BACKEND ARCHITECTURE OVERVIEW

The system uses a **multi-layered analysis pipeline** combining deterministic statistical analysis with optional AI-powered insights. The main entry point is `AnalysisService.analyze_dataset()` which orchestrates 10 sequential steps.

HOW DATASET ANALYSIS WORKS

Step 1-2: Target Detection & Problem Type Classification

- Auto-detects target column if not specified (looks for columns named 'target', 'label', 'y', 'class', etc.)
- Determines problem type (classification vs regression) based on target characteristics:
    - **Classification**: ≤20 unique values OR object/category dtype
    - **Regression**: >20 unique values AND numeric dtype

Step 3: Data Profiling (`DataProfiler`)

Generates comprehensive dataset profile including:

- **Basic info**: shape, memory usage, column names
- **Missing values**: percentage per column, overall completeness
- **Duplicates**: count and percentage of duplicate rows
- **Column types**: categorizes as numerical, categorical, datetime, boolean
- **Statistical summary**: mean, std, quartiles, skewness, kurtosis for numerical columns
- **Correlations**: correlation matrix, identifies high correlation pairs (>0.8)
- **Outliers**: IQR-based detection, counts per column
- **Feature analysis**: cardinality for categorical, variance for numerical
- **Target analysis**: class distribution/balance (classification) or statistical distribution (regression)
- **Feature importance**: mutual information scores between features and target
- **Data leakage detection**: identifies suspicious columns with perfect correlations or leakage patterns
- **Data consistency**: detects mixed types, format inconsistencies, semantic nulls

---

HOW QUALITY SCORING WORKS

Quality Score Calculation (`QualityScorer`)

The system calculates **5 independent quality dimensions** using deterministic heuristics:

1. Data Quantity (20% weight)

- Scores based on **samples-per-feature ratio**:
    - ≥100 samples/feature: 90-100 points (excellent)
    - 50-100: 70-89 points (good)
    - 20-50: 50-69 points (acceptable)
    - <20: <50 points (insufficient)
- **Penalties applied for**:
    - Small dataset size (<100 rows: -20 points, <500: -10 points)
    - Class imbalance (>20:1 ratio: -15 points, >10:1: -10 points)
    - High dimensionality risk (<10 samples/feature: -15 points)

2. Data Completeness (20% weight)

- Starts at 100 points
- **Penalties**:
    - Missing values: 2 points per 1% missing (max -40)
    - Columns with >40% missing: -5 points each
    - Duplicate rows: 1.5 points per 1% duplicated (max -20)

3. Data Consistency & Validity (20% weight)

- Uses `ConsistencyChecker` to detect:
    - Mixed data types in same column
    - Format inconsistencies (e.g., "30" vs 30)

    - Semantic nulls ("N/A", "null", "missing")
    - String noise (extra spaces, special characters)
- **Penalties**:
    - Consistency issues: based on checker score
    - Excessive outliers (>15% per column): -5 points per column (max -20)
    - High correlation pairs: -2 points per pair (max -15)

4. Signal-to-Noise Ratio (25% weight - highest!)

- **Most important dimension** - uses actual ML model performance!
- Uses `BaselineModelTrainer` to train quick baseline model:
    - **Classification**: LogisticRegression with 5-fold CV
    - **Regression**: Ridge with 5-fold CV
- Converts CV score to points:
    - Accuracy/R^2 >0.8: 90+ points (strong signal)
    - 0.6-0.8: 70-89 points (moderate signal)
    - <0.6: <70 points (weak signal)
- **Additional penalties**:
    - Low variance features: -3 points each (max -15)
    - Near-zero mutual information: -20 points (very weak)

5. Target Suitability (15% weight)

- Only applicable if target column exists
- Starts at 90 points
- **Penalties**:
    - Missing target values: -3 points per missing (max -30)
    - Class imbalance: -8 to -25 points based on severity
    - Data leakage detected: -30 (high), -15 (medium), -5 (low)
    - Too many classes for classification: -25 points
    - Zero variance target: -30 points

Final ML Readiness Score

Weighted average: `(Quantity×0.20) + (Completeness×0.20) + (Consistency×0.20) + (Signal×0.25) + (Target×0.15)`

Converts to letter grade: A (≥90), B (≥80), C (≥70), D (≥60), F (<60)

HOW ISSUES ARE DETECTED

Statistical Issue Detection (`AnalysisService._generate_detected_issues()`)

Issues are detected using threshold-based rules:

1. **Missing Values** (severity: high if >20%, medium if >5%)
2. **Duplicates** (severity: medium if >10%, low if >5%)
3. **Class Imbalance** (severity: high if >10:1, medium if >5:1)
4. **High Correlations** (medium severity if pairs >0.8)
5. **High Cardinality** (low severity if >50 unique values in categorical)
6. **Low Variance** (low severity if variance <0.01)

Each issue includes:

- Severity level (critical/high/medium/low)
- Category (e.g., "missing_values", "class_imbalance")
- Description with specific percentages
- Affected columns list
- Impact on ML models
- Recommended fix

WHERE & HOW LLM IS INVOLVED

LLM Integration Architecture

The system has **TWO LLM clients**:

1. **Basic LLMClient** (`llm_client.py`) - Original simple insights

2. **EnhancedLLMClient** (`enhanced_llm_client.py`) - Advanced quality assessment

When LLM is Used

LLM analysis is **optional** and only runs if:

- `use_llm_insights=True` (user enabled it)
- OpenAI API key is configured
- `settings.enable_llm=True`

What LLM Does: Enhanced Quality Assessment

The system uses **GPT-4o** (model: `gpt-4o-2024-08-06`) with:

- Temperature: 0.2 (consistent, factual analysis)
- Response format: JSON object
- Max tokens: 2× normal (for detailed analysis)

Smart Sampling Process (`SmartRowSampler`)

Instead of analyzing entire dataset, LLM receives **50 carefully selected problematic rows**:

- Samples rows with CRITICAL issues (mixed types, semantic nulls)
- Samples rows with HIGH issues (format inconsistencies)
- Samples rows with MEDIUM issues (outliers, minor problems)
- Includes baseline clean rows for comparison

This reduces token usage by 95%+ while maintaining accuracy!

The Exact LLM Prompt (from `enhanced_llm_client.py`):

System Prompt:

```
You are an expert ML data quality analyst with deep expertise in identifying data issues that impact machine learning pipelines.

Your role is to provide accurate, calibrated, and evidence-based quality assessments.

Key responsibilities:
1. Identify issues that prevent or degrade ML model training
2. Detect semantic data problems (mixed types, non-standard nulls, inconsistent formats)
3. Provide fair and accurate quality scores based on actual impact to ML readiness
4. Recommend specific, actionable fixes with Python code
5. Estimate realistic cleanup effort in hours

Guidelines for severity assessment:
- CRITICAL: Issues that prevent model training (mixed data types, semantic nulls in >20% of data)
- HIGH: Issues that significantly reduce model accuracy (format inconsistencies affecting >10%, high missing data >30%)
- MEDIUM: Issues that moderately impact performance (outliers, minor inconsistencies affecting 5-10%)
- LOW: Issues that have minimal impact (few statistical outliers <5%, cosmetic issues)

Quality scoring guidelines:
- Score based on actual ML impact, not theoretical perfection
- Datasets with CRITICAL issues blocking training: <50%
- Datasets with HIGH severity issues significantly impacting accuracy: 50-70%
- Datasets with MEDIUM issues requiring moderate cleanup: 70-85%
- Datasets with only minor issues: 85-95%
- Near-perfect datasets (rare): 95-100%
- Adjust scores UP or DOWN based on what you find - be accurate
- Provide confidence level: HIGH (very certain), MEDIUM (somewhat certain), LOW (uncertain)

Always respond with valid JSON in the exact structure specified.
```

User Prompt Structure:

```
Analyze this dataset's quality issues using the provided problematic sample rows.

## Dataset Information:
- Filename, total rows, columns list

## Current Deterministic Quality Score:
- Overall score/100, grade

## Analyzed 50 Representative Samples:

### CRITICAL ISSUES (X samples):
Example 1: Mixed types in Age column
Row data: {'Age': 'thirty', 'Income': 50000}
Additional context: {dtype inconsistency detected}

### HIGH SEVERITY ISSUES (Y samples):
Example 1: Format inconsistency
Row data: {'Date': '2023-01-01', 'Amount': '1,234.56'}

### MEDIUM SEVERITY ISSUES (Z samples):
[similar format]

### BASELINE SAMPLES (clean rows):
[clean examples for comparison]

## Original Issues Detected by Heuristics:
- HIGH: Missing Values - 15% of data points missing
- MEDIUM: Duplicate Rows - 8.2% duplicates

## Your Task:
Review the sample data and determine if the deterministic score is accurate, too high, or too low.

Provide your assessment in JSON format:
{
  "confidence_level": "HIGH|MEDIUM|LOW",
  "adjusted_quality_score": number 0-100,
  "adjusted_grade": "A-F",
  "score_adjustment_reasoning": "2-3 sentences",
  "blocking_issues": [
    {
      "issue_type": "mixed_data_types",
      "severity": "CRITICAL",
      "description": "Age column contains both text ('thirty') and numbers (30)",
      "affected_columns": ["Age"],
      "sample_evidence": ["row 5: has 'thirty' mixed with 30"],
      "estimated_rows_affected": "15%",
      "impact_on_ml": "Will cause training failure",
      "fix_priority": 1
    }
  ],
  "recommended_fixes": [
    {
      "issue_type": "mixed_data_types",
      "fix_description": "Convert text numbers to numeric",
      "python_code": "import pandas as pd\nage_map = {'thirty': 30, 'twenty': 20}\ndf['Age'] = df['Age'].map(age_map).fillna(pd.to_n
      "estimated_effort_hours": "2-3 hours",
      "data_loss_risk": "Low"
    }
  ],
  "total_cleanup_effort_hours": "4-6 hours",
  "ml_readiness_assessment": "Needs Major Work",
  "biggest_concerns": ["Mixed data types will crash training"]
}
```

LLM Response Validation (`LLMFindingValidator`)

The system **validates every LLM finding** against actual data:

- Checks if reported issues actually exist in the data
- Verifies affected row counts and percentages
- Confirms column names are correct
- Calculates **validation reliability**: RELIABLE, MODERATELY_RELIABLE, QUESTIONABLE, UNRELIABLE
- Adjusts confidence based on evidence quality

Blended Scoring (`BlendedScorer`)

The system doesn't blindly trust LLM scores. Instead it uses **intelligent blending**:

```
final_score = (deterministic_score × weight_det) + (llm_score × weight_llm)
```

Weights depend on:

- **LLM confidence** (HIGH/MEDIUM/LOW from LLM itself)
- **Validation reliability** (how many findings were verified)
- **Consistency** (how close LLM score is to deterministic)

Blending strategies:

1. **High confidence + reliable**: 40% deterministic, 60% LLM
2. **Medium confidence**: 60% deterministic, 40% LLM
3. **Low confidence**: 80% deterministic, 20% LLM (mostly ignore LLM)

This prevents the LLM from being too pessimistic OR too optimistic!

HOW MODELS ARE RECOMMENDED

Model Recommendation (`ModelRecommender`)

Uses **rule-based fingerprinting** - NO LLM involved!

Dataset Fingerprinting

Creates a comprehensive profile:

- Size category (tiny/small/medium/large)
- Samples per feature ratio
- Missing data level (none/low/medium/high)
- Has categorical features
- Has outliers
- Class imbalance severity
- Signal strength (from feature importance)
- Preprocessing complexity

Model Scoring Rules

For each candidate model, calculates score (0-100) based on:

Rule 1 - Dataset Size Matching:

- XGBoost/LightGBM: +20 for medium/large, -15 for tiny
- Random Forest: +15 for small/medium/large
- Linear/Logistic: +15 for tiny/small, +20 if high-dimensional
- SVM: +15 for tiny/small, -10 for large

Rule 2 - Missing Data:

- Model handles missing: +15
- Model doesn't handle missing: -15

Rule 3 - Categorical Data:

- Model handles categorical: +10 (XGBoost/LightGBM: +20 if mostly categorical)
- Model doesn't handle: -10 to -20 penalty

Rule 4 - Class Imbalance:

- Model handles imbalance: +15 (severe) or +10 (moderate)
- Model doesn't handle: -15 (severe) or -8 (moderate)

Rule 5 - Robustness:

- Tree-based models (XGBoost/RF): +15 if outliers present
- Linear models: -10 if outliers present

Rule 6 - Interpretability:

- Interpretable models: +20 if dataset needs it (small size or weak signal)

Rule 7 - High Dimensions:

- ElasticNet/SVM: +25
- Linear/Logistic: +15
- Others: -5

Rule 8 - Signal Strength:

- Weak signal: interpretable models +10
- Strong signal: complex models +15

Rule 9 - Speed Requirements:

- LightGBM: +20 for large datasets
- SVM: -15 for large datasets

Final ranking: Top 3 models by score

KEY TECHNICAL DETAILS FOR YOUR MANAGER

Technology Stack:

- **Language**: Python 3.9+
- **Framework**: FastAPI (async)
- **Data Processing**: pandas, numpy, scipy
- **ML Libraries**: scikit-learn (baseline models, feature selection)
- **AI**: OpenAI GPT-4o
- **Statistical Analysis**: Custom algorithms + sklearn

Performance Optimizations:

1. **Smart sampling** reduces LLM cost by 95%
2. **Baseline model training** uses fast models (Ridge, LogisticRegression) with 5-fold CV
3. **Async operations** for LLM calls (doesn't block)
4. **Caching** of profile data between steps

Reliability Measures:

1. **LLM validation** - every finding is verified against actual data
2. **Blended scoring** - combines deterministic + AI scores intelligently
3. **Confidence tracking** - system knows when to trust LLM
4. **Fallback logic** - if LLM fails, uses deterministic analysis only

API Token Usage:

- Basic analysis: **0 tokens** (all deterministic)
- Enhanced LLM analysis: **1,500-3,000 tokens** per dataset
    - Input: ~1,200 tokens (samples + context)
    - Output: ~800 tokens (analysis + recommendations)
- Cost: ~$0.01-0.02 per analysis (GPT-4o pricing)

WORKFLOW SUMMARY

```
1. Upload dataset →
2. Auto-detect target & problem type →
3. Generate comprehensive profile (statistics, correlations, outliers) →
4. Calculate 5-dimension quality score (deterministic) →
5. Detect issues using thresholds →
6. [OPTIONAL] Sample 50 problematic rows →
7. [OPTIONAL] Send to GPT-4o for analysis →
8. [OPTIONAL] Validate LLM findings →
9. [OPTIONAL] Blend deterministic + LLM scores →
10. Recommend top 3 models using rule-based matching →
11. Generate preprocessing recommendations →
12. Return comprehensive analysis report
```

This system combines the **reliability of statistical analysis** with the **semantic understanding of AI** to provide accurate, actionable insights for ML practitioners!