcomplex model is making predictions for specific data points.
4. Identify important features that contribute most to individual predictions.

CHAPTER 2

LITERATURE SURVEY

[1] **Title:** Credit risk analysis using machine learning classifiers

Authors: Trilok Nath Pandey; Alok Kumar Jagadev; Suman Kumar Mohapatra

Year: 2023

Description

Banking industry has the major activity of lending money to those who are in need of money. In order to payback the principle borrowed from the depositor bank collects the interest made by the principle borrowers. Credit risk analysis is becoming an important field in financial risk management. Many credit risk analysis techniques are used for the evaluation of credit risk of the customer dataset. The evaluation of the credit risk datasets leads to the decision to issue the loan of the customer or reject the application of the customer is the difficult task which involves the deep analysis of the customer credit dataset or the data provided by the customer. In this paper we are surveying different techniques for the credit risk analysis which are used for the evaluation for the credit risk datasets.

[2] **Title:** Explainable AI in Credit Risk Management

Authors: Branka Hadji Misheva, Joerg Osterrieder, Ali Hirsa, Onkar Kulkarni, Stephen Fung Lin

Year : 2021

Description:

Artificial Intelligence (AI) has created the single biggest technology revolution the world has ever seen. For the finance sector, it provides great opportunities to enhance customer experience, democratize financial services, ensure consumer

protection and significantly improve risk management. While it is easier than ever to run state-of-the-art machine learning models, designing and implementing systems that support real-world finance applications have been challenging. In large part because they lack transparency and explainability which are important factors in establishing reliable technology and the

research on this topic with a specific focus on applications in credit risk management. In this paper, we implement a advanced post-hoc model agnostic explainability technique called Local Interpretable Model Agnostic Explanations (LIME) to machine learning (ML)-based credit scoring models applied to the open-access data set offered by the US-based P2P Lending Platform, Lending Club. Specifically, we use LIME to explain instances locally and. We also discuss the practical challenges associated with the implementation of these state-of-art explainable AI (XAI) methods and document them for future reference. We have made an effort to document every technical aspect of this research, while at the same time providing a general summary of the conclusions.

[3] **Title:** Explainable Machine Learning in Credit Risk Management

Authors: Niklas Bussmann, Paolo Giudici, Dimitri Marinelli & Jochen Papenbrock

Year : 2022

Description:

The paper proposes an explainable Artificial Intelligence model that can be used in credit risk management and, in particular, in measuring the risks that arise when credit is borrowed employing peer to peer lending platforms. The model applies correlation networks to Shapley values so that Artificial Intelligence predictions are grouped according to the similarity in the underlying explanations. The empirical analysis of 15,000 small and medium companies asking for credit reveals that both risky and not risky borrowers can be grouped according to a set of similar financial characteristics,

which can be employed to explain their credit score and, therefore, to predict their future behaviour.

[4] **Title:** Explainable AI in Fintech Risk Management

Authors: Niklas Bussmann1,2 Paolo Giudici3* Dimitri Marinelli1 Jochen Papenbrock1

Year : 2023

Description:

The following paper is about weather prediction using python programming language and concepts of machine learning. This report is an attempt to predict the weather of the next five days at the interval of three hours. The concepts of python help in extracting the required data from the API (open weather API), wherein API refers to the Application Programming Interface. Through coding in python and using the library requests we have requested thesite to get the predicted weather and we have extracted and cleaned the data to make the correct use. The basic data types list and dictionary are used to extract the required data from the raw API. The libraries „Requests", „Matplotlib", and „Pandas" are used for the coding in python to implement the prediction of weather. The demonstration and use of these libraries are clearly depicted in the report. These libraries play a very important role in the execution of the program.

[5] **Title:** Enabling Machine Learning Algorithms for Credit Scoring -- Explainable Artificial Intelligence (XAI) methods for clear understanding complex predictive models

Authors: Przemysław Biecek, Marcin Chlebus, Janusz Gajda, Alicja Gosiewska, AnnaKozak, Dominik Ogonowski, Jakub Sztachelski, Piotr Wojewnik

Year : 2021

Description:

Rapid development of advanced modelling techniques gives an opportunity to develop tools that are more and more accurate. However as usually, everything comes with a price and inthis case, the price to pay is to loose interpretability of a model while gaining on its accuracy and precision. For managers to control and effectively manage credit risk and for regulators to be convinced with model quality the price to pay is too high. In this paper, we show how to take credit scoring analytics in to the next level, namely we present comparison of various predictive models (logistic regression, logistic regression with weight of evidence transformations and modern artificial intelligence algorithms) and show that advanced tree based models give best results in prediction of client default. What is even more important and valuable we also show how to boost advanced models using techniques which allow to interpret them and made them more accessible for credit risk practitioners, resolving the crucial obstacle in widespread deployment of more complex, 'black box' models like random forests, gradient boosted or extreme gradient boosted trees. All this will be shown on the large dataset obtained from the Polish Credit Bureau to which all the banks and most of thelending companies in the country do report the credit files. In this paper the data from lending

companies were used. The paper then compares state of the art best practices in credit risk modelling with new advanced modern statistical tools boosted by the latest developments in the field of interpretability and explainability of artificial intelligence algorithms. We believethat this is a valuable contribution when it comes to presentation of different modelling tools but what is even more important it is showing which methods might be used to get insight and understanding of AI methods in credit risk context.

CHAPTER 3

SOFTWARE ENVIRONMENT

3.1 REQUIREMENT ANALYSIS

Requirement analysis is a critical phase in the project's development process. It involves gathering, documenting, and understanding the specific needs and expectations ofstakeholders for the weather prediction application. This phase focuses on identifying the functional and non-functional requirements, user expectations, system constraints, and desired features. By analyzing the requirements, the project team can define a clear scope and roadmap for the application's development, ensuring that it meets the users' needs and aligns with the project's objectives..

3.1.1 PYTHON

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language.

For example, x=10. Here, x can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operatingsystems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus.The source code is freely available and open for modification and reuse. Python has asignificant number of users.

Features in Python

There are many features in Python, some of which are discussed below

- Easy to code

- Free and Open Source
- Object-Oriented Language
- GUI Programming Support
- High-Level Language
- Extensible feature
- Python is Portable language
- Python is Integrated language
- Interpreted Language

3.1.2 ANACONDA

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from Py PI as well as the anaconda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between anaconda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and thereason anaconda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a packageand any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google TensorFlow, can find that itstops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, anaconda analyses the current environment including everything currently

installed, and, together with any version limitations specified (e.g., the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies,and shows a warning if this cannot be done.

Open source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the anaconda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into anaconda environment using pip, and anaconda will keep track of what it has installed itself and whatpip has installed.

Custom packages can be made using the anaconda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with anaconda.

3.1.3 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage anaconda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole

- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

3.1.4 JUPYTER NOTEBOOK

Jupyter Notebook (formerly Python Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application,Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release(October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

3.2 RESOURCE REQUIREMENTS

3.2.1HARDWARE REQUIREMENTS

- **Processor :** intel core i5 or AMD Ryzen 5 (or equivalent)
- **RAM:** 8 GB (minimum)
- **Hard Disk:** 256 GB SSD (or larger)

3.2.2SOFTWARE REQUIREMENTS:

- **Operating System :** windows 10 or later version
- **Software :** Python IDE (Python IDLE or VS Code)

3.3 EXIXTING SYSTEM

Traditional loan approval systems in financial institutions rely heavily on opaque machine learning models, which often lack transparency and interpretability. These systems utilize complex algorithms to predict loan statuses based on various applicant attributes, income, credit history, and loan details. However, their inherent complexity leads to a lack of understanding regarding to the decision-making process. This opacity raises concerns among stakeholders, including loan applicants and financial institutions, as it hampers trust, accountability, and compliance with regulatory norms.

3.3.1 DISADVANTAGES:

1. Lack of Transparency
2. Limited Explainability
3. Potential Bias
4. Regulatory Compliance Challenge

3.4 PROPOSED SYSTEM

The proposed system seeks to address the limitations of the existing system by integrating advanced interpretability technique like LIME. This integration aims to provide detailed insights into complex machine learning models, enhancing transparency and interpretability. By leveraging these methods, the system aims to generate understandable explanations for model predictions, facilitating user comprehension and trust. The goal is to ensure a more transparent decision-making process and mitigate potential biases, ultimately improving model explainability and accountability.

3.2.3 ADVANTAGES

1. Enhanced model interpretability
2. Improved transparency in decision-making
3. Mitigation of potential biases
4. Facilitated user comprehension
5. Increased model accountability.

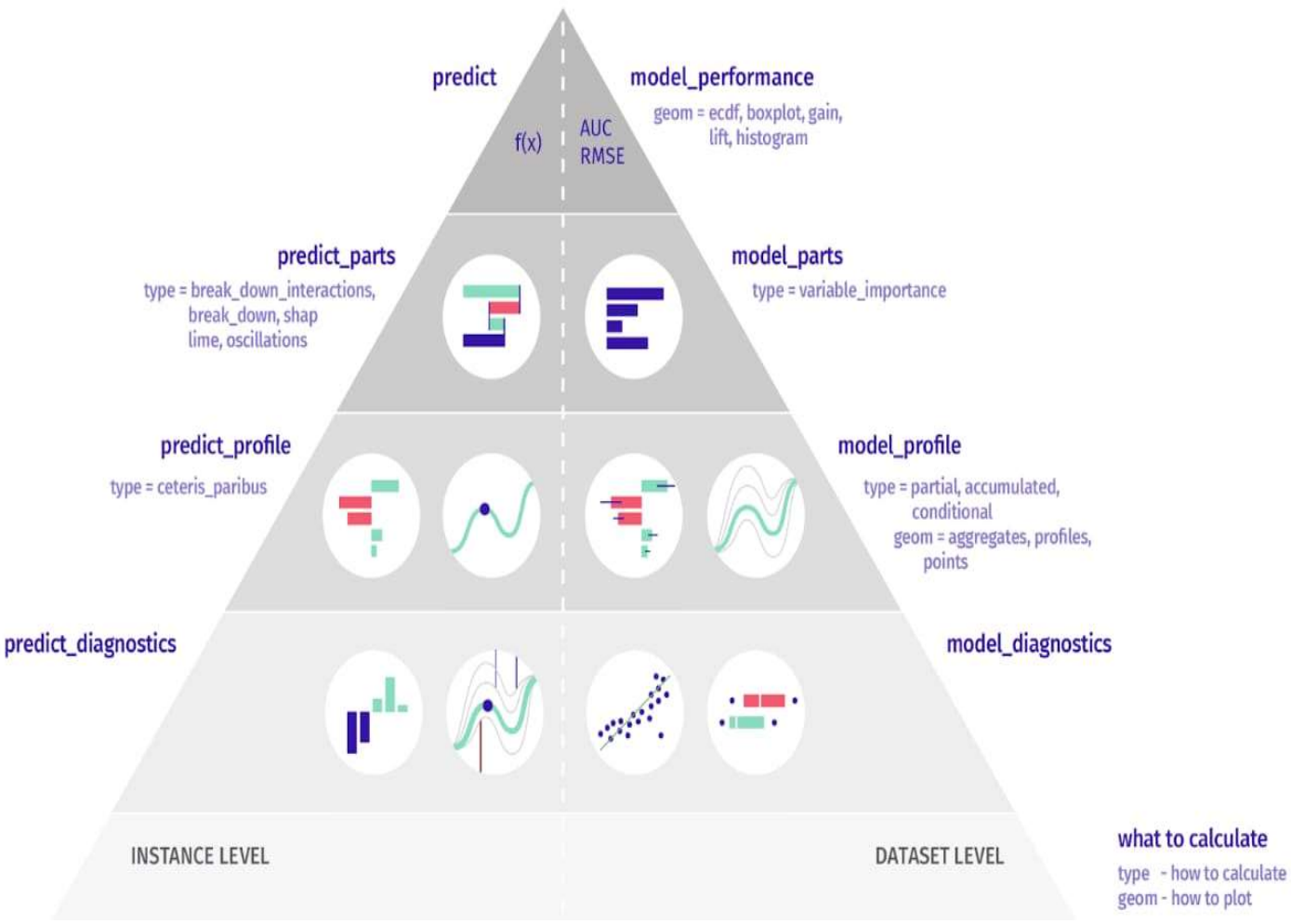


FIGURE 3.3

THE COMPLETE ML TECHNIQUE FOR PROVIDING EXPLAINABILITY

CHAPTER 4
ARCHITECTURE

4.1 SYSTEM ARCHITECTURE

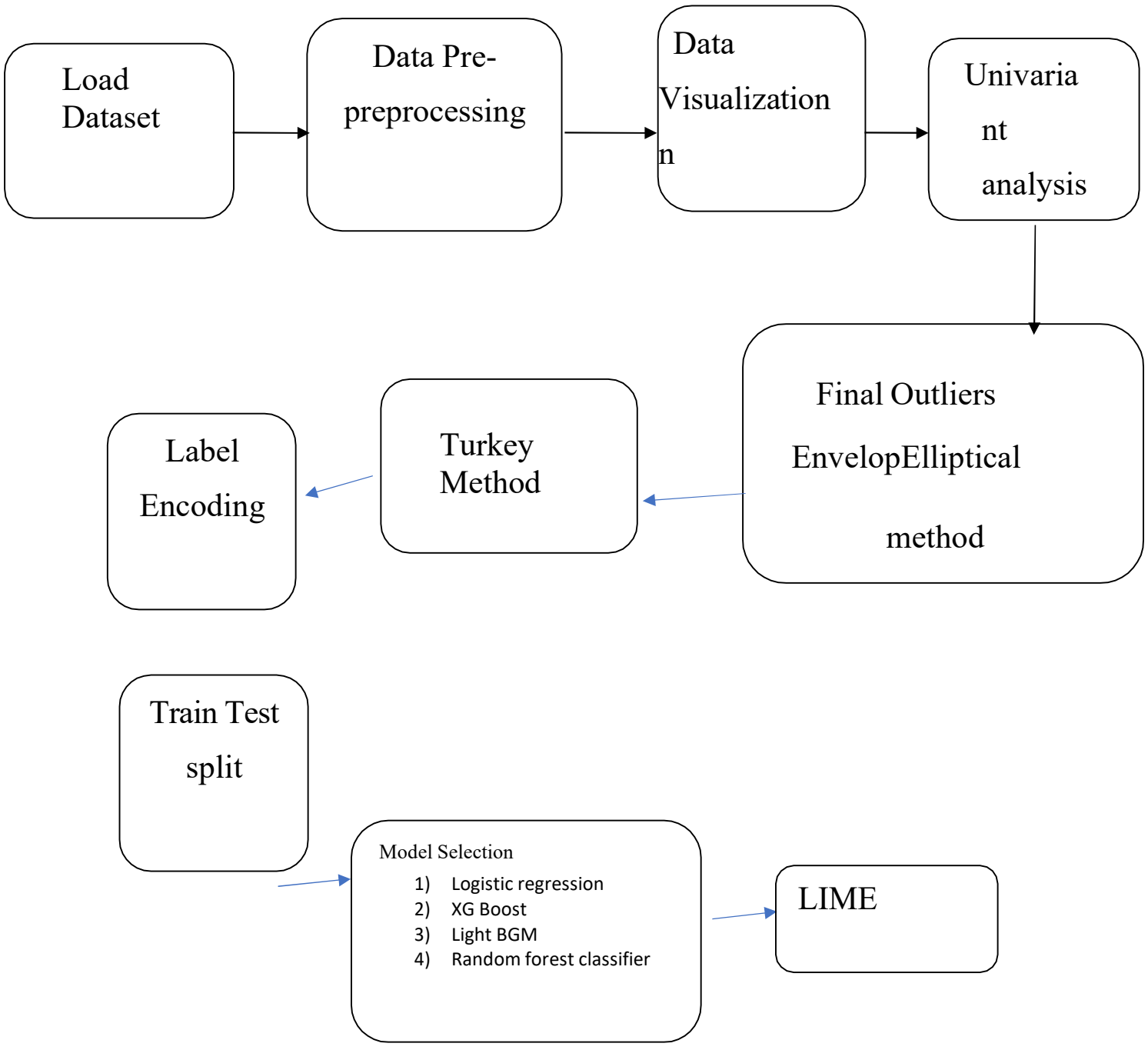


FIGURE 4.1

4.2 MODULE DESCRIPTION

Module 1: Data Preprocessing

Module 2: Outlier Detection

Module 3: Model Implementation

Module 4: Model Interpretability

Module 5: Evaluation

Module 6: Visualization

MODULE 1: **Data Preprocessing**

This segment focuses on data refinement, covering tasks like data cleaning, handling missing values, scaling, and encoding categorical variables. It ensures the dataset is properly formatted and ready for modeling. By identifying and addressing issues within the data, such as inconsistencies or missing information, this module aims to enhance the quality of inputfor subsequent analysis.

MODULE 2: Outlier Detection

This module aims to identify and manage outliers in the dataset, which could potentially skew the models' performance. Techniques such as Elliptical Envelopes and Tukey's method are applied to recognize and handle these anomalies. By addressing outliers, this module ensures the model's robustness and reliability by reducing the influence of aberrant data points.

MODULE 3: Model Implementation

In this phase, various machine learning algorithms are employed to build predictive models. Algorithms like Logistic Regression, XGBoost, LightGBM, and Random Forest Classifier are implemented to train models on the prepared dataset. Each algorithm's suitability and performance in predicting the target variable are evaluated to determine the most effective one for the given problem.

MODULE 4: Model Interpretability

LIME technique is integrated into this section to provide interpretability for the trained models. LIME (Local Interpretable Model-agnostic Explanations) generates locally faithful explanations for individual predictions, offering insights into model decisions at an instancelevel.

MODULE 5: Evaluation

This phase evaluates the performance of the developed models using various metrics like accuracy, precision, recall, and F1-score. It quantitatively assesses the models' predictive capabilities, comparing their performance to determine which model best suits the given problem statement.

MODULE 6: Visualization

Visual representation of data insights, model performance metrics, and interpretability results is the focus here. This module employs graphical representations like plots, charts, and interactive dashboards to present findings and facilitate better comprehension of the model behaviour and data patterns.

4.3 LIME

LIME Locally Interpretable Model Agnostic Explanations is a post-hoc model-agnostic explanation technique which aims to approximate any black box machine learning model with a local, interpretable model to explain each individual prediction . By model agnostic explanations, the authors suggest that it can be used for explaining any classifier, irrespective of the algorithm used for predictions as LIME is independent of the original classifier. Finally, LIME works locally which in essence means that it is observation specific and similarly as SHAP, will give explanations for every specific observation it has. In terms of the methodology, the way LIME works is that it tries to fit a local model using sample data points that are similar to the instance

being explained. The local model can be from the class of potentially interpretable models such as linear models, decision trees, etc. For further details, please see . The explanations provided by LIME for each observation x is obtained as follows: $\xi(x) = \text{argming}\in G\ L(f, g, \pi x) + \Omega(g)$ (1) where G is the class of potentially interpretable models such as linear models and decision trees, $g \in G$: An explanation considered as a model $f : R\ d \rightarrow R$: The main classifier being explained $\pi x(z)$: Proximity measure of an instance z from x $\Omega(g)$: A measure of complexity of the explanation $g \in G$ The goal is to minimize the locality aware loss L without making any assumptions about f, since a key property of LIME is that it is model agnostic. L is the measure of how unfaithful g is in approximating f in the locality defined by $\pi(x)$.

4.4 MODEL SELECTION

Logistic Regression

This is a simple yet effective model commonly used for binary classification tasks like credit risk analysis. It provides good interpretability, as you can easily understand the impact of each feature on the predicted outcome. Coefficients associated with each feature can be directly interpreted as the log-odds of the target variable.

XGBoost (Extreme Gradient Boosting)

XGBoost is a powerful ensemble learning algorithm that often outperforms other algorithms in terms of predictive accuracy. However, it's not as inherently explainable as logistic regression. While it offers feature importance scores, understanding the exact relationship between features and the target variable might require additional techniques or tools.

LightGBM (Light Gradient Boosting Machine)

Similar to XGBoost, LightGBM is another gradient boosting framework known for its efficiency and accuracy. Like XGBoost, it provides feature importance scores,

but interpreting the model's decisions might not be as straightforward as logistic regression.

Random Forest Classifier

Random forests are an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes as the prediction. While random forests offer good predictive performance and feature importance measures, the interpretability of individual trees might be challenging.

CHAPTER 5

SYSTEM STUDY

5.1. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth N with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

i. Economic Feasibility

ii. Technical Feasibility

iii. Social Feasibility

5.1.1. Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2. Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed systemmust have a modest requirement, as only minimal or null changes are

required for implementing this system.

5.1.3. Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user ofthe system.

CHAPTER 6

TESTING

6.1. TYPES OF TESTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub – assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1.1. UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.1.2. INTEGRATION TESTING

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

testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must beconsidered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.3. SYSTEM TEST

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

6.1.4. WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge ofthe inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.1.5. BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1.6. UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conductedas two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.

- The entry screen, messages and responses must not be delayed.

integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

CHAPTER 7

APPENDIX

7.1 SOURCE CODE

```
# Import libraries

    # basic

    libraries

    import numpy

    as npimport

    pandas as pd#

    visulazation

    import seaborn as sns

    import matplotlib.pyplot as

    plt import plotly.express as

    px import

    plotly.graph_objects as go

    from plotly.subplots import make_subplots

    from plotly.offline import

    init_notebook_mode

    # outliers

    from sklearn.linear_model import

    LinearRegressionfrom sklearn.metrics import

    mean_squared_error from sklearn.covariance

    import EllipticEnvelope
```

```
# SMOTE
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Model in Machine
Learningimport xgboost
as xgb
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import
cross_val_score from sklearn.model_selection
import  cross_val_predictimport statsmodels.api as
sm
import xgboost as
xgbimport
lightgbm as lgb
from sklearn.linear_model import
LogisticRegression from sklearn.ensemble import
RandomForestClassifierfrom sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
# Deep Learning
import tensorflow
as tf
from tensorflow.keras.models import
```

```
Sequentialfrom tensorflow.keras.layers
import Dense import os

# lime library
import lime
import lime.lime_tabular
from sklearn.linear_model import LinearRegression
#Remove warnings
import warnings
warnings.filterwarnings("igno
re")
 # Load dataset
credit = pd.read_csv('credit_risk.csv')
# data pre-processing
credit.he
ad()
credit.tai
l()
credit.sh
ape

credit.describ
e()
credit.info()
credit.columns
credit.sample(n=20)
```

```
credit.nunique()

credit.select_dtypes(include='object')

credit.isnull().sum()

credit['loan_intent'].value_counts()

credit = credit.dropna()

 credit.isnull().sum()

print('Non Default', round(credit['loan_status'].value_counts()[0]/len(credit) *

100,2), '% of the dataset')

print('Default', round(credit['loan_status'].value_counts()[1]/len(credit) * 100,2),

'% of the dataset')

# data visualization

# Create a box plot to visualize outliers

numeric_data = credit.select_dtypes(include=['number'])

# Create a figure to hold all the box plots

plt.figure(figsize=(20, 10))

# Loop through each column in the numeric data and create a box

plotfor i, column in enumerate(numeric_data.columns):

plt.subplot(1, len(numeric_data.columns), i

+ 1)plt.boxplot(numeric_data[column])

plt.title(column)

plt.ylabel('Value')
```

```
# Adjust
layout
plt.tight_layo
ut()

# Show the box
plotsplt.show()

# correlation plot
plt.figure(figsize=(10,
10))
data_num =
credit.select_dtypes(include=['number'])corr =
data_num.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corr * 100, cmap = 'cividis', annot= True, fmt='.2f', mask=mask)
plt.title('Confusion Matrix')
ply. Show()
# temperature
temp=dict(layout=go.Layout(font=dict(family="Franklin Gothic", size=12),
height=500, width=1000))
target=credit.loan_status.value_counts(normalize=True)
target.rename(index={1:'Default',0:'non default'},inplace=True)
pal, color=['#016CC9','#DEB078'], ['#8DBAE2','#EDD3B3']
```

```
fig=go.Figure()
fig.add_trace(go.Pie(labels=target.index, values=target*100,
hole=.45, showlegend=True,sort=False,
marker=dict(colors=color,line=dict(color=pal,width=2.5)),
hovertemplate = "%{label} Accounts:
%{value:.2f}%<extra></extra>"))fig.update_layout(template=temp,
title='Target Distribution',
legend=dict(traceorder='reversed',y=1.05,x=0),
uniformtext_minsize=15, uniformtext_mode='hide',width=700)
fig.show()

# level counts
fig=px.pie(values=level_counts.values,
names=level_counts.index,
color_discrete_sequence=px.colors.sequential.Mint,
title= 'person_home_ownership'
)
fig.update_traces(textinfo='label+percent+value', textfont_size=13,
marker=dict(line=dict(color='#102000', width=0.2)))

fig.data[0].marker.line.width =
2
fig.data[0].marker.line.color='g
ray'fig.show()

#univarient analysis
```

```
#MAX AND MIN AGE
max_ =
credit['person_age'].max()
min_ =
credit['person_age'].min()
print(f"maximum Age
{max_}") print(f"minimum
Age {min_}")

# people with an age between x
and ydef age_group(arr):
lenarr = len(arr)
for i in range(0,lenarr-1):
next = arr[i]+1
num_people = credit['person_age'].between(next,arr[i+1]).sum()
print(f'Age between {next} and {arr[i+1]}: Number of people {num_people}')

age_group([0 ,18, 26, 36, 46, 56, 66])

#max and min income
max_ =
credit['person_income'].max()
min_ =
credit['person_income'].min()

print(f"maximum Income
```

```
{max_}")print(f"minimum
Income {min_}")

#people with an income between x
and ydef income_group(arr):
lenarr = len(arr)
for i in range(0,lenarr-1):
next = arr[i]+1
num_people = credit['person_income'].between(next,arr[i+1]).sum()
print(f'Income between {next} and {arr[i+1]}: Number of people
{num_people}')

income_group([0, 25000, 50000, 75000, 100000,float('inf')])
minimum Income 4000

Income between 1 and 25000: Number of people 1972
Income between 25001 and 50000: Number of people 10198
Income between 50001 and 75000: Number of people 8532
Income between 75001 and 100000: Number of people
4199Income between 100001 and inf: Number of people
3737 #min and max loan amount
max_loan_amount =
credit['loan_amnt'].max()
min_loan_amount =
credit['loan_amnt'].min()

print(f"maximum Loan Amount {max_loan_amount}")
```

```
print(f"minimum Loan Amount {min_loan_amount}")

# people with an income between x

and ydef loan_amount_group(arr):

lenarr = len(arr)

for i in range(0,lenarr-1):

next = arr[i]+1

num_people = credit['loan_amnt'].between(next,arr[i+1]).sum()

print(f'Loan Amount between {next} and {arr[i+1]}: Number of

people
{num_people}')

loan_amount_group([0, 5000, 10000, 15000, float('inf')])

#finding outliers

numeric_data =

credit.select_dtypes(include=['number'])x =

numeric_data.drop(['loan_status'], axis=1)

y = credit['loan_status']

model =

LinearRegression()

model.fit(x, y)

y_pred = model.predict(x)

# Calculating the mean squared

error mse =

mean_squared_error(y, y_pred)

# Detecting outliers using the Elliptic Envelope
```

```
method outlier_detector =
EllipticEnvelope(contamination=0.1)
outlier_detector.fit(x)
# Identifying outliers
outliers = outlier_detector.predict(x)
== -1 # Removing outliers from the
dataset
data =
credit[~outliers]#
Printing the results
print("Mean Squared Error:",
mse)print("Outliers:",
sum(outliers))
# Columns with suspected outliers
columns_with_outliers = ['person_age', 'person_income', 'person_emp_length',
'loan_amnt', 'loan_percent_income']
# Creating box plots for each
column for column in
columns_with_outliers:
plt.figure(figsize=(8, 6))
sns.boxplot(x=credit[column])
plt.title(f'Boxplot of {column}')
plt.xlabel(column)
plt.show()
```

```
# Define a function to remove outliers using Tukey's
methoddef remove_outliers(credit):
cleaned_df = credit

# Iterate through each
columnfor column in
credit.columns:
if
pd.api.types.is_numeric_dtype(credit[column])
:Q1 = credit[column].quantile(0.25)
Q3 = credit[column].quantile(0.75)
IQR = Q3 - Q1
 # Define outlier
 boundsk = 1.5
        lower_bound = Q1 - (k *
        IQR)upper_bound = Q3 +
        (k * IQR)

# Remove outliers
cleaned_df = cleaned_df[(cleaned_df[column] >= lower_bound) &
(cleaned_df[column] <= upper_bound)]

return cleaned_df

# Call the function to remove outliers from all columns
filtered_df = remove_outliers(credit)
# Statistical Summary
```

```
print("Before Outlier
Removal:")credit.describe()

#person_age max 144 (issue)
#person_emp_length max 123
(issue)credit.reset_index(inplace
= True)
verti =
credit['person_age'].value_counts().valueshori
= credit['person_age'].value_counts().index fig
= plt.figure(figsize = (15, 5))
plt.bar(hori,
verti)
plt.show()
# after 80 it is rare
(credit['person_age'].value_counts().values>90).su
m()# dropping rows that have age greater than 9
# Box Plots
plt.figure(figsize=(25
, 7))
plt.subplot(1, 2, 2)
sns.boxplot(data=filtered_df)
plt.title('Box Plot - After Outlier
Removal')plt.tight_layout()
plt.show()
```

```
# ENCODING
 # Drop columns with missing
 valuescredit.dropna(axis=1,
 inplace=True)
 from sklearn.preprocessing import
 LabelEncoder# Create LabelEncoder object
 label_encoder = LabelEncoder()
 # Apply label encoding to each categorical column
 categorical_cols = ['person_home_ownership', 'loan_intent', 'loan_grade',
'cb_person_default_on_file']
 for col in categorical_cols:
 credit[col] =
 label_encoder.fit_transform(credit[col])# Check
 the updated DataFrame
 credit.head()
 Train & test split
 # Split data into features (X) and target variable
 (y)X = credit.drop('loan_status', axis=1)
 y = credit['loan_status']
 X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
 Model selection
 XGB CLASSIFIER
 # Perform GridSearchCV for hyperparameter
```

```
tuningxgbmodel = xgb.XGBClassifier()

param_dist = {

"max_depth":

[3, 5],

"min_child_weight": [3, 6],

"n_estimators": [100],

"learning_rate": [0.05, 0.16]

}

grid_search = GridSearchCV(xgbmodel, param_grid=param_dist, cv=3,
verbose=10, n_jobs=-1, scoring="f1")

grid_search.fit(X_train, Y_train)

# Get the best estimator from

GridSearchCVbest_model =

grid_search.best_estimator_


# Train the model with the best hyperparameters on the

training seteval_set = [(X_train, Y_train), (X_test, Y_test)]

best_model.fit(X_train, Y_train, verbose=True, eval_metric=["error", "auc"],
eval_set=eval_set)

# Plotting the training and test errors

results = best_model.evals_result()

error_train =

results['validation_0']['error']error_test

= results['validation_1']['error']

plt.plot(range(len(error_train)), error_train, label='Training Error')
```

```
plt.plot(range(len(error_test)), error_test, label='Test Error')
plt.xlabel('Iterations')
plt.ylabel('Error'
)plt.legend()
plt.title('Training and Test
Errors')plt.show()
# Predict on the test set
y_pred =
best_model.predict(X_test)#
Generate classification report
class_report = classification_report(Y_test, y_pred)
print("\nClassification Report:")
print(class_report)
# Generate confusion matrix
conf_matrix = confusion_matrix(Y_test,
y_pred)# Plotting the confusion matrix as a
heatmap plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
xticklabels=['Predicted 0', 'Predicted 1'],
yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted
Label')plt.ylabel('True
Label')
plt.title('Confusion
```

Matrix')plt.show()

GLM (Logistic regression)

Add a constant term for the

interceptX_train =

sm.add_constant(X_train)

Fit a GLM (logistic regression) to the training data

glm_model = sm.GLM(Y_train, X_train, family=sm.families.Binomial())

glm_results = glm_model.fit()

Display the summary of the

model

print(glm_results.summary())

Predictions on test set

X_test =

sm.add_constant(X_test) y_pred

= glm_results.predict(X_test)

Example of evaluating performance (e.g., accuracy)

y_pred_class = np.where(y_pred > 0.5, 1, 0) # Convert predicted probabilities toclasses

accuracy = np.mean(y_pred_class ==

Y_test)print(f"Accuracy: {accuracy}")

LOGISTIC REGRESSION

Initialize and fit the logistic regression model

logistic_model = LogisticRegression()

logistic_model.fit(X_train, Y_train)

```
# Make predictions on the test set
y_pred =
logistic_model.predict(X_test)#
Confusion matrix
conf_matrix = confusion_matrix(Y_test,
y_pred)print("XGBoost Confusion Matrix:")
# Plotting the confusion matrix as a
heatmapplt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
xticklabels=['Predicted 0', 'Predicted 1'],
yticklabels=['Actual 0', 'Actual
1'])plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
# Classification report
class_report = classification_report(Y_test,
y_pred)print("\nlogistic regression Report:")
print(class_report)
plt.show()
# Calculate accuracy
accuracy = accuracy_score(Y_test,
y_pred)print(f"Accuracy: {accuracy}")
```

XGBoost

classifier# Initialize and fit the XGBoost

classifier xgb_model =

xgb.XGBClassifier()

xgb_model.fit(X_train, Y_train)

Predict on the test set

y_pred =

xgb_model.predict(X_test)#

Confusion matrix

conf_matrix = confusion_matrix(Y_test,

y_pred)print("XGBoost Confusion Matrix:")

Plotting the confusion matrix as a

heatmapplt.figure(figsize=(8, 6))

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',

xticklabels=['Predicted 0', 'Predicted 1'],

yticklabels=['Actual 0', 'Actual

1'])plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

Classification report

class_report = classification_report(Y_test, y_pred)

print("\nXGBoost Classification Report:")

print(class_report)

plt.show()

```
# Calculate accuracy

accuracy = accuracy_score(Y_test,

y_pred)print(f"Accuracy: {accuracy}")
```

LightGBM

```
classifier# Initialize and fit the LightGBM

classifier lgb_model =

lgb.LGBMClassifier()

lgb_model.fit(X_train, Y_train)

# Predict on the test set

y_pred =

lgb_model.predict(X_test)#

Confusion matrix

conf_matrix = confusion_matrix(Y_test, y_pred)

print("LightGBM Confusion Matrix:")

# Plotting the confusion matrix as a

heatmapplt.figure(figsize=(8, 6))

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
xticklabels=['Predicted 0', 'Predicted 1'],

yticklabels=['Actual 0', 'Actual

1'])plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

# Classification report
```

class_report = classification_report(Y_test, y_pred)

print("\nLightGBM Classification Report:")

print(class_report)

Calculate accuracy

accuracy = accuracy_score(Y_test,

y_pred)print(f"Accuracy: {accuracy}")

Random Forest

classifier# Initialize and fit the Random Forest

classifier rf_model = RandomForestClassifier()

rf_model.fit(X_train, Y_train)

Predict on the test set

y_pred =

rf_model.predict(X_test)#

Confusion matrix

conf_matrix = confusion_matrix(Y_test, y_pred)

print("Random Forest Confusion Matrix:")

Plotting the confusion matrix as a

heatmapplt.figure(figsize=(8, 6))

sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',

xticklabels=['Predicted 0', 'Predicted 1'],

yticklabels=['Actual 0', 'Actual

1'])plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

```
plt.show()

# Classification report

class_report = classification_report(Y_test, y_pred)

print("\nRandom Forest Classification Report:")

print(class_report)

# Plotting a tree from Random Forest

plt.figure(figsize=(20, 10))

tree.plot_tree(rf_model.estimators_[0], filled=True)

plt.show()
```

ElasticNet regression

```
from sklearn.linear_model import ElasticNet

from sklearn.metrics import mean_squared_error, r2_score

# Standardize the features (optional but recommended for regularized

regression)scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Create the ElasticNet regression model

elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)  # You can adjust alpha and
l1_ratio as needed

# Fit the model to the training data

elastic_net.fit(X_train_scaled,

Y_train)# Make predictions

y_pred =

elastic_net.predict(X_test_scaled)#
```

Evaluate the model

mse = mean_squared_error(Y_test,

y_pred)r2 = r2_score(Y_test, y_pred)

print(f"Mean Squared Error (MSE):

{mse}")print(f"R-squared (R2): {r2}")

Lasso regression

from sklearn.linear_model import Lasso

Standardize the features (optional but recommended for regularized

regression)scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

Create the Lasso regression model

lasso = Lasso(alpha=0.1) # You can adjust alpha as

needed# Fit the model to the training data

lasso.fit(X_train_scaled, Y_train)

Make predictions

y_pred =

lasso.predict(X_test_scaled)#

Evaluate the model

mse = mean_squared_error(Y_test, y_pred)

r2 = r2_score(Y_test, y_pred)

print(f"Mean Squared Error (MSE):

{mse}")print(f"R-squared (R2): {r2}")

```
from sklearn.tree import
DecisionTreeClassifierfrom catboost import
CatBoostClassifier
# Train decision trees (you can tune
hyperparameters)tree1 =
DecisionTreeClassifier(max_depth=3)
tree2 =
DecisionTreeClassifier(max_depth=5)
tree1.fit(X_train, Y_train)
tree2.fit(X_train, Y_train)
# Get predictions from decision trees
tree1_preds =
tree1.predict_proba(X_test)
tree2_preds =
tree2.predict_proba(X_test)
# Train CatBoost (you can tune hyperparameters)
catboost_model = CatBoostClassifier(iterations=100, depth=6,
learning_rate=0.1)catboost_model.fit(X_train, Y_train, verbose=False)
# Get predictions from CatBoost
catboost_preds = catboost_model.predict_proba(X_test)
# Combine predictions (for example, averaging for classification)
ensemble_preds = (tree1_preds + tree2_preds +
catboost_preds) / 3# Evaluate ensemble model
ensemble_accuracy = accuracy_score(Y_test, np.argmax(ensemble_preds, axis=1))
print(ensemble_accuracy)
```

Neural network# Define the neural

network architecture NNmodel =

Sequential([

Dense(64, input_shape=(X_train.shape[1],), activation='relu'),

Dense(32, activation='relu'),

Dense(16, activation='relu'), # Additional hidden

layer Dense(8, activation='relu'), # Additional

hidden layer Dense(1, activation='sigmoid')

])

Compile the model

NNmodel.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

Define TensorBoard callback with an absolute log directory path

log_dir = os.path.join(r"D:\Mo Project", "logs") # Use 'r' before the string to treatit as a raw string

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

# Define other callbacks (EarlyStopping and ModelCheckpoint)
callbacks = [

tf.keras.callbacks.EarlyStopping(patience=3, monitor='val_loss'),

tf.keras.callbacks.ModelCheckpoint(filepath='best_model.keras', monitor='val_accuracy', save_best_only=True)

]

Train the model with TensorBoard callback

history = NNmodel.fit(X_train, Y_train, epochs=20, batch_size=32,

validation_data=(X_test, Y_test),

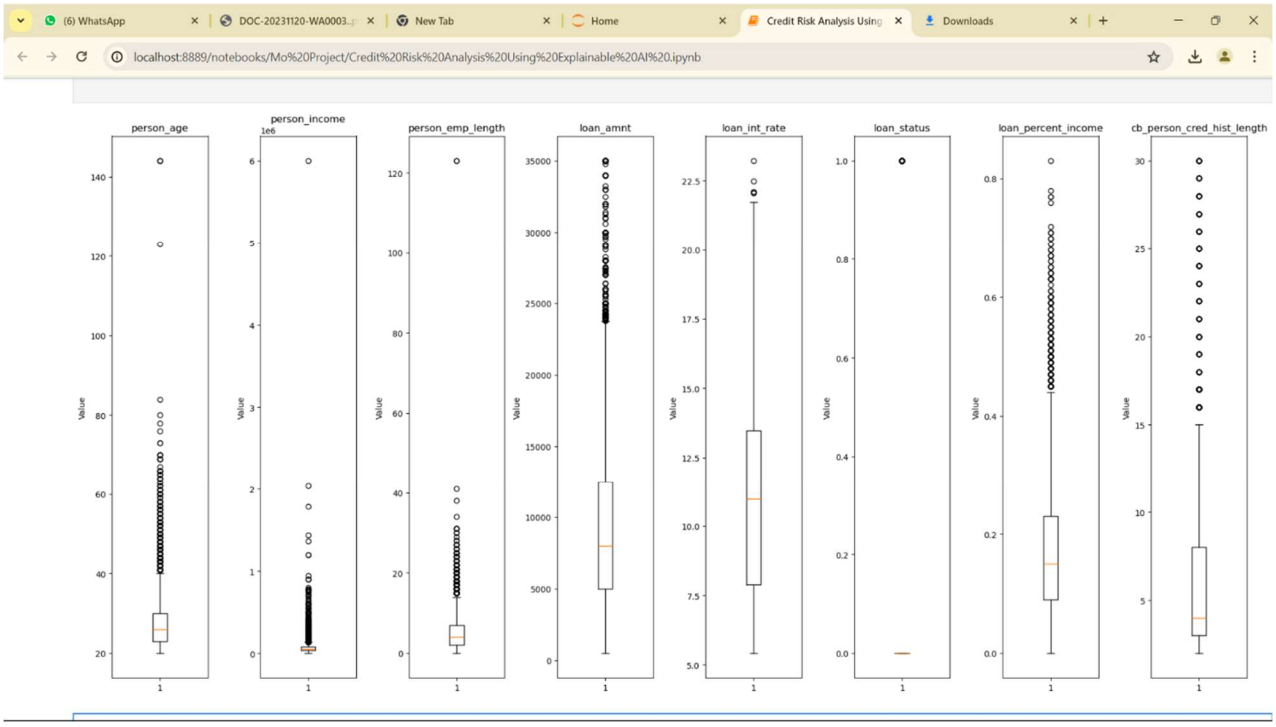
```
callbacks=[tensorboard_callback] +
callbacks)# Evaluate the model on the test
set
test_loss, test_accuracy = NNmodel.evaluate(X_test, Y_test)
print(f"Test Accuracy: {test_accuracy}")
```

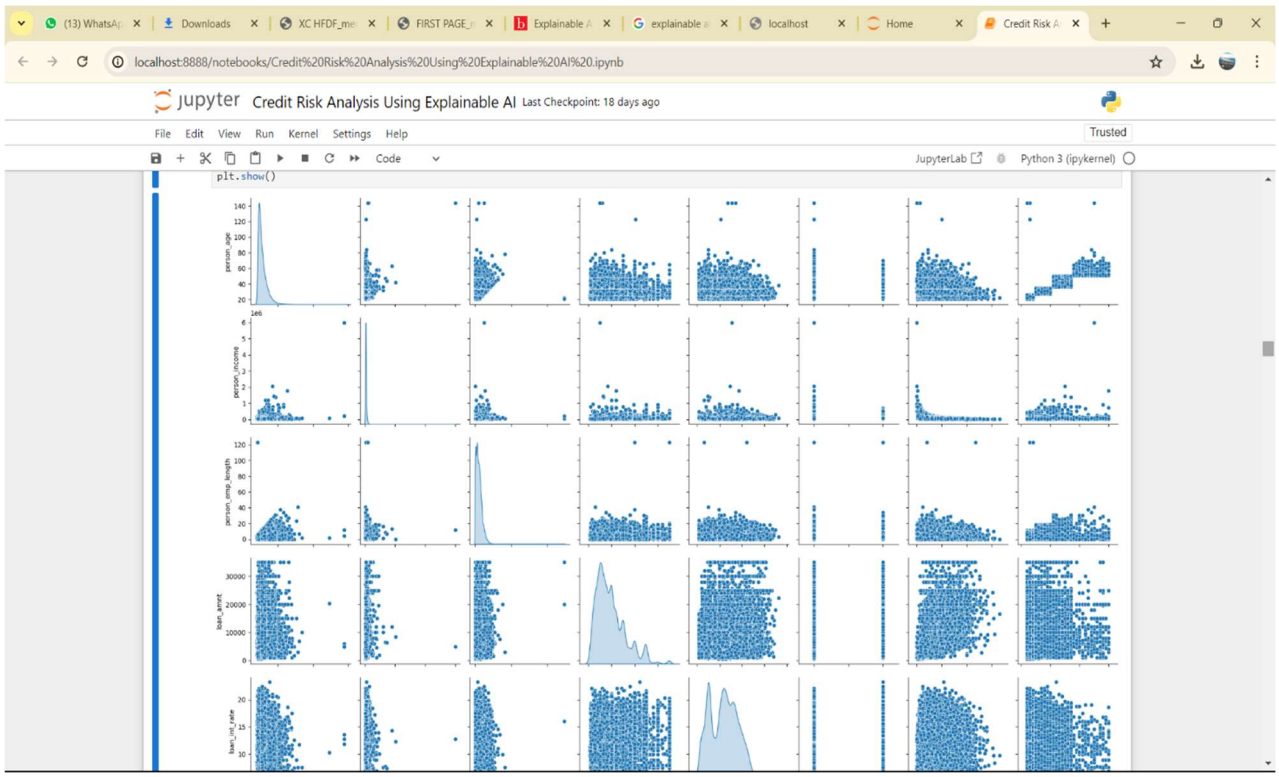
10.LIME

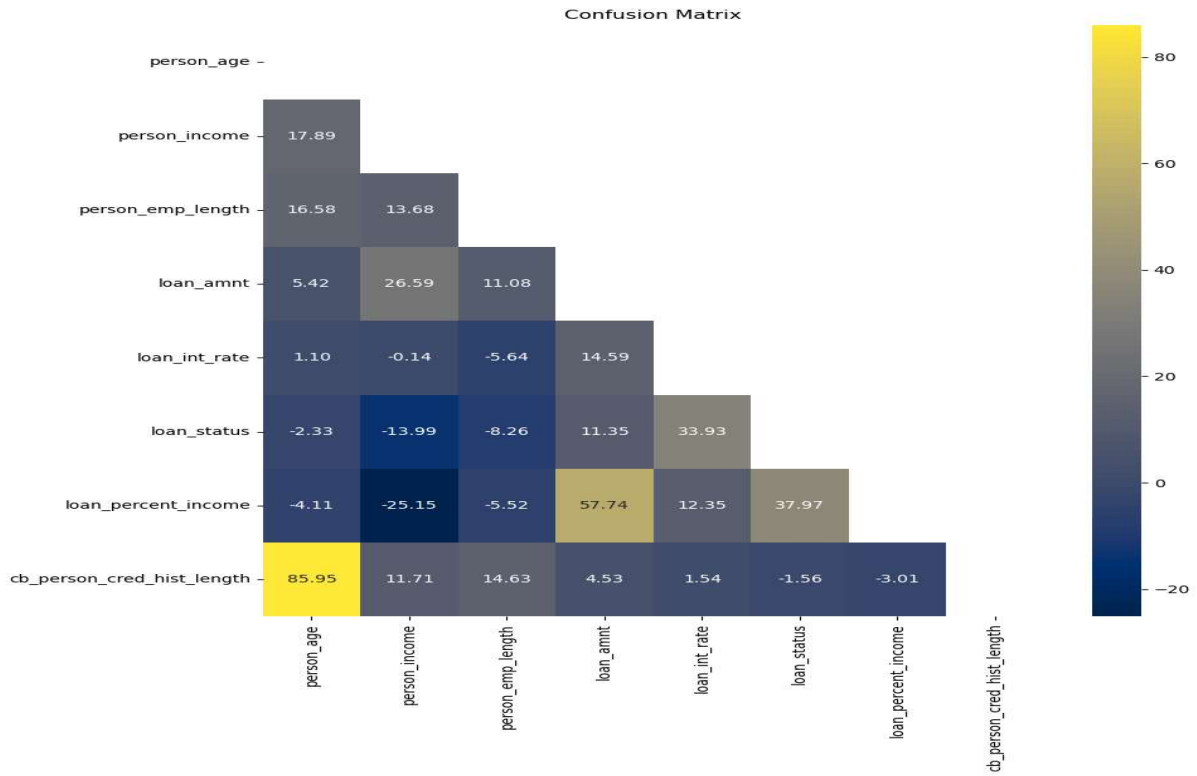
```
Model only used for
LIMEA: Logistic
model.
B: XG
Boost. C:
Light
BGM.
D: Random Forest.
 Initialize LIME explainer
explainer =
lime.lime_tabular.LimeTabularExplainer(
X_train.values,
feature_names=X_train.columns.tolist(),
class_names=['Non-Risky', 'Risky'],  # Replace with your class names
discretize_continuous=True
)
# Choose an instance from the test set to explain
```

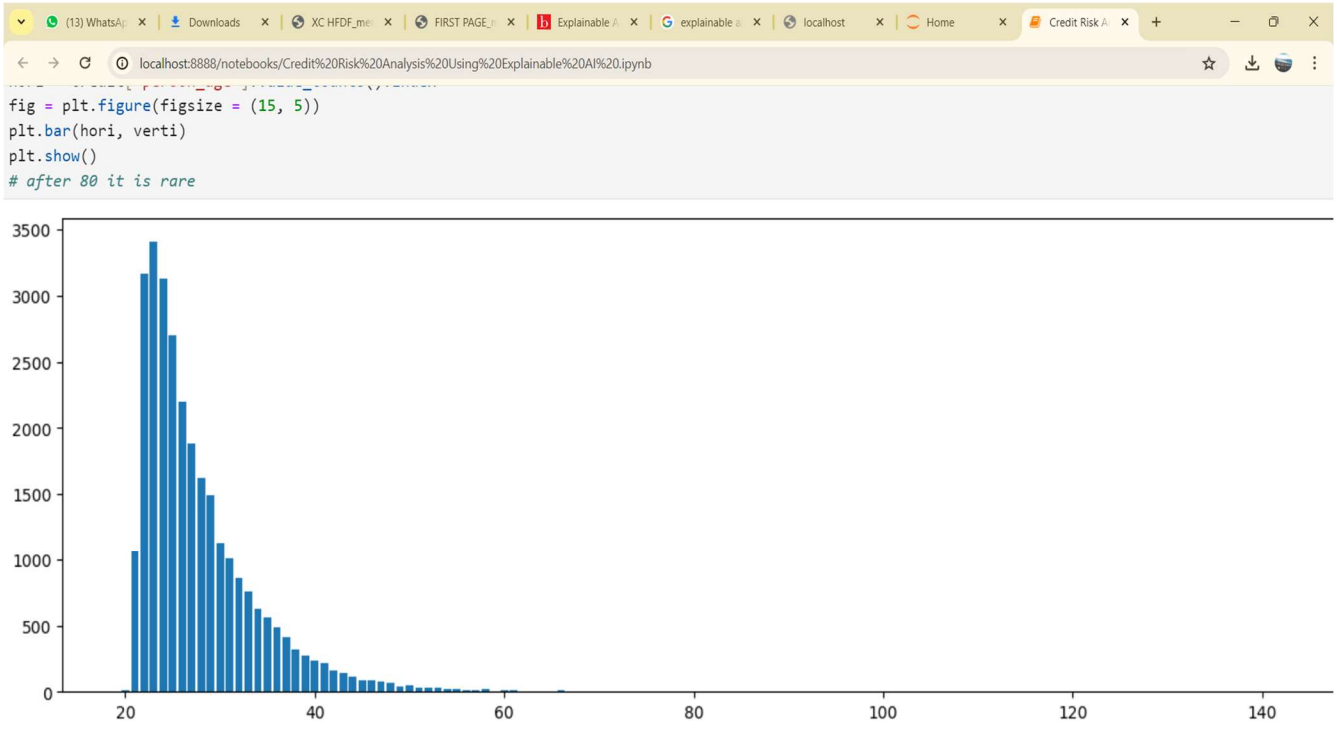
```
instance_to_explain = X_test.iloc[11]
# Step 4: Explain Predictions and Visualize
Explanations# Generate explanation for the selected
instance explanation = explainer.explain_instance(
instance_to_explain,
 rf_model.predict_proba,
  num_features=5
 )
# Visualize explanations
explanation.show_in_notebook() # Replace with appropriate visualization
method# Generate the bar chart without the 'title' argument
explanation.as_pyplot_figure(label=True)
# If you need a title, add it separately using matplotlib:
import matplotlib.pyplot as plt
plt.title("LIME Explanation (Bar
Chart)")plt.show()
```

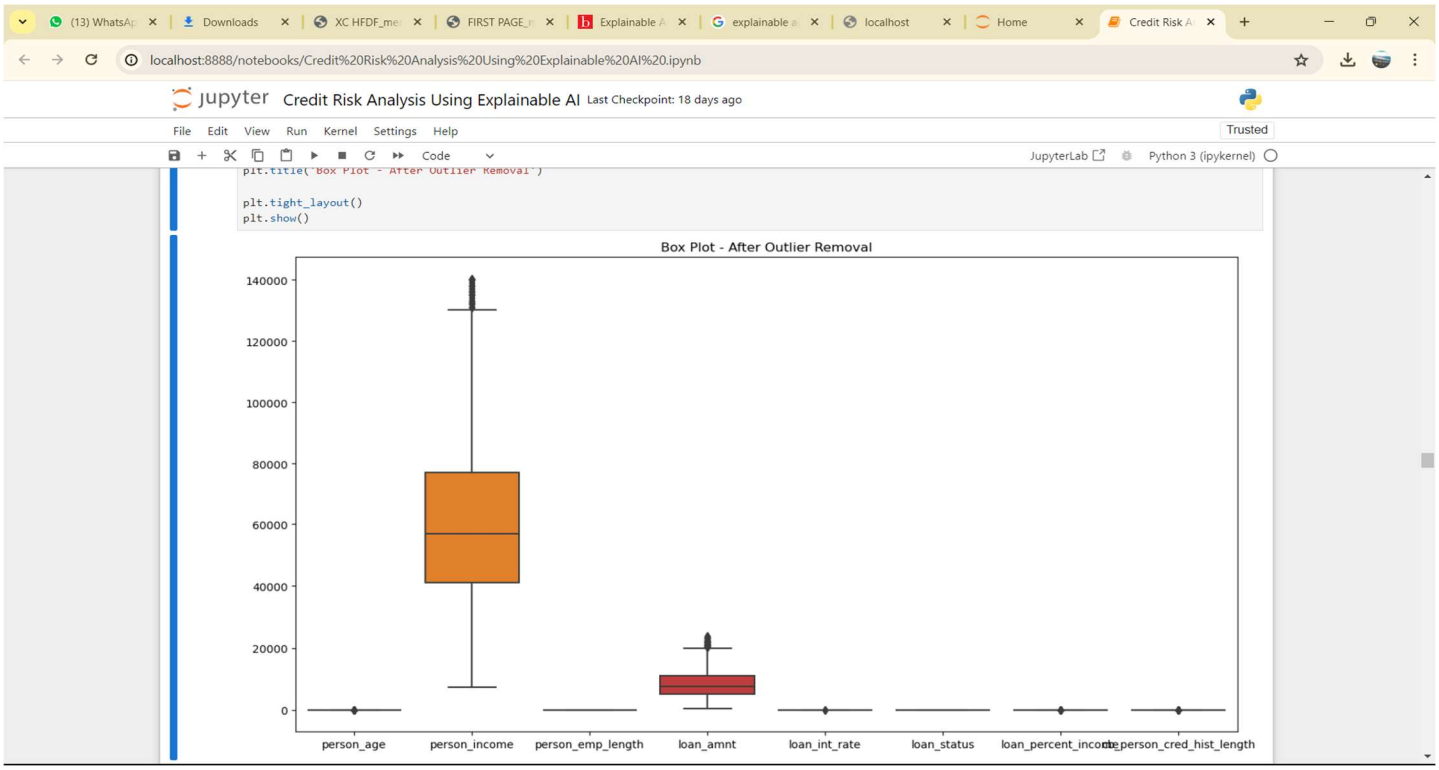
7.1 RESULT

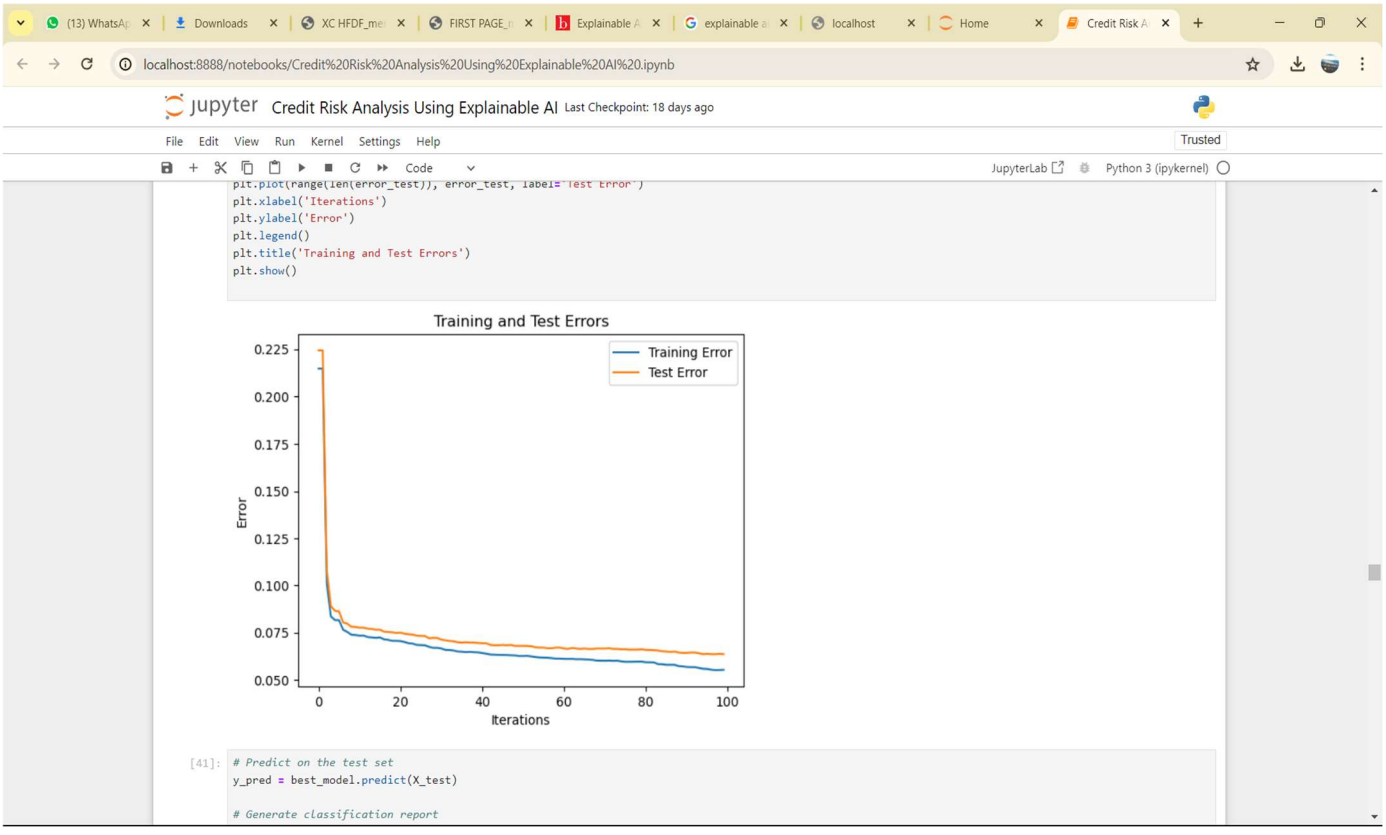

localhost:8889/notebooks/Mo%20Project/Credit%20Risk%20Analysis%20Using%20Explainable%20AI%20.ipynb
person_age
person_income
person_emp_length
loan_amnt
loan_int_rate
loan_status
loan_percent_income
cb_person_cred_hist_length
Value

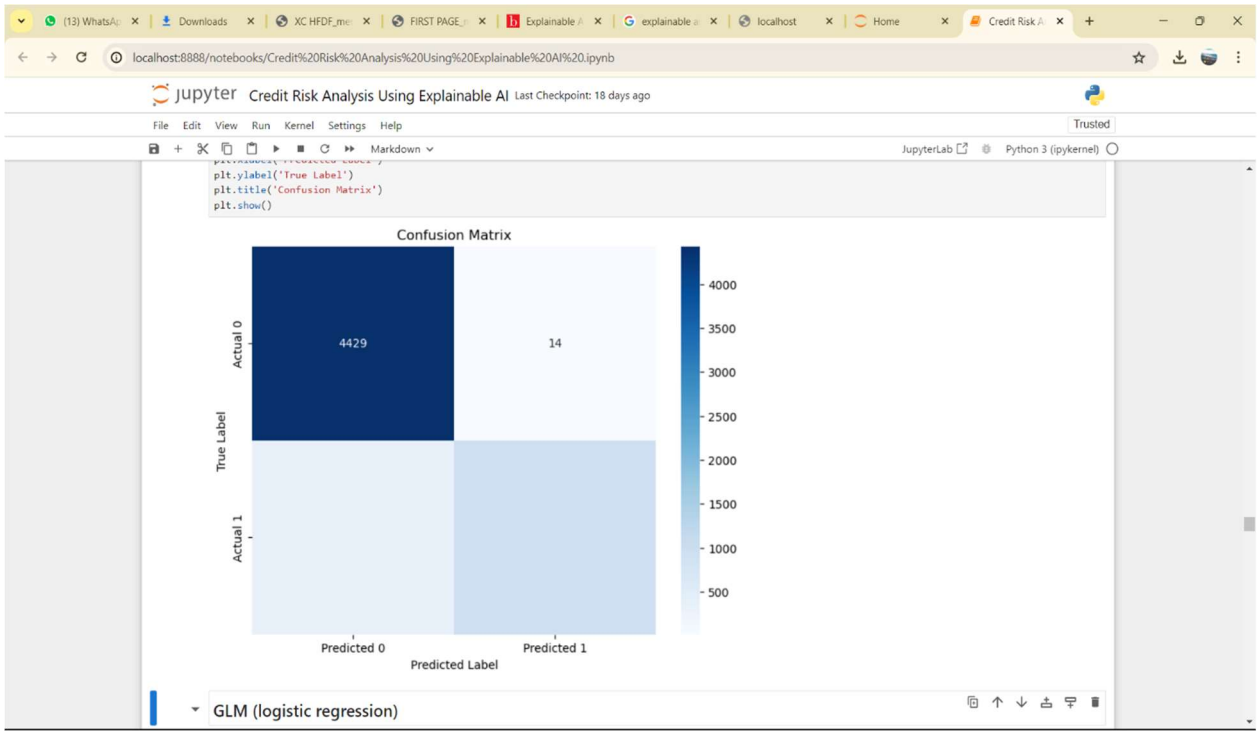

localhost:8888/notebooks/Credit%20Risk%20Analysis%20Using%20Explainable%20AI%20.ipynb
jupyter
Credit Risk Analysis Using Explainable AI
Last Checkpoint: 18 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
plt.show()

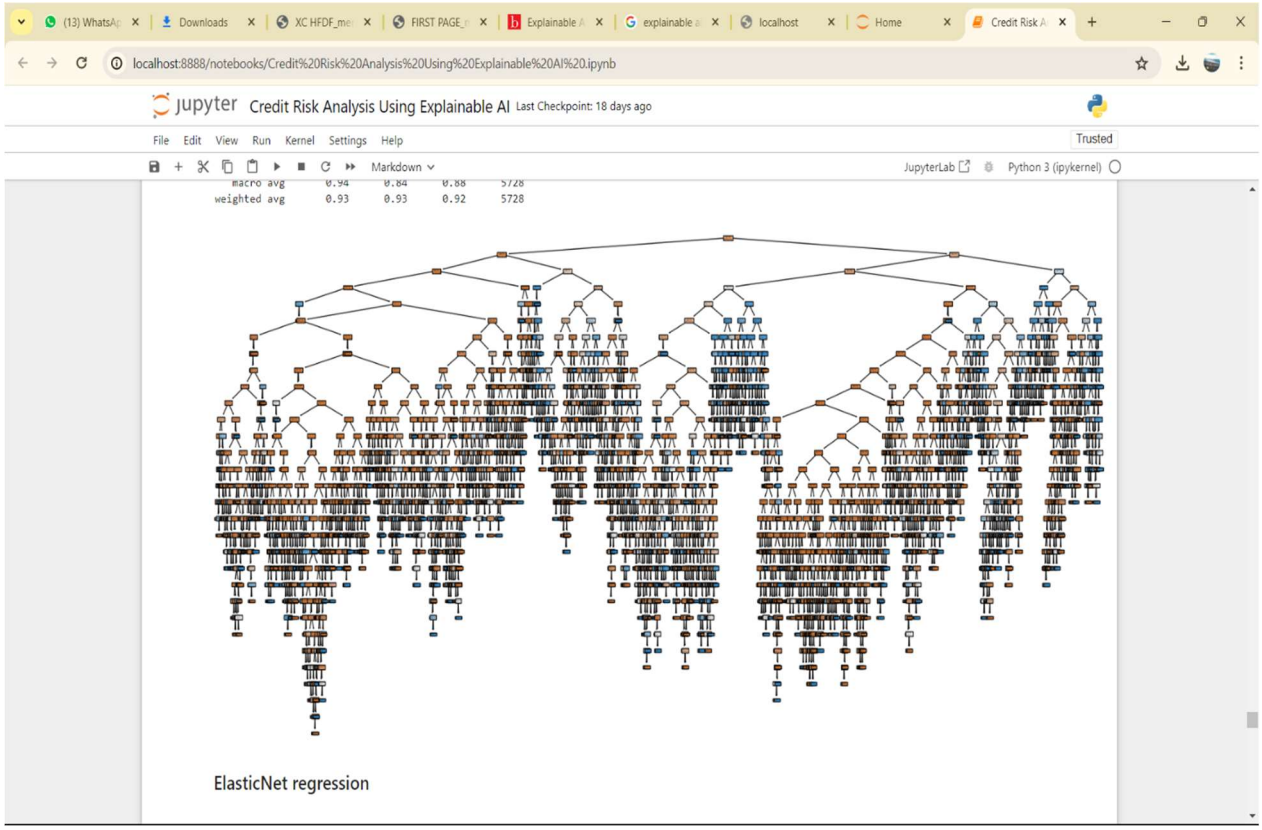

Confusion Matrix
person_age
person_income
person_emp_length
loan_amnt
loan_int_rate
loan_status
loan_percent_income
cb_person_cred_hist_length
17.89
16.58 13.68
5.42 26.59 11.08
1.10 -0.14 -5.64 14.59
-2.33 -13.99 -8.26 11.35 33.93
-4.11 -25.15 -5.52 57.74 12.35 37.97
85.95 11.71 14.63 4.53 1.54 -1.56 -3.01
80
60
40
20
0
−20

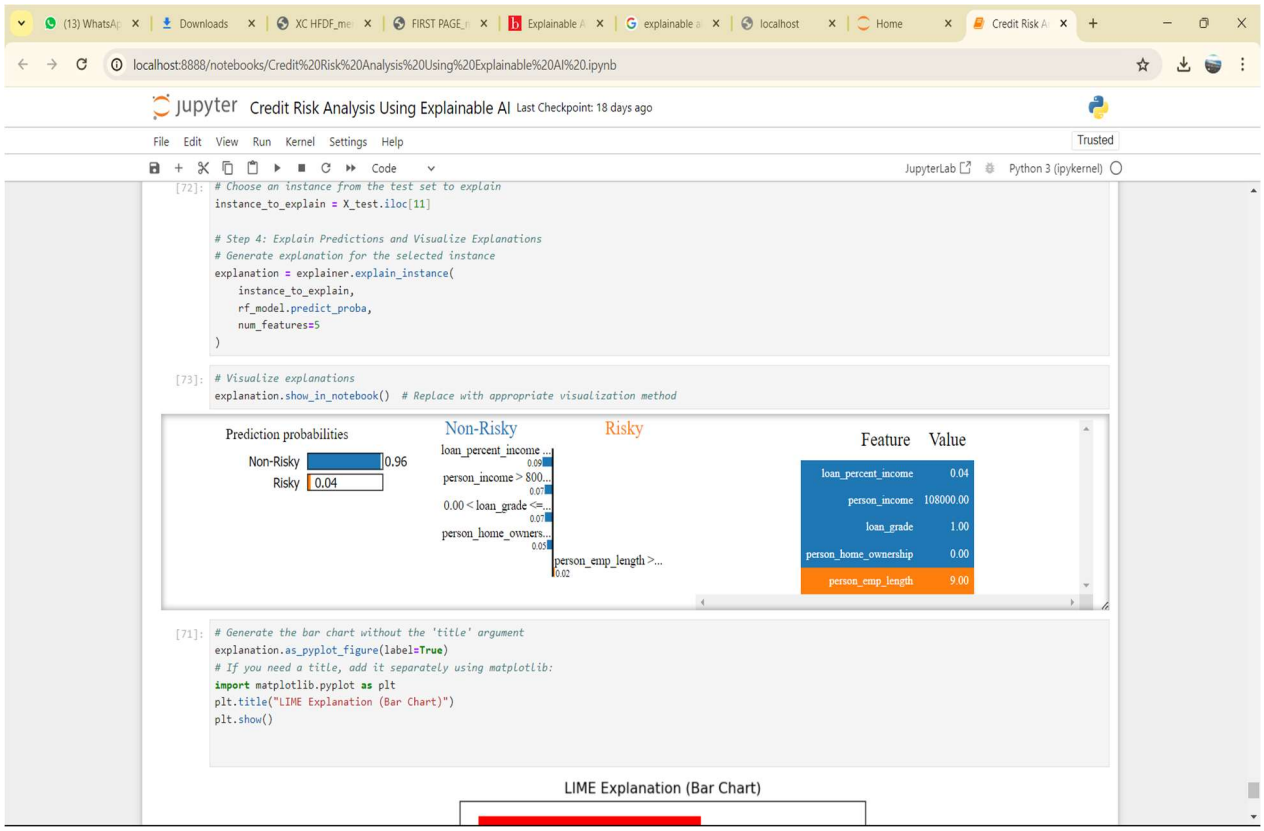

localhost:8888/notebooks/Credit%20Risk%20Analysis%20Using%20Explainable%20AI%20.ipynb
fig = plt.figure(figsize = (15, 5))
plt.bar(hori, verti)
plt.show()
# after 80 it is rare
3500
3000
2500
2000
1500
1000
500
0
20
40
60
80
100
120
140
(credit['person_age'].value_counts().values>90).sum()
dropping rows that have age greater than 9

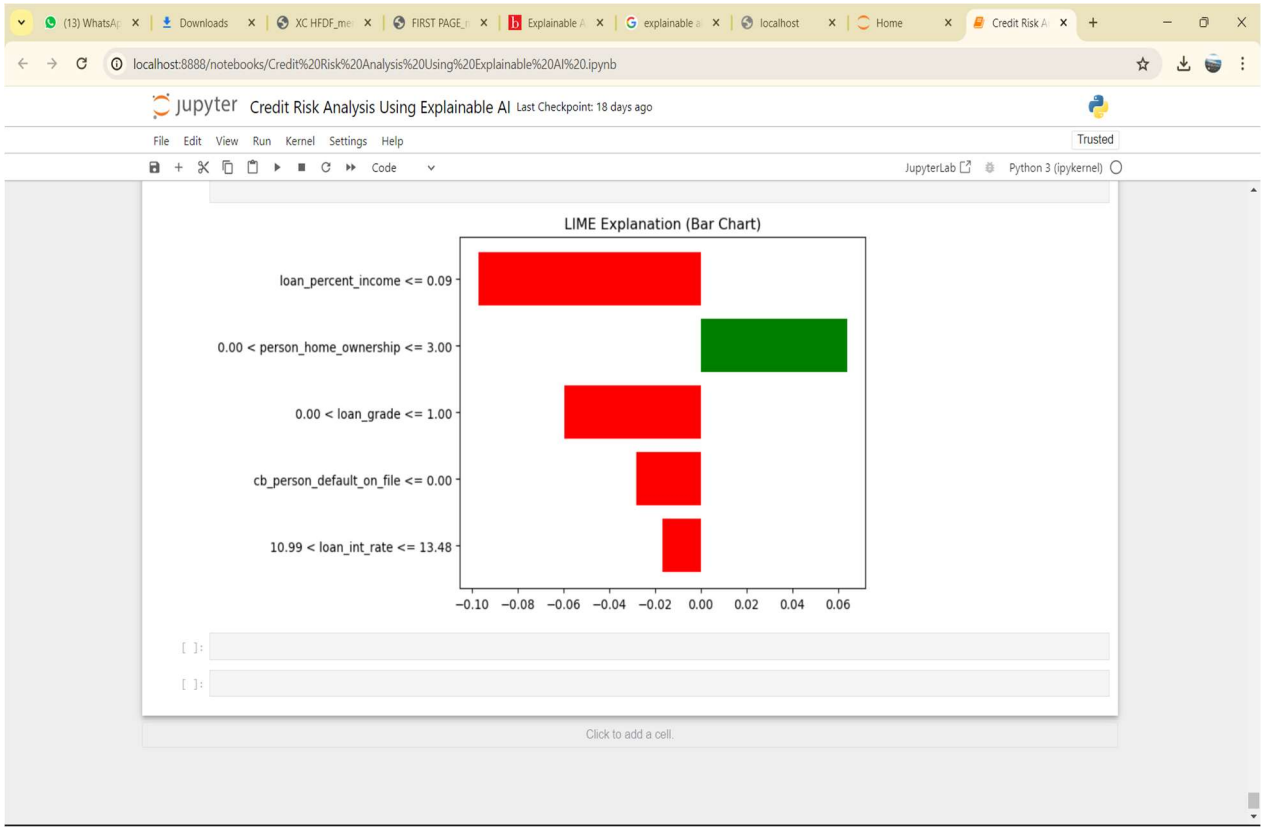

localhost:8888/notebooks/Credit%20Risk%20Analysis%20Using%20Explainable%20AI%20.ipynb
jupyter Credit Risk Analysis Using Explainable AI Last Checkpoint: 18 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
plt.tight_layout()
plt.show()
Box Plot - After Outlier Removal
140000
120000
100000
80000
60000
40000
20000
0
person_age
person_income
person_emp_length
loan_amnt
loan_int_rate
loan_status
loan_percent_income
person_cred_hist_length


localhost:8888/notebooks/Credit%20Risk%20Analysis%20Using%20Explainable%20AI%20.ipynb
jupyter Credit Risk Analysis Using Explainable AI Last Checkpoint: 18 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
plt.xlabel('Iterations')
plt.ylabel('Error')
plt.legend()
plt.title('Training and Test Errors')
plt.show()
Training and Test Errors
Training Error
Test Error
Error
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0
20
40
60
80
100
Iterations
[41]:
# Predict on the test set
y_pred = best_model.predict(X_test)
# Generate classification report


Credit Risk Analysis Using Explainable AI
Confusion Matrix
4429
14
Actual 0
Actual 1
True Label
Predicted 0
Predicted 1
Predicted Label
4000
3500
3000
2500
2000
1500
1000
500
GLM (logistic regression)


Credit Risk Analysis Using Explainable AI
macro avg 0.94 0.84 0.88 5728
weighted avg 0.93 0.93 0.92 5728
ElasticNet regression


Credit Risk Analysis Using Explainable AI
Last Checkpoint: 18 days ago
Trusted
Python 3 (ipykernel)
[72]:
# Choose an instance from the test set to explain
instance_to_explain = X_test.iloc[11]
# Step 4: Explain Predictions and Visualize Explanations
# Generate explanation for the selected instance
explanation = explainer.explain_instance(
instance_to_explain,
rf_model.predict_proba,
num_features=5
)
[73]:
Visualize explanations
explanation.show_in_notebook() # Replace with appropriate visualization method
Prediction probabilities
Non-Risky
Risky
0.96
0.04
Non-Risky
Risky
loan_percent_income ...
person_income > 800...
0.00 < loan_grade <=...
person_home_owners...
person_emp_length >...
Feature
Value
loan_percent_income 0.04
person_income 108000.00
loan_grade 1.00
person_home_ownership 0.00
person_emp_length 9.00
[71]:
# Generate the bar chart without the 'title' argument
explanation.as_pyplot_figure(label=True)
# If you need a title, add it separately using matplotlib:
import matplotlib.pyplot as plt
plt.title("LIME Explanation (Bar Chart)")
plt.show()
LIME Explanation (Bar Chart)


Credit Risk Analysis Using Explainable AI
Last Checkpoint: 18 days ago
Trusted
Python 3 (ipykernel)
LIME Explanation (Bar Chart)
loan_percent_income <= 0.09
0.00 < person_home_ownership <= 3.00
0.00 < loan_grade <= 1.00
cb_person_default_on_file <= 0.00
10.99 < loan_int_rate <= 13.48
−0.10 −0.08 −0.06 −0.04 −0.02 0.00 0.02 0.04 0.06
[ ]:
[ ]:
Click to add a cell.

CHAPTER 8

CONCLUSION

The core findings of this study revealed the significant impact of data preprocessing on model performance. Efficient handling of missing values, categorical encoding, and scaling proved pivotal in enhancing the quality of input data for subsequent modelling. The comparative analysis of machine learning algorithms—Logistic Regression, XGBoost,LightGBM, and Random Forest Classifier—highlighted their varying performances in predicting .Addressing outliers through Elliptical Envelopes and Tukey's method significantly improved the models' robustness by mitigating the influence of anomalous data points. Additionally, the integration of LIME technique for model interpretability provided valuable insights into individual predictions and feature importance, empowering better understanding and trust in the model's decision-making.

Answering the first research question, we find that LIME methodologie significantly enhance the clarity and transparency of complex ML models for external customers. By breaking down predictions into individual feature contributions, these methodologies demystify the model’s decision-making process and provide tangible insights. As demonstrated through the visualizations and discussions in this paper, LIME and SHAP allow customers to see exactly which factors have influenced a credit decision, there by making the opaque workings of sophisticated ML models more accessible.

8.1 FUTURE ENHANCEMENT

Future enhancements can focus on the development of an interactive tool, like a dashboard that will allow customers to engage with the model's predictions more directly. This dashboard will provide a hands-on experience, enabling users to modify feature values and immediately see the impact on their credit risk assessment. Concurrently, efforts can be made fine-tune the delivery of explanations and personalized financial advice. This shall involve identifying the most effective ways to communicate complex credit information to customers. Improving these communication strategies, we can help customers make more informed financial decisions and enhance their trust in AI-driven credit evaluations.

REFERENCES

[1] Bastos, J. A., & Matos, S. M. (2022). Explainable models of credit losses. European Journal of Operational Research, 301(1), 386-394.

[2] Biecek, P., Chlebus, M., Gajda, J., Gosiewska, A., Kozak, A., Ogonowski, D., ... & Wojewnik, P. (2021). Enabling machine learning algorithms for credit scoring--explainable artificial intelligence (XAI) methods for clear understanding complex predictive models. arXiv preprint arXiv:2104.06735.

[3] Burgt, J. V. D. (2020). Explainable AI in banking. Journal of Digital Banking, 4(4), 344-350.

[4]Bussmann, N., Giudici, P., Marinelli, D., & Papenbrock, J. (2021). Explainable machinelearning in credit risk management. Computational Economics, 57, 203-216.

[5] Davis, R., Lo, A. W., Mishra, S., Nourian, A., Singh, M., Wu, N., & Zhang, R. (2022).Explainable machine learning models of consumer credit risk. Available at SSRN 4006840.

[6] De Lange, P. E., Melsom, B., Vennerød, C. B., & Westgaard, S. (2022). Explainable AI for Credit Assessment in Banks. Journal of Risk and Financial Management, 15(12), 556.

[7] Demajo, L. M., Vella, V., & Dingli, A. (2020). Explainable ai for interpretable credit scoring. arXiv preprint arXiv:2012.03749.

[8] De Lange, P. E., Melsom, B., Vennerød, C. B., & Westgaard, S. (2022). Explainable AI for Credit Assessment in Banks. Journal of Risk and Financial

Management, 15(12), 556.

[9] Dessain, J., Bentaleb, N., & Vinas, F. (2023, July). Cost of Explainability in AI: An Example with Credit Scoring Models. In World Conference on Explainable Artificial Intelligence (pp. 498-516). Cham: Springer Nature Switzerland

[10] Egan, C. (2021). Improving Credit Default Prediction Using Explainable AI (Doctoraldissertation, Dublin, National College of Ireland).

[11] Fritz-Morgenthal, S., Hein, B., & Papenbrock, J. (2022). Financial risk management and explainable, trustworthy, responsible AI. Frontiers in artificial intelligence, 5, 779799.

[12] Gramespacher, T., & Posth, J. A. (2021). Employing explainable AI to optimize the return target function of a loan portfolio. Frontiers in Artificial Intelligence, 4, 693022.

[13] Heng, Y. S., & Subramanian, P. (2022, October). A Systematic Review of Machine Learning and Explainable Artificial Intelligence (XAI) in Credit Risk Modelling. In Proceedings of the Future Technologies Conference (pp. 596-614). Cham: Springer International Publishing.

[14] Hu, B., & Wu, Y. (2023). Unlocking Causal Relationships in Commercial Banking RiskManagement: An Examination of Explainable AI Integration with Multi-Factor Risk Models. Journal of Financial Risk Management, 12(3), 262-274.

[15] Kuiper, O., van den Berg, M., van der Burgt, J., & Leijnen, S. (2022). Exploring explainable AI in the financial sector: Perspectives of banks and supervisory

authorities. In Artificial Intelligence and Machine Learning: 33rd Benelux Conference on Artificial Intelligence, BNAIC/Benelearn 2021, Esch-sur-Alzette, Luxembourg, November 10–12, 2021, Revised Selected Papers 33 (pp. 105-119). Springer International Publishing.

[16] Li, W., Paraschiv, F., & Sermpinis, G. (2022). A data-driven explainable case-based reasoning approach for financial risk detection. Quantitative Finance, 22(12), 2257-2274.

[17] Misheva, B. H., Osterrieder, J., Hirsa, A., Kulkarni, O., & Lin, S. F. (2021). ExplainableAI in credit risk management. arXiv preprint arXiv:2103.00949.

[18] Pandey, T. N., Jagadev, A. K., Mohapatra, S. K., & Dehuri, S. (2017, August). Credit risk analysis using machine learning classifiers. In 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) (pp. 1850-1854). IEEE.

[19] Tyagi, S. (2022). Analyzing Machine Learning Models for Credit Scoring with Explainable AI and Optimizing Investment Decisions. arXiv preprint arXiv:2209.093

[20] Walambe, R., Kolhatkar, A., Ojha, M., Kademani, A., Pandya, M., Kathote, S., & Kotecha, K. (2020, December). Integration of explainable AI and blockchain for secure storage of human readable justifications for credit risk assessment. In International Advanced Computing Conference (pp. 55-72). Singapore: Springer Singapore.