HIGH LEVEL DOCUMENT

CREDIT CARD DEFAULTER PREDICTION

DATA VERSION CONTROL

DATE ISSUED	VERSION	DISCRIPTION	AUTHOR
24.June.2024	1	Initial HLD-V1.0	Rahul Yadav

CONTENT

Abstract

The financial industry has witnessed significant advancements, leading to the emergence of various financial threats, one of which is the credit risk faced by commercial banks. Accurate prediction of credit default risk has become crucial for these institutions to mitigate potential losses. This study aims to predict the probability of credit default based on the characteristics and payment history of credit card owners. By leveraging data-driven techniques, the research seeks to develop a predictive model that can accurately assess the creditworthiness of clients, thereby enabling banks to make informed lending decisions and manage their credit risk more effectively.

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:

a. Present all of the design aspects and define them in detail
b. Describe the user interface being implemented Describe the hardware and software interfaces
c. Describe the performance requirements Include design features and the architecture of the project
d. List and describe the non-functional attributes like:

1.Security
2. Reliability
3.Maintainability
4.Portability
5.Reusability
6. Application compatibility
7. Resource utilization
8. 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
DATABASE	Collection of all information monitored by this system
IDE	integrated Development Enviromeny
AWS	Amazon web services

General Description

2.1 Product Perspective

The Credit Card Defaulter Prediction System is designed to assist commercial banks in assessing and managing credit risk by predicting the likelihood of credit card defaults. This system leverages advanced machine learning techniques to analyze credit card owner's characteristics and payment history, providing banks with actionable insights to make informed lending decisions.

2.2 Problem statement

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.

2.3 PROPOSED SOLUTION

The proposed solution for predicting credit card defaulters involves developing a sophisticated machine learning-based system that integrates seamlessly with the existing infrastructure of commercial banks. This system will utilize historical data on credit card owner's characteristics and payment history to predict the probability of credit default. The solution consists of several key components: data collection and preprocessing, model development and training, integration, and user interface.

2.4 Technical Requirements

1. Data Requirements:

- **Data Sources:**
  - Customer profiles (demographics, employment details, etc.).
  - Transaction histories (payment records, purchase patterns, etc.).
  - External credit reports and scores.
  - Historical default data for model training.
- **Data Formats:**
  - Structured data formats (e.g., CSV, SQL databases).
  - Support for integration with banking systems using APIs.
- **Data Volume:**
  - Scalable to handle large volumes of historical and real-time data.

2.5 Data requirement

The UCI Credit Card Default dataset contains information about credit card clients and their payment history, which is used to predict the likelihood of default. Below are the detailed data requirements for utilizing this dataset effectively:

1. Data Sources:

- **Kaggle dataset:**
    - The dataset can be directly downloaded from the Kaggle and name of data set is Default of credit card datasers

2. Data Components:

- The dataset typically includes the following features:
    - **ID**: Unique identifier for each customer.
    - **LIMIT_BAL**: Amount of the given credit (includes individual and family/supplementary credit).
    - **SEX**: Gender (1 = male, 2 = female).
    - **EDUCATION**: Education level (1 = graduate school, 2 = university, 3 = high school, 4 = others).
    - **MARRIAGE**: Marital status (1 = married, 2 = single, 3 = others).
    - **AGE**: Age in years.
    - **PAY_0 to PAY_6**: History of past payments (September 2005 - April 2005). The scale ranges from -1 (pay duly) to 8 (payment delay for 8 months or more).
    - **BILL_AMT1 to BILL_AMT6**: Amount of bill statement (September 2005 - April 2005).
    - **PAY_AMT1 to PAY_AMT6**: Amount of previous payment (September 2005 - April 2005).
    - **default.payment.next.month**: Default payment (1 = yes, 0 = no).

3. Data Formats:

- **Structured Data:**
    - CSV file format typically provided by the UCI repository.
    - Example filename: default of credit card clients.csv.

4. Data Volume:

- **Number of Records:**
    - The dataset contains 30,000 records (observations).
- **Number of Features:**
    - The dataset includes 24 features (columns).

2.6 TOOL USED

Programming Language:

- Python 3.x

Libraries and Tools:

- **Data Manipulation and Analysis:**
  - pandas for data manipulation and analysis.
  - numpy for numerical operations.
- **Machine Learning:**
  - scikit-learn for model building and evaluation.
  - catboost and xgboost for advanced gradient boosting algorithms.
- **Visualization:**
  - matplotlib for basic plotting.
  - seaborn for advanced statistical plots.
- **Web Framework:**
  - flask for building the web interface and API.
- **Experiment Tracking and Version Control:**
  - mlflow==2.2.2 for experiment tracking.
  - dvc for data version control.
- **Database:**
  - pymongo and pymongo[srv]==3.8 for MongoDB integration.
- **IDE:**
  - Visual Studio Code (VSCode) for development.

2.7 Hardware Requirements:

- **Development Environment:**
  - Local machine with sufficient computational resources (CPU, RAM).
  - Optionally, access to cloud-based services (e.g., AWS, Azure) for scalable processing.
- **Servers:**
  - High-performance servers for model training and deployment.
  - Cloud infrastructure (e.g., AWS EC2, Azure VMs) for handling larger workloads.

2.8 Constraints

The Credit card defaulter prediction system must be user friendly, as automated as possible and user should not be required to know any of the workings

2.9 ASSUMPTIONS

Assume the UCI Credit Card Default dataset is representative and clean. Sufficient computational resources and budget are available. Users are trained, and the system integrates seamlessly with existing infrastructure, complying with relevant data privacy and security regulations.

3. Design Details

3.1. System Architecture:

- **Data Ingestion:**
  - Source: UCI Credit Card Default dataset (CSV format).
  - Tools: pandas for reading and initial preprocessing.
- **Data Preprocessing:**
  - Handling missing values, outliers, and noise.
  - Normalization and encoding of categorical variables.
  - Tools: pandas, numpy, scikit-learn.
- **Feature Engineering:**
  - Creating new features (e.g., ratios, aggregates).
  - Feature selection based on importance and correlation.
  - Tools: pandas, scikit-learn.
- **Model Development:**
  - Initial model building using scikit-learn (e.g., logistic regression, decision trees).
  - Advanced model training with catboost and xgboost.
  - Hyperparameter tuning using grid search or random search.
  - Tools: scikit-learn, catboost, xgboost.
- **Experiment Tracking and Version Control:**
  - Tracking experiments, parameters, metrics, and artifacts.
  - Version control for data and models.
  - Tools: mlflow==2.2.2, dvc.
- **Model Evaluation:**
  - Evaluating model performance using accuracy, precision, recall, F1-score, and AUC-ROC.
  - Cross-validation and performance monitoring.
  - Tools: scikit-learn, matplotlib, seaborn.
- **Integration and Deployment:**
  - Developing a web interface using flask.
  - Exposing prediction functionality via RESTful APIs.
  - Integration with MongoDB for data storage.
  - Tools: flask, pymongo, pymongo[srv]==3.8.
- **User Interface:**
  - Interactive dashboard for visualizing predictions and performance metrics.
  - Automated alerts for high-risk customers.
  - Tools: flask, matplotlib, seaborn.

3.2 Data Flow:

1. **Data Ingestion:**
   - Load dataset from CSV using pandas.
2. **Data Preprocessing:**
   - Clean and preprocess data (handle missing values, normalize, encode).
3. **Feature Engineering:**
   - Generate new features and select relevant ones.

4. **Model Training:**
    - Split data into training and testing sets.
    - Train models using scikit-learn, catboost, xgboost.
    - Track experiments with mlflow.
5. **Model Evaluation:**
    - Evaluate and validate model performance.
    - Tune hyperparameters and retrain models.
6. **Integration and Deployment:**
    - Deploy model as a RESTful API using flask.
    - Store and manage data with MongoDB using pymongo.
7. **User Interaction:**
    - Visualize results on a dashboard.
    - Generate and send alerts for high-risk cases.

3.3. Data Preprocessing Details:

- **Missing Values:**
    - Strategy: Imputation or removal.
    - Tools: pandas.
- **Normalization:**
    - Standardization using StandardScaler or MinMaxScaler.
    - Tools: scikit-learn.
- **Encoding:**
    - One-hot encoding for categorical variables.
    - Tools: pandas, scikit-learn.

3.4. Model Development Details:

- **Initial Models:**
    - Logistic Regression, Decision Trees.
    - Tools: scikit-learn.
- **Advanced Models:**
    - CatBoost, XGBoost for gradient boosting.
    - Tools: catboost, xgboost.
- **Hyperparameter Tuning:**
    - Grid Search, Random Search.
    - Tools: scikit-learn.

3.5 Experiment Tracking and Version Control:

- **Experiment Tracking:**
    - Log parameters, metrics, artifacts.
    - Tools: mlflow.

3.6. Integration and Deployment (AWS EC2):

- **Deploy on AWS EC2:**

    - Launch an EC2 instance and configure the environment.
    - Deploy Flask application and MongoDB.
    - Secure the instance and manage access.
- **Database:**
    - Use MongoDB for data storage.
    - Tools: pymongo.

3.7 Visualization and User Interface:

- **Dashboard:**
    - Interactive visualizations for model insights.
    - Tools: matplotlib, seaborn.
- **Alerts:**
    - Automated notifications for high-risk predictions.
    - Tools: Custom scripts in flask.

4. conclusion

- In conclusion, the High-Level Design (HLD) for the UCI Credit Card Defaulter Prediction system outlines a comprehensive architecture that leverages machine learning models to predict credit default risks. The system utilizes various tools and technologies, including pandas, scikit-learn, Flask, MongoDB, and AWS EC2, to preprocess data, train models, and deploy a web interface for user interaction. Experiment tracking, version control, and integration with existing banking systems are also addressed. Overall, the HLD provides a solid foundation for implementing an effective and efficient credit risk prediction system.