Credit card default prediction

**Abstract**

Financial threats are displaying a trend about the credit risk of commercial banks as the

incredible improvement in the financial industry has arisen. In this way, one of the

biggest threats faces by commercial banks is the risk prediction of credit clients. The

goal is to predict the probability of credit default based on credit card owner's

characteristics and payment history.

**Introduction**

**Why this High-Level Design Document?**  
This High-Level Design (HLD) Document is created to provide a comprehensive overview of the Credit Card Default Prediction Project. It outlines the system architecture, design aspects, and functional requirements necessary for the successful implementation of the project. This document serves as a blueprint for developers and stakeholders, ensuring clarity and alignment before proceeding to the coding phase. It also helps in identifying potential contradictions or challenges early in the process.

The HLD will:

* Present all design aspects and define them in detail.
* Describe the user interface being implemented.
* Outline the hardware and software interfaces.
* Define performance requirements.
* Include design features and the architecture of the project.
* List and describe the non-functional attributes, including security, reliability, maintainability, portability, reusability, application compatibility, resource utilization, and serviceability.

**Scope**

The HLD documentation provides the structural details of the system, covering database architecture, application architecture (layers), application flow (navigation), and technology architecture. The language used is non-technical to mildly technical, making it accessible to system administrators and other stakeholders.

**Definitions**

| **Term** | **Description** |
| --- | --- |
| CI/CD | Continuous Integration/Continuous Deployment |
| Dataset | Credit Card Default dataset from Kaggle |
| IDE | Integrated Development Environment (e.g., VSCODE) |
| AWS | Amazon Web Services |

**General Description**

**Product Perspective**  
The Credit Card Default Prediction system is a machine learning-based solution aimed at predicting whether a customer will default on their credit card payment. The system leverages historical data from a well-known Kaggle dataset, using various features related to customer demographics, credit history, and payment behavior to make predictions.

**Problem Statement**  
The goal is to develop a robust and scalable CI/CD pipeline that automates the process of building, testing, and deploying a credit card default prediction model. The pipeline should ensure that the model is always up-to-date with the latest data and can be easily integrated into a production environment.

**Proposed Solution**  
The solution involves creating a CI/CD pipeline that integrates various tools and technologies to automate the model's lifecycle. The pipeline will include stages for data ingestion, preprocessing, model training, testing, and deployment. The model will be deployed on AWS, with a user-friendly interface for managing and monitoring predictions.

**Technical Requirements**

**Data Requirements**

* The dataset includes features such as LIMIT\_BAL, SEX, EDUCATION, MARRIAGE, AGE, and payment history (PAY\_0 to PAY\_6).
* Data preprocessing steps include handling missing values, encoding categorical variables, and scaling numerical features.
* The dataset will be split into training and testing sets to evaluate the model's performance.

**Tools Used**

* **Programming Languages & Frameworks**: Python, NumPy, Pandas, Scikit-learn, TensorFlow.
* **IDE**: VsCode.
* **Visualization**: Matplotlib, Seaborn, Plotly.
* **CI/CD Tools**: GitHub Actions.
* **Cloud Services**: AWS (for deployment).
* **Front-End Development**: HTML/CSS.
* **Back-End Development**: Flask
* **Version Control**: GitHub.

**Hardware Requirements**

* A machine with at least 8 GB of RAM for model training and testing.
* Access to AWS for cloud-based deployment and scaling.
* GPU support for TensorFlow model training (optional but recommended).

**Design Details**

**Process Flow**

1. **Data Ingestion**: Data is extracted from the Kaggle dataset and loaded into the system.
2. **Data Validation**: Data is cleaned, encoded, and scaled to ensure it is suitable for model training.
3. **Data transformation**: Train/Test split and scaling is done
4. **Model Training**: A machine learning model is trained using TensorFlow. Hyperparameters are tuned for optimal performance.
5. **Model Evaluation**: The model is evaluated using metrics such as accuracy, precision, recall, and F1-score.
6. **Deployment**: The trained model is deployed on AWS using a Flask based API.
7. **Monitoring & Maintenance**: The model is monitored for performance and retrained as needed using the CI/CD pipeline.

**Event Log**  
The system logs all events, including data processing steps, model training progress, and deployment status. Logs are stored in a centralized database for easy access and troubleshooting.

**Error Handling**  
The system provides detailed error messages in case of failures during any stage of the pipeline. Errors are logged, and notifications are sent to the development team for prompt resolution.

**Performance**  
The CI/CD pipeline ensures that the model is continuously updated with new data, maintaining high accuracy and reliability. The system is designed to handle large datasets efficiently, utilizing cloud resources as needed.

**Reusability**  
The codebase is modular, allowing components to be reused for other machine learning projects. The CI/CD pipeline can be adapted for different models with minimal changes.

**Application Compatibility**  
The system is compatible with various platforms and environments, thanks to the use of Python and containerization technologies like Docker.

**Resource Utilization**  
The system is optimized to make efficient use of computational resources, scaling up or down based on the workload.

**Deployment**

The model is deployed on AWS, with endpoints exposed via a RESTful API. The API allows external systems to interact with the model for real-time predictions. A dashboard is provided for monitoring the system's performance and managing updates.

**Dashboards**

Dashboards are implemented to display key performance indicators (KPIs) such as model accuracy, number of predictions made, and system uptime. The dashboards also provide insights into data trends and customer behavior.

**KPIs (Key Performance Indicators)**

* Model accuracy and other evaluation metrics.
* Number of predictions made over time.
* Time taken for model updates and retraining.
* System uptime and reliability.

**Conclusion**

The Credit Card Default Prediction system, built with a robust CI/CD pipeline, provides a scalable and reliable solution for predicting customer defaults. By automating the entire model lifecycle, the system ensures that predictions are always based on the latest data, helping financial institutions mitigate risks and make informed decisions.