High Level Design

(HLD)

CREDIT CARD DEFAULT DETECTION

SATYAM SHARMA

DATA SCIENTIST PWSKILLS

Document Version Control

Date Issued	Descriptions	Author
Aug 21 2023	Reviewed project objectives, established a plan for data exploration	SATYAM
Aug 26 2023	Explored dataset structure , identify key features and initial data statistics	SATYAM
Sept 5 2023	Started data preprocessing, including missing values handling and data cleaning	SATYAM
Sept 12, 2023	Began feature Engineering, creating new variables and transforming existing ones to improve model performance	SATYAM
Sept 20, 2023	Evaluated various machine learning algorithms to choose the most suitable for credit default prediction	SATYAM
Oct 3, 2023	Evaluated model performance using various metrics and fine-tuned model parameters	SATYAM
Oct 18, 2023	Prepared the system for deployment, including setting up AWS resources, and testing deployments process	SATYAM

Oct 25, 2023	Developed the user interface using Flask, focusing on user experience and input validation	SATYAM
Nov 2 ,2023	Successfully completed the project, conducted final testing and documented the entire machine learning projects	SATYAM

1. Introduction

In the realm of financial institutions and risk management, accurate credit card default prediction plays a pivotal role in shaping sound lending decisions. This High-Level Design (HLD) document is a vital milestone in our journey towards the creation of a robust and intuitive Credit Card Default Prediction System. It serves as the foundational blueprint for the system's architectural framework, outlining the high-level design principles and key considerations that will guide us through the development process.

a. Why this High-level design Document?

The creation of this HLD document is indispensable for several reasons:

- Clarity of Purpose: By defining the high-level architecture and design, this document ensures that all stakeholders, from developers to business analysts, share a common understanding of the project's scope and objectives.

- Structural Framework: It establishes the architectural foundation, outlining the system's major components, data flow, interfaces, and overall design structure. This clarity is essential for cohesive development.

- Alignment with Objectives: The HLD document reaffirms our commitment to meeting the project's objectives and aligns our design choices with these goals.

- Transparency and Collaboration: It fosters transparency and encourages collaboration among multidisciplinary teams by providing a structured and shared reference point for discussions.

- Risk Mitigation: By addressing technical, architectural, and design considerations at this stage, we are better equipped to identify and mitigate potential risks early in the project lifecycle.

b. Scope

The scope of this High-Level Design (HLD) document extends to defining the boundaries and constraints within which the Credit Card Default Prediction System will be designed and developed. It encompasses the project's objectives, functional boundaries, and the context in which the system will operate.

- Scope of the Credit Card Default Prediction System:

- Objective: The primary objective of this system is to predict credit card default events accurately, providing valuable insights for making informed lending decisions. The system will analyze historical transaction data and assess the likelihood of a cardholder defaulting on their credit card payments.

- User Base: The system's end-users include financial analysts, underwriters, and other personnel responsible for assessing creditworthiness. The predictions generated by the system will assist these users in their decision-making processes.

- Functional Boundaries: The system will focus on the prediction aspect and not encompass credit card issuance or transaction processing. It will accept transaction data as input and provide predictions as output.

- Data Sources: The system will rely on historical transaction data, including credit limits, balances, payment history, and other relevant financial information. Data integration with external systems for continuous data updates will be considered within the scope.

- Out of Scope:

- Credit Card Issuance: The system will not be involved in credit card issuance, account management, or transaction processing.

- Regulatory Compliance: While the system will contribute to risk assessment, it will not address regulatory compliance or adherence to specific financial regulations. Compliance requirements will be separate and distinct.

- Operational Decision-Making: The system's predictions will serve as decision support, but the final credit decisions and actions will be determined by human operators.

- Data Gathering: The process of acquiring transaction data from external sources or maintaining data quality is beyond the system's scope.

Infrastructure Management: The document primarily focuses on the system's design and architecture, with infrastructure management and hosting considerations being addressed separately.

c. Definitions

- High-Level Design (HLD): High-Level Design is a phase in software development that provides a conceptual framework for the architecture and structure of a system. It focuses on defining major components, their interactions, and key considerations without diving into specific code or implementation details.

- Credit Card Default Prediction System: The Credit Card Default Prediction System is a software solution designed to assess the likelihood of a credit cardholder defaulting on their payments. It uses historical transaction data and predictive models to make informed credit decisions.

- Architectural Overview: An architectural overview outlines the high-level structure and organization of a software system. It defines the system's components, their relationships, and the flow of data or information between them.

- User-Centric Design: User-centric design is an approach that prioritizes the needs and preferences of the end-users when designing a system or interface. It aims to create products that are intuitive, user-friendly, and tailored to user expectations.

- Data Flow: Data flow refers to the movement of data within a system. It encompasses data sources, data processing, storage, and data destinations, providing an understanding of how information is processed and utilized.

- Interfaces: Interfaces in the context of software systems refer to the points of interaction between different components or systems. They define how data and communication flow between these entities.

- Scalability: Scalability is the system's ability to handle increased workloads or growing data volumes without a significant decrease in performance. It ensures that a system can grow as the demand for its services increases.

- Regulatory and Compliance Requirements: Regulatory and compliance requirements refer to the legal and industry-specific rules and regulations that a system must adhere to. This can include data privacy laws, financial regulations, and other relevant compliance standards.

- Technology Stack: A technology stack is a combination of software tools, programming languages, databases, and frameworks used to build a software system. It defines the technical components and infrastructure of the system.

- Data Storage and Database Design: Data storage and database design refer to how data is structured and stored within a system. It includes decisions about database management systems, schema design, and data storage methods.

2. General Descriptions

a. Product Perspective

1. Interaction with Existing Systems: The Credit Card Default Prediction System will interact with various existing systems within HFDC, such as data repositories, analytics tools, and user interfaces. It will seamlessly integrate with these systems to retrieve and process relevant data for credit assessments.

2. External Interfaces: The system may have external interfaces with data providers, financial institutions, or credit bureaus to fetch real-time or updated credit-related information. These interfaces will be carefully managed to ensure data accuracy and security.

3. Dependency on Data Sources: A key aspect of the product perspective is its reliance on accurate and timely data sources. The system's performance and predictions are directly influenced by the quality and availability of data from internal and external sources.

4. Alignment with HFDC's Software Ecosystem: The Credit Card Default Prediction System is designed to align with HFDC's broader software ecosystem, adhering to technology standards, security protocols, and architectural guidelines in place within the organization.

5. Impact on Decision-Making: The system's predictions will impact HFDC's decision-making processes, aiding in the assessment of creditworthiness and influencing credit-related decisions made by financial analysts and underwriters.

6. Future Integration Opportunities: As the system evolves, opportunities for integration with additional HFDC systems and external data providers will be explored, allowing for enhanced data enrichment and improved prediction accuracy.

b. Problem statement

- Risk Assessment Accuracy: The current credit assessment processes at HFDC lack the precision and granularity required to assess credit cardholders' risk accurately. This leads to increased instances of both false positives and false negatives in credit decisions.

- Manual Decision-Making: HFDC relies heavily on manual assessments by financial analysts and underwriters, which can be time-consuming and subject to human bias. Automation is required to improve efficiency and consistency.

- Data Management Challenges: Managing, updating, and ensuring data quality from multiple sources are complex and error-prone tasks. Data integration and maintenance pose significant challenges.

- Regulatory Compliance: HFDC is subject to regulatory requirements in credit assessment. Ensuring compliance with these regulations, such as data protection laws and fair lending practices, is a crucial concern.

- Customer Satisfaction: Inaccurate credit assessments may result in declining credit card applications from potentially creditworthy customers or approving credit for high-risk applicants. This can impact customer satisfaction and retention.

- Economic Impact: Credit defaults can have a severe economic impact on HFDC. Reducing the default rate through accurate predictions is critical for the financial health of the institution.

- Data-Driven Decision-Making: HFDC seeks to transition towards a more data-driven approach to credit assessment. The absence of a reliable, data-driven predictive model is a significant problem that the system aims to address

c. Proposed Solution

- Predictive Analytics: The core of the proposed solution is predictive analytics. The system will employ advanced machine learning and data analysis techniques to assess historical transaction data and generate accurate predictions regarding credit card defaults.

- Automation: The system will automate the credit assessment process, reducing the reliance on manual evaluations. This automation streamlines decision-making and ensures consistency and efficiency.

- Data Integration: The system will establish robust data integration pipelines, enabling the retrieval of up-to-date information from both internal and external sources. This ensures that the credit assessments are based on the most recent data.

- Regulatory Compliance: The system will be designed with built-in mechanisms to ensure regulatory compliance, incorporating data protection and fairness practices. This alignment with legal requirements will be a key feature.

- User-Friendly Interface: To make the system accessible to a wide range of users, it will include a user-friendly interface for financial analysts and underwriters. This interface will present predictions and supporting data in an understandable format.

- Continuous Improvement: The system will be equipped with mechanisms for continuous improvement. It will learn and adapt to changing credit patterns and user feedback, enhancing its prediction accuracy over time.

- Data-Driven Decision-Making: The proposed solution fully embraces the concept of data-driven decision-making, embedding it into HFDC's credit assessment processes. The system will provide data-backed insights to enhance the decision-making framework.

d. Further Improvements

- Enhanced Prediction Models: The system will continually evolve its prediction models by incorporating more sophisticated machine learning algorithms, allowing for even more accurate credit assessments.

- Expanded Data Sources: To further enrich the prediction process, the system will explore opportunities for integrating additional data sources, such as social and economic indicators, to gain a more comprehensive view of creditworthiness.

- User Feedback Mechanisms: The incorporation of user feedback mechanisms will be a priority. Continuous user feedback will inform system improvements, making it more user-centric and aligned with the evolving needs of financial analysts and underwriters.

- Advanced Visualization: The system will enhance data visualization features, offering more intuitive and informative ways to present predictions and supporting data. Visual representations will aid users in interpreting results.

- 5. Real-time Prediction: As technology permits, real-time prediction capabilities will be explored, allowing for instantaneous credit risk assessments and quicker decision-making.

- 6. Scalability: The system will be designed with scalability in mind, ensuring it can handle increased data volumes and user loads as HFDC's operations expand.

- 7. Security Enhancements: The ongoing improvement of security measures will be a key focus, addressing emerging threats and ensuring data protection and system integrity.

- 8. Regulatory Updates: As financial regulations evolve, the system will be updated to remain compliant with changing legal requirements, ensuring HFDC's adherence to industry standards.

- 9. Integration with Other HFDC Systems: Opportunities for integration with other HFDC systems will be explored to further streamline and automate credit assessment processes, creating a more interconnected and efficient financial ecosystem.

e. Technical Requirements

1. Hardware Requirements:

- Server Infrastructure: The system will require server infrastructure capable of handling data processing and analysis. Hardware specifications will be determined based on the expected data volumes and processing demands.
- Storage: Sufficient storage capacity is necessary to accommodate historical transaction data, models, and system backups.
- Redundancy: To ensure system availability, redundancy and failover mechanisms will be in place to minimize downtime.

2. Software Requirements:

- Operating System: The system will be compatible with a specific operating system or a range of supported operating systems.
- Python Programming Languages: The system will be developed using programming languages best suited for machine learning, data analysis, and web development.
- Development Frameworks: Frameworks for machine learning and web development will be utilized to streamline development processes.
- Security Software: Security software, including firewalls, intrusion detection systems, and encryption tools, will be implemented to safeguard data and system integrity.

3. Data Requirements:

Data Sources: The system will require access to historical transaction data, which may be obtained from internal databases, data providers, or credit bureaus.

Data Quality: Data quality measures will be implemented to ensure data accuracy, consistency, and completeness.

4. Security and Compliance Requirements:

Authentication and Authorization: Robust user authentication and authorization mechanisms will be in place to control access to sensitive data.

Data Encryption: Data in transit and at rest will be encrypted to protect against unauthorized access.

Compliance Frameworks: The system will adhere to relevant data protection laws, financial regulations, and industry standards.

5. Performance and Scalability:

Performance Optimization: Performance tuning will be ongoing to minimize response times and maximize system efficiency.

Scalability: The system will be designed for scalability, allowing it to handle increased workloads as user demand grows.

6. Integration:

Data Integration: Integration with internal and external data sources will be established to maintain data currency.

API Compatibility: The system's interfaces will adhere to API standards for integration with other HFDC systems

f. Data Requirements

2. Data Quality:

g. Tools Used

Machine Learning Frameworks: Scikit-learn

Data Analysis Tools: Python (Pandas, NumPy), Jupyter Notebooks.

Web Development Framework: Flask.

Security Tools: Firewall software, intrusion detection systems, SSL certificates.

Visualization Libraries: Matplotlib, Seaborn.

i. Hardware Requirements

Server Infrastructure: A dedicated server or cloud-based instance with sufficient computing power and memory to support data processing and machine learning tasks. While the system does not rely on a database for data storage, the server's capacity should accommodate data processing needs efficiently.

Storage: High-capacity storage devices or network-attached storage (NAS) for storing model files, user data, and system backups. The storage requirements are primarily driven by the need to maintain model versions and user data.

Redundancy: While the absence of a traditional database reduces complexity, redundancy remains a crucial consideration. Redundant servers or instances should be in place to ensure system availability, especially given the importance of continuous service in the financial domain.

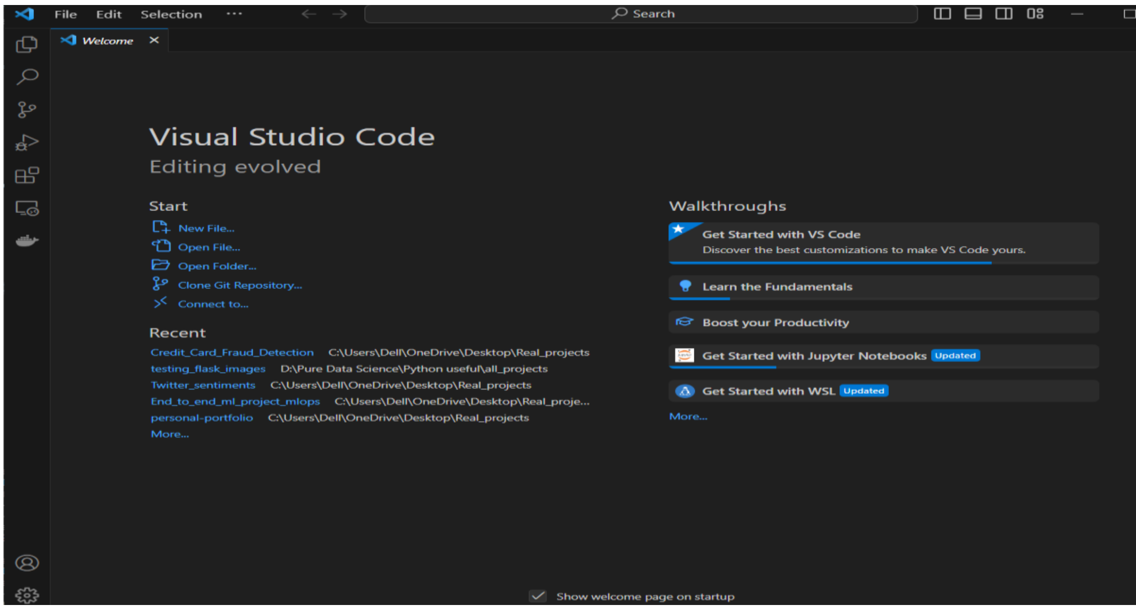
ii. Software Requirements

Operating System: A Linux-based operating system, such as Ubuntu Server, is suitable for hosting the system. It provides a stable and secure environment for system operations.

Web Framework: Flask is the web framework employed for developing the user interface of the Credit Card Default Prediction System. It offers a lightweight and flexible approach for building web applications.

Programming Language: Python is the primary programming language used for both machine learning and web development. It provides a rich ecosystem of libraries and tools for data analysis and application development.

Development and IDE Tools: Visual Studio Code (VS Code) is used as the integrated development environment for managing machine learning tasks and application development. Additionally, Jupyter notebooks (.ipynb) are employed for experimentation and prototyping.



Visual Studio Code
Editing evolved

Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...

Recent
Credit_Card_Fraud_Detection C:\Users\Dell\OneDrive\Desktop\Real_projects
testing_flask_images D:\Pure Data Science\Python useful\all_projects
Twitter_sentiments C:\Users\Dell\OneDrive\Desktop\Real_projects
End_to_end_ml_project_mlops C:\Users\Dell\OneDrive\Desktop\Real_proje...
personal-portfolio C:\Users\Dell\OneDrive\Desktop\Real_projects
More...

Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Boost your Productivity
Get Started with Jupyter Notebooks Updated
Get Started with WSL Updated
More...

✓ Show welcome page on startup

Security Software: Security measures will include firewall software and intrusion detection systems to protect the system from threats and unauthorized access. SSL certificates may be used to ensure secure data transmission.

h. Constraints

- Data Privacy and Compliance: The system must adhere to data privacy regulations and financial compliance standards. It should ensure the secure handling of sensitive user data and comply with relevant laws governing data protection and fair lending practices.

- Hardware Limitations: The choice of hardware may be subject to budget constraints, impacting the selection of servers, storage, and redundancy options. Cost-effectiveness should be considered while ensuring adequate performance.

- Data Availability: The system's predictions are contingent on the availability and quality of historical transaction data. Data sources must be reliable and accessible for the system to operate effectively.

- Resource Scalability: The scalability of resources may be limited by budget considerations or the technology stack chosen. Scalability planning should account for potential resource constraints.

- Real-time Data: If real-time data integration is a requirement, the availability and reliability of real-time data sources can be a constraint, impacting the system's ability to provide instant assessments.

- User Data Entry: As the system relies on user-provided data directly, the quality and accuracy of data entered by users can be a constraint. Data validation and quality control are essential.

- Regulatory Changes: Evolving financial regulations and data privacy laws may introduce new constraints that the system must adapt to in order to maintain compliance.

- Budgetary Constraints: Budget limitations may affect the choice of software, hardware, and other resources. Optimization and cost-effective solutions are essential.

i. Assumptions

- Data Quality: The system assumes that the data provided by users for credit assessment is accurate, complete, and representative of their credit history. Adequate data validation and quality control measures will be in place to address any discrepancies.

- User Data Entry: Users are expected to provide accurate and relevant information during the data input process. User interfaces will be designed to facilitate clear and correct data submission.

- Data Privacy and Compliance: The system operates under the assumption that it will adhere to data privacy regulations and financial compliance standards. Necessary security and compliance measures will be implemented to safeguard user data and ensure regulatory compliance.

- Availability of Historical Data: The system assumes the availability and accessibility of historical transaction data for credit assessment. Data sources will be in place to provide the necessary transaction records for analysis.

- Resource Scalability: The project assumes that resource scalability is feasible within budget constraints. Measures will be taken to design and plan for resource expansion as necessary to accommodate increased data volumes and user loads.

- Security Measures: Assumptions include the presence of security measures to protect the system from unauthorized access and security threats. Ongoing vigilance in monitoring and addressing security concerns is expected.

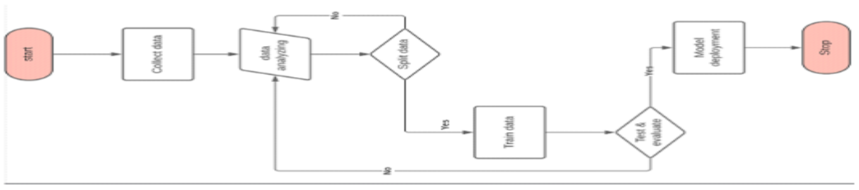
3. Design details

a. Process Flow

i. Model Training and Evaluation

Model Training:

- Data Collection: Historical transaction data is collected from various sources, including user-provided data and external sources.
- Data Preprocessing: The collected data undergoes preprocessing, which includes data cleansing, imputation, and transformation to prepare it for modelling.
- Feature Engineering: Relevant features are selected and engineered to create input variables for the model.
- Model Selection: Machine learning models, such as logistic regression, decision trees, or neural networks, are selected for credit default prediction.
- Training: The chosen model is trained on the preprocessed data, learning patterns and relationships within the data.



Model Evaluation:

Data Split: The dataset is split into training and validation sets to assess the model's performance.

Evaluation Metrics: Various metrics like accuracy, precision, recall, and F1-score are used to evaluate the model's performance in predicting credit defaults.

Hyperparameter Tuning: Model hyperparameters are fine-tuned to optimize predictive accuracy.

Validation: The model is validated on the validation dataset to ensure it generalizes well to unseen data.

Model Deployment: Once a satisfactory model is achieved, it is ready for deployment within the system.

ii. Deployment Process

Model Export: The trained machine learning model is exported in a format compatible with deployment on AWS Elastic Beanstalk.

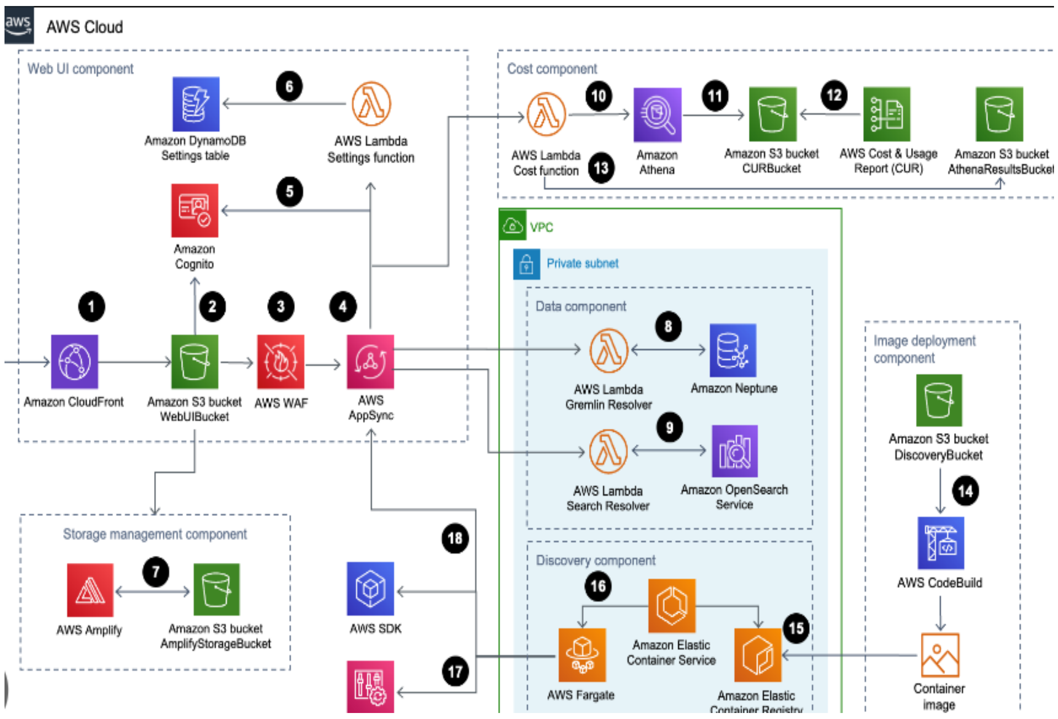
AWS Elastic Beanstalk Setup: An AWS Elastic Beanstalk environment is provisioned to facilitate model hosting and real-time inference. This environment allows for the management of machine learning models and their deployment.

Model Upload: The exported model is deployed to the AWS Elastic Beanstalk environment, making it available for real-time inference requests.

API Development: A RESTful API is developed using AWS API Gateway or integrated directly into the AWS Elastic Beanstalk environment. This API provides a user-friendly and standardized interface for accessing the model. It acts as an intermediary for user data input and model output.

Authentication and Authorization: Security measures, such as authentication and authorization, are implemented to control access to the API and model hosted on AWS Elastic Beanstalk. Only authorized users and systems can interact with the model.

Monitoring and Logging: AWS CloudWatch and other monitoring tools are set up to track the system's performance, detect anomalies, and generate logs for auditing and troubleshooting.



AWS Cloud
Web UI component
Amazon DynamoDB Settings table
AWS Lambda Settings function
Amazon Cognito
Amazon CloudFront
Amazon S3 bucket WebUIBucket
AWS WAF
AWS AppSync
Cost component
AWS Lambda Cost function
Amazon Athena
Amazon S3 bucket CURBucket
AWS Cost & Usage Report (CUR)
Amazon S3 bucket AthenaResultsBucket
VPC
Private subnet
Data component
AWS Lambda Gremlin Resolver
Amazon Neptune
AWS Lambda Search Resolver
Amazon OpenSearch Service
Image deployment component
Amazon S3 bucket DiscoveryBucket
AWS CodeBuild
Container image
Storage management component
AWS Amplify
Amazon S3 bucket AmplifyStorageBucket
AWS SDK
Discovery component
Amazon Elastic Container Service
AWS Fargate
Amazon Elastic Container Registry

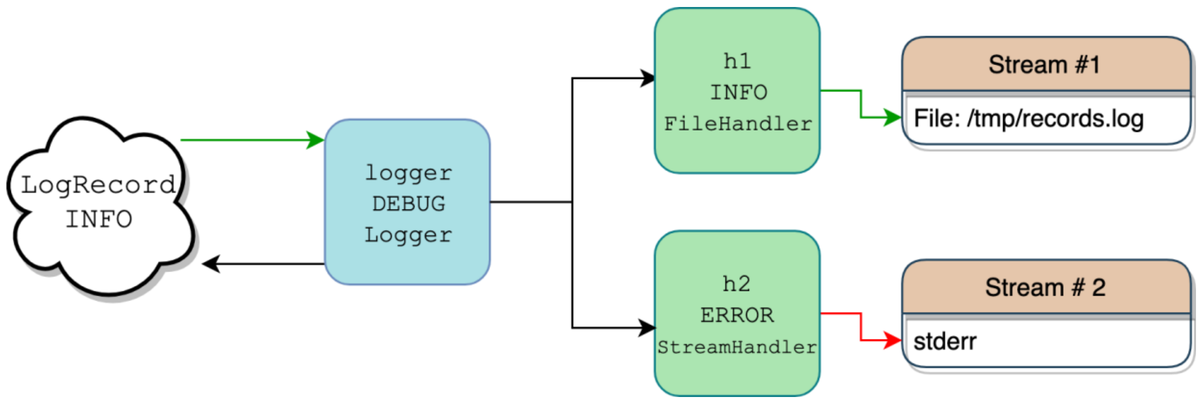
b. Event log

Logging Configuration: The Python logging library is configured to specify where and how log messages are handled. It may include log levels (e.g., INFO, WARNING, ERROR), log file locations, and log format.

Logging Events: Various events and activities within the system trigger log messages. For example, when a user submits data for credit assessment, when a model makes a prediction, or when an error occurs, log messages are generated.

Log Levels: Different log levels are used to categorize the importance of log messages. INFO might be used for routine activities, while ERROR is reserved for critical issues.

Log Destination: Log messages can be directed to different destinations, including log files, console output, or external logging services.



c. Error Handling

Try and Except Blocks: Throughout the system's codebase, try and except blocks are strategically placed to encapsulate potentially problematic code sections. When an exception is raised, it is caught by the except block, preventing the system from crashing.

Exception Types: Specific exception types are used to handle different error scenarios. For example, ValueError may be used to handle issues related to incorrect data format, while Exception may be used for general error handling.

Custom Error Messages: The system provides custom error messages within the except blocks to clearly communicate the nature of the error and assist with debugging.

Logging: The system logs errors and exceptions using the Python logging library, creating an event log that records error details and context

4. Performance

The "Performance" section evaluates various aspects related to the performance of the Credit Card Default Prediction System. In this section, we will discuss the system's reusability, which is essential for extending and adapting its components for future use.

a. Reusability

Reusability Strategies:

Modular Design: The system is designed with a modular structure, allowing individual components to be reused or replaced without affecting the entire system. For instance, machine learning models can be swapped or upgraded without altering the user interface.

API Integration: The system provides an API for external integration, enabling other applications or systems to make credit assessments using the predictive models. This promotes the reusability of the system's core functionality.

Configurability: Many aspects of the system, including model hyperparameters and data sources, are configured to be adjustable without code changes. This configurability supports future adaptation and reuse.

Version Control: The system's codebase is managed using version control systems (e.g., Git). This enables tracking changes, rolling back to previous versions, and collaborating on system improvements

b. Application Compatibility
    - Application compatibility is a crucial consideration in the design and performance evaluation of the Credit Card Default Prediction System. It assesses the system's ability to seamlessly interact with other applications and environments
    - Data Interoperability: The system is designed to ingest and process data from various sources and formats. It is compatible with common data exchange formats such as JSON, YAML, and joblib. This flexibility enables data integration from diverse systems.
    - Security Compatibility: The system integrates security measures such as authentication and authorization, ensuring that external applications adhere to security standards. It is compatible with industry best practices for safeguarding sensitive information.
    - Scalability: The system's architecture is designed to be horizontally scalable, making it compatible with the dynamic resource requirements of applications experiencing fluctuations in user traffic or data input volumes.
    - Platform Independence: The system's core functionality is platform-independent, allowing it to run on various operating systems and cloud platforms. This ensures compatibility with different hosting environments.

c. Resource Utilization
    - Scalability: The system employs horizontal scalability, allowing it to efficiently allocate additional computing resources to accommodate increased user traffic and data processing requirements. This dynamic scaling minimizes resource wastage during periods of low demand.
    - Resource Monitoring: Continuous resource monitoring and performance analysis are conducted to identify underutilized or overutilized resources. This enables proactive adjustments to optimize resource allocation.
    - Load Balancing: Load balancing mechanisms distribute incoming requests evenly across multiple servers, ensuring that each server's resources are effectively utilized.
    - Resource Allocation Policies: Resource allocation policies are established to prioritize critical system components. For example, machine learning model instances are allocated more resources during prediction requests.

- Caching: Caching is employed to store frequently accessed data, reducing the need for repeated data processing and resource consumption.

Containerization: The system utilizes containerization technologies like Docker to isolate components, making efficient use of resources and facilitating consistent deployment across environments.

d. Deployment

- Amazon EC2: Used for hosting web interface and machine learning models, providing scalability to accommodate varying workloads.

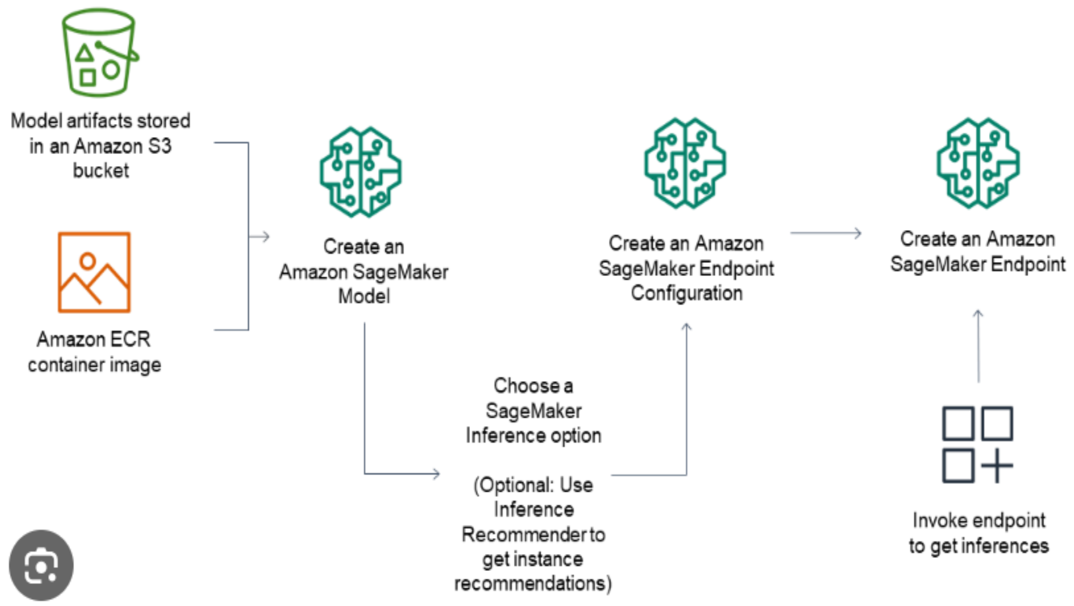
- Amazon Elastic Beanstalk: Deployed machine learning models for real-time predictions via APIs, leveraging a Platform as a Service (PaaS) environment for simplified deployment and management.

- Amazon API Gateway: Managed and exposed the system's RESTful API to external applications and users, offering security, monitoring, and control over API endpoints.

- Amazon RDS: Employed for specific data storage and reporting needs, offering managed database instances with scalability and reliability.

- Auto Scaling: Configured to adjust Elastic Beanstalk instances dynamically based on user traffic, ensuring resource efficiency and system availability.

- CI/CD Pipeline: Implemented to automate deployment to AWS Elastic Beanstalk, streamlining the process of building, testing, and deploying model, web interface, and API updates.



Model artifacts stored in an Amazon S3 bucket
Amazon ECR container image
Create an Amazon SageMaker Model
Choose a SageMaker Inference option
(Optional: Use Inference Recommender to get instance recommendations)
Create an Amazon SageMaker Endpoint Configuration
Create an Amazon SageMaker Endpoint
Invoke endpoint to get inferences

5. Conclusion

In conclusion, the Credit Card Default Prediction System represents a powerful tool for evaluating credit risk and enhancing the decision-making process for credit approvals. This high-level design document has provided an overview of the system's key aspects and considerations

6. References

Scikit-Learn (sklearn):

https://scikit-learn.org/stable/

Reference: Scikit-Learn Documentation

Amazon Web Services (AWS):

https://aws.amazon.com/documentation-overview/?nc2=h_ql_doc_do

Reference: Amazon Web Services Documentation

Python Logging Library:

https://docs.python.org/3/library/logging.html

Reference: Python Logging Library Documentation

Docker:

https://docs.docker.com/

Reference: Docker Documentation

Elastic Load Balancing (ELB-):

https://docs.aws.amazon.com/elasticloadbalancing/