

**A PROJECT REPORT ON**  
**FORECASTING THE SUCCESS OF BANK MARKETING CAMPAIGNS**  
**USING MACHINE LEARNING AND FLASK**

**Submitted in partial fulfilment for the completion of BE- V SEMESTER**

**in**  
**INFORMATION TECHNOLOGY**

**By**  
**M SAI NIGAM (160121737183)**  
**V SAMUEL SACHIN (160121737198)**

**Under the guidance of**  
**Ms. T. MADHURI**  
**Assistant Professor,**  
**Department of Information Technology**



**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**

**(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)**

**GANDIPET, HYDERABAD – 500 075**

**Website: [www.cbit.ac.in](http://www.cbit.ac.in)**

**2022-2023**



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**

Kokapet ( Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**43**  
years



## **CERTIFICATE**

This is to certify that the project work entitled “**Forecasting the Success of Bank Marketing Campaigns**” submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**, in partial fulfilment of the requirements for the award of the completion of V semester of B.E in Information Technology, during the academic year 2023-2024, is a record of original work done by **M Sai Nigam (160121737183), V Samuel Sachin (160121737198)** during the period of study in the Department of IT, CBIT, HYDERABAD, under our supervision and guidance.

**Project Guide**

**Ms. T. MADHURI**

Assistant Professor,  
CBIT, Hyderabad.

**Head of the Department**

**Dr. RAJANIKANTH ALUVALU**

Professor, Dept. of IT,  
CBIT, Hyderabad.

## **DECLARATION**

This is to declare that the work reported in the present report titled "**Forecasting the Success of Bank Marketing Campaigns**" submitted in partial fulfilment for the completion of B.E., the IV semester, in the department of Information Technology, Chaitanya Bharathi Institute of Technology, Hyderabad, is a record of original work. No part of the report is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred. The reported results are based on the project work done entirely by us and not copied from any other sources.

**M Sai Nigam**  
**160121737183**

**V Samuel Sachin**  
**1601217373198**

## **ACKNOWLEDGEMENT**

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success. First of all I would like to express my profound sense of gratitude to the principal Dr. **Prof. C. V. NARASIMHULU** for giving me the permission to carry out this project.

I express my deep gratitude and regards to Dr. Rajanikanth Aluvalu, Head of Department of Information Technology, and project guide Ms. T. Madhuri, Assistant Professor, Department in Information Technology for their encouragement and valuable guidance in bringing shape to this dissertation.

I am thankful to all the Technical Staff and Non-teaching staff in the department for their teachings and academic support.

**M Sai Nigam**  
**160121737183**

**V Samuel Sachin**  
**1601217373198**

## **ABSTRACT:**

In the dynamic world of banking, the ability to connect with customers effectively through marketing campaigns is a strategic imperative. This project focuses on harnessing the potential of data-driven insights to revolutionize bank marketing. Targeted towards the Indian market, the project capitalizes on a comprehensive dataset containing customer demographics, communication channel preferences, historical campaign outcomes, and various other relevant attributes. The central goal is to construct a robust predictive model using Flask, a Python web framework, capable of dissecting customer profiles, scrutinizing campaign timings, evaluating communication strategies, and assessing product offerings. Through this model, banks gain the power to anticipate the success of upcoming marketing initiatives, enabling them to strategically identify and prioritize promising customer segments.

By leveraging the predictive capabilities of our model, banks can make data-informed decisions, optimize resource allocation, and personalize marketing messages for specific customer cohorts. This precision-driven approach is primed to maximize both response and conversion rates, ultimately leading to elevated levels of customer engagement and acquisition. The project's ultimate mission is to furnish banks with a potent tool that not only elevates their marketing campaigns but also empowers them to cultivate enduring customer relationships in the ever-evolving financial landscape. In an era where data reigns supreme, our project stands as a beacon guiding banks towards a more effective and efficient marketing future.

# INDEX

## PRELIMINARY PAGES

TITLE PAGE	01
CERTIFICATE	02
DECLARATION	03
ACKNOWLEDGEMENT	04
ABSTRACT	05

## CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION	09 – 10
1.2 OBJECTIVE	11
1.3 EXISTING SYSTEM	11 – 12
1.4 DISADVANTAGES	12
1.5 PROPOSED SYSTEM	13

## CHAPTER 2: TECHNOLOGIES

1.1 PYTHON	15 – 16
1.2 FLASK	16
1.3 MODULES OF PYTHON USED IN THE APPLICATION	17 – 19
1.4 MACHINE LEARNING CLASSIFIERS	19 - 20

## CHAPTER 3: SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS	22
3.2 SOFTWARE REQUIREMENTS	23 - 24

## **CHAPTER 4: SYSTEM DESIGN**

4.1 SYSTEM DESIGN	26
4.2 UML DIAGRAMS	26

## **CHAPTER 5: IMPLEMENTATION AND TESTING**

5.1 IMPLEMENTATION DESCRIPTION	28
5.2 SYSTEM TESTING	29 - 30

## **CHAPTER 6: SOURCE CODE**

1.1 PYTHON	32 – 36
1.2 HTML	37 – 39
1.3 FLASK	40

## **CHAPTER 7: RESULTS** **42**

## **CONCLUSION** **44**

## **BIBLIOGRAPHY** **45**

# INTRODUCTION



## 1.1 INTRODUCTION

In today's data-centric landscape, the success of marketing campaigns is the linchpin of business growth, especially for banking institutions seeking to engage their customer base effectively. The "Marketing Campaign Success Prediction" project represents a fusion of cutting-edge machine learning and web development, poised to redefine how banks approach marketing strategies. At its core, this project features a sophisticated machine learning model, meticulously trained on historical campaign data encompassing customer demographics, campaign timing, communication channels, and product offerings. Through an intuitive Flask-based web interface, this model empowers banking professionals to peer into the future, predicting campaign outcomes with precision. Beyond mere prediction, our focus on interpretability ensures that users not only receive predictions but also comprehend the underlying rationale. This project is a transformative leap towards data-driven marketing, offering banks the ability to allocate resources judiciously, customize messaging, and elevate customer engagement to unprecedented heights. Here's an introduction to bank campaigns success prediction, encompassing several key parameters:

### **Age:**

Age, a fundamental demographic factor, plays a pivotal role in our campaign success prediction model. It serves as a critical parameter, guiding marketing strategies. Understanding the age distribution of the target audience enables tailored messaging, ensuring campaigns resonate with specific generational preferences, thereby enhancing the likelihood of success.

### **Job:**

This key demographic feature informs marketing strategies profoundly. Different job roles exhibit distinct interests and needs. Incorporating job-related data enables precise campaign customization, ensuring resonance with specific professional demographics, ultimately influencing campaign success.

### **Balance:**

Average yearly balance is a critical parameter within our campaign success prediction model. It provides essential insights into a customer's financial stability and capacity. By factoring in this parameter, our model tailors campaigns to match the financial behaviours of individuals,

optimizing offerings and messaging to maximize success based on their specific financial circumstances.

**Loan:**

Loan status is a pivotal parameter in our campaign success prediction model. It holds significant sway in shaping marketing strategies. Understanding whether a customer has an existing loan informs tailored campaigns – offering relevant financial products or adjusting messaging to address loan-related concerns, thereby influencing campaign outcomes.

**Campaign:**

The number of contacts made during a campaign is a crucial parameter in our campaign success prediction model. It signifies the intensity of engagement with a client. Incorporating this parameter guides campaign strategies, helping to strike a balance between maintaining client interest and avoiding overexposure, thus influencing the campaign's overall effectiveness.

**Campaign:**

The number of contacts made during a campaign is a crucial parameter in our campaign success prediction model. It signifies the intensity of engagement with a client. Incorporating this parameter guides campaign strategies, helping to strike a balance between maintaining client interest and avoiding overexposure, thus influencing the campaign's overall effectiveness.

## 1.2 OBJECTIVE

The primary objective of this project is to empower banking institutions with a sophisticated and data-driven approach to their marketing endeavours. At its core, the project seeks to predict the success of marketing campaigns with a high degree of accuracy. By harnessing historical campaign data and leveraging advanced machine learning algorithms, the project aims to uncover hidden patterns and insights that can significantly enhance campaign outcomes. This predictive capability allows banks to make informed decisions, allocate resources more efficiently, and tailor their marketing efforts precisely to meet the unique preferences and behaviours of their customer base.

Furthermore, the project aspires to bridge the gap between data science and practical marketing applications. It places a strong emphasis on interpretability, ensuring that the predictions generated by the model are not just numerical outputs but actionable insights. By understanding why a particular campaign is likely to succeed or fail, banking professionals can fine-tune their strategies, optimizing factors such as customer profiles, communication channels, and product offerings. Ultimately, the overarching goal is to enable banks to achieve higher customer engagement, improved acquisition rates, and a more significant return on their marketing investments, transforming marketing from a gamble into a strategic advantage.

## 1.3 EXISTING SYSTEM

Pre-existing systems, in the context of our "Marketing Campaign Success Prediction" project, refer to the traditional methods and tools that banks may have relied on for making marketing decisions before the implementation of our data-driven solution. These systems often lack the advanced predictive capabilities and real-time insights offered by our project. Here's a description of pre-existing systems:

**1. Manual Decision-Making :** Before the advent of data-driven systems, banks typically relied on manual decision-making processes. Marketing teams would analyse past campaigns' outcomes and use their experience and intuition to make decisions about the next campaign. This approach can be time-consuming, subjective, and less accurate due to the inability to process large datasets effectively.

**2. Basic Analytics Tools:** Some banks may have used basic analytics tools such as spreadsheets or rudimentary database queries to extract insights from historical campaign data. While these tools can provide basic statistics and trends, they often lack the predictive capabilities and visualizations needed for sophisticated marketing optimization.

**3. Limited Segmentation:** Pre-existing systems might have employed basic customer segmentation techniques based on demographic information or broad categories. However, these segments may not capture the nuanced preferences and behaviours of different customer groups accurately.

**4. Lack of Predictive Modelling:** Traditional systems typically lack predictive modelling capabilities. They might provide historical data but do not use it to forecast future campaign outcomes, limiting the bank's ability to proactively optimize strategies.

**5. Inefficient Resource Allocation:** Without robust data-driven insights, banks may allocate resources inefficiently, spreading their efforts across various channels and customer segments without a clear understanding of which ones are most likely to yield positive results.

**6. Reactive Approach:** Pre-existing systems often lead to a reactive marketing approach. Banks respond to campaign outcomes after they occur, making adjustments for future campaigns based on past failures or successes. This approach can be less agile and may result in missed opportunities.

## **1.4 DISADVANTAGES**

The primary disadvantage of pre-existing marketing decision systems in banking lies in their limited capacity to harness data for predictive and proactive decision-making. These systems often rely on manual analysis and basic tools, making them inefficient in processing vast datasets and identifying intricate patterns. As a result, banks using such systems may struggle with resource allocation, as they lack precise insights into which customer segments or strategies are likely to succeed. Additionally, their reactive nature means that adjustments are made retrospectively, potentially leading to missed opportunities and suboptimal campaign outcomes. In today's data-driven landscape, these limitations hinder banks from fully capitalizing on the potential of tailored marketing strategies and predictive analytics.

## 1.5 PROPOSED SYSTEM

The proposed "Marketing Campaign Success Prediction" system represents a pioneering leap in the realm of data-driven marketing for banking institutions. At its core, this system harnesses the power of advanced machine learning techniques to predict the outcomes of marketing campaigns with an unprecedented degree of accuracy. By analysing historical campaign data encompassing customer demographics, communication channels, product offerings, and more, the system extracts invaluable insights that guide the formulation of future marketing strategies.

Unlike traditional, manual decision-making processes, this system operates in real-time, offering marketers a dynamic and agile platform for optimizing their campaigns. Through a user-friendly Flask-based web interface, banking professionals can input campaign parameters and receive instant predictions, empowering them to make informed decisions on-the-fly. This shift from intuition-based decisions to data-informed strategies represents a monumental advantage, enabling banks to allocate resources judiciously, tailor messaging precisely, and maximize customer engagement and acquisition rates.

Furthermore, the system prioritizes interpretability, ensuring that its predictions are not enigmatic outputs but actionable insights. Marketers gain a profound understanding of why a campaign is likely to succeed or falter, allowing for strategic adjustments to enhance outcomes continually. In summary, the proposed "Marketing Campaign Success Prediction" system is a game-changer, offering banks the tools they need to navigate the complex marketing landscape with precision, transforming marketing from a gamble into a strategic asset.

# TECHNOLOGIES

## 2.1 PYTHON:

Python is a high-level, interpreted programming language that is widely used in a variety of applications. It is known for its simplicity, readability, and versatility, and is a popular choice for beginners and experts alike.

One of the key advantages of Python is its ease of use. The language is designed to be straightforward and intuitive, with a syntax that is easy to read and write. This makes it an excellent choice for learning programming, as it allows users to focus on the core concepts of programming without getting bogged down in syntax and other details.

Another advantage of Python is its versatility. It is used in a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, and automation. Python has a large and active community of developers who have created a vast ecosystem of libraries and tools that extend the functionality of the language.

Python is also known for its strong support for object-oriented programming, functional programming, and other programming paradigms. This makes it a flexible language that can be used in a variety of programming styles.

Overall, Python is a powerful and popular programming language that is well-suited to a wide range of applications. Its ease of use, versatility, and strong community support make it an excellent choice for beginners and experts alike.

Python is widely used for machine learning because it offers several key advantages forth is field:

- 1. Large and Active Community:** Python has a large and active community of developers who have created a vast ecosystem of libraries and tools that are specifically designed for machine learning. Some of the most popular machine learning libraries in Python include TensorFlow, PyTorch, Scikit-learn, Keras, and Pandas.
- 2. Simple and Readable Syntax:** Python is known for its simple and readable syntax, which makes it easy to write and understand machine learning code. This can save time and reduce the risk of errors when working on complex machine learning projects.
- 3. Flexibility:** Python is a highly flexible language that can be used for a wide range of applications, including machine learning. It supports a variety of programming paradigms,

including object-oriented programming and functional programming, and can be used with both structured and unstructured data.

**4. Rapid Prototyping:** Python's ease of use and flexibility make it well-suited for rapid prototyping in machine learning. With Python, developers can quickly experiment with different algorithms and approaches, and iterate on their models until they find the best solution.

Overall, Python's combination of a large and active community, simple and readable syntax, flexibility, and rapid prototyping capabilities make it an ideal language for machine learning.

## **2.2 FLASK:**

Flask is a lightweight and flexible web framework for Python. It is designed to make it easy to build web applications quickly and with minimal code. Flask provides a set of tools and libraries for building web applications, including support for URL routing, templates, and request/response handling.

One of the key advantages of Flask is its simplicity. Flask is easy to learn and use, and is a popular choice for small to medium-sized web applications. Flask also provides a lot of flexibility, allowing developers to choose which components they want to use and how they want to structure their application.

Flask is also highly extensible, with a large and active community of developers who have created a wide range of extensions and plugins for the framework. These extensions provide additional functionality for things like database integration, authentication, and form handling. Another advantage of Flask is its lightweight nature. Flask has a small codebase and minimal dependencies, making it easy to deploy and scale. This makes it a popular choice for developers who want to build simple and efficient web applications.

Overall, Flask is a powerful and flexible web framework that is well-suited for building small to medium-sized web applications. Its simplicity, flexibility, and extensibility make it an ideal choice for developers who want to build web applications quickly and with minimal code.



## 2.3 MODULES OF PYTHON USED IN THE APPLICATION

### **NUMPY:**

NumPy (short for "Numerical Python") is a popular Python library used for scientific computing and data analysis. It provides a powerful array processing capability that allows for efficient operations on large, multi-dimensional arrays and matrices.

NumPy provides a set of tools for working with arrays, including a wide range of mathematical functions for performing element-wise operations, linear algebra operations, and statistical analysis. It also provides tools for working with complex numbers and Fourier transforms, as well as tools for generating random numbers.

One of the key advantages of NumPy is its performance. NumPy is designed to be fast and efficient, with optimized code that allows for quick execution of mathematical operations on arrays. This makes it well-suited for scientific computing applications that require intensive computation on large datasets.

Another advantage of NumPy is its ease of use. NumPy provides a simple and intuitive interface for working with arrays, making it easy to get started with data analysis and scientific computing. It also integrates well with other Python libraries, such as Pandas and Matplotlib, making it a key component of the scientific Python ecosystem.

Overall, NumPy is a powerful and versatile library that is widely used for scientific computing and data analysis in Python. Its performance, ease of use, and rich set of tools make it a key component of the scientific Python ecosystem.

### **PANDAS:**

Pandas is a popular open-source Python library used for data manipulation and analysis. It provides a flexible and powerful set of tools for working with structured data, including data frames and series.

Pandas provides a wide range of features for data manipulation and analysis, including:

1. Data structures: Pandas provides two main data structures, Data Frames and Series, for working with structured data. Data Frames are two-dimensional tables with rows and columns, while Series are one-dimensional arrays with labeled indexes.
2. Data cleaning: Pandas provides a range of tools for cleaning and transforming data, including handling missing values, filtering, sorting, and merging data sets.
3. Data analysis: Pandas provides a range of tools for data analysis, including statistical functions, time series analysis, and data visualization.
4. Data input/output: Pandas provides support for reading and writing data in a wide range of formats, including CSV, Excel, SQL, and JSON, among others. Pandas is widely used in data science and analysis because of its ease of use, flexibility, and powerful set of tools. It integrates well with other Python libraries, such as NumPy and Matplotlib, making it a key component of the scientific Python ecosystem.

Overall, Pandas is a versatile and powerful library that is essential for working with structured data in Python. Its rich set of tools and easy-to-use interface make it a popular choice for data analysis and manipulation.

## **SKLEARN:**

Scikit-learn, also known as sklearn, is a popular machine learning library in Python. It provides a wide range of tools for various tasks in machine learning, including data preprocessing, classification, regression, clustering, model selection, and more. Scikit-learn provides a simple and consistent interface for working with various machine learning models. It includes a wide range of models, such as linear regression, logistic regression, decision trees, random forests, support vector machines, k-nearest neighbors, and more. It also provides tools for evaluating the performance of these models using various metrics.

One of the strengths of Scikit-learn is its ease of use and accessibility.

It provides a consistent and intuitive API for working with machine learning models, which makes it easy for beginners to get started with machine learning. At the same time, it also provides advanced features and tools for more experienced users, such as hyperparameter tuning and model selection.

Scikit-learn also integrates well with other popular Python libraries for data manipulation and visualization, such as Pandas, NumPy, and Matplotlib.

Overall, Scikit-learn is a powerful and versatile machine learning library in Python that simplifies the process of building and evaluating machine learning models. Its ease of use, consistency, and wide range of features make it a popular choice for data scientists and machine learning practitioners. And many other machine learning and data-visualization packages are used.

## **2.4 Machine Learning Classifiers:**

The machine learning-based classifiers have been trained based on the training data and tested the performance using the test data. The best classifier will be selected on the basis of their classification performance. Finally, the best performing classifier has been used on the validation data to forecast the rainy days based on the feature set. The sklearn library has been used to import these classifiers. The models have been trained using the X\_train and y\_train sets and tested on the X\_test data for predicting whether the day will be rainy or not.

### **Random Forest :**

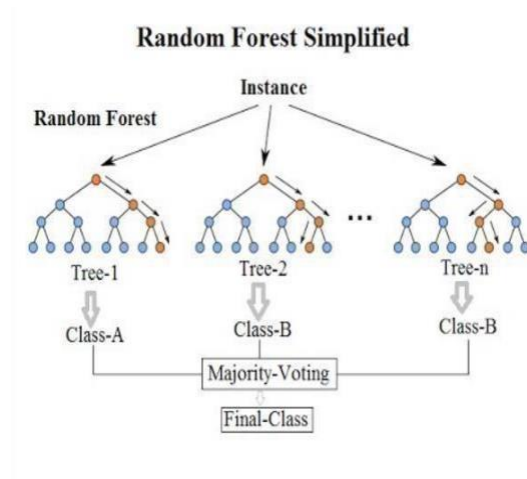
Random forest is a machine learning algorithm that is supervised. The bagging approach is used to train a forest, which is a collection of decision trees. Random forest generates a large number of decision trees and then combines them to get a consistent and accurate classification. The Random Forest algorithm has the advantage of being able to be used for both classification and regression analysis. Random Forest Algorithm follows the rules of Decision Tree. The difference in them is decision tree algorithm gives the output by considering only one factor whereas Random Forest Algorithm compares many no of decision trees and gives the result satisfying majority no of decision trees.

Random Forest Algorithm builds a strong model which satisfies the models many no of decision tree, this model is applied on the testing dataset for getting the required output. Firstly, for building a model we require a training dataset, from that training dataset we consider the subset of training dataset known as bootstrap dataset1. This bootstrap dataset1 consists of set of variables. From these variables we consider only two variables from these two variables we

make a root node for one variable which produces accurate output than the other variable. In this format we build a decision tree from the bootstrap dataset1. We consider another subset of dataset from the training dataset, let us say bootstrap dataset2. As we build a decision tree for the bootstrap dataset 1 one in the same way we build a dataset for the bootstrap dataset2. We have to follow these steps until we get many no of decision tree. After getting many no of decision trees we compare the entire decision tree and build a model which satisfies these entire decision trees. The obtained model is known as Strong Model. In this way a model is built from the training dataset.

This model is applied on the testing data set and it produces a required output. The obtained output is accurate because a strong model is built from many no. of decision trees.

In this format, when we upload a training dataset then the system builds a model using Random forests Algorithm then when upload the testing dataset by using the model system provides the required output. The output consists of two class labels yes/no. Yes indicates that the client is eligible for approval of loan and no indicates that the client is not eligible for the approval of loan. Like this, the system provides the required output.



It builds the number of decision tree models from the trained dataset and all these models are combined and form a new model which satisfies all the tree models. The obtained model is said to be known as a strong model since it satisfies all the decision tree models.

# **SYSTEM REQUIREMENTS**

### **3.1 HARDWARE REQUIREMENTS:**

For a project Forecasting Water Quality, you will need system software that can handle data analysis and machine learning tasks. Here are some software requirements to consider:

1. **Operating System:** The operating system should be a version of Windows, Linux, or macOS that supports data analysis and machine learning tools.
2. **Programming Language:** The project will require programming skills, so you will need a programming language that is well-suited for data analysis and machine learning. Python is a popular choice for this purpose, with libraries like NumPy, Pandas, Scikit-learn, and TensorFlow.
3. **Integrated Development Environment (IDE):** You will need an IDE to write, run, and debug your code. Some popular options for Python development include PyCharm, Spyder, and Jupyter Notebooks.
4. **Data Analysis Tools:** You will need tools for data manipulation, visualization, and exploration. This could include software like Excel, Tableau, or Matplotlib.
5. **Machine Learning Framework:** You will need a machine learning framework to build, train, and evaluate your models. Some popular options include TensorFlow, PyTorch, and Scikit-learn.
6. **Database Management System:** You will need a database management system to store and retrieve your data. This could include software like MySQL, PostgreSQL, or MongoDB.
7. **Version Control:** You will need version control software to manage changes to your codebase. Git is a popular option for this purpose.

#### **PROGRAMMING LANGUAGES:**

- **HTML**
- **CSS**
- **BOOTSTRAP**
- **PYTHON**
- **FLASK**

## **3.2 SOFTWARE REQUIREMENTS:**

### **FUNCTIONAL REQUIREMENTS :**

Requirement analysis is a software engineering that is composed of the various tasks that determine the needs or conditions that are to be met for a new or altered product, taking into consideration the possible conflicting requirements of the various users. Functional requirements are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. The functional requirements identified are:

- **USERS INPUT :**

In order for the application to work the user should provide some inputs of which specifications they want. They can give inputs or can select from different options.

- **OUTPUT:**

The model then works on the given inputs and perform the random forest model to predict whether the campaign will be a success or a failure according to the given parameters.

- **ALERTS:**

If user enters any wrong or nil input then application sends an alert to the user.

### **NON-FUNCTIONAL REQUIREMENTS :**

- **RESPONSE TIME:**

The system should have high performance rate when executing user's input and should be able to provide feedback or response within a short time span usually 50 seconds for highly complicated task.

- **ERROR HANDLING:**

Error should be considerably minimized and an appropriate error message that guides the user to recover from an error should be provided. Validation of user's input is highly essential. Also, the standard time taken to recover from an error should be 15 to 20 seconds.

**INPUT DESIGN:**

Input design is responsible for converting the user given inputs into the modal required parameters. This is done internally in the backend in the flask application. It typically converts the given inputs into 0 and 1 according to the specifications.

**OUTPUT DESIGN:**

Computer output is the most important and direct source of information to the user. Output design is a very important phase since the output needs to be in an efficient manner. Efficient and intelligible output design improves the system relationship with the user and helps in decision making.



# **SYSTEM DESIGN**

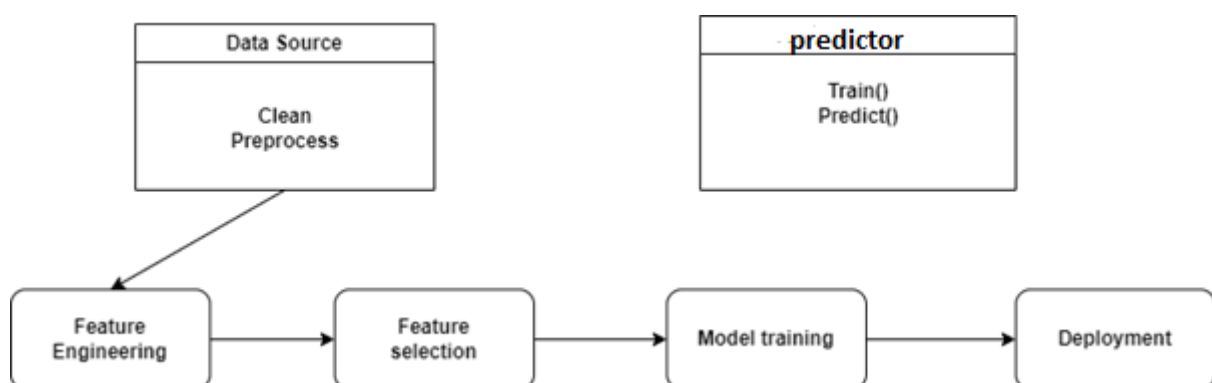
## 4.1 SYSTEM DESIGN:

System Design is the most creative and challenging. The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic and external interfaces. The system is designed to satisfy the user requirements based on analysis of the system. In system design we move from logical to the physical aspects. The requirements identified in the requirements analysis phase are transformed into a System Design Document that accurately describes the design of the system and that can be used as an input to system development in the next phase. During the Design Phase we have to consider about a key question “**HOW SHOULD BE PROBLEM SOLVED?**”

## 4.2 UML DIAGRAMS :

UML diagrams show a collection of classes, interfaces, associations, collaborations and constraints. It is also known as Structural diagram. The purpose of the UML diagrams is to model the static view of an application. UML diagrams are the only diagrams which can be directly mapped with object-oriented languages such that widely used at the time of construction.

UML diagrams like Activity diagram, Sequence diagram can only give the sequence flow of the application however UML diagrams is a bit different. It is the most popular UML diagram in the coder community.



# **IMPLEMENTATION AND TESTING**

Predicting the success of bank campaigns can be done using machine learning techniques. Here are the steps you can follow to implement prediction:

## **5.1 IMPLEMENTATION DESCRIPTION :**

- **Data Collection and Monitoring:**

Set up a robust data collection system to gather Bank campaigns data from various sources, including banks, government websites and historical records. Implement real-time monitoring systems to continuously collect data, ensuring the availability of up-to-date information.

- **Data Preprocessing:**

Clean and preprocess collected data to handle missing values, outliers, and inconsistencies. Standardize data formats and units to ensure consistency across datasets.

- **Feature Selection and Engineering:**

Identify relevant features (parameters) that affect Bank campaigns success. Perform feature engineering to create new features or transform existing ones to improve model performance.

- **Choice of Prediction Models:**

Select appropriate prediction models, which can include statistical, machine learning, or mathematical models. Consider the complexity of models and their suitability for different parameters.

- **Data Splitting and Validation:**

Divide the dataset into training, validation, and test sets to train, tune, and evaluate the models. Implement cross-validation techniques to assess model robustness and generalization.

- **Model Training and Tuning:**

Train prediction models using collected data, adjusting hyperparameters for optimal performance. Regularly update and retrain models to adapt to changing bank campaigning conditions.

## **5.2 SYSTEM TESTING :**

- **UNIT TESTING :**

Each module is considered independently, it focuses on each unit of software as implemented in the source code, it is white box testing.

- **INTEGRATION TESTING :**

Integration testing aims at constructing the program structure while at the same constructing tests to uncover errors associated with interfacing the modules. Modules are integrated by using the top-down approach.

- **FUNCTIONAL TESTING :**

Functional testing is a type of software testing that validates the software system against the functional specifications. The purpose of Functional test is to test each function of the software application, by providing appropriate functional requirements. Functional testing mainly involves black box testing and it is concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication.

- **SYSTEM TESTING :**

It is executing programs to check logical changes made in it with intention of finding errors in system is tested for online response, volume of transaction, recovery from failure etc. System testing is done to ensure that the system satisfies all the user requirements.

- **WHITE BOX TESTING :**

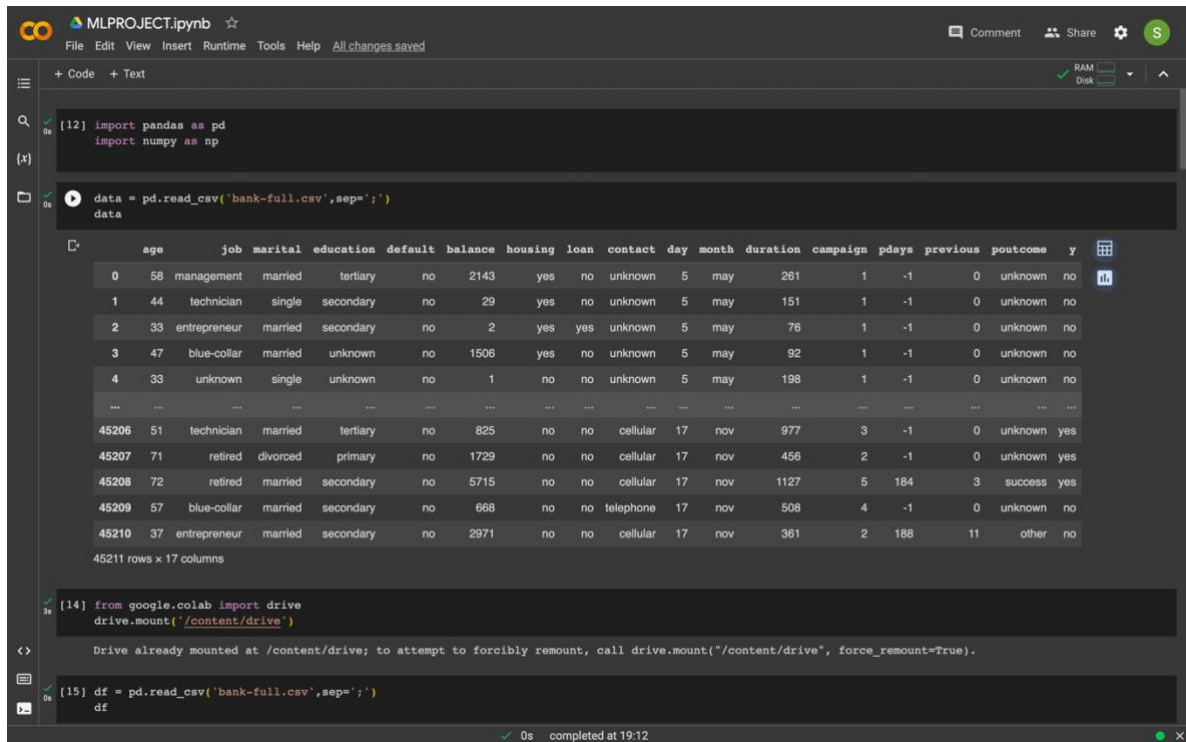
In this technique, the close examination of the logical parts through the software is tested by cases that exercise specific sets of conditions or loops. All logical parts of the software checked once. Errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. When the box testing tests all the independent part within a module a logical decision on their true and the false side are exercised, all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

- **BLACK BOX TESTING :**

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. Black box testing tests the input, the output and the external data. It checks whether the input data is correct and whether we are getting the desired output.

# **SOURCE CODE**

## 6.1 Python:



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
[12] import pandas as pd
import numpy as np

data = pd.read_csv('bank-full.csv', sep=';')
data
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

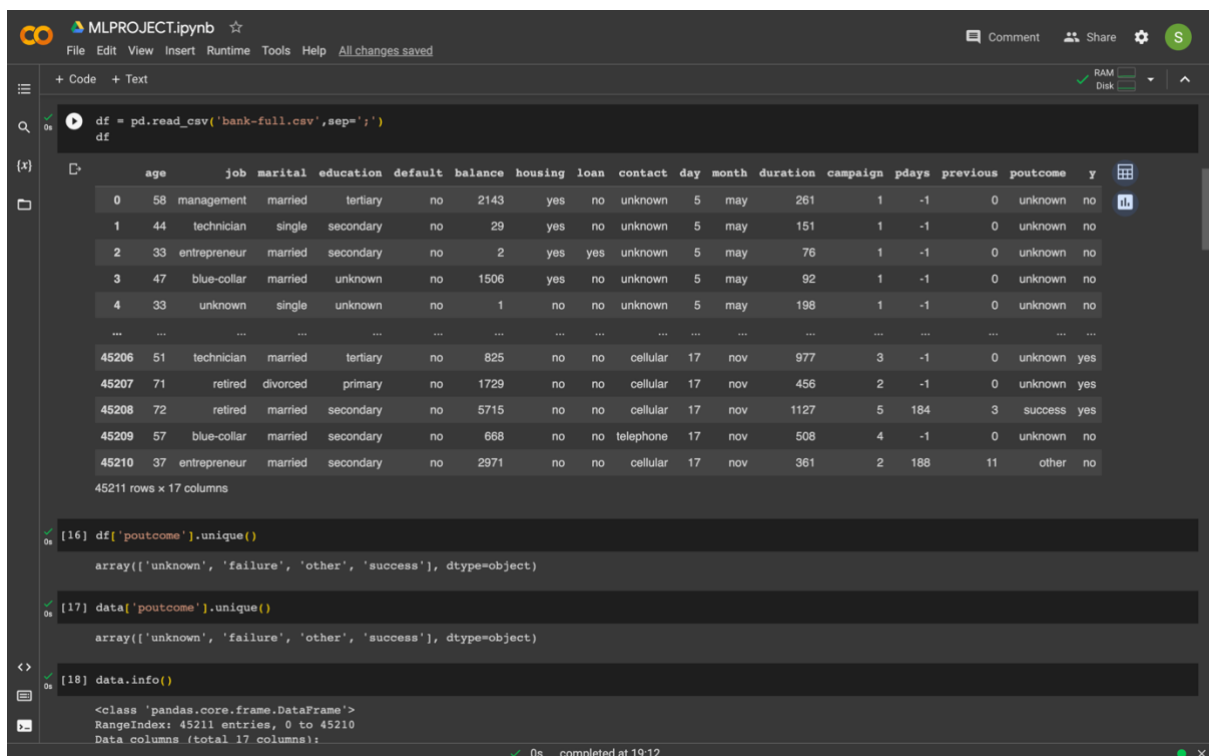
45211 rows x 17 columns

```
[14] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[15] df = pd.read_csv('bank-full.csv', sep=';')
df
```

completed at 19:12



The screenshot shows the same Jupyter Notebook with additional code and output:

```
df = pd.read_csv('bank-full.csv', sep=';')
df
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	3	-1	0	unknown	yes
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	2	-1	0	unknown	yes
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	5	184	3	success	yes
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	4	-1	0	unknown	no
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	2	188	11	other	no

45211 rows x 17 columns

```
[16] df['poutcome'].unique()

array(['unknown', 'failure', 'other', 'success'], dtype=object)
```

```
[17] data['poutcome'].unique()

array(['unknown', 'failure', 'other', 'success'], dtype=object)
```

```
[18] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
```

completed at 19:12



MLPROJECT.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

```
[22] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    age         45211 non-null  int64
1    job         45211 non-null  int64
2    marital     45211 non-null  int64
3    education   45211 non-null  int64
4    default     45211 non-null  int64
5    balance     45211 non-null  int64
6    housing     45211 non-null  int64
7    loan        45211 non-null  int64
8    contact     45211 non-null  int64
9    day         45211 non-null  int64
10   month       45211 non-null  int64
11   duration    45211 non-null  int64
12   campaign    45211 non-null  int64
13   pdays      45211 non-null  int64
14   previous    45211 non-null  int64
15   poutcome    45211 non-null  int64
16   y           45211 non-null  int64
dtypes: int64(17)
memory usage: 5.9 MB

x = data.drop('y',axis=1);
y = data['y'];
x
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	3
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	3
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	3

0s completed at 19:12

MLPROJECT.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

```
[18] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    age         45211 non-null  int64
1    job         45211 non-null  object
2    marital     45211 non-null  object
3    education   45211 non-null  object
4    default     45211 non-null  object
5    balance     45211 non-null  int64
6    housing     45211 non-null  object
7    loan        45211 non-null  object
8    contact     45211 non-null  object
9    day         45211 non-null  int64
10   month       45211 non-null  object
11   duration    45211 non-null  int64
12   campaign    45211 non-null  int64
13   pdays      45211 non-null  int64
14   previous    45211 non-null  int64
15   poutcome    45211 non-null  object
16   y           45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB

[19] from sklearn.preprocessing import LabelEncoder

[20] encoder = LabelEncoder()

for col in data.columns:
    if data[col].dtype=='object':
        data[col]=encoder.fit_transform(data[col])

[22] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
```

0s completed at 19:12

MLPROJECT.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[24] y

```
0 0
1 0
2 0
3 0
4 0
..
45206 1
45207 1
45208 1
45209 0
45210 0
Name: y, Length: 45211, dtype: int64
```

[25] from sklearn.model\_selection import train\_test\_split

[26] x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,train\_size=0.2,random\_state=123);

[27] from sklearn.ensemble import RandomForestClassifier

[28] model = RandomForestClassifier()

model.fit(x\_train,y\_train)

RandomForestClassifier

RandomForestClassifier()

[30] data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
# Column Non-Null Count Dtype
---
0 age 45211 non-null int64
```

0s completed at 19:12

MLPROJECT.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

x = data.drop('y',axis=1);  
y = data['y'];  
x

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	3
2	33	2	1	0	2	1	1	2	5	8	76	1	-1	0	3	
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	-1	0	3
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	-1	0	3
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3	2
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	-1	0	3
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11	1

45211 rows x 16 columns

[24] y

```
0 0
1 0
2 0
3 0
4 0
..
45206 1
45207 1
45208 1
45209 0
45210 0
Name: y, Length: 45211, dtype: int64
```

0s completed at 19:12

MLPROJECT.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    age         45211 non-null  int64
1    job         45211 non-null  int64
2    marital     45211 non-null  int64
3    education   45211 non-null  int64
4    default     45211 non-null  int64
5    balance     45211 non-null  int64
6    housing     45211 non-null  int64
7    loan        45211 non-null  int64
8    contact     45211 non-null  int64
9    day         45211 non-null  int64
10   month       45211 non-null  int64
11   duration    45211 non-null  int64
12   campaign    45211 non-null  int64
13   pdays      45211 non-null  int64
14   previous    45211 non-null  int64
15   poutcome    45211 non-null  int64
16   y           45211 non-null  int64
dtypes: int64(17)
memory usage: 5.9 MB
```

[31] data

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3	0
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	3	0
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	3	0
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3	0
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	-1	0	3	1

0s completed at 19:12

MLPROJECT.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

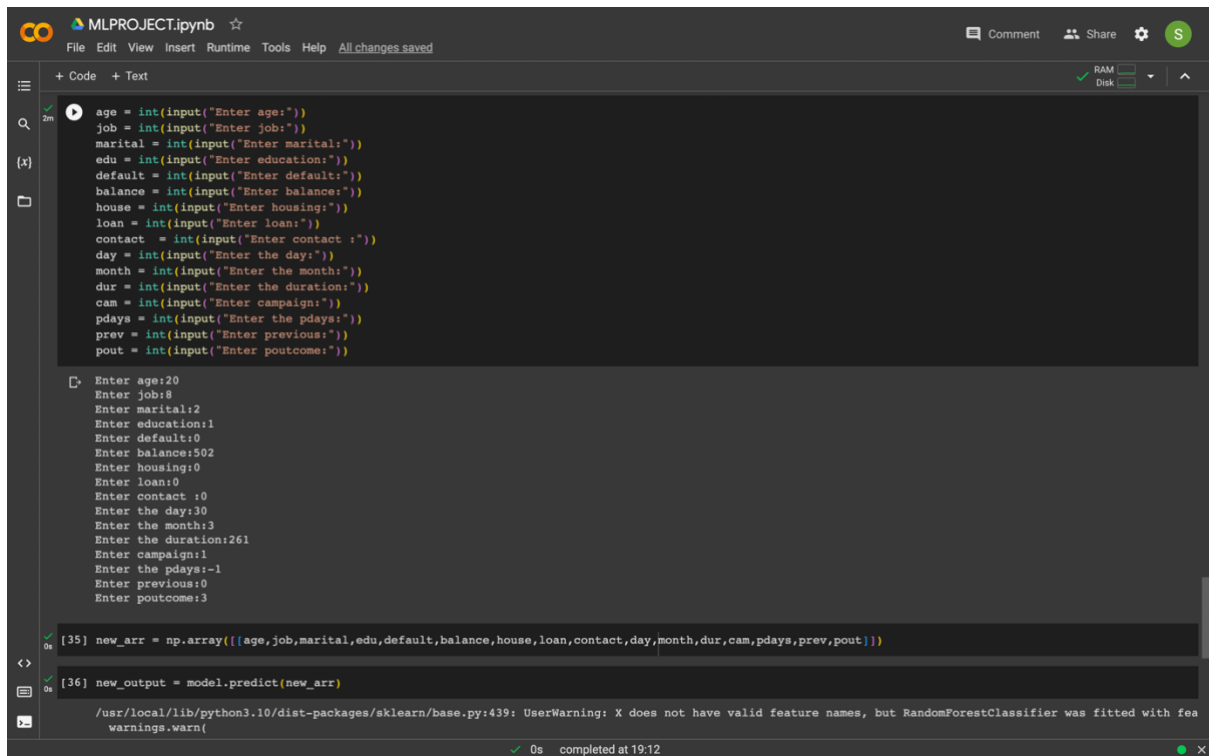
data

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3	0
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	3	0
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	3	0
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3	0
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51	9	1	2	0	825	0	0	0	17	9	977	3	-1	0	3	1
45207	71	5	0	0	0	1729	0	0	0	17	9	456	2	-1	0	3	1
45208	72	5	1	1	0	5715	0	0	0	17	9	1127	5	184	3	2	1
45209	57	1	1	1	0	668	0	0	1	17	9	508	4	-1	0	3	0
45210	37	2	1	1	0	2971	0	0	0	17	9	361	2	188	11	1	0

45211 rows x 17 columns

```
[34] age = int(input("Enter age:"))
job = int(input("Enter job:"))
marital = int(input("Enter marital:"))
edu = int(input("Enter education:"))
default = int(input("Enter default:"))
balance = int(input("Enter balance:"))
house = int(input("Enter housing:"))
loan = int(input("Enter loan:"))
contact = int(input("Enter contact :"))
day = int(input("Enter the day:"))
month = int(input("Enter the month:"))
dur = int(input("Enter the duration:"))
cam = int(input("Enter campaign:"))
pdays = int(input("Enter the pdays:"))
prev = int(input("Enter previous:"))
```

0s completed at 19:12



The image shows a Jupyter Notebook interface with the title 'MLPROJECT.ipynb'. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating 'All changes saved'. The left sidebar shows icons for file explorer, search, and other notebook functions. The main area contains a code cell with the following Python code:

```
age = int(input("Enter age:"))
job = int(input("Enter job:"))
marital = int(input("Enter marital:"))
edu = int(input("Enter education:"))
default = int(input("Enter default:"))
balance = int(input("Enter balance:"))
house = int(input("Enter housing:"))
loan = int(input("Enter loan:"))
contact = int(input("Enter contact :"))
day = int(input("Enter the day:"))
month = int(input("Enter the month:"))
dur = int(input("Enter the duration:"))
cam = int(input("Enter campaign:"))
pdays = int(input("Enter the pdays:"))
prev = int(input("Enter previous:"))
pout = int(input("Enter poutcome:"))
```

Below the code, there is a text input area showing the following values entered:

```
Enter age:20
Enter job:8
Enter marital:2
Enter education:1
Enter default:0
Enter balance:502
Enter housing:0
Enter loan:0
Enter contact :0
Enter the day:30
Enter the month:3
Enter the duration:261
Enter campaign:1
Enter the pdays:-1
Enter previous:0
Enter poutcome:3
```

The code cell is followed by two more cells:

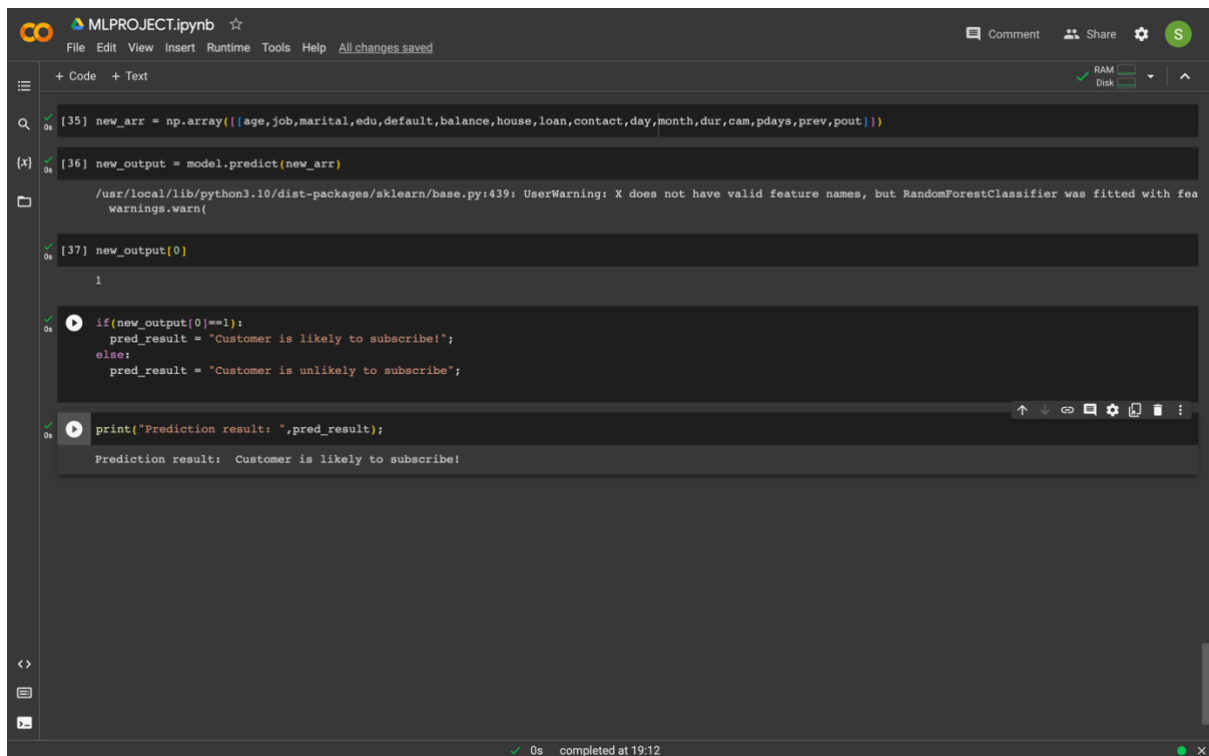
```
[35] new_arr = np.array([[age,job,marital,edu,default,balance,house,loan,contact,day,month,dur,cam,pdays,prev,pout]])
```

```
[36] new_output = model.predict(new_arr)
```

The output of the second cell shows a warning message:

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with fea
warnings.warn(
```

The bottom status bar indicates '0s completed at 19:12'.



The image shows a Jupyter Notebook interface with the title 'MLPROJECT.ipynb'. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating 'All changes saved'. The left sidebar shows icons for file explorer, search, and other notebook functions. The main area contains a code cell with the following Python code:

```
[35] new_arr = np.array([[age,job,marital,edu,default,balance,house,loan,contact,day,month,dur,cam,pdays,prev,pout]])
```

```
[36] new_output = model.predict(new_arr)
```

The output of the second cell shows a warning message:

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with fea
warnings.warn(
```

The code cell is followed by two more cells:

```
[37] new_output[0]
```

```
1
```

```
if(new_output[0]==1):
    pred_result = "Customer is likely to subscribe!";
else:
    pred_result = "Customer is unlikely to subscribe";
```

```
print("Prediction result: ",pred_result);
```

The output of the last cell shows the prediction result:

```
Prediction result: Customer is likely to subscribe!
```

The bottom status bar indicates '0s completed at 19:12'.

## 6.2 HTML:

```
templates/index.html - ML-Proj x +
replit.com/@SaiNigam/ML-Project#templates/index.html
ML-Project Stop
index.html app.py prediction.html requirements.txt +
templates > index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Forecasting Bank Campaign</title>
5 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVglZdg1CS+VKNBQMNGChEKQm+PtmoHDEuppvndJzQiu9" crossorigin="anonymous">
6 </head>
7 <body>
8 <div class="container mt-5">
9 <h1 class="mb-4">Forecasting Bank Campaign</h1>
10 <form action="/predict" method="post">
11 <div class="row">
12 <!-- Age -->
13 <div class="col-md-6 mb-3">
14 <label for="age">Enter the age:</label>
15 <input type="number" class="form-control" id="age" name="age" placeholder="Age" required>
16 </div>
17 <!-- Job -->
18 <div class="col-md-6 mb-3">
19 <label for="job">Select job:</label>
20 <select class="form-select" id="job" name="job" required>
21 <option selected>Choose a job:</option>
22 <option value="0">admin</option>
23 <option value="1">blue-collar</option>
24 <option value="2">entrepreneur</option>
25 <option value="3">house-maid</option>
26 <option value="4">management</option>
27 <option value="5">retired</option>
28 <option value="6">self-employed</option>
29 <option value="7">services</option>
30 <option value="8">student</option>
31 <option value="9">technician</option>
```

```
templates/index.html - ML-Proj x +
replit.com/@SaiNigam/ML-Project#templates/index.html
ML-Project Stop
index.html app.py prediction.html requirements.txt +
templates > index.html
38 <div class="row">
39 <!-- Marital Status -->
40 <div class="col-md-6 mb-3">
41 <label for="marital">Select marital status:</label>
42 <select id="marital" class="form-select" name="marital" required>
43 <option value="0">Divorced</option>
44 <option value="1">Married</option>
45 <option value="2">Single</option>
46 </select>
47 </div>
48 <!-- Education -->
49 <div class="col-md-6 mb-3">
50 <label for="education">Select education status:</label>
51 <select id="education" class="form-select" name="education" required>
52 <option value="0">Primary</option>
53 <option value="1">Secondary</option>
54 <option value="2">Tertiary</option>
55 <option value="3">Unknown</option>
56 </select>
57 </div>
58 </div>
59
60 <div class="row">
61 <!-- Default -->
62 <div class="col-md-6 mb-3">
63 <label for="default">Select default status:</label>
64 <select id="default" class="form-select" name="default" required>
65 <option value="0">No</option>
66 <option value="1">Yes</option>
67 </select>
68 </div>
69 <!-- Balance -->
```

```
templates/index.html - ML-Proj x +
replit.com/@SaiNigam/ML-Project#templates/index.html

ML-Project
Stop

index.html x app.py x prediction.html x requirements.txt x +
templates > index.html

69 <!-- Balance -->
70 <div class="col-md-6 mb-3">
71   <label for="balance">Enter balance:</label>
72   <input type="number" id="balance" name="balance" class="form-control" placeholder="Balance" required>
73 </div>
74 </div>
75
76 <div class="row">
77   <!-- Housing -->
78   <div class="col-md-6 mb-3">
79     <label for="house">Select housing details:</label>
80     <select id="house" class="form-select" name="house" required>
81       <option value="0">No</option>
82       <option value="1">Yes</option>
83     </select>
84   </div>
85
86   <!-- Loan -->
87   <div class="col-md-6 mb-3">
88     <label for="loan">Select loan details:</label>
89     <select id="loan" class="form-select" name="loan" required>
90       <option value="0">No</option>
91       <option value="1">Yes</option>
92     </select>
93   </div>
94 </div>
95
96 <div class="row">
97   <!-- Number of Days -->
98   <div class="col-md-6 mb-3">
99     <label for="day">Number of days:</label>
100    <input type="number" class="form-control" id="day" name="day" placeholder="Number of days" required>

```

```
templates/index.html - ML-Proj x +
replit.com/@SaiNigam/ML-Project#templates/index.html

ML-Project
Stop

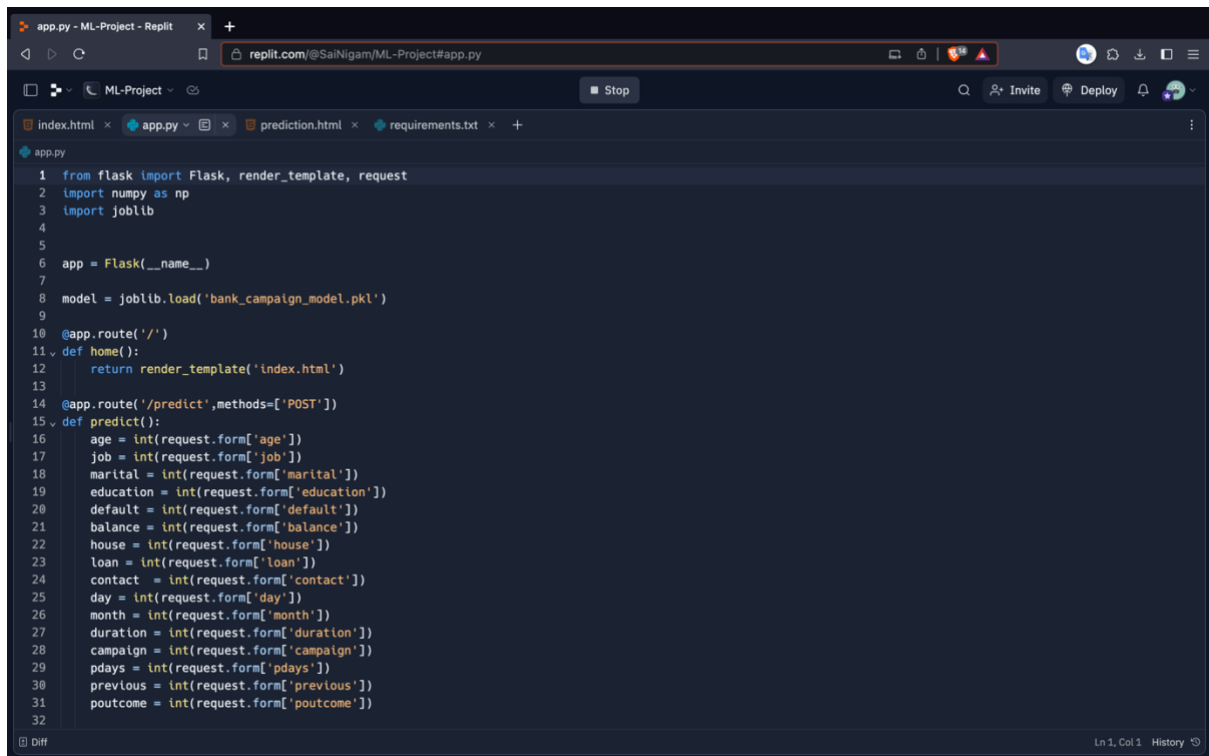
index.html x app.py x prediction.html x requirements.txt x +
templates > index.html

102 <!-- MONTH -->
103 <div class="col-md-6 mb-3">
104   <label for="month">Select the month:</label>
105   <select id="month" class="form-select" name="month" required>
106     <option value="0">january</option>
107     <option value="1">february</option>
108     <option value="2">march</option>
109     <option value="3">april</option>
110     <option value="4">may</option>
111     <option value="5">june</option>
112     <option value="6">july</option>
113     <option value="7">august</option>
114     <option value="8">september</option>
115     <option value="9">october</option>
116     <option value="10">november</option>
117     <option value="11">december</option>
118   </select>
119 </div>
120 </div>
121
122 <div class="row">
123   <!-- Duration -->
124   <div class="col-md-6 mb-3">
125     <label for="duration">Enter the duration of the campaign:</label>
126     <input type="number" class="form-control" id="duration" name="duration" placeholder="Duration" required>
127   </div>
128   <!-- Campaign -->
129   <div class="col-md-6 mb-3">
130     <label for="campaign">Enter the campaign value:</label>
131     <input type="number" id="campaign" name="campaign" class="form-control" placeholder="Campaign Value" required>
132   </div>
133 </div>

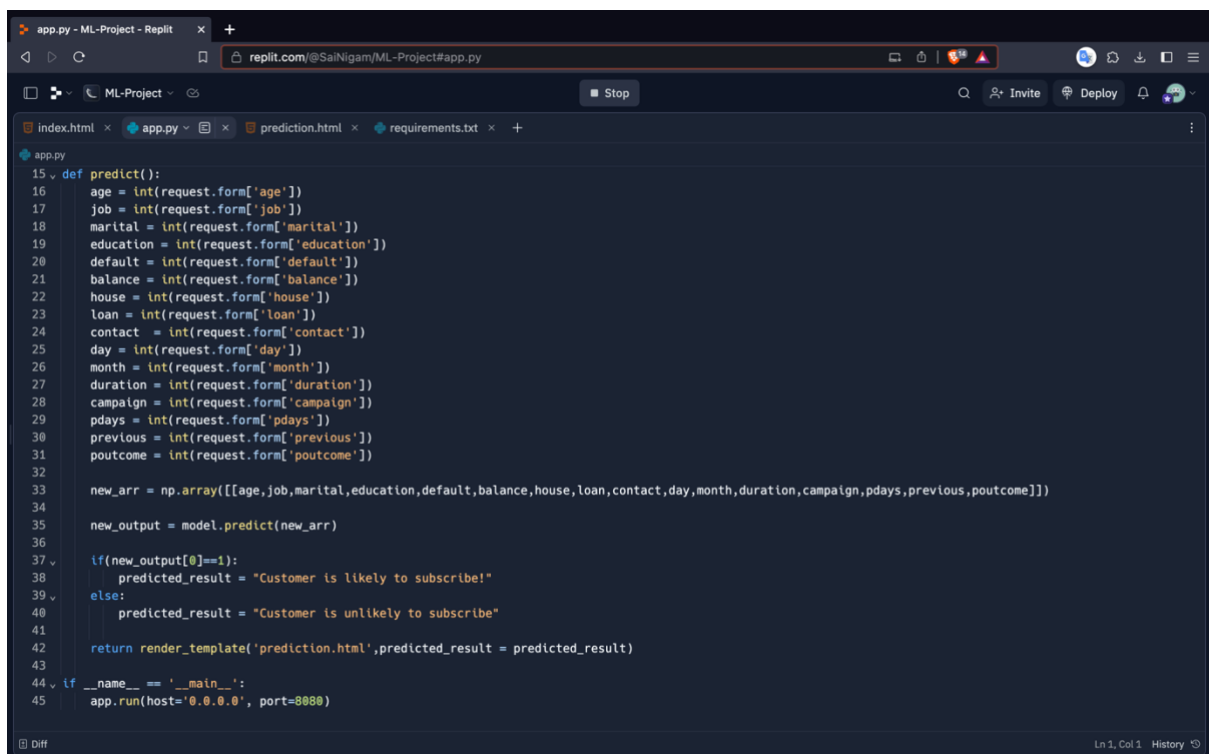
```

```
templates/index.html - ML-Proj x +
replit.com/@SaiNigam/ML-Project#templates/index.html
ML-Project ML-Project Stop
index.html x app.py x prediction.html x requirements.txt x +
templates > index.html
...
148 <div class="row">
149 <!-- Contact -->
150 <div class="col-md-6 mb-3">
151 <label for="contact">Select contact details:</label>
152 <select id="contact" class="form-select" name="contact" required>
153 <option value="0">Cellular</option>
154 <option value="1">Telephone</option>
155 <option value="2">Unknown</option>
156 </select>
157 </div>
158
159 <!-- Poutcome -->
160 <div class="col-md-6 mb-3">
161 <label for="poutcome">Choose the poutcome:</label>
162 <select id="poutcome" class="form-select" name="poutcome" required>
163 <option value="0">failure</option>
164 <option value="1">other</option>
165 <option value="2">success</option>
166 <option value="3">unknown</option>
167 </select>
168 </div>
169 </div>
170
171 <button type="submit" class="btn btn-primary">Predict</button>
172 </form>
173 </div>
174
175 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
HwwvtgBN03bZJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInIXGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>
176 </body>
177 </html>
178
Diff Ln 1, Col 1 History
```

## 6.3 FLASK:



```
1 from flask import Flask, render_template, request
2 import numpy as np
3 import joblib
4
5
6 app = Flask(__name__)
7
8 model = joblib.load('bank_campaign_model.pkl')
9
10 @app.route('/')
11 def home():
12     return render_template('index.html')
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     age = int(request.form['age'])
17     job = int(request.form['job'])
18     marital = int(request.form['marital'])
19     education = int(request.form['education'])
20     default = int(request.form['default'])
21     balance = int(request.form['balance'])
22     house = int(request.form['house'])
23     loan = int(request.form['loan'])
24     contact = int(request.form['contact'])
25     day = int(request.form['day'])
26     month = int(request.form['month'])
27     duration = int(request.form['duration'])
28     campaign = int(request.form['campaign'])
29     pdays = int(request.form['pdays'])
30     previous = int(request.form['previous'])
31     poutcome = int(request.form['poutcome'])
32
```



```
33
34 new_arr = np.array([age, job, marital, education, default, balance, house, loan, contact, day, month, duration, campaign, pdays, previous, poutcome])
35
36 new_output = model.predict(new_arr)
37
38 if new_output[0]==1:
39     predicted_result = "Customer is likely to subscribe!"
40 else:
41     predicted_result = "Customer is unlikely to subscribe"
42
43 return render_template('prediction.html', predicted_result = predicted_result)
44
45 if __name__ == '__main__':
46     app.run(host='0.0.0.0', port=8080)
```



# RESULTS

Forecasting Bank Campaign

ml-project-qxo8.onrender.com

## Forecasting Bank Campaign

Enter the age: <input type="text" value="Age"/>	Select job: <input type="text" value="Choose a job:"/>
Select marital status: <input type="text" value="Divorced"/>	Select education status: <input type="text" value="Primary"/>
Select default status: <input type="text" value="No"/>	Enter balance: <input type="text" value="Balance"/>
Select housing details: <input type="text" value="No"/>	Select loan details: <input type="text" value="No"/>
Number of days: <input type="text" value="Number of days"/>	Select the month: <input type="text" value="january"/>
Enter the duration of the campaign: <input type="text" value="Duration"/>	Enter the campaign value: <input type="text" value="Campaign Value"/>
Enter the number of pdays: <input type="text" value="Number of pdays"/>	Enter the previous value: <input type="text" value="Previous Value"/>
Select contact details: <input type="text" value="Cellular"/>	Choose the poutcome: <input type="text" value="failure"/>

Predict

Prediction

ml-project-qxo8.onrender.com/predict

## Forecast of Campaign Using the Given Data

Predicted value: Customer is likely to subscribe!

# CONCLUSION

In summary, the "Marketing Campaign Success Prediction" project stands as a groundbreaking leap forward in the realm of banking marketing. This initiative combines cutting-edge machine learning capabilities with a user-friendly interface, empowering banks to harness the full potential of their data for marketing success. It effectively addresses the limitations of traditional marketing decision-making, where past experiences and gut feelings often guided campaign strategies.

With this innovative system in place, banks gain a substantial competitive advantage. They can anticipate campaign outcomes with remarkable precision, allocate resources judiciously, and fine-tune messaging to align with the specific preferences of their diverse customer base. This transition from reactive marketing to a proactive, data-driven approach not only enhances campaign success but also optimizes resource utilization, reducing waste. Moreover, the system's commitment to interpretability fosters trust among marketing professionals, enabling them to understand the rationale behind each prediction and make strategic refinements. The "Marketing Campaign Success Prediction" project is not just a tool; it is a strategic compass, guiding banks through the dynamic landscape of modern marketing in the banking sector.

In conclusion, this project signifies the next evolution of marketing in the banking industry, offering a pathway to thrive in the age of data. It not only enhances the efficiency and effectiveness of marketing campaigns but also elevates customer engagement, acquisition rates, and, ultimately, financial performance. As a testament to the power of data-driven decision-making, it is more than just a tool in the marketing toolkit; it is a catalyst for banking institutions to redefine their strategies, achieve unprecedented success, and maintain a leading position in the competitive landscape.

## **BIBLIOGRAPHY :**

### **PYTHON - REFERENCE**

- ❖ TUTORIAL - CODEGNAN IT SOLUTIONS
- ❖ RESOURCES - <https://tinyurl.com/cgmlcbit>

### **FLASK, MACHINE LEARNING – REFERENCE**

- ❖ TUTORIAL - CODEGNAN IT SOLUTIONS
- ❖ RESOURCES - <https://tinyurl.com/cgmlcbit>
- ❖ YOUTUBE TUTORIAL - <https://youtu.be/yBDHkveJUf4?feature=shared>

**Data set web-scraped from different websites.**