Credit Card Default Prediction

High-Level Design

Ritik Patel

High-Level Design Document for Credit Card Default Prediction with Machine Learning

1. Introduction

This document outlines the high-level design for a credit card default prediction system using machine learning. This system aims to predict whether a client will default on their credit card payment based on historical data.

2. System Architecture

2.1. Data Ingestion

- Source: Various data sources including transactional data, customer profiles, and historical payment information.
- Process: ETL (Extract, Transform, Load) process to clean and preprocess the data.
- Tools: MongoDB for data storage.

2.2. Data Processing

- Preprocessing: Handling missing values, normalization, and feature engineering.
- Tools: Python, Pandas, Scikit-Learn.

2.3. Model Training

- Algorithm Selection: Evaluate and select machine learning algorithms such as Logistic Regression, Decision Trees, Random Forest, and Gradient Boosting.
- Training: Split the data into training and test sets. Train the models on the training set and validate on the test set.
- Tools: Scikit-Learn, TensorFlow, Keras.

2.4. Model Evaluation

- Metrics: Use metrics such as Accuracy, Precision, Recall, F1 Score, and AUC-ROC to evaluate the model's performance.
- Validation: Cross-validation techniques to ensure model robustness.

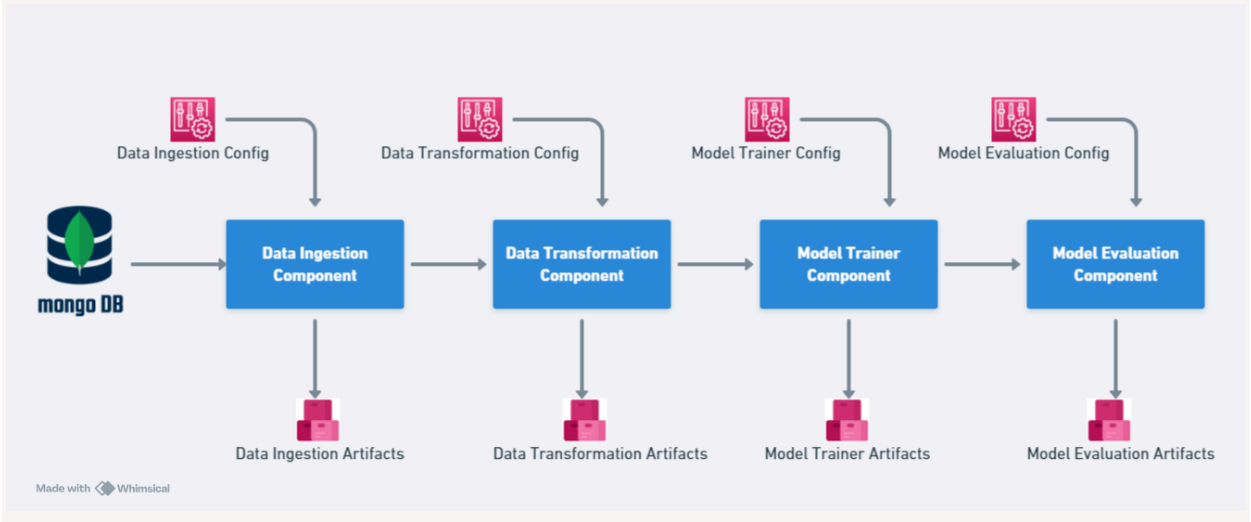
2.5. Model Deployment

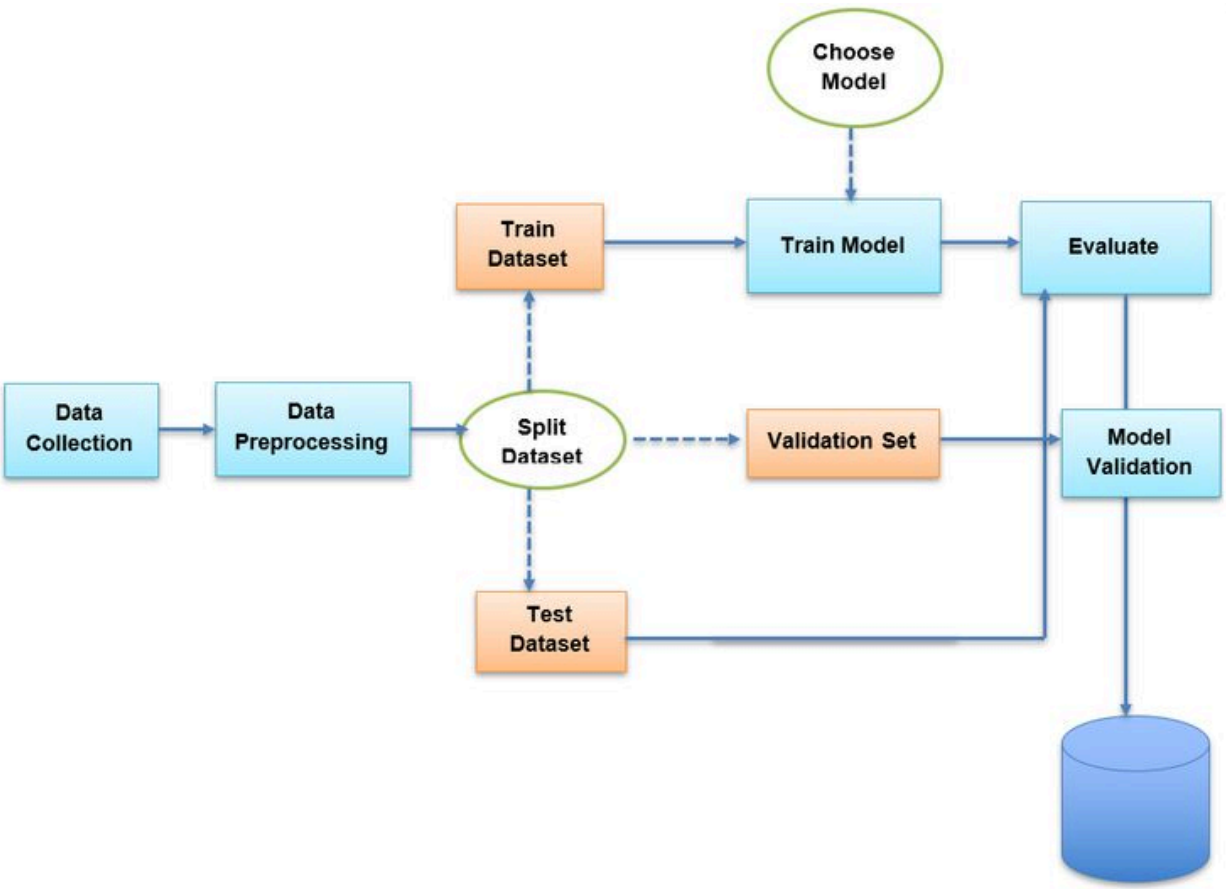
- Containerization: Dockerize the model for deployment.
- Deployment Platform: AWS SageMaker for model hosting and deployment.
- API: Expose the model via RESTful APIs using Flask or FastAPI.

2.6. Monitoring and Maintenance

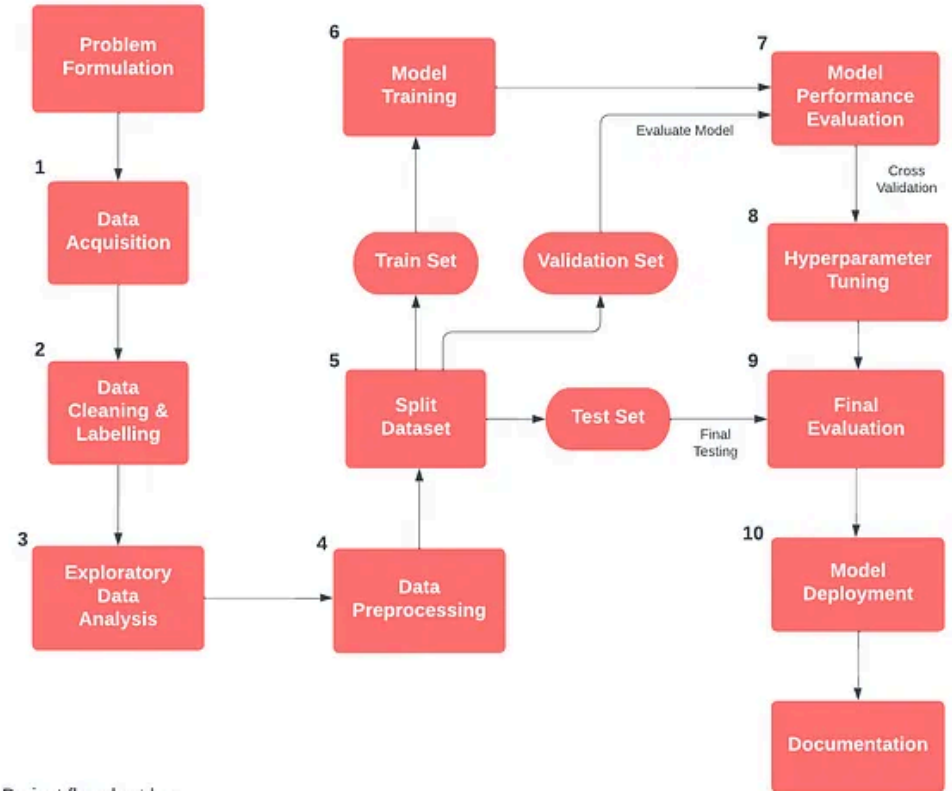
- Monitoring: Implement monitoring for model performance, data drift, and system health.
- Tools: Prometheus, Grafana, AWS CloudWatch.

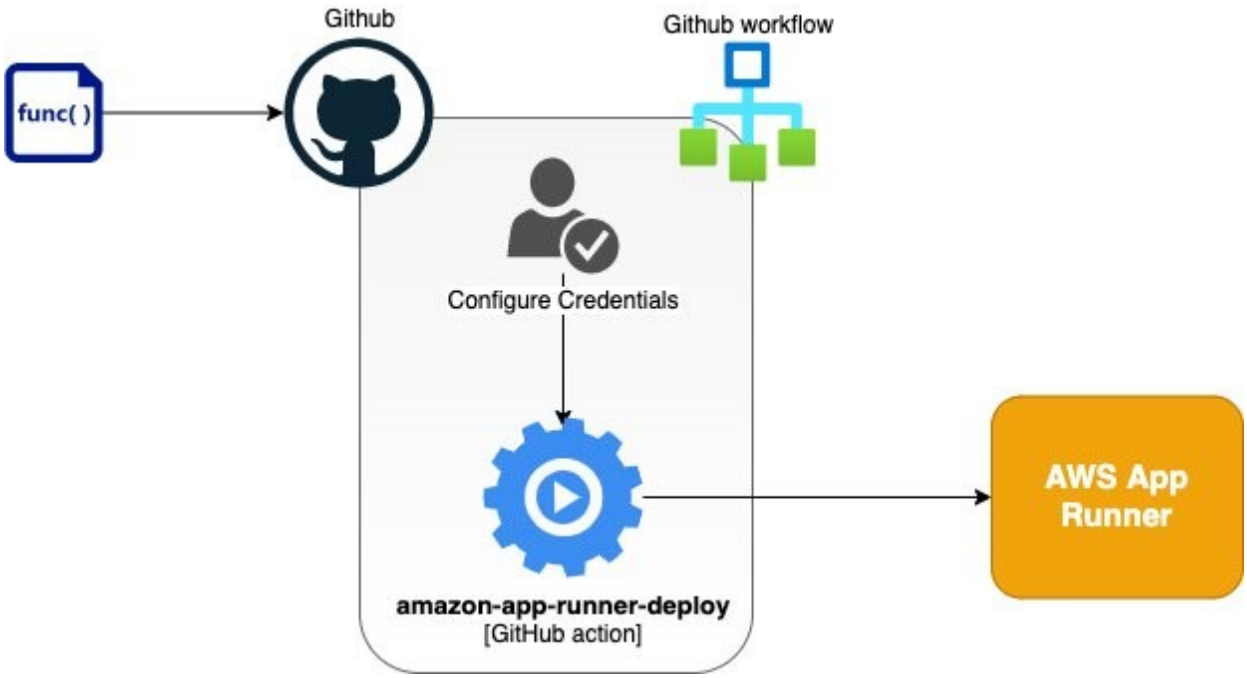
3. Data Flow Diagram




Choose Model
Train Dataset
Train Model
Evaluate
Data Collection
Data Preprocessing
Split Dataset
Validation Set
Model Validation
Test Dataset

Machine Learning Project Life Cycle





4. Detailed Component Design

4.1. Data Ingestion

- Data Sources:
    - Transactional Data: Information on customer transactions.
    - Customer Profiles: Demographic and behavioral data.
    - Historical Payments: Past payment records.
- ETL Process:
    - Extract: Collect data from various sources.
    - Transform: Clean and preprocess data, handle missing values, and encode categorical variables.
    - Load: Store data in a centralized data lake (e.g., Mongodb).

4.2. Data Processing

- Data Cleaning: Remove duplicates, and handle missing values.
- Normalization: Scale numerical features to a standard range.
- Feature Engineering: Create new features from existing data to improve model performance.

4.3. Model Training

- Algorithms:
    - Logistic Regression: For baseline performance.
    - Decision Trees: To handle non-linear relationships.
    - Random Forest: For ensemble learning and reducing overfitting.
    - Gradient Boosting: For improving model accuracy.
- Training and Validation:
    - Split Data: Into training (70%) and test (30%) sets.
    - Cross-Validation: K-fold cross-validation for robust performance metrics.

4.4. Model Evaluation

- Metrics:

    - Accuracy: Proportion of correctly predicted instances.
    - Precision: Proportion of positive identifications that are correct.
    - Recall: Proportion of actual positives that are correctly identified.
    - F1 Score: Harmonic mean of precision and recall.
    - AUC-ROC: Area under the Receiver Operating Characteristic curve.

4.5. Model Deployment

- Containerization: Use Docker to package the model with its dependencies.
- Deployment: Deploy the model on AWS SageMaker.
- API: Develop a RESTful API to serve predictions.

4.6. Monitoring and Maintenance

- Performance Monitoring: Track model accuracy and other metrics in real time.
- Data Drift Detection: Identify changes in data patterns that could affect model performance.
- System Health: Monitor infrastructure and application performance.

5. Security and Compliance

- Data Security: Encrypt data at rest and in transit.
- Access Control: Implement role-based access control (RBAC).
- Compliance: Ensure compliance with relevant regulations (e.g., GDPR, CCPA).

6. Dataset Information

This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.

Content

There are 25 variables:

- ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit)

- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years
- PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2: Repayment status in August, 2005 (scale same as above)
- PAY_3: Repayment status in July, 2005 (scale same as above)
- PAY_4: Repayment status in June, 2005 (scale same as above)
- PAY_5: Repayment status in May, 2005 (scale same as above)
- PAY_6: Repayment status in April, 2005 (scale same as above)
- BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar
- **default.payment.next.month**: Default payment (1=yes, 0=no)

7. Conclusion

This high-level design document outlines the architecture, data flow, and detailed component design for a credit card default prediction system. Following this blueprint will help ensure a robust, scalable, and maintainable solution.