**“FRAUD DETECTION IN BANKING DATA BY**

**MACHINE LEARNING TECHNIQUES”**

***An Industrial Oriented mini project report submitted to JNTUH in the partial fulfillment of the requirements for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

**Batch No. - 01**

|  |  |
| --- | --- |
| **SHAIK SUMAYA** | **- 21631A0554** |
| **MOHAMMED FARAAZ** | **- 21631A0520** |
| **TELAPUTTA RAJU** | **- 21631A0539** |
| **VEESALA PAVAN** | **- 21631A0535** |

***Under the Esteemed guidance of***

**Mr. P. Rathaiah, M.Tech, Sr.Assistant Professor in CSE Dept.**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# SRI VENKATESWARA ENGINEERING COLLEGE

(SPONSORED BY THE EXHIBITION SOCIETY, HYDERABAD) (Affiliated to Jawaharlal Nehru Technological University, Hyderabad) (Approved by AICTE, NEW DELHI) SURYAPET – 508213,December-2023

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# SRI VENKATESWARA ENGINEERING COLLEGE SURYAPET-508213



# CERTIFICATE

This is to certify that the project report entitled **“FRAUD DETECTION IN BANKING DATA BY MACHINE LEARNING TECHNIQUES”** is the bonafide work done by **SK.SUMAYA(21631A0554)**,**MD.FARAAZ(21631A0520)**,**T.RAJU(21631A0539)**,**V.PAVAN (21631A0535)** in partial fulfilment of the requirements for the award of **BACHELOR OF TECHNOLOGY** in **Computer Science and Engineering** by JNTUH during the academic year 2025.

**Guide Head of the Department**

**Mr. P. Rathaiah**M.Tech (CSE) **Mr. P. RATHAIAH**M.Tech (CSE)

Sr. Assistant Professor Sr. Assistant Professor

Sri Venkateshwara Engineering College, Sri Venkateshwara Engineering College,

Suryapet Suryapet

**Principal External Examiner**

**Dr. Raju Muthyala**, M Tech, Ph. D,

Sri Venkateshwara Engineering College, Suryapet



## DECLARATION

To the best of our knowledge and belief, we hereby declare that this project bears no resemblance to any other project submitted at Sri Venkateswara Engineering College, Surya pet or any other college We, SHAIK SUMAYA, MOHAMMED FARAAZ, TELAPUTTA RAJU,VEESALA PAVAN, students of B. Tech with CSE of Sri Venkateswara Engineering College, Suryapet, being H.T.No21631A0554,21631A0520,21631A0539,21631A0535 respectively hereby declare that the project with title “FRAUD DETECTION IN BANKING DATA BY MACHINE LEARNING TECHNIQUES” is the original work done by us. Affiliated to Jawaharlal Nehru Technological University, Hyderabad for the award of the degree.

Place: Suryapet

Date:

**Signature of the candidate**

1.SHAIK SUMAYA,

2.MOHAMMED FARAAZ,

3.TELAPUTTA RAJU,

4.VEESALA PAVAN

## ACKNOWLEDGEMENT

We thank the almighty for giving us the courage and perseverance in complete the project. This project itself is an acknowledgment for all those people who have given this project a success.

We take this opportunity to express our deep and sincere gratitude to the project guide **Mr. P.RATHAIAH SIR**, Assistant Professor for his valuable advice at every stage of this work. Without his supervision and valuable guidance, I never have come out in this form.

We are also thankful to **Mr. P. RATHAIAH SIR** Head of the department of computer science and engineering, and, **Dr. M. RAJU** Principal, and **Dr. D. KIRAN KUMAR** Director S.V.E.S for providing excellent facilities, motivation, and encouragement to complete the project work on time.

Last but not least we would like to express our deep sense of gratitude and earnest thanksgiving to our parents for their moral support and heartfelt cooperation in doing the project. We would also like to thank all the teaching and non-teaching staff and my friends whose direct or indirect help has enabled us to complete this work successfully.

**Project Associate**

**SK.SUMAYA -21631A0554**

**MD.FARAAZ -21631A0520**

**T.RAJU -21631A0539**

**V.PAVAN -21631A0535**

**ABSTRACT**

As technology developed and e-commerce services expanded, credit cards became one of the most popular payment methods, resulting in a rise in the number of banking transactions. In addition, the significant rise in fraud requires high banking transaction costs. As a result detecting fraudulent activities has become a fascinating topic. In this study, we examine the use of class weight-tuning hyper parameters to control the weight of legitimate and fraudulent transactions. Specifically, we use Bayesian optimization to optimize the hyper parameters while preserving practical issues such as unbalanced data. We propose weight-tuning as a pre-process for unbalanced data, as well as Cat Boost and XG Boost to enhance the efficiency of the LightGBM method by taking into account for the voting mechanism. To enhance performance even further, we apply deep learning to fine-tune the hyper parameters, particularly our proposed weight-tuning technique. We conduct experiments using real-world data to test the proposed methods. In addition to the standard ROC-AUC, we utilize recall-precision metrics to better cover unbalanced datasets. Cat Boost, LightGBM, and XGBoost , logistic regression are evaluated individually using a 5-fold cross-validation method. In addition, the majority voting ensemble learning technique is used to evaluate the performance of the combined algorithms. The results show that the proposed methods outperform the cutting-edge methods and achieve a significant improvement in performance.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study  2.1 Economic feasibility  2.2 Technical feasibility  2.3 Social feasibility |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

**1. INTRODUCTION**

In recent years, there has been a significant increase in the volume of financial transactions due to the expansion of financial institutions and the popularity of web-based e-commerce. Fraudulent transactions have become a growing problem in online banking, and fraud detection has always been challenging. Along with credit card development, the pattern of credit card fraud has always been updated.An ideal fraud detection system should detect more fraudulent cases, and the precision of detecting fraudulent cases should be high, i.e., all results should be correctly detected, which will lead to the trust of customers in the bank, and on the other hand, the bank will not suffer losses due to incorrect detection. So, this paper discusses the problem of fraudulent transactions in online banking. There is lot of challenges for detecting fraudulent activities and the need for effective fraud detection methods. The paper proposes the use of machine learning techniques, such as Cat Boost, LightGBM, and XGBoost, to improve the performance of fraud detection, as well as deep learning and hyper parameter settings. We employ Bayesian optimization for fraud detection and suggest using the weight-tuning hyperparameter as a pre-process step to address the issue of unbalanced data. In order to enhance performance, we also advise using CatBoost and XGBoost in addition to LightGBM. Using the XGBoost algorithm because of the quick training both in large data and the word "regularization," which overfitting is avoided by evaluating the complexity of it takes little time to set the tree up, and hyper parameters. Cat boost is another algorithm we employ. since it's unnecessary to change the hyper parameters for over fitting management, and it also produces effective outcomes. not modifying the hyper parameters in comparison to other algorithmic learning processes. We suggest an ensemble learning system with majority vote. Combination of Cat Boost, XG Boost, and LightGBM methodology and examine the impact of the combined techniques on the fraud detection performance on actual, unbalanced data. Additionally, we suggest utilizing deep learning for altering adjusting the hyper parameters, etc. • To assess the effectiveness of the suggested techniques, We conduct in-depth tests using data from the real world. In addition to the often used ROC-AUC, we also employ recall precision to better cover the unbalanced datasets. Additionally, we assess performance utilizing F1\_score and MCC measures. The findings indicate that the suggested strategies perform better than the tried-and-true techniques. For we make use of publically accessible data sets and also public access to the source codes should be published by more scholars.

* 1. **OBJECTIVE**

The main objective of this project is to predict credit card fraud detection by using different types of Machine learning and Deep learning Models like LightGBM, XG Boost, Cat Boost, Neural Network and Hybrid model like LG + XG+ CAT, LG + XG, LG + CAT, XG + CAT.

* 1. **PROBLEM STATEMENT**

The proposed Project for credit card fraud detection including the dataset, pre-processing, feature extraction and feature selection, algorithms, framework, and evaluation metrics, is presented and discusses the evaluation results of the experiments performed, and finally concludes the project with framework predict of credit card fraud.

* 1. **SOFTWARE REQUIREMENTS**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer’s architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

1. **Anaconda**

**1.2 HARDWARE REQUIREMENTS**

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

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1) Operating System : Windows**

**2) Processor : i5 and above**

**3) Ram : 8gb and above**

**4) Hard Disk : 25 GB in local drive**

**2. FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth 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

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**2.1 ECONOMICAL 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.

### **2.2 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 system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**2.3 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 of the system.

**3. LITERATURE SURVEY**

**3.1 Ensemble Learning in Credit Card Fraud Detection Using Boosting Methods:**

[Ensemble Learning in Credit Card Fraud Detection Using Boosting Methods | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9463278)

**ABSTRACT:** With the ceaseless thriving of the monetary market, MasterCard volume has forever been blasting these years. The extortion organizations are likewise rising quickly. Under this situation, extortion discovery has turned into an increasingly more significant issue. However, the extent of the misrepresentation is totally much lower than the virtuoso exchange, so the unevenness dataset makes this issue significantly more testing. In this paper we principally advise how to adapt to the Visa misrepresentation identification issue by utilizing supporting strategies and furthermore gave a commitment of the concise examination between these helping techniques.

**3.2 Ecommerce Fraud Detection through Fraud Islands and Multi-layer Machine Learning Model:**

[Ecommerce Fraud Detection Through Fraud Islands and Multi-layer Machine Learning Model | Semantic Scholar](https://www.semanticscholar.org/paper/Ecommerce-Fraud-Detection-Through-Fraud-Islands-and-Nanduri-Liu/263e1b23669c14c4d4dd034eee6c755cb930b4dd)

**ABSTRACT:** Principal challenge for web-based business exchange extortion counteraction is that misrepresentation designs are fairly unique and various. This paper presents two inventive techniques, extortion islands (interface investigation) and multi-facet AI model, which can actually handle the test of distinguishing assorted misrepresentation designs. Extortion Islands are framed utilizing join examination to explore the connections between various fake elements and to reveal the secret complex misrepresentation designs through the shaped organization. Multi-facet model is utilized to manage the generally assorted nature of misrepresentation designs. As of now, the extortion not entirely set in stone through various channels which are banks' declination choice, manual survey specialists' dismissal choices, banks' misrepresentation alarm and clients' chargeback demands. It tends to be sensibly accepted that different misrepresentation examples could be gotten however unique extortion risk anticipation powers (for example bank, manual audit group and misrepresentation AI model). The analyses showed that by incorporating not many different AI models which were prepared utilizing various sorts of misrepresentation marks, the exactness of extortion choices can be fundamentally moved along.

**3.3 Detecting Credit Card Fraud Using Selected Machine Learning Algorithms:**

[Detecting Credit Card Fraud Using Selected Machine Learning Algorithms | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/8757212)

**ABSTRACT:** Because of the gigantic development of internet business and expanded web based installment prospects, Visa misrepresentation has become profoundly significant worldwide issue. As of late, there has been significant interest for applying AI calculations as information digging method for charge card extortion location. Be that as it may, number of difficulties show up, for example, absence of freely accessible informational collections, exceptionally imbalanced class sizes, variation deceitful way of behaving and so on. In this paper we analyze execution of three AI calculations: Arbitrary Woodland, Backing Vector Machine and Strategic Relapse in recognizing extortion on genuine information containing Visa exchanges. To alleviate imbalanced class sizes, we use Destroyed examining technique. The issue of consistently changing misrepresentation designs is considered with utilizing gradual learning of chosen ML calculations in tests. The presentation of the methods is assessed in view of normally acknowledged measurement: accuracy and review.

**3.4 Credit card fraud detection using Ada Boost and majority voting:**

[Credit card fraud detection using AdaBoost and majority voting — Monash University](https://research.monash.edu/en/publications/credit-card-fraud-detection-using-adaboost-and-majority-voting)

**ABSTRACT:** Visa extortion is a difficult issue in monetary administrations. Billions of dollars are lost because of charge card misrepresentation consistently. There is an absence of exploration concentrates on dissecting genuine Visa information inferable from secrecy issues. In this paper, AI calculations are utilized to identify charge card misrepresentation. Standard models are first utilized. Then, at that point, cross breed strategies which use Ada Boost and greater part casting a ballot techniques are applied. To assess the model viability, an openly accessible charge card informational index is utilized. Then, at that point, a true charge card informational index from a monetary establishment is broke down. Also, clamor is added to the information tests to additionally survey the power of the calculations. The trial results decidedly demonstrate that the greater part casting a ballot strategy accomplishes great precision rates in recognizing misrepresentation cases in MasterCard.

**3.5 Feature engineering strategies for credit card fraud detection:**

[Feature engineering strategies for credit card fraud detection - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0957417415008386)

**ABSTRACT:** Consistently billions of Euros are lost overall because of Visa misrepresentation. Subsequently, constraining monetary foundations to further develop their extortion recognition frameworks consistently. As of late, a few investigations have proposed the utilization of AI and information mining methods to resolve this issue. Be that as it may, most examinations utilized some kind of misclassification measure to assess the various arrangements, and don't consider the genuine monetary expenses related with the extortion recognition process. Besides, while developing a charge card extortion identification model, it is vital how to separate the right elements from the conditional information. This is typically finished by conglomerating the exchanges to notice the spending personal conduct standards of the clients. In this paper we grow the exchange conglomeration procedure, and propose to make another arrangement of elements in light of examining the occasional way of behaving of the hour of an exchange utilizing the von Mises conveyance. Then, utilizing a genuine Visa extortion dataset given by a huge European card handling organization, we look at cutting edge MasterCard misrepresentation location models, and assess what the various arrangements of elements have a mean for on the outcomes. By including the proposed occasional elements into the strategies, the outcomes show a typical expansion in reserve funds of 13%.

**Tabular Format of Literature Survey**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Title & Authors** | **Methodology** | **Proposed system** | **Cons of Proposed System** | **Conclusion** |
|  | Title: A Sequence Mining-Based Novel Architecture for Detecting Fraudulent Transactions in Healthcare Systems, [PDF] A Sequence Mining-Based Novel Architecture for Detecting Fraudulent Transactions in Healthcare Systems | Semantic Scholar, Authors: Irum Matloob, S. A. Khan, Rukaiya Rukaiya, M. K. Khattak, Arslan Munir, 2022 | Develop a healthcare fraud detection model using sequence mining. Analyze service sequences in specialty areas, calculate confidence values, and compare them with actual patient data to identify anomalies using a rule engine. Validate on five years of hospital transactional data. | Propose a healthcare fraud detection system utilizing sequence mining. It identifies insurance claim-related frauds by analyzing service sequences in specialties, calculating confidence values, and comparing them with actual patient data. Validated on five years of local hospital transactional data. | Implementing a sequence mining-based healthcare fraud detection system may face challenges such as false positives, privacy concerns, and limited adaptability to evolving fraud schemes. | In conclusion, the presented process-based fraud detection methodology, utilizing sequence mining, offers a promising approach to identify insurance claim-related healthcare frauds. It enhances transparency, cost optimization, and accuracy in fraud detection, validated on real-world hospital data. |
|  | Title: Healthcare Fraud Data Mining Methods: A Look Back and Look Ahead, [Healthcare Fraud Data Mining Methods: A Look Back and Look Ahead - PubMed (nih.gov)](https://pubmed.ncbi.nlm.nih.gov/35440932/), Authors: Nishamathi Kumaraswamy, Mia K Markey, Tahir Ekin, Jamie C Barner, Karen Rascati, 2022 | The methodology involves reviewing literature on healthcare fraud detection, summarizing objectives, conclusions, and data characteristics. Identifying gaps in real-world implementation and proposing future research topics to address these issues. | The proposed system entails a comprehensive review of healthcare fraud detection methods and systems from existing literature. It includes data analysis, summarization of research findings, identification of implementation gaps, and suggesting future research directions to enhance real-world healthcare fraud detection. | This proposed system may be time-consuming, relying on potentially outdated literature, and might not provide immediate solutions to current healthcare fraud challenges. | In conclusion, healthcare fraud is a costly societal issue in the US, impacting both individuals and the healthcare system. This article highlights the importance of evolving digital fraud detection systems, emphasizing the complexity of healthcare data. The review of existing literature offers valuable insights into current methods and identifies gaps in their real-world application. The proposed research topics aim to guide future efforts in enhancing healthcare fraud detection, ultimately safeguarding the system and its beneficiaries. |
|  | Title: Machine Learning based Credit Card Fraud Detection - A Review, [Machine Learning based Credit Card Fraud Detection - A Review | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9792653), Authors: Kavya Gupta; Kirtivardhan Singh; Gaurav Vikram Singh; Mohd. Hassan; Himani, 2022 | The methodology involves analyzing and addressing evolving digital transaction frauds through a sequence of machine learning models, selecting optimal methods via evaluation, and implementing real-time fraud detection with predictive analysis using an API module. Efficient data handling strategies are employed. | The proposed system is an advanced real-time fraud detection solution for digital transactions, employing a sequence of machine learning models and API modules to identify and prevent credit card frauds effectively. | Costly development, false positives, privacy concerns, evolving tactics challenges, and imperfect machine learning pose cons for real-time fraud detection systems. | In conclusion, the digital era has brought numerous advantages but also ushered in evolving fraud challenges, particularly in online transactions like credit card frauds. This review emphasizes using machine learning and predictive analysis through API modules for real-time fraud detection and highlights the need for efficient data handling strategies. |
|  | Title: Analyzing Credit Card Fraud Detection based on Machine Learning Models, [Analyzing Credit Card Fraud Detection based on Machine Learning Models | Semantic Scholar](https://www.semanticscholar.org/paper/Analyzing-Credit-Card-Fraud-Detection-based-on-Almutairi-Godavarthi/1c84442f623513569c90951bfd15daf71eb51245), Authors: Raghad Almutairi, Abhishek Godavarthi, Arthi Reddy Kotha, Ebrima N. Ceesay, 2022 | Utilize machine learning and data science to detect credit card fraud by analyzing historical transaction data, employing various modeling techniques, and implementing real-time fraud detection algorithms. | Proposed system: Implement a comprehensive fraud detection system utilizing machine learning and data science to analyze past transaction data and enhance real-time monitoring, mitigating credit card fraud risks across various payment scenarios. | Drawbacks of implementing a machine learning-based fraud detection system include cost, false positives, slow adaptation, privacy concerns, limited effectiveness against evolving fraud, and potential disruption to payment systems. | In conclusion, the prevalence of credit card usage, especially in online transactions, poses a significant risk of fraud. Leveraging machine learning and data science is essential for effective fraud detection and prevention. |
|  | Title: Evaluation of Naïve Bayes and Voting Classifier Algorithm for Credit Card Fraud Detection, [Evaluation of Naïve Bayes and Voting Classifier Algorithm for Credit Card Fraud Detection | Semantic Scholar](https://www.semanticscholar.org/paper/Evaluation-of-Na%C3%AFve-Bayes-and-Voting-Classifier-for-Vairam-Sarathambekai/be5a7ed4faed34e5bcf7321b6bae9cf8ed468425), Authors: T. Vairam, S. Sarathambekai, S. Bhavadharani, A. Kavi Dharshini, N. Nithya Sri, Tarika Sen, 2022 | The methodology involves leveraging machine learning techniques for Credit Card Fraud Detection. Data is analyzed by a Data Science team to build a model that effectively identifies and prevents fraudulent transactions in the growing online payment landscape. | Proposed System: Implementing a Machine Learning-based Credit Card Fraud Detection solution to combat the rising fraudulent transactions in the digital era, enhancing security for online payments and safeguarding financial transactions effectively. | Implementing a Machine Learning-based Credit Card Fraud Detection solution may face challenges such as false positives, data privacy concerns, and evolving fraud tactics, potentially impacting user experience. | In conclusion, the increasing reliance on online payments and the surge in fraudulent transactions have driven the demand for machine learning-based Credit Card Fraud Detection systems, effectively enhancing security in the digital payment landscape. |
|  | Title: Analysis of Supervised Machine Learning Algorithms in the Context of Fraud Detection, [Analysis of Supervised Machine Learning Algorithms in the Context of Fraud Detection - IOPscience](https://iopscience.iop.org/article/10.1149/10701.7189ecst), Authors: Pradeep Verma and Poornima Tyagi, 2022 | The research employs a supervised machine learning approach to combat credit card fraud. It utilizes an imbalanced dataset and evaluates five classification techniques, with Supervised Vector Classifier and Logistic Regression Classifier demonstrating superior fraud detection performance. | Develop a credit card fraud detection solution using Supervised Machine Learning, focusing on an imbalanced dataset, and compare five classification techniques, with Supervised Vector Classifier and Logistic Regression Classifier as top performers. | Using Supervised Machine Learning for credit card fraud detection with imbalanced data may lead to overfitting and biased results. Comparing only five techniques may overlook better alternatives. | The study addresses the critical issue of credit card fraud in the digital age, highlighting the effectiveness of Supervised Vector Classifier and Logistic Regression Classifier in detecting fraud on imbalanced datasets. |
|  | Title: An Ensemble-Based Credit Card Fraud Detection Algorithm Using an Efficient Voting Strategy, [An Ensemble-Based Credit Card Fraud Detection Algorithm Using an Efficient Voting Strategy | Semantic Scholar](https://www.semanticscholar.org/paper/An-Ensemble-Based-Credit-Card-Fraud-Detection-Using-Rakhshaninejad-Fathian/93d85d7692d6a6d7ff29b6ae8ac893bf66ccfc4c), Authors: Morteza Rakhshaninejad, M. Fathian, Babak Amiri, Navid Yazdanjue, 2022 | The study employed an ensemble-based approach, addressing credit card fraud detection challenges. It balanced imbalanced data, ranked features, utilized seven base classifiers, and introduced a weighted voting strategy for optimal fraud detection. | The proposed system is an ensemble-based credit card fraud detection method with a weighted voting strategy, addressing imbalanced data, feature ranking, and using seven base classifiers for improved fraud detection performance. | The need for feature ranking may require manual intervention and domain expertise, making the approach less automated and potentially less adaptable to changing fraud patterns. | In conclusion, the proposed ensemble-based method with weighted voting effectively addresses credit card fraud detection challenges, achieving superior performance compared to existing methods, ensuring financial security and customer trust. |
|  | Title: Ensemble Learning in Credit Card Fraud Detection Using Boosting Methods, [Ensemble Learning in Credit Card Fraud Detection Using Boosting Methods | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9463278), Authors: Haonan Feng, 2021 | The methodology involves addressing credit card fraud detection amid a booming financial market. It focuses on combating the challenge of imbalanced datasets using boosting methods and provides a concise comparison of these techniques. | The proposed system addresses the growing challenge of credit card fraud in a booming financial market, leveraging boosting methods to tackle imbalanced datasets, with a brief comparative analysis of these techniques. | The proposed system may have limitations, including potential false positives, increased computational requirements, and limited adaptability to emerging fraud tactics. | In conclusion, this study highlights the increasing importance of credit card fraud detection due to a thriving financial market. It emphasizes the challenge of imbalanced datasets and discusses the efficacy of boosting methods, offering a concise comparative analysis. |
|  | Title: Elucidation of big data analytics in banking: a four-stage Delphi study, [[PDF] Elucidation of big data analytics in banking: a four-stage Delphi study | Semantic Scholar](https://www.semanticscholar.org/paper/Elucidation-of-big-data-analytics-in-banking%3A-a-Delgosha-Hajiheydari/f7466e5c94dc22e281f421ac9c2557c34dfd8d19), Authors: Mohammad Soltani Delgosha, Nastaran Hajiheydari, S. Fahimi, 2022 | This study employed a four-round Delphi study, engaging 25 experts, to investigate strategic applications, drivers, and challenges of big data analytics in banking, providing valuable insights for academia and industry. | The proposed system entails conducting a four-round Delphi study involving 25 experts to investigate and prioritize strategic applications, key drivers, and challenges of implementing big data analytics in the banking industry. | Implementing a four-round Delphi study with 25 experts might lead to prolonged decision-making, increasing costs, and potential participant fatigue | In conclusion, this study underscores the significance of big data analytics in banking, emphasizing its critical role in fraud detection and credit risk analysis. Decision-making enhancement and innovation drive its adoption, yet information silos pose significant challenges, calling for strategic solutions. These findings contribute to enhancing the understanding of big data's managerial implications and offer actionable insights for scholars and decision-makers in navigating the dynamic business landscape. |
|  | Title: Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection, [Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection | SpringerLink](https://link.springer.com/article/10.1007/s41870-020-00430-y), Fayaz Itoo, Meenakshi & Satwinder Singh, 2022 | This study focuses on addressing credit card fraud in the context of imbalanced data. It involves data preprocessing using random under-sampling, employs machine learning algorithms (logistic regression, Naïve Bayes, and K-nearest neighbor), and assesses model performance using accuracy, sensitivity, specificity, precision, F-measure, and AUC metrics. Logistic regression outperforms other models, and under-sampling enhances predictions. The implementation is in Python. | The proposed system addresses rising financial fraud in credit card transactions by utilizing machine learning (logistic regression, Naïve Bayes, K-nearest neighbor) on imbalanced data, enhanced through random under-sampling, implemented in Python. | The utilization of logistic regression, Naïve Bayes, and K-nearest neighbor on imbalanced data with random under-sampling in Python to combat credit card fraud may suffer from limitations such as potential oversimplification, reduced adaptability to evolving fraud tactics, and a risk of overfitting. | In conclusion, financial fraud poses a growing threat due to technological advancements. Addressing skewed credit card data through under-sampling and employing logistic regression enhances fraudulent transaction prediction, outperforming Naïve Bayes and K-nearest neighbor models. |

**4. SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

In literature they developed a transaction aggregation strategy and created a new set of features based on the periodic behaviour analysis of the transaction time by using the von Mises distribution. In addition, they propose a new cost-based criterion for evaluating credit card fraud detection’s models and then, using a real credit card dataset, examine how different feature sets affect results. More precisely, they extend the transaction aggregation strategy to create new offers based on an analysis of the periodic behaviour of transactions. In another study the application of machine learning algorithms to detect fraud in credit cards. They first use Naive Bayes, stochastic forest and decision trees, neural networks, linear regression (LR), and logistic regression, as well as support vector machine standard models, to evaluate the available datasets. Further, they propose a hybrid method by applying AdaBoost and majority voting. In addition, they add noise to the data samples for robustness evaluation. They perform experiments on publicly available datasets and show that majority voting is effective in detecting credit card fraud cases.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

1. Other hyper parameter tuning methods like gridsearchcv and RandomizedSearchCV that spend more time to reach the highest accuracy model.
2. They used over sampling techniques to address the data imbalance.
3. They considered accuracy rather than precision-recall metrics for evaluating the model performance when the data is imbalance.
4. New set of features are created based on the periodic behavior analysis of the transaction time by using the von mises distribution.
5. Does not use any deep learning to fine-tune the hyperparameters, particularly the proposed weight-tuning one.

# 4.2 PROPOSED SYSTEM:

The proposed work in the paper is a fraud detection approach using machine learning techniques. We consider the use of class weight-tuning hyperparameters to control the weight of fraudulent and legitimate transactions. We use Bayesian optimization to optimize the hyperparameters while preserving practical issues such as unbalanced data. We propose weight-tuning as a pre-process for unbalanced data, as well as CatBoost and XGBoost to improve the performance of the LightGBM method by accounting for the voting mechanism. Finally, in order to improve performance even further, they use deep learning to fine-tune the hyperparameters, particularly the proposed weight-tuning one.Then examine the algorithms using different evaluation metrics, including accuracy, precision, recall, the Matthews correlation coefficient (MCC), the F1- score, and AUC diagrams

# 4.2.1ADVANTAGES OF PROPOSED SYSTEM:

1. The proposed approach uses class weight-tuning hyper parameters to control the weight of fraudulent and legitimate transactions, which helps to address the problem of unbalanced data.
2. We use Bayesian optimization to optimize the hyper parameters, which helps to improve the performance of the model while preserving practical issues such as unbalanced data.
3. The proposed approach combines multiple machine learning techniques, including Cat Boost, LightGBM, XGBoost, and deep learning, which helps to improve the performance of the model compared to cutting-edge methods.

### **4.3 FUNCTIONAL REQUIREMENTS**

1. Data Collection

2. Data Pre-processing

3. Training and Testing

4. Modiling

5. Predicting

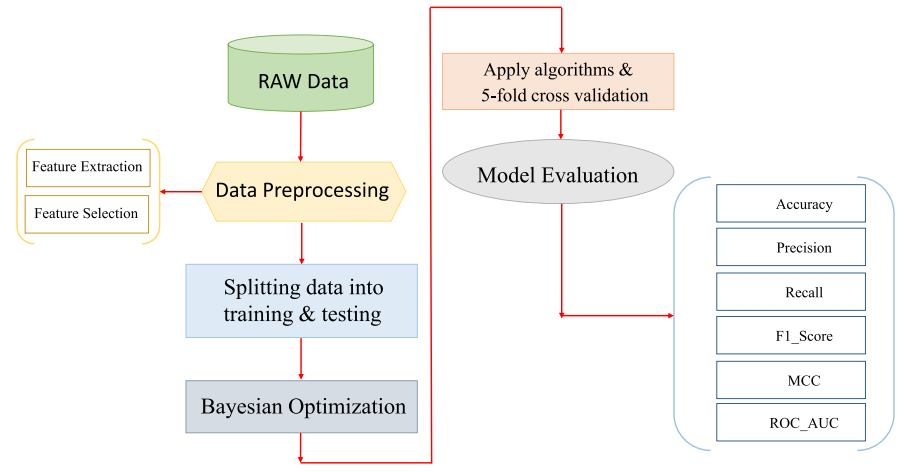
### **4.4 NON FUNCTIONAL REQUIREMENTS**

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

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**5. SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**

****

**Fig.5.1.1 System architecture**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**Import libraries**

**VERIFY**

**NO PROCESS**

**YES NO**

**Importing the dataset**

**Data processing**

**Feature Selection**

**Splitting the data into train & test**

**Building the model – LightGBM, XG Boost, Cat Boost, Neural Network – 1st experiment**

**Hybrid model like LG + XG+ CAT, LG + XG, LG + CAT, XG + CAT – 2nd experiment**

**Training the model**

**Signup & sign in**

**User input**

**End process**

**Final Outcome**

**5.2 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

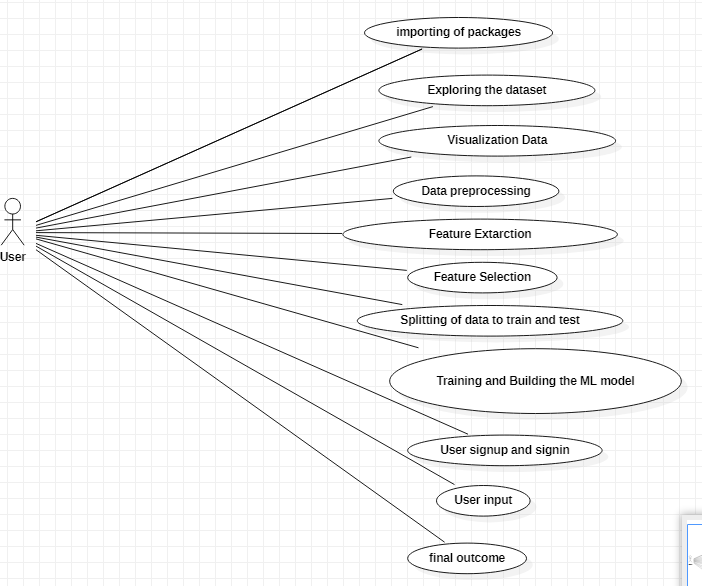
**GOALS:**

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

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

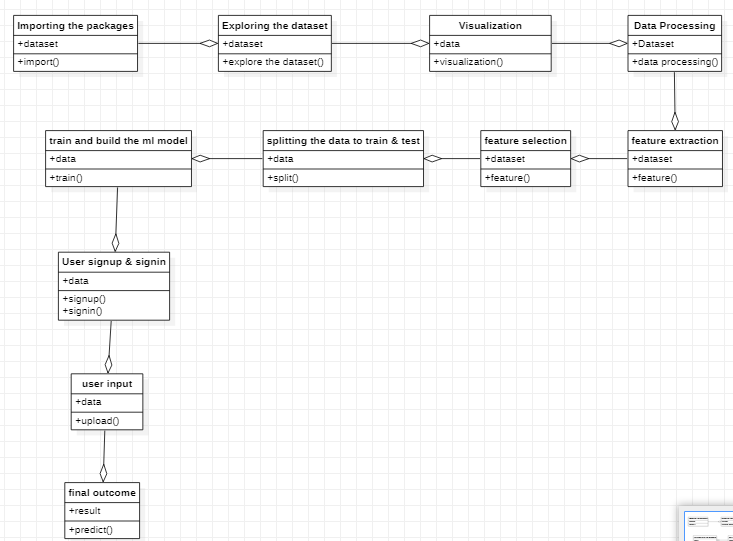
**Use case diagram:**

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



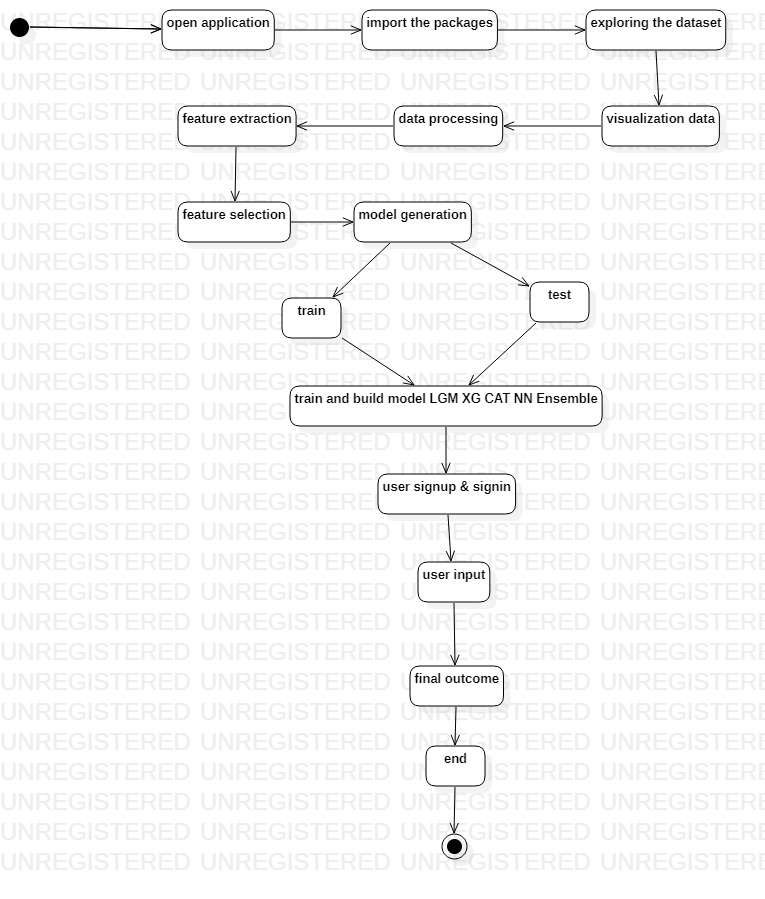
**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

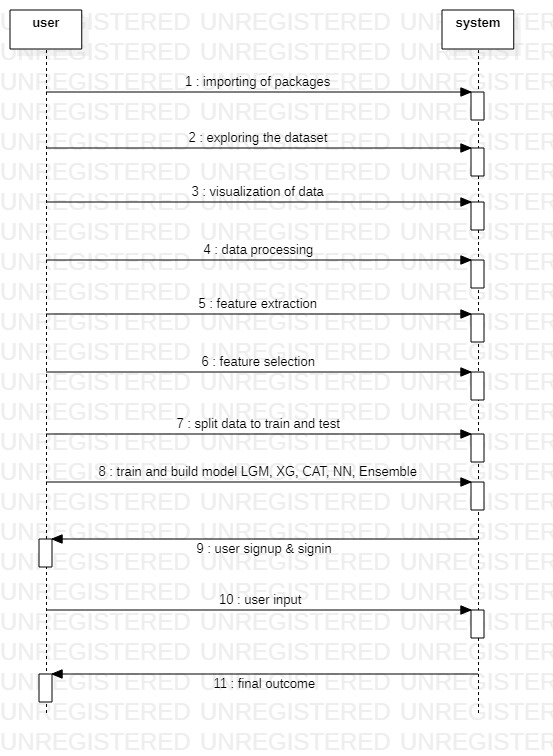


**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

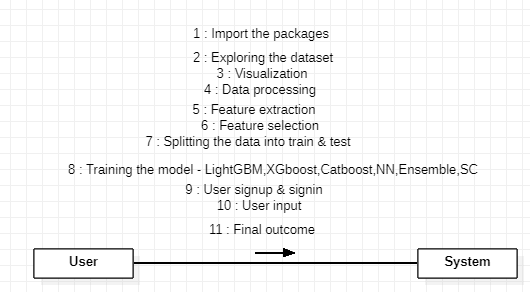
**Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



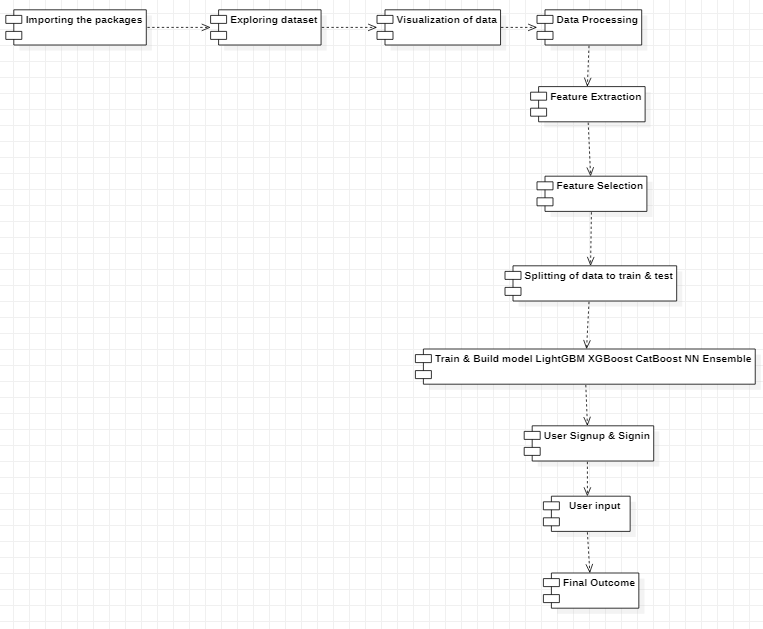
**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

****

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

****

1. **IMPLEMENTATION**

**MODULES:**

* + Data exploration: using this module we will load data into system
  + Processing: Using the module we will read data for processing
  + Splitting data into train & test: using this module data will be divided into train & test
  + Model generation: Model building
* Bayesian Optimization : LightGBM – XGBoost – CatBoost - Neural Network
* CV Stratified Kfold : LightGBM – XGBoost – CatBoost - Neural Network
* Smote Sampling (over and under sampling): LightGBM – XGBoost – CatBoost - Neural Network
* Hyper parameter Tuning : LightGBM – XGBoost – CatBoost - Ensemble of LG + XG + CAT - Ensemble of LG + XG - Ensemble of XG + CAT - Ensemble of LG + CAT - Neural Network - Stacking Classifier (Gradient Boosting with RF + LightGBM).
* User signup & login: Using this module will get registration and login
* User input: Using this module will give input for prediction
* Prediction: final predicted displayed

**Note:** As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction.

However, we can further enhance the performance by exploring other ensemble techniques such as Stacking Classifier with RF + LightGBM With Gradient Boosting which got 100% accuracy.

**Algorithms:**

Bayesian Optimization: Bayesian Optimization provides a principled technique based on Bayes Theorem to direct a search of a global optimization problem that is efficient and effective. It works by building a probabilistic model of the objective function, called the surrogate function that is then searched efficiently with an acquisition function before candidate samples are chosen for evaluation on the real objective function.

CV StratifiedKfold: Stratified k-fold cross-validation is the same as just k-fold cross-validation, But Stratified k-fold cross-validation, it does stratified sampling instead of random sampling.

Smote Sampling (over and under sampling**):** [SMOTE is a technique that**oversamples the minority class by synthetically generating data points**](https://www.bing.com/ck/a?!&&p=de88a440a1d29c4cJmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTgxNQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&psq=-+Smote+Sampling+(over+and+under+sampling)&u=a1aHR0cHM6Ly93d3cuaXJqZXQubmV0L2FyY2hpdmVzL1Y0L2k4L0lSSkVULVY0STg1Ny5wZGY&ntb=1)**.** [It uses k nearest neighbours to create new examples that are similar to the existing ones](https://www.bing.com/ck/a?!&&p=2be3908cb0172d80JmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTgxNw&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&psq=-+Smote+Sampling+(over+and+under+sampling)&u=a1aHR0cHM6Ly93d3cuaXJqZXQubmV0L2FyY2hpdmVzL1Y0L2k4L0lSSkVULVY0STg1Ny5wZGY&ntb=1). [SMOTE can be combined with random under sampling of the majority class to balance the class distribution and improve the performance of the classifier](https://www.bing.com/ck/a?!&&p=defc27cddb10c995JmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTgxOQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&psq=-+Smote+Sampling+(over+and+under+sampling)&u=a1aHR0cHM6Ly9tYWNoaW5lbGVhcm5pbmdtYXN0ZXJ5LmNvbS9zbW90ZS1vdmVyc2FtcGxpbmctZm9yLWltYmFsYW5jZWQtY2xhc3NpZmljYXRpb24v&ntb=1)

**Hyper parameters: Hyper parameters** that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

Light GBM: LightGBM, short for light gradient-boosting machine, is a [free and open-source](https://www.bing.com/ck/a?!&&p=21f508325248f603JmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc1NQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPUZyZWUlMjBhbmQlMjBvcGVuJTIwc291cmNlJTIwd2lraXBlZGlhJmZvcm09V0lLSVJF&ntb=1) distributed [gradient-boosting](https://www.bing.com/ck/a?!&&p=03d66b7bb5f3fa49JmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc1Ng&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPUdyYWRpZW50JTIwYm9vc3RpbmclMjB3aWtpcGVkaWEmZm9ybT1XSUtJUkU&ntb=1) framework for [machine learning](https://www.bing.com/ck/a?!&&p=3c2967e98a9f5822JmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc1Nw&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPU1hY2hpbmUlMjBsZWFybmluZyUyMHdpa2lwZWRpYSZmb3JtPVdJS0lSRQ&ntb=1), originally developed by [Microsoft](https://www.bing.com/ck/a?!&&p=277e31d4b1260e4aJmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc1OA&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPU1pY3Jvc29mdCUyMHdpa2lwZWRpYSZmb3JtPVdJS0lSRQ&ntb=1). It is based on [decision tree](https://www.bing.com/ck/a?!&&p=5e8c8057e9b505c9JmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc1OQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPURlY2lzaW9uJTIwdHJlZSUyMHdpa2lwZWRpYSZmb3JtPVdJS0lSRQ&ntb=1) algorithms and used for [ranking](https://www.bing.com/ck/a?!&&p=a4462fe2c12871deJmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc2MA&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPUxlYXJuaW5nJTIwdG8lMjByYW5rJTIwd2lraXBlZGlhJmZvcm09V0lLSVJF&ntb=1), [classification](https://www.bing.com/ck/a?!&&p=b84f7114449e03afJmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTc2MQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&u=a1L3NlYXJjaD9xPVN0YXRpc3RpY2FsJTIwY2xhc3NpZmljYXRpb24lMjB3aWtpcGVkaWEmZm9ybT1XSUtJUkU&ntb=1) and other machine learning tasks. The development focus is on performance and scalability.

XG Boost: [XGBoost](https://xgboost.ai/), which stands for Extreme Gradient Boosting, is a scalable, distributed [gradient-boosted](https://en.wikipedia.org/wiki/Gradient_boosting) decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

Cat Boost: [CatBoost is an**open-source boosting library developed by Yandex**](https://www.bing.com/ck/a?!&&p=a91abb957cd0d69dJmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTg2OQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&psq=Cat+Boost+model+definition&u=a1aHR0cHM6Ly93d3cuZ2Vla3Nmb3JnZWVrcy5vcmcvY2F0Ym9vc3QtbWwv&ntb=1)**.** [It is designed for use on problems like regression and classification having a very large number of independent features](https://www.bing.com/ck/a?!&&p=cb8e4b94bf24971fJmltdHM9MTY5MjgzNTIwMCZpZ3VpZD0wYmVjNTU2Ni05ZGVlLTY5ZTEtMjE3OS00NjM4OWM0MzY4ODcmaW5zaWQ9NTg3MQ&ptn=3&hsh=3&fclid=0bec5566-9dee-69e1-2179-46389c436887&psq=Cat+Boost+model+definition&u=a1aHR0cHM6Ly93d3cuZ2Vla3Nmb3JnZWVrcy5vcmcvY2F0Ym9vc3QtbWwv&ntb=1).

Neural Network: A neural network (NN), in the case of artificial neurons called artificial neural network (ANN) or simulated neural network (SNN), is an interconnected group of natural or artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation.

Ensemble Methods: The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions. These methods follow the same principle as the example of buying an air-conditioner cited above. In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms.

Stacking Classifier (Gradient Boosting with RF + LightGBM): A stacking classifier is an ensemble learning method that combines multiple classification models to create one “super” model. This can often lead to improved performance, since the combined model can learn from the strengths of each individual model.

**6.2 SAMPLE CODE:**

# Necessary imports

## Data loading, processing and for more

import pandas as pd

import numpy as np

from imblearn.over\_sampling import SMOTE

## Visualization

import seaborn as sns

import matplotlib.pyplot as plt

# set seaborn style because it prettier

sns.set()

## Metrics

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, classification\_report

from sklearn.metrics import roc\_curve, auc

## Models

import xgboost as xgb

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import VotingClassifier

# read the data and show first 5 rows

data = pd.read\_csv("../input/bs140513\_032310.csv")

data.head(5)

data.info()

# Create two dataframes with fraud and non-fraud data

df\_fraud = data.loc[data.fraud == 1]

df\_non\_fraud = data.loc[data.fraud == 0]

sns.countplot(x="fraud",data=data)

plt.title("Count of Fraudulent Payments")

plt.show()

print("Number of normal examples: ",df\_non\_fraud.fraud.count())

print("Number of fradulent examples: ",df\_fraud.fraud.count())

#print(data.fraud.value\_counts()) # does the same thing above

print("Mean feature values per category",data.groupby('category')['amount','fraud'].mean())

# Create two dataframes with fraud and non-fraud data

pd.concat([df\_fraud.groupby('category')['amount'].mean(),df\_non\_fraud.groupby('category')['amount'].mean(),\

data.groupby('category')['fraud'].mean()\*100],keys=["Fraudulent","Non-Fraudulent","Percent(%)"],axis=1,\

sort=False).sort\_values(by=['Non-Fraudulent'])

# Plot histograms of the amounts in fraud and non-fraud data

plt.figure(figsize=(30,10))

sns.boxplot(x=data.category,y=data.amount)

plt.title("Boxplot for the Amount spend in category")

plt.ylim(0,4000)

plt.legend()

plt.show()

# Plot histograms of the amounts in fraud and non-fraud data

plt.hist(df\_fraud.amount, alpha=0.5, label='fraud',bins=100)

plt.hist(df\_non\_fraud.amount, alpha=0.5, label='nonfraud',bins=100)

plt.title("Histogram for fraudulent and nonfraudulent payments")

plt.ylim(0,10000)

plt.xlim(0,1000)

plt.legend()

plt.show()

print((data.groupby('age')['fraud'].mean()\*100).reset\_index().rename(columns={'age':'Age','fraud' : 'Fraud Percent'}).sort\_values(by='Fraud Percent'))

print("Unique zipCodeOri values: ",data.zipcodeOri.nunique())

print("Unique zipMerchant values: ",data.zipMerchant.nunique())

# dropping zipcodeori and zipMerchant since they have only one unique value

data\_reduced = data.drop(['zipcodeOri','zipMerchant'],axis=1)

data\_reduced.columns

# turning object columns type to categorical for easing the transformation process

col\_categorical = data\_reduced.select\_dtypes(include= ['object']).columns

for col in col\_categorical:

data\_reduced[col] = data\_reduced[col].astype('category')

# categorical values ==> numeric values

data\_reduced[col\_categorical] = data\_reduced[col\_categorical].apply(lambda x: x.cat.codes)

data\_reduced.head(5)

X = data\_reduced.drop(['fraud'],axis=1)

y = data['fraud']

print(X.head(),"\n")

print(y.head())

y[y==1].count()

sm = SMOTE(random\_state=42)

X\_res, y\_res = sm.fit\_resample(X, y)

y\_res = pd.DataFrame(y\_res)

print(y\_res[0].value\_counts())

# I won't do cross validation since we have a lot of instances

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_res,y\_res,test\_size=0.3,random\_state=42,shuffle=True,stratify=y\_res)

# %% Function for plotting ROC\_AUC curve

def plot\_roc\_auc(y\_test, preds):

'''

Takes actual and predicted(probabilities) as input and plots the Receiver

Operating Characteristic (ROC) curve

'''

fpr, tpr, threshold = roc\_curve(y\_test, preds)

roc\_auc = auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc\_auc)

plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0, 1])

plt.ylim([0, 1])

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

# The base score should be better than predicting always non-fraduelent

print("Base accuracy score we must beat is: ",

df\_non\_fraud.fraud.count()/ np.add(df\_non\_fraud.fraud.count(),df\_fraud.fraud.count()) \* 100)

# %% K-ello Neigbors

knn = KNeighborsClassifier(n\_neighbors=5,p=1)

knn.fit(X\_train,y\_train)

y\_pred = knn.predict(X\_test)

print("Classification Report for K-Nearest Neighbours: \n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix of K-Nearest Neigbours: \n", confusion\_matrix(y\_test,y\_pred))

plot\_roc\_auc(y\_test, knn.predict\_proba(X\_test)[:,1])

# %% Random Forest Classifier

rf\_clf = RandomForestClassifier(n\_estimators=100,max\_depth=8,random\_state=42,

verbose=1,class\_weight="balanced")

rf\_clf.fit(X\_train,y\_train)

y\_pred = rf\_clf.predict(X\_test)

print("Classification Report for Random Forest Classifier: \n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix of Random Forest Classifier: \n", confusion\_matrix(y\_test,y\_pred))

plot\_roc\_auc(y\_test, rf\_clf.predict\_proba(X\_test)[:,1])

XGBoost\_CLF = xgb.XGBClassifier(max\_depth=6, learning\_rate=0.05, n\_estimators=400,

objective="binary:hinge", booster='gbtree',

n\_jobs=-1, nthread=None, gamma=0, min\_child\_weight=1, max\_delta\_step=0,

subsample=1, colsample\_bytree=1, colsample\_bylevel=1, reg\_alpha=0, reg\_lambda=1,

scale\_pos\_weight=1, base\_score=0.5, random\_state=42, verbosity=True)

XGBoost\_CLF.fit(X\_train,y\_train)

y\_pred = XGBoost\_CLF.predict(X\_test)

print("Classification Report for XGBoost: \n", classification\_report(y\_test, y\_pred))

print("Confusion Matrix of XGBoost: \n", confusion\_matrix(y\_test,y\_pred))

plot\_roc\_auc(y\_test, XGBoost\_CLF.predict\_proba(X\_test)[:,1])

**7. SOFTWARE ENVIRONMENT**

**MACHINE LEARNING:**

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

## Challenges in Machines Learning:-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

**Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of over fitting & under fitting** − If the model is over fitting or under fitting, it cannot be represented well for the problem.

**Curse of dimensionality** − another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

**DEEP LEARNING**

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain’s biological neurons, and they are designed to learn from large amounts of data.

## What is Anaconda for Python?

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over [thirty](https://www.anaconda.com/blog/10-years-of-data-science-innovation-anacondas-commitment-to-the-open-source-python-community)[million people](https://www.anaconda.com/blog/10-years-of-data-science-innovation-anacondas-commitment-to-the-open-source-python-community) worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

**What is Anaconda for Python?**

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

Now that you know what Anaconda Python is, let's look at how to install it.

**How to install Anaconda for Python?**



To install Anaconda, just head to the Anaconda Documentation website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

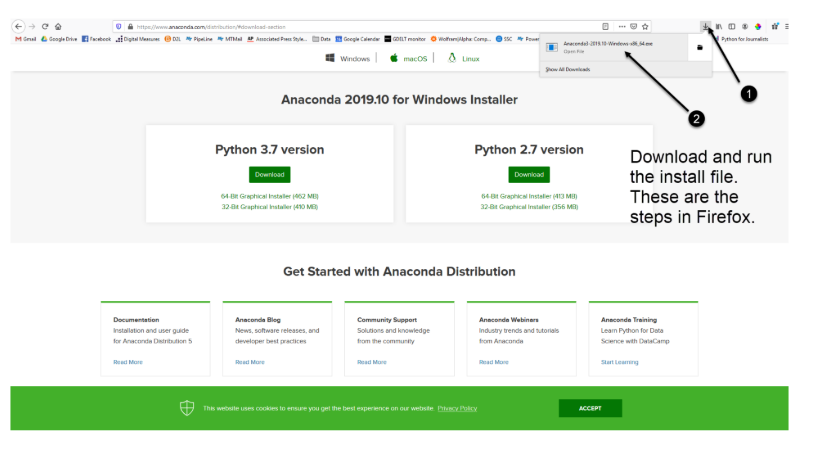
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

### **Python Anaconda Installation**

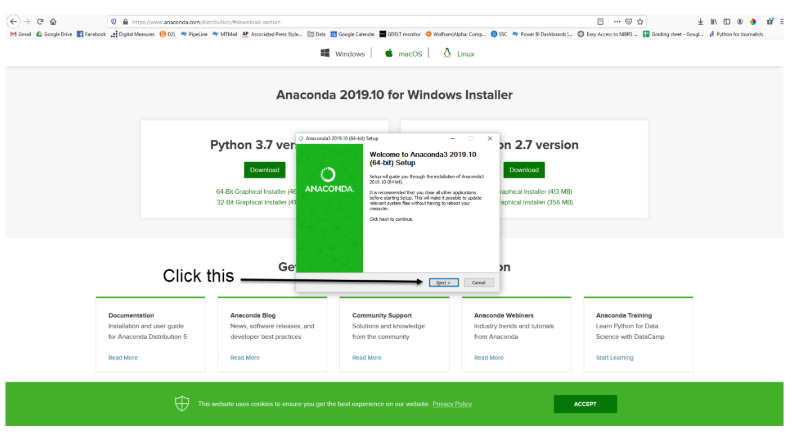
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – [Download anaconda](https://www.anaconda.com/distribution/)

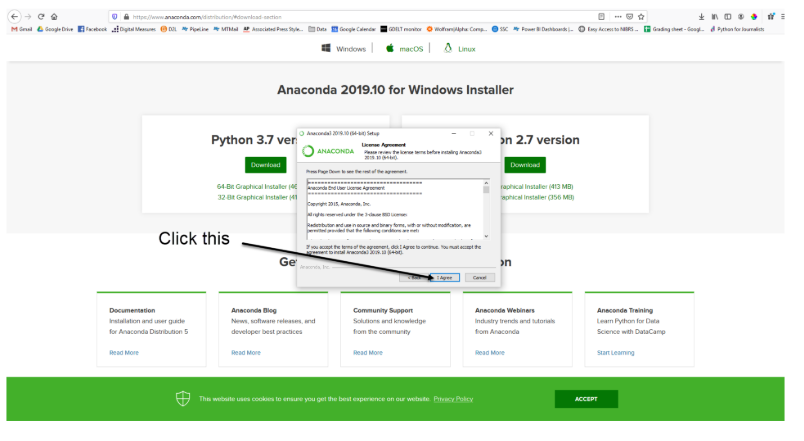


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

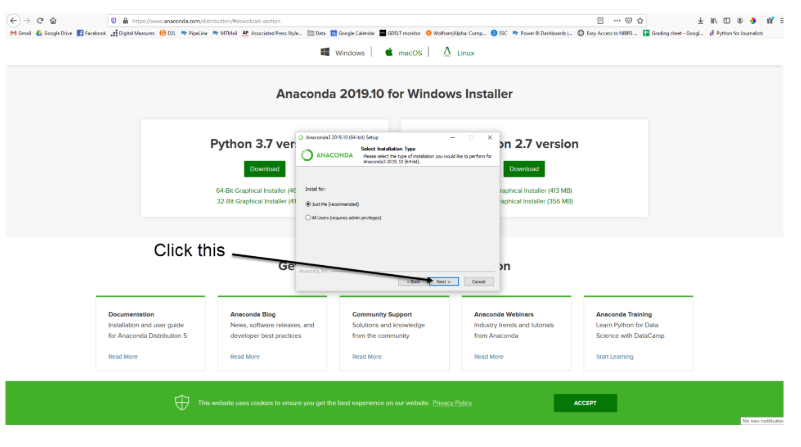
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



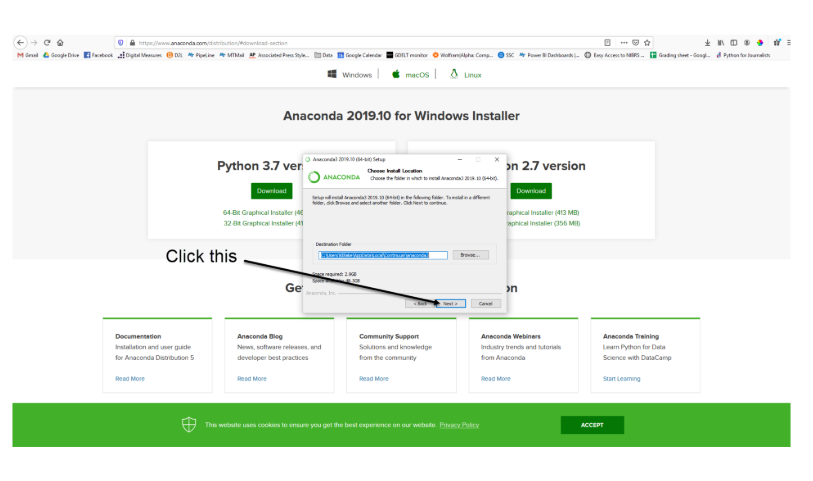
3. Now, you’ll see the license agreement. Click on ‘I Agree’.



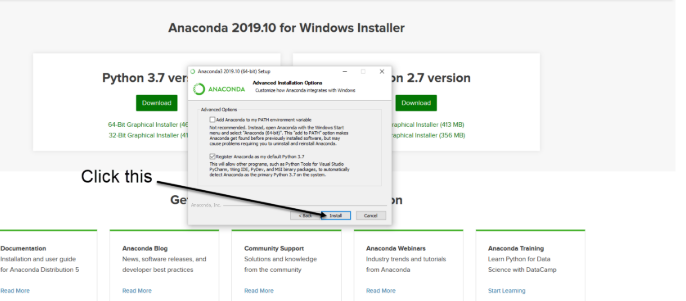
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



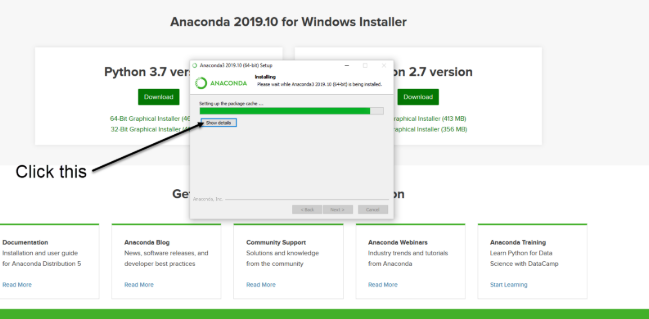
5. Choose where you want to install it. Here, you can see the available space and how much you need.



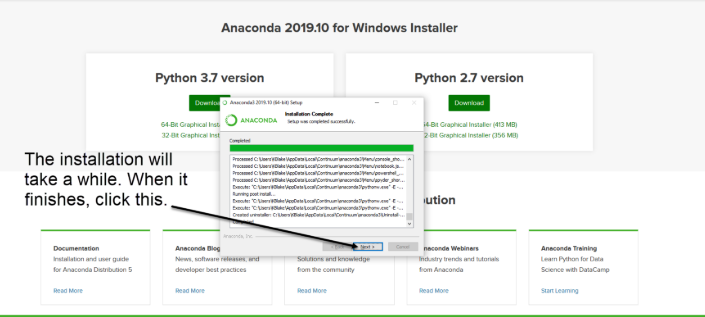
6. Now, you’ll get some advanced options. You can add Anaconda to your system’s PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on ‘Install’.



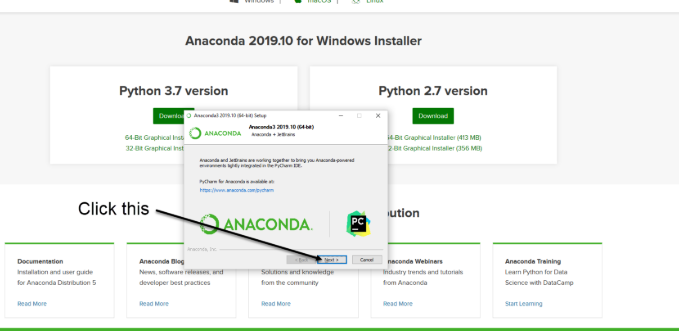
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



8. The installation is complete. Click Next.



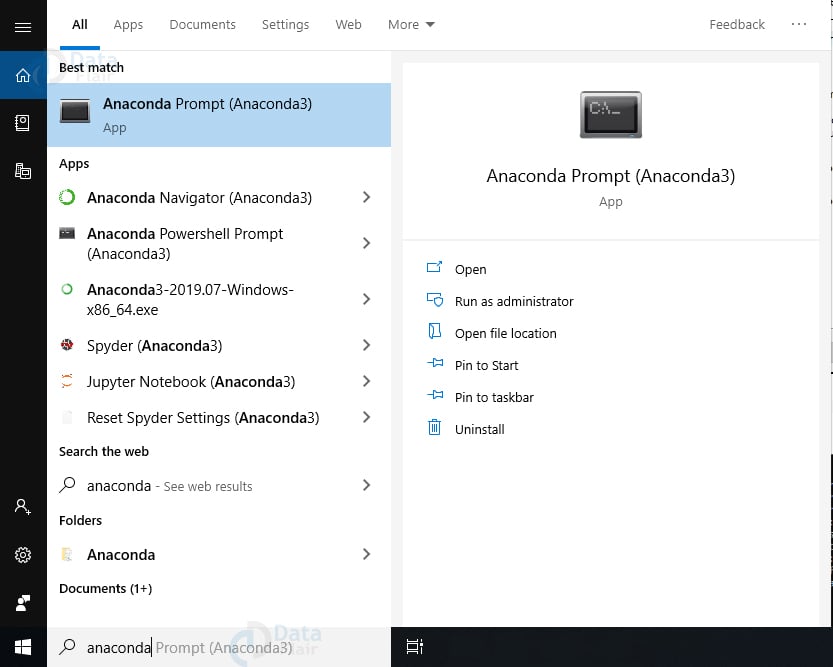
9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.



11. If you search for Anaconda now, you will see the following options:



**PYTHON LANGUAGE:**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

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.

## Features in Python:

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

### **1. Free and Open Source**

[Python](https://www.geeksforgeeks.org/python-programming-language/)language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. [Download Python](https://www.python.org/downloads/) Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### **2. Easy to code**

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of [Python is Object-Oriented programming](https://www.geeksforgeeks.org/python-oops-concepts/). Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Extensible feature**

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an integrated language**

Python is also an integrated language because we can easily integrate Python with other languages like C, [C++](http://www.geeksforgeeks.org/c-plus-plus/), etc.

### **11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, [Java](https://www.geeksforgeeks.org/java/), etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **byte code**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as [regular expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-testing/), web browsers, etc.

### **13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

### **14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### 15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

**LIBRARIES/PACKGES:-**

**Tensor flow**

Tensor Flow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**8. SYSTEM TESTING**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

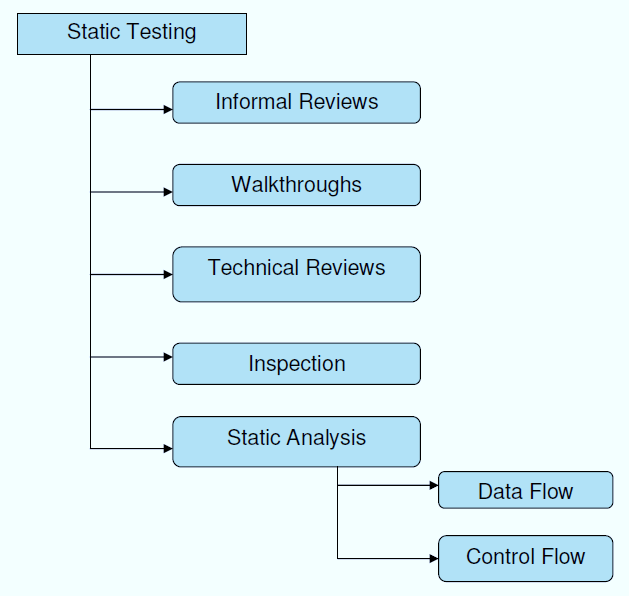
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**8.1Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

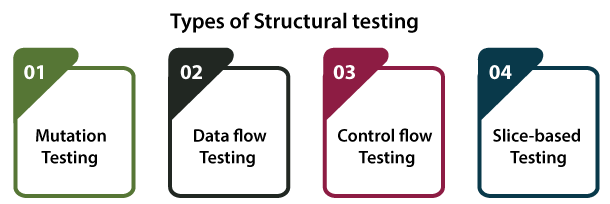
Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



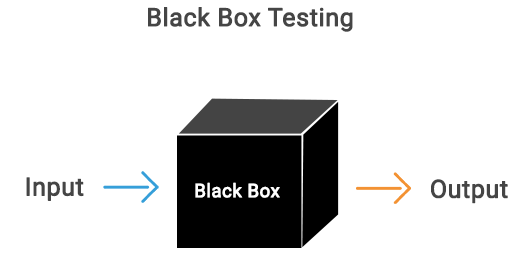
**Structural Testing:**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software’s structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system’s behavior by comparing the tests’ outcomes with the results of previous iterations (control flow testing).



**Behavioral Testing:**

The final stage of testing focuses on the software’s reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user’s point of view. QA engineers usually have some specific information about a business or other purposes of the software (‘the black box’) to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.

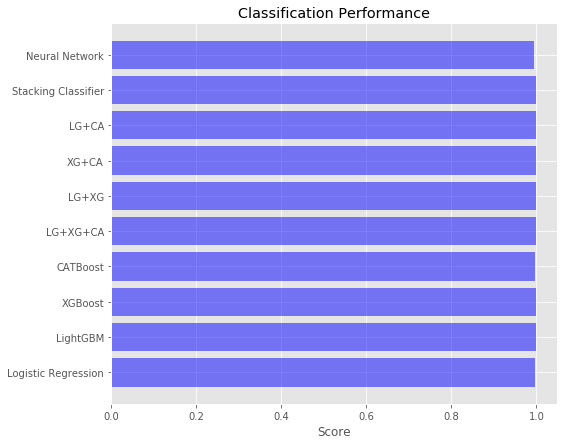


**8.2 TEST CASES:**

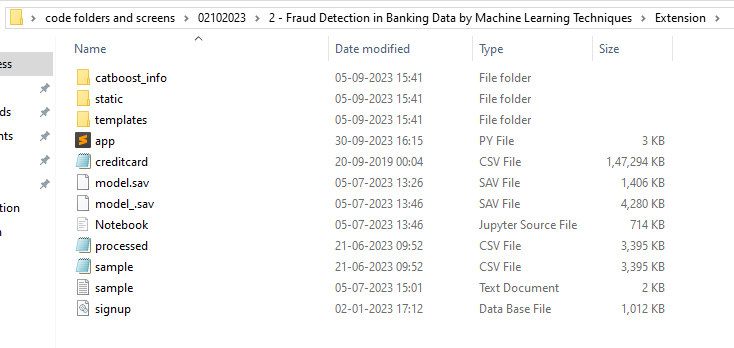
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User sign in | User get login into the application | There is no process |
| 3 | Enter input for prediction | Prediction result displayed | There is no process |

1. **SCREENSHOTS**

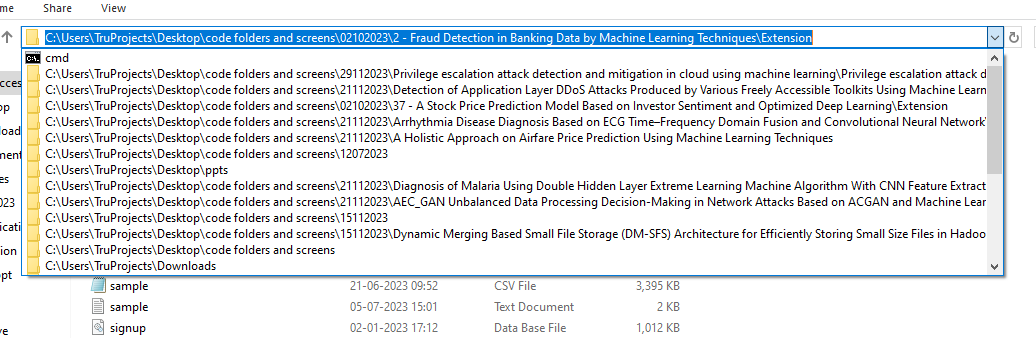
**SCREENS:**



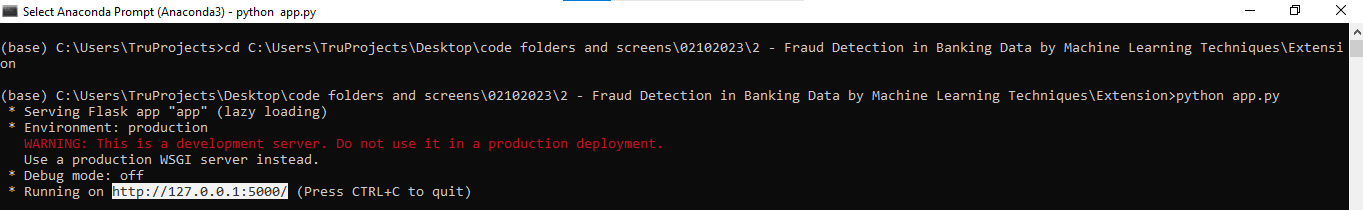
**Comparison Graphs**



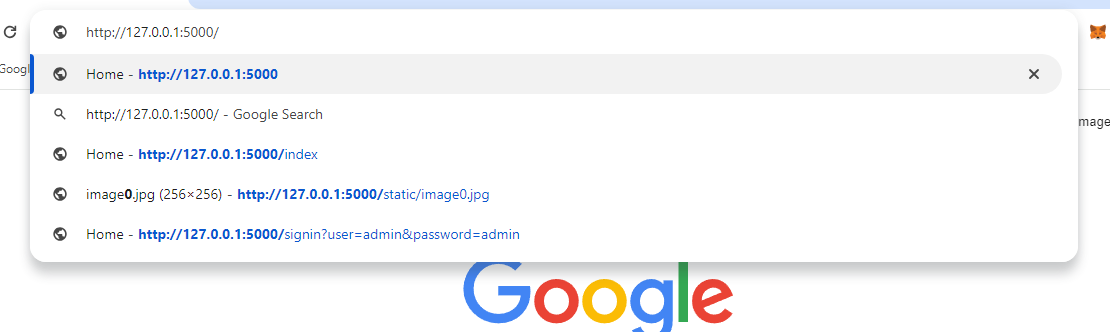
Step 1



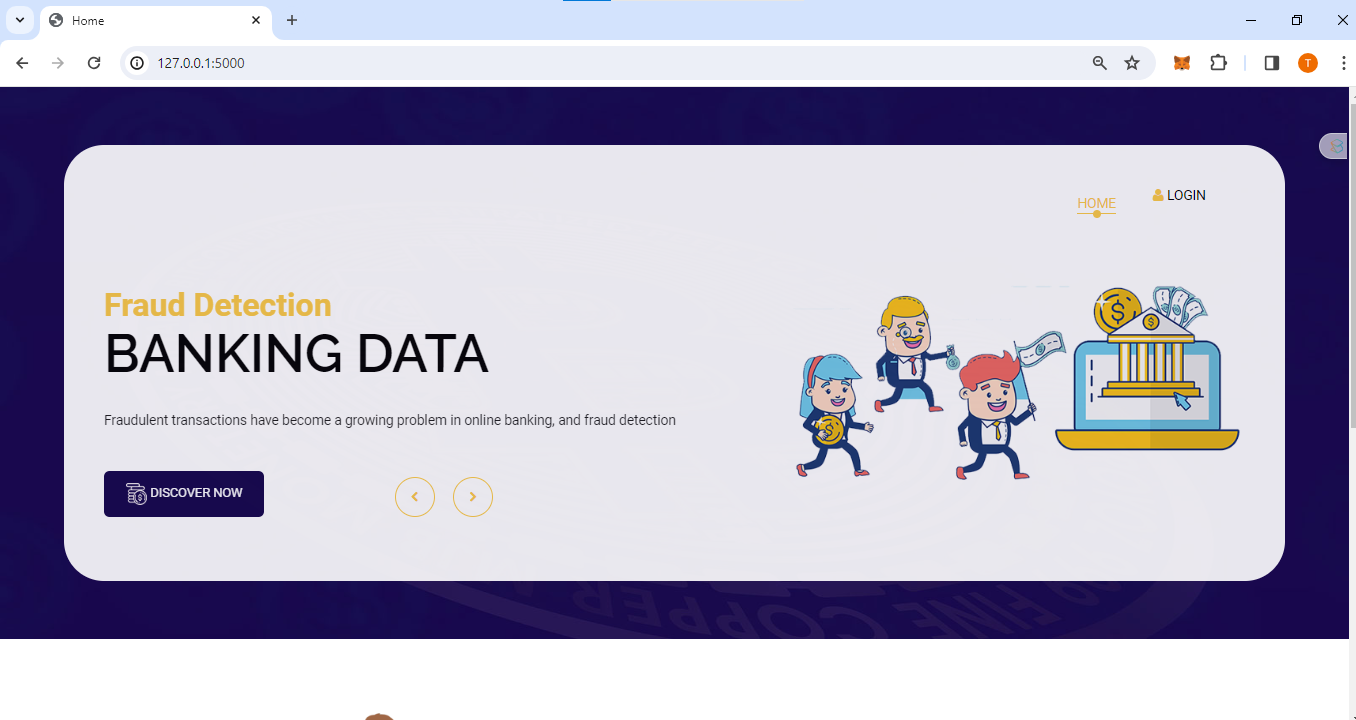
Step 2



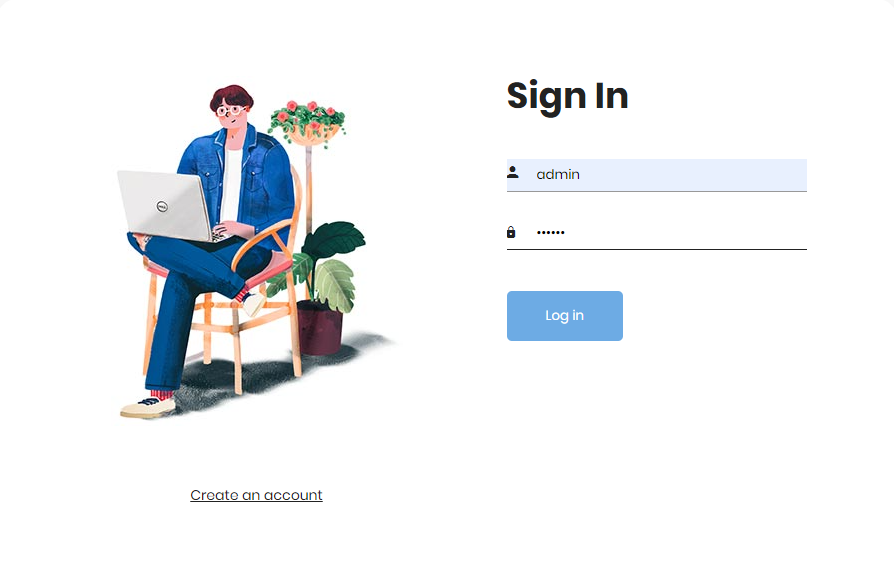
Step 3



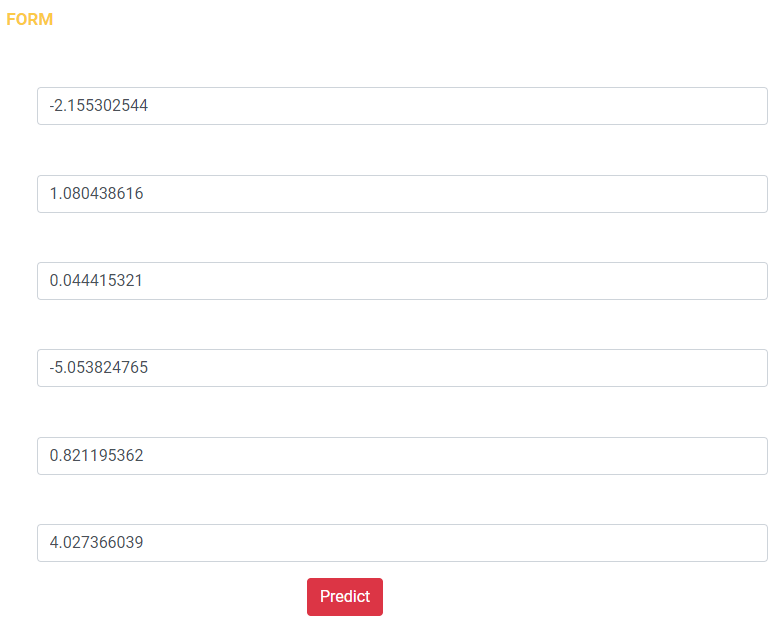
Step 4



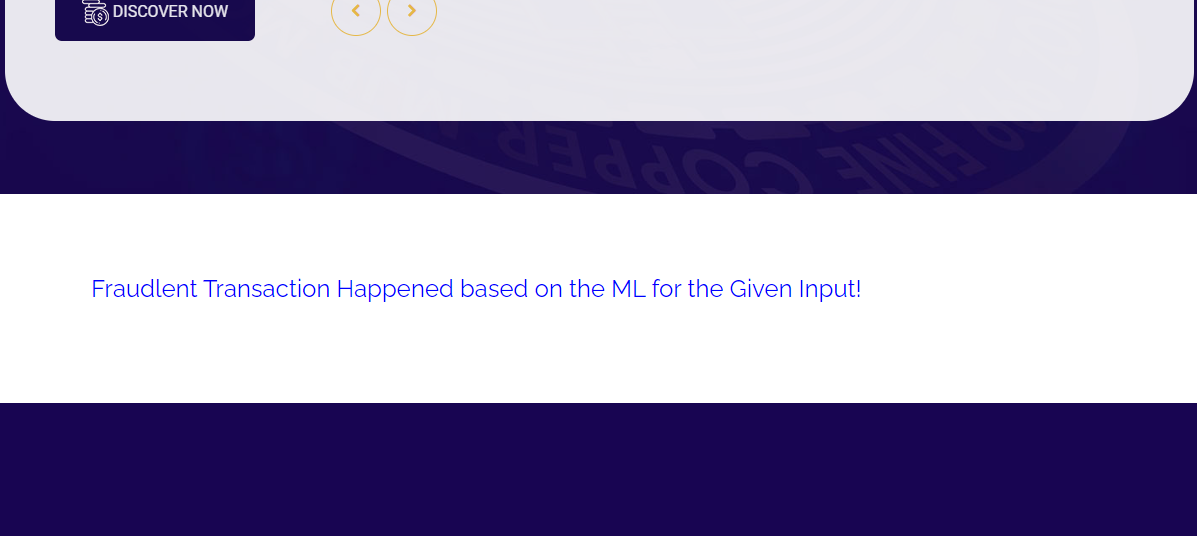
Step 5



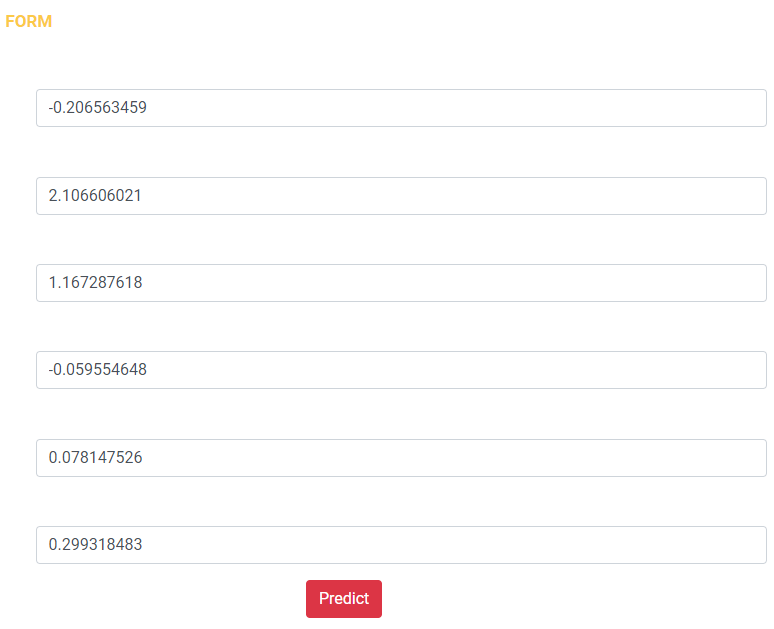
Step 6



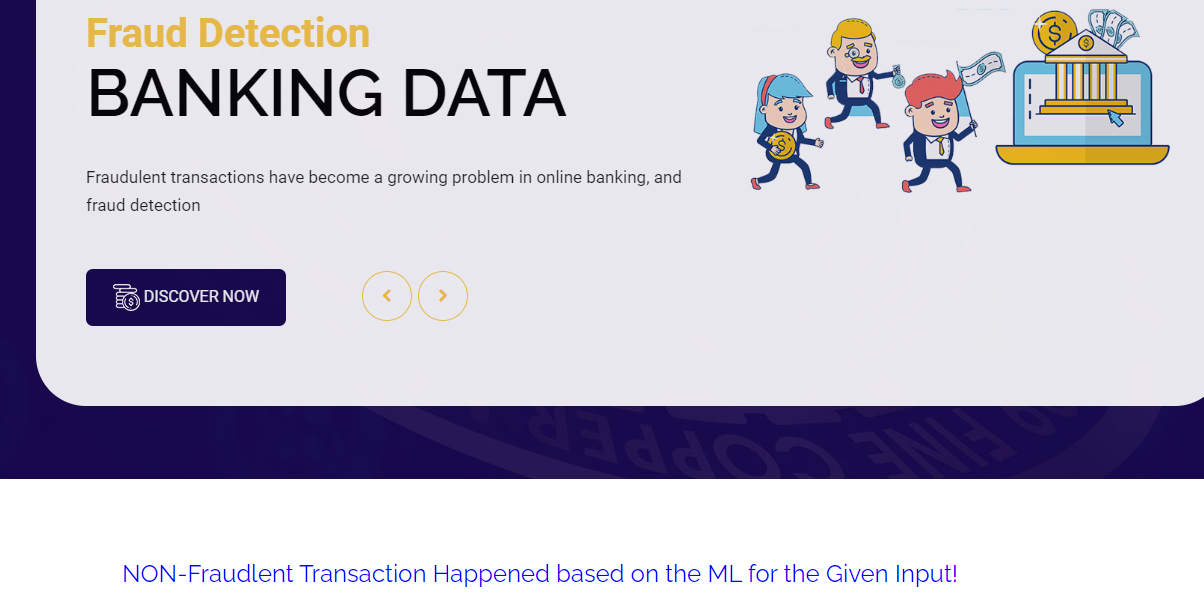
Step 7



Step 8



Step 9



Step 10

**10. CONCLUSION**

The conclusion of the paper is that the proposed machine learning approach using class weight-tuning hyper parameters, Cat Boost, LightGBM, and XG Boost algorithms, and deep learning to fine-tune the hyper parameters significantly improves the performance of fraud detection in real unbalanced datasets. This proposed approach can be used to detect fraudulent activities in banking data and reduce the high banking transaction costs associated with fraud. The result shows that the proposed methods outperform the other cutting-edge methods and achieve a significant improvement in performance. For the Future work, this paper suggests that using other hybrid models and working specifically in the field of Cat Boost by changing more hyper parameters, especially the number of trees having a chance of increase in the performance of this proposed model. The assurance of the results of MCC for unbalanced data proved that, compared to other criteria of evaluation, it’s stronger. In this paper, by combining the LightGBM and XG Boost methods, we obtained 0.79 and 0.81 for the deep learning method. Using hyper parameters to address data unbalance compared. to sampling methods, in addition to reducing memory and time needed to evaluate algorithms, also has better results. For future studies and work, we propose using other hybrid models as well as working specifically in the field of Cat Boost by changing more hyper parameters, especially the hyper parameter number of trees. Also, due to hardware limitations in this study, the use of stronger and better hardware may bring better results that can ultimately be compared with the results of this study.

**11. REFERENCES**

[1] J. Nanduri, Y.-W. Liu, K. Yang, and Y. Jia, ‘‘Ecommerce fraud detection through fraud islands and multi-layer machine learning model,’’ in Proc. Future Inf. Commun. Conf., in Advances in Information and Communication. San Francisco, CA, USA: Springer, 2020, pp. 556–570.

[2] I. Matloob, S. A. Khan, R. Rukaiya, M. A. K. Khattak, and A. Munir, ‘‘A sequence mining-based novel architecture for detecting fraudulent transactions in healthcare systems,’’ IEEE Access, vol. 10, pp. 48447–48463, 2022.

[3] H. Feng, ‘‘Ensemble learning in credit card fraud detection using boosting methods,’’ in Proc. 2nd Int. Conf. Comput. Data Sci. (CDS), Jan. 2021, pp. 7–11.

[4] M. S. Delgosha, N. Hajiheydari, and S. M. Fahimi, ‘‘Elucidation of big data analytics in banking: A four-stage delphi study,’’ J. Enterprise Inf. Manage., vol. 34, no. 6, pp. 1577–1596, Nov. 2021.

[5] M. Puh and L. Brkić, ‘‘Detecting credit card fraud using selected machine learning algorithms,’’ in Proc. 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), May 2019, pp. 1250–1255.

[6] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, ‘‘Credit card fraud detection using AdaBoost and majority voting,’’ IEEE Access, vol. 6, pp. 14277–14284, 2018.

[7] N. Kumaraswamy, M. K. Markey, T. Ekin, J. C. Barner, and K. Rascati, ‘‘Healthcare fraud data mining methods: A look back and look ahead,’’ Perspectives Health Inf. Manag., vol. 19, no. 1, p. 1, 2022.

[8] E. F. Malik, K. W. Khaw, B. Belaton, W. P. Wong, and X. Chew, ‘‘Credit card fraud detection using a new hybrid machine learning architecture,’’ Mathematics, vol. 10, no. 9, p. 1480, Apr. 2022.

[9] K. Gupta, K. Singh, G. V. Singh, M. Hassan, G. Himani, and U. Sharma, ‘‘Machine learning based credit card fraud detection—A review,’’ in Proc. Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC), 2022, pp. 362–368.

10] R. Almutairi, A. Godavarthi, A. R. Kotha, and E. Ceesay, ‘‘Analyzing credit card fraud detection based on machine learning models,’’ in Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS), Jun. 2022, pp. 1–8.

[11] N. S. Halvaiee and M. K. Akbari, ‘‘A novel model for credit card fraud detection using artificial immune systems,’’ Appl. Soft Comput., vol. 24, pp. 40–49, Nov. 2014.

[12] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, ‘‘Feature engineering strategies for credit card fraud detection,’’ Expert Syst. Appl., vol. 51, pp. 134–142, Jun. 2016.

[13] U. Porwal and S. Mukund, ‘‘Credit card fraud detection in e-commerce: An outlier detection approach,’’ 2018, arXiv:1811.02196.

[14] H. Wang, P. Zhu, X. Zou, and S. Qin, ‘‘An ensemble learning framework for credit card fraud detection based on training set partitioning and clustering,’’ in Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Oct. 2018, pp. 94–98.

[15] F. Itoo, M. Meenakshi, and S. Singh, ‘‘Comparison and analysis of logistic regression, Naïve Bayes and knn machine learning algorithms for credit card fraud detection,’’ Int. J. Inf. Technol., vol. 13, no. 4, pp. 1503–1511, 2021.

[16] T. A. Olowookere and O. S. Adewale, ‘‘A framework for detecting credit card fraud with cost-sensitive meta-learning ensemble approach,’’ Sci. Afr., vol. 8, Jul. 2020, Art. no. e00464.

[17] A. A. Taha and S. J. Malebary, ‘‘An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine,’’ IEEE Access, vol. 8, pp. 25579–25587, 2020.

[18] X. Kewei, B. Peng, Y. Jiang, and T. Lu, ‘‘A hybrid deep learning model for online fraud detection,’’ in Proc. IEEE Int. Conf. Consum. Electron. Comput. Eng. (ICCECE), Jan. 2021, pp. 431–434.

[19] T. Vairam, S. Sarathambekai, S. Bhavadharani, A. K. Dharshini, N. N. Sri, and T. Sen, ‘‘Evaluation of Naïve Bayes and voting classifier algorithm for credit card fraud detection,’’ in Proc. 8th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS), Mar. 2022, pp. 602–608.

[20] P. Verma and P. Tyagi, ‘‘Analysis of supervised machine learning algorithms in the context of fraud detection,’’ ECS Trans., vol. 107, no. 1, p. 7189, 2022.