High-Level Design (HLD)

Credit Card Default Prediction

Revision Number: 1.0

Last date of revision: 30/07/2023

Document Version Control

Date Issued	Version	Description	Author
29/07/2023	1	Initial HLD	ABHISHEK

Contents

Abstract

This High-Level Design Document provides an overview of the project titled "Credit Card default prediction". It outlines the purpose, scope, and technical details of the project, including its general description, proposed solution, and further improvements. The document also presents the design details, process flow, error handling, performance considerations, and deployment strategy. Additionally, it covers the key performance indicators displayed in the dashboards and concludes with a summary of the project.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
  - Security
  - Reliability
  - Maintainability
  - Portability
  - Reusability
  - Application compatibility
  - Resource utilization
  - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	Description
Defaults	Credit card defaults prediction
Database	Collection of all the information monitored by this system
IDE	Integrated Development Environment
AWS	Amazon Web Services

2 General Description

2.1 Product Perspective

The project is a standalone machine learning pipeline that includes both training and deployment components. The model is trained offline using historical data, and the trained model is deployed as a web application for credit card default prediction

2.3 PROPOSED SOLUTION

The proposed solution involves the following steps:

- Data Collection: Gather historical credit card transaction data, including customer demographics and payment history.
- Data Preprocessing: Perform data cleaning, handle missing values, and scale numerical features.
- Feature Engineering: Select relevant features and engineer new features for model training.
- Model Training: Train a machine learning model on the preprocessed data to predict credit card default.
- Model Evaluation: Evaluate the model's performance using appropriate metrics.
- Web Application Development: Develop a user-friendly web application for users to input their data and receive predictions.

2.4 FURTHER IMPROVEMENTS

Future enhancements to the project could include:

- Incorporating more advanced machine learning algorithms for improved prediction accuracy.
- Implementing a real-time prediction feature to handle incoming credit card transactions.
- Enhancing the web application's user interface for better user experience.

2.5 Technical Requirements

The project is developed using the following technologies and tools:

- Programming Language: Python
- Machine Learning Libraries: Scikit-learn, XGBoost

- Web Framework: Flask
- Frontend: HTML, CSS, JavaScript
- Data Storage: CSV files, Pickle files
- IDE: Virtual Studio Code (VS Code)

2.6 Data Requirements

The project requires historical credit card transaction data, including customer information, payment history, and default labels. The data must be properly formatted and preprocessed for training the machine learning model.

Data requirements completely depend on our problem statement.

- We need data that is balanced and must have at least 500 rows .
- Some features related to the problem.

2.7 Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn are used to build the whole model.

- VS Code is used as IDE.
- For visualization of the plots, Matplotlib, Seaborn and Plotly are used.

- AWS is used for deployment of the model.
- Tableau/Power Bl is used for dashboard creation.
- MySQL/MongoDB is used to retrieve, insert, delete, and update the database.
- Front end development is done using HTML/CSS
- GitHub is used as version control system.

2.7.1Hardware Requirements

The project can be executed on a standard computer with the following minimum hardware specifications:

- CPU: Dual-core processor
- RAM: 8GB
- Storage: 100GB

2.7.2 ROS (Robotic Operating System)

The Robotic Operating System (ROS) is used for deploying the web application on a cloud server or virtual machine.

2.8 Constraints

The project's constraints include:

- Limited computing resources for training large models or handling massive datasets.
- Sensitivity of customer data, requiring proper security measures during data handling and storage.

2.9 Assumptions

The project assumes that:

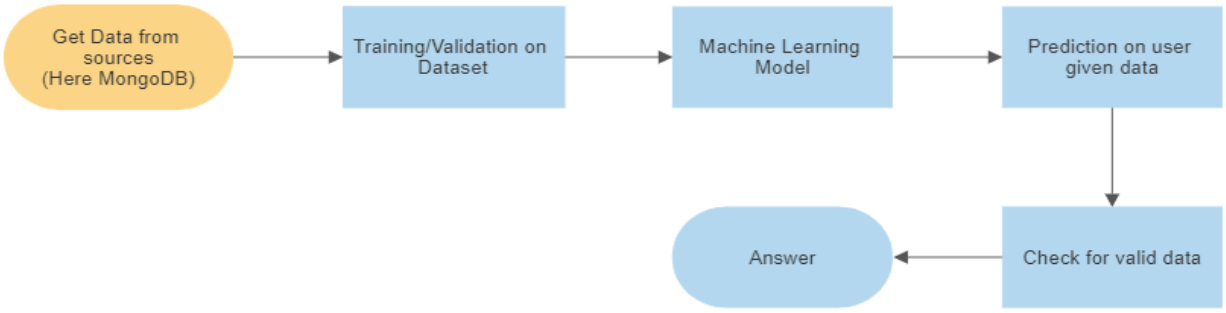
- The provided historical credit card data is reliable and relevant for credit card default prediction.
- Users input accurate data in the web application for prediction

3 Design Details

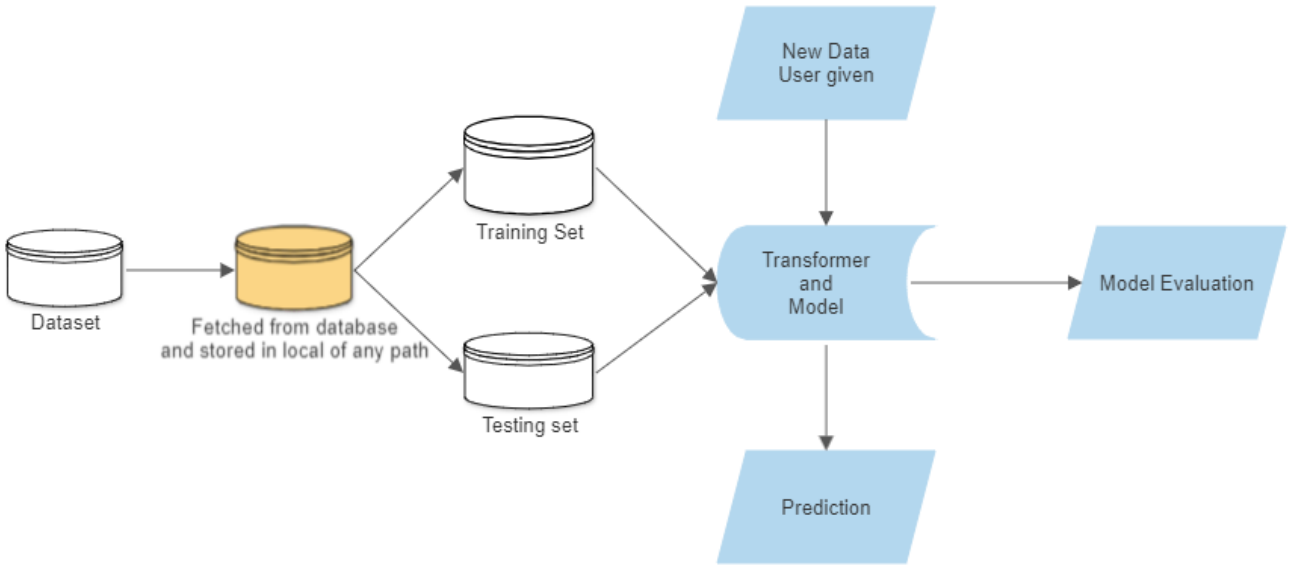
3.1 Process Flow

The process flow of the project consists of two main stages:

- Model Training
- Deployment



3.1.1 Model Training and Evaluation

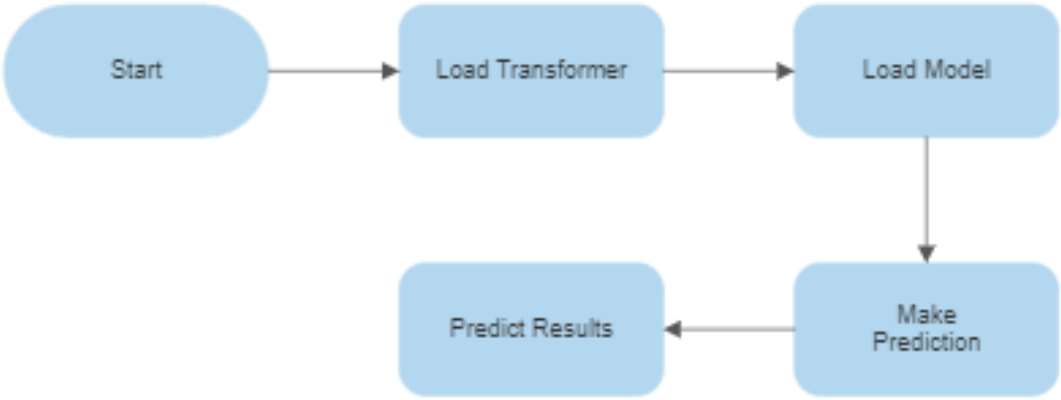


- Data Collection: Historical credit card transaction data is collected from reliable sources.

- Data Preprocessing: The raw data is cleaned, missing values are handled, and numerical features are scaled.
- Feature Engineering: Relevant features are selected, and new features are engineered based on domain knowledge.
- Model Training: The preprocessed data is used to train a machine learning model (e.g., XGBoost).
- Model Evaluation: The trained model's performance is evaluated using appropriate metrics (e.g., accuracy, precision, recall).

3.1.2 Deployment Process

- Web Application Development: A user-friendly web application is developed using the Flask web framework.
- Data Preprocessing: The user's input data is preprocessed using the saved data transformer.
- Prediction: The preprocessed data is fed into the trained model to make credit card default predictions.
- Result Display: The prediction results are displayed on the web application as "Default Payment" or "No Default Payment."



3.2 Event log

The project maintains an event log to record critical events and errors during model training and deployment.

Initial Step-By-Step Description:

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3 Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

3.4 Performance

Performance considerations involve optimizing the model for faster prediction and minimizing resource consumption.

4.1 Reusability

The project aims to maximize code reusability by organizing components into reusable functions and modules.

4.2 Application Compatibility

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

4.3 Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

4.4 Deployment

Amazon Web Services

5. Conclusion

This High-Level Design Document presents an overview of the "Credit Card Default Prediction" project, detailing its objectives, scope, technical details, and design considerations

References

[1] Scikit-learn: Machine Learning in Python. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Journal of Machine Learning Research, 12, 2825-2830.

[2] XGBoost: A Scalable Tree Boosting System. Chen, T., & Guestrin, C. (2016). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

[3] Flask: Web Development with Python. Ronacher, A. (2018). O'Reilly Media.

[4] Python. https://www.python.org/

[5] Robotic Operating System (ROS). https://www.ros.org/

[6] HTML, CSS, and JavaScript. MDN Web Docs. https://developer.mozilla.org/

[7] Credit Card Default Dataset. https://www.kaggle.com/

[8] Machine Learning Mastery. https://machinelearningmastery.com/

[9] Towards Data Science. https://towardsdatascience.com/

[10] DataCamp. https://www.datacamp.com/

[11] iNeuron. https://ineuron.ai/

Please note that the list of references may not be exhaustive and additional sources may have been used during the development of this project.