TECHNICAL ARCHITECTURE DOCUMENT

CUSTOMER INTELLIGENCE PLATFORM

Date: December 2025

Project by:
Hemant Borana

Course:
Data Analytics and Reporting

Technical Architecture Document

Customer Intelligence Platform
Date: December 2025
Course: Data Analytics and Reporting

Contents

System Overview

Purpose

The Customer Intelligence Platform is an end-to-end business intelligence solution designed to transform raw multi-source data into actionable customer insights through advanced analytics and interactive reporting.

Key Objectives

- Integrate structured, semi-structured, and unstructured data sources
- Perform comprehensive exploratory data analysis
- Build predictive models for customer behavior
- Deliver interactive dashboards for business users
- Enable self-service analytics capabilities

System Architecture Pattern

Layered Architecture:

- **Data Layer:** SQLite database with normalized schema
- **Integration Layer:** ETL pipelines with error handling
- **Analytics Layer:** ML models and statistical analysis
- **Presentation Layer:** Streamlit-based interactive dashboards

Data Flow

End-to-End Data Pipeline

Phase 1: Data Ingestion

Raw Data Sources

↓

CSV/JSON/XML Files

↓

ETL Extract Module

↓

Staging Area (In-Memory)

Phase 2: Data Transformation

Raw Data

↓

Data Cleaning (nulls, duplicates)

↓

Type Conversion & Validation

↓

Business Logic Application

↓

Quality Score Calculation

↓

Transformed Data

Phase 3: Data Loading

Transformed Data

↓

Database Connection

↓

Bulk Insert Operations

↓

Index Optimization

↓

Metadata Logging

↓

SQLite Database

Phase 4: Analytics Processing

Database Tables

↓

Data Aggregation & Feature Engineering

↓

Statistical Analysis (EDA)

↓

ML Model Training & Validation

↓

Results Storage (JSON/PKL)

Phase 5: Reporting & Visualization

Processed Data + ML Results

↓

Streamlit Application

↓

Interactive Dashboards

↓

User Insights & Actions

Database Schema

Table Specifications

customers (Dimension Table)

- Primary Key: customer_id (INTEGER)
- Foreign Key: segment_id → customer_segments
- Indexes: email, segment_id, signup_date
- Records: ~5,000

transactions (Fact Table)

- Primary Key: transaction_id (INTEGER)
- Foreign Keys: customer_id → customers, product_id → products
- Indexes: customer_id, transaction_date, product_id
- Records: ~25,000

products (Dimension Table)

- Primary Key: product_id (INTEGER)
- Unique Constraint: product_name
- Records: ~12

customer_segments (Lookup Table)

- Primary Key: segment_id (INTEGER)
- Values: Premium, Regular, Occasional
- Records: 3

web_analytics (Fact Table)

- Primary Key: session_id (TEXT)
- Foreign Key: customer_id → customers
- Indexes: customer_id, timestamp
- Records: ~3,000

Metadata Tables:

- data_quality_log: Quality metrics tracking
- etl_metadata: ETL execution history

ETL Pipeline

Pipeline Architecture

1. Extract Phase

# Supported formats and parsers

CSV → pandas.read_csv()

JSON → json.load() + pd.DataFrame()

XML → xml.etree.ElementTree + parsing logic

2. Transform Phase

# Data quality operations

- Remove duplicates: drop_duplicates()

- Handle nulls: fillna() / dropna()

- Outlier treatment: IQR method, Z-score

- Type conversion: pd.to_datetime(), astype()

- Feature engineering: derived columns

- Validation: schema checks, range validation

3. Load Phase

# Database operations

- Bulk insert: df.to_sql()

- Transaction management: COMMIT/ROLLBACK

- Index creation: CREATE INDEX

- Foreign key validation

4. Quality Monitoring

# Metrics tracked

- Completeness: null count / total records

- Accuracy: data type validation

- Consistency: referential integrity

- Timeliness: ETL execution time

- Quality Score: weighted aggregate (0-100)

Error Handling Strategy

Level 1: Record-Level Errors

- Action: Skip record, log error
- Example: Invalid date format in single transaction

Level 2: Table-Level Errors

- Action: Partial load, flag affected records
- Example: 10% of records fail validation

Level 3: Pipeline-Level Errors

- Action: Rollback transaction, alert admin
- Example: Database connection failure

Performance Optimization

Batch Processing:

- Chunk size: 1,000 records per batch
- Memory management: Process large files in chunks

Indexing Strategy:

- Create indexes on foreign keys
- Create indexes on frequently queried columns
- Avoid indexes on high-cardinality columns

Query Optimization:

- Use prepared statements
- Leverage covering indexes
- Optimize JOIN operations

Analytics Engine

Exploratory Data Analysis (EDA)

1. Missing Data Analysis

- Method: Column-wise null count
- Output: Missing percentage per column
- Action: Imputation strategy recommendation

2. Outlier Detection

- Methods: IQR, Z-score (3σ), Percentile (1%-99%)
- Visualization: Box plots, histograms
- Treatment: Winsorization at 5th and 95th percentiles

3. Trend Analysis

- Time series decomposition
- Month-over-month growth calculation
- Seasonality detection (day-of-week patterns)

4. Correlation Analysis

- Pearson correlation matrix
- Strong correlation threshold: $|r| > 0.5$
- Multicollinearity detection

5. Pattern Discovery

- Category distribution analysis
- Channel performance metrics
- Payment method preferences

Machine Learning Pipeline

Regression Models (Customer LTV)

Model 1: Linear Regression

- Features: age, transaction_count, avg_amount, recency, tenure, segment
- Target: total_spent
- Performance: $R^2 = 0.9309$, RMSE = $677.76

Model 2: Ridge Regression (L2)

- Alpha: 1.0
- Performance: R^2 = 0.9309, RMSE = $677.74

Model 3: Lasso Regression (L1)

- Alpha: 1.0
- Performance: R^2 = 0.9310, RMSE = $677.70

Model 4: XGBoost (Best Model)

- Hyperparameters: n_estimators=100, max_depth=5
- Performance: R^2 = 0.9991, RMSE = $75.63
- Feature importance tracking enabled

Classification Models (Churn Prediction)

Model 1: Logistic Regression

- Features: Same as regression
- Target: churned (binary)
- Performance: Accuracy = 74.3%, F1 = 0.0993

Model 2: Decision Tree

- Max depth: 5
- Performance: Accuracy = 76.4%, F1 = 0.2252

Model 3: SVM with Grid Search

- Best params: C=10, gamma='scale'
- Performance: CV F1 = 0.1963

Model 4: XGBoost (Best Classifier)

- Performance: Accuracy = 71.6%, F1 = 0.2663

Clustering Models (Customer Segmentation)

K-Means Clustering

- Optimal K determination: Silhouette score
- K range tested: 2-7
- Optimal K: 2
- Silhouette Score: 0.2836
- Features: age, total_spent, transaction_count, recency

Model Validation

Cross-Validation

- Method: 5-fold stratified CV
- Scoring: R^2 for regression, F1 for classification
- Results logged for comparison

Train-Test Split

- Ratio: 80% train, 20% test
- Stratification: Yes (for classification)
- Random seed: 42 (reproducibility)

Feature Scaling

- Method: StandardScaler
- Applied to: All numeric features
- Fit on train, transform on test

Reporting Layer

Dashboard Architecture

Page 1: Executive Dashboard

- Purpose: High-level KPIs for leadership
- Refresh: On page load (cached)
- Drill-down: Clickable charts to detailed views

Page 2: Analytics Deep Dive

- Purpose: Detailed statistical analysis
- Tabs: Trends, Outliers, Correlations, Patterns
- Interactivity: Filter by date range, segment

Page 3: ML Models & Predictions

- Purpose: Model performance and predictions
- Features: Interactive prediction forms
- Real-time: Yes, predictions on-demand

Page 4: Ad-Hoc Query Builder

- Purpose: Self-service analytics
- Components: Dimension/metric selector
- Export: CSV download enabled

Page 5: Data Quality Monitor

- Purpose: ETL and data health tracking
- Metrics: Quality scores, ETL logs
- Alerts: Visual indicators for issues

Page 6: Reports Library

- Purpose: Pre-built exportable reports
- Types: Executive, Sales, Customer, Product, Marketing
- Format: CSV export

Visualization Standards

Chart Types:

- Line charts: Trends over time
- Bar charts: Categorical comparisons
- Pie charts: Composition analysis
- Scatter plots: Correlation analysis
- Box plots: Distribution analysis

Color Palette:

- Primary: #1f77b4 (blue)
- Secondary: #ff7f0e (orange)
- Success: #2ca02c (green)
- Warning: #d62728 (red)

Interactivity:

- Hover tooltips: Detailed values
- Click events: Drill-down navigation
- Zoom: Pan and zoom on charts
- Export: Download chart as PNG

Technology Stack

Core Technologies

Programming Language

- Python 3.8+
- Reason: Rich data science ecosystem

Database

- SQLite 3
- Reason: Lightweight, serverless, file-based

Web Framework

- Streamlit 1.29.0
- Reason: Rapid prototyping, Python-native

Libraries & Frameworks

Data Processing

- pandas 2.1.4: Data manipulation
- numpy 1.26.2: Numerical operations
- sqlalchemy 2.0.25: Database ORM

Machine Learning

- scikit-learn 1.3.2: ML algorithms
- xgboost 2.0.3: Gradient boosting
- statsmodels 0.14.1: Statistical models
- scipy 1.11.4: Scientific computing

Visualization

- plotly 5.18.0: Interactive charts
- matplotlib 3.8.2: Static plots
- seaborn 0.13.0: Statistical visualizations

Data Generation

- faker 22.0.0: Synthetic data generation

Utilities

- requests 2.31.0: HTTP library
- beautifulsoup4 4.12.2: XML/HTML parsing

Development Tools

Version Control

- Git
- GitHub for repository hosting

Virtual Environment

- venv (Python built-in)
- Isolates project dependencies

Code Quality

- Type hints for key functions
- Docstrings following Google style
- Error handling with try-except blocks

Security & Performance

Security Measures

Data Protection

- No hardcoded credentials
- Environment variables for sensitive config
- SQL injection prevention via parameterized queries

Access Control

- Currently: Single-user desktop application
- Future: Role-based access control (RBAC)

Data Privacy

- Synthetic data used (no real customer data)
- PII handling guidelines documented

Performance Optimization

Caching Strategy

- Streamlit @cache_data decorator
- Cache database queries (5 min TTL)
- Cache ML model results

Database Optimization

- Indexes on foreign keys
- Indexes on frequently queried columns
- Query optimization via EXPLAIN QUERY PLAN

Memory Management

- Chunked processing for large files
- Garbage collection after heavy operations
- Efficient data structures (numpy arrays)

Load Time Targets

- Dashboard initial load: < 3 seconds
- Query execution: < 1 second
- ML prediction: < 500ms

Scalability Considerations

Current Limitations

- Single-user desktop application
- SQLite max size: ~281 TB (theoretical)
- Practical limit: ~100K customers, 1M transactions

Future Scaling Options

- Database: Migrate to PostgreSQL/MySQL
- Deployment: Cloud hosting (AWS, Azure, GCP)
- Caching: Redis for distributed caching
- Load balancing: Multiple Streamlit instances

Deployment Architecture

Current Deployment (Local)

User's Machine

↓

Python Virtual Environment

↓

Streamlit Server (localhost:8501)

↓

SQLite Database (local file)

Recommended Production Deployment

User Browser

↓

Load Balancer

↓

Streamlit Cloud / Docker Containers

↓

PostgreSQL Database (RDS/Cloud SQL)

↓

Object Storage (S3/Cloud Storage) for ML models

THANK YOU

for your time and review.

Email:
hemantpb123@gmail.com

Phone:
9284494154