odens

Odens - Hiring Test: Development of AI-Powered Pricing Engine

Deadline: This test/system need to be finished before Tuesday, June 3, 2025, 16:00 CET

Welcome. This test assesses your ability to design and develop a sophisticated AI-powered pricing engine prototype. It's an opportunity to showcase your technical skills, problem-solving approach, and how you might contribute to our team. Security is a key design consideration we value.

Test Goal:

Develop a functional prototype of an intelligent pricing engine. The system should learn from historical user data to quote custom metal profiles.

Core Requirements & Evaluation Focus:

1. **Accuracy:** Aim for high predictive accuracy (e.g., ~98% Mean Absolute Percentage Error - MAPE, or a well-justified alternative). Your analysis and metric justification are as important as the score itself.
2. **Testability:** Your solution must be fully testable, allowing review of its architecture, data handling, model, and functionality.
3. **Submission:** A GitHub repository with all necessary components.
4. **Technology Choices:** You have autonomy in selecting the tech stack and AI/ML solutions. Justify your choices clearly, as this is a key evaluation point.

Task for this Test: Develop the AI Quoting System Prototype

Implement the following core functionalities, demonstrating your approach to the outlined considerations:

1. Data Handling

- **Input & Schema:**
    - Simulate user input of historical quotes and product specifications (e.g., versioned CSV/JSON). A dataset of ~500-1000 simulated quotes is advisable.
      Here is 50 quote files for reference, please create the rest yourself.
      : Test files
    -
    - Design a flexible, evolving schema for variables: Material Properties (alloy, finish), Geometric Specifications (dimensions, cross-sectional profile representation – devise and document your abstraction), Manufacturing Details (tolerances including a conceptual encoding for key GD&T aspects, order quantity), Cost & Contextual Factors (LME price, weight, customer attributes, lead time), and the Target Variable (Historical Quoted Price).
    - Implement robust schema validation and discuss schema evolution handling.
- **Preprocessing:**
    - Implement comprehensive steps: imputation for missing values (justify beyond simple methods), robust outlier detection/treatment, and advanced encoding for categorical variables (especially high-cardinality).
- **Feature Engineering:**
    - **Geometric Data & Profile Complexity:** Devise and implement numerical features for "Profile Complexity" from your cross-section representation (e.g., derived geometric features, DFM rule proxies). Justify thoroughly.
    - **Manufacturing Tolerances:** Develop a numerical representation for dimensional tolerances and selected GD&T features.
    - **LME Price:** Incorporate time-series features (e.g., moving averages, lags).
- **Data Storage:**
    - Implement user-specific, isolated, and secure data storage.
    - Demonstrate conceptual design for scalability and multi-tenancy (e.g., for at least two distinct "Users," managing potential schema variations). Ensure efficient data retrieval.

2. Model Development, Training & Evaluation

- **AI/ML Pipeline:**
    - Develop logic to train models on user-specific data, learning complex price relationships.

    - Explore and justify advanced model architectures (e.g., NNs for tabular data, Transformers, Ensembles). Detail feature fusion if used.
- **Training & Evaluation:**
    - Implement a robust training pipeline with appropriate cross-validation (use time-series CV if temporal dependencies exist, e.g., LME price).
    - Provide comprehensive performance metrics (MAPE, RMSE, R-squared).
    - Conduct detailed error analysis (e.g., performance on specific profile types, bias discussion, impact of under/over-prediction).
    - Evaluate model robustness and generalization.
- **Hyperparameter Optimization:** Demonstrate an automated approach (e.g., Bayesian optimization).

3. Model Management

- **Persistence & Security:** Implement reliable and secure methods to save/retrieve trained user-specific models (consider format, storage, access control, encryption).
- **Versioning & Retrieval:** Implement model versioning and ensure efficient, isolated model loading for predictions.

4. Quote Generation & Uncertainty Quantification

- **Interface:** Allow input of new product specifications for a user (e.g., via a simple API).
- **Prediction & Uncertainty:**
    - Generate price predictions using the user's model.
    - Implement a mechanism for uncertainty quantification (e.g., MC dropout, quantile regression, Bayesian methods, ensemble variance).
    - Derive a Final Price from the resulting distribution (justify choice) and provide the uncertainty measure (e.g., CI/PI). Discuss its interpretation.

5. Continuous Learning (Conceptual Design & Partial Implementation)

- **Feedback Loop:** Design (and ideally, demonstrate) how new validated quotes (with data validation checks) update a user's training data.
- **Retraining Strategy:** Outline (and ideally, implement) a model retraining/fine-tuning strategy.
- **Concept Drift:** Briefly discuss potential monitoring/handling.
- **Isolation:** Ensure the continuous learning pipeline is isolated per user and maintains security.

Key Definitions (Contextual)

- **User:** A hypothetical company with unique data/pricing.
- **Product Specification:** Detailed attributes of the metal profile.
- **Mathematical Model:** The AI/ML algorithm trained on user data.
- **Final Price (FP):** Predicted price, ideally from an uncertainty distribution.

Deliverables & Evaluation Criteria

- **GitHub Repository:** All source code, configuration, sample data.
    - **README.md:** Comprehensive documentation covering:
        - System architecture, data flow (including security considerations).
        - Justifications for technology/model choices.
        - Setup, build, and execution instructions for all functionalities.
        - Explanation of approaches to complex challenges (geometric/tolerance encoding, complexity, multi-tenancy, uncertainty, security).
        - Accuracy calculation methodology and results.
- **Testability:** We must be able to run and verify all functionalities.
- **Accuracy:** Verifiable achievement of high accuracy on a well-defined test set (include evaluation scripts/data).
- **Functionality & Robustness:** Implementation of all core functionalities; handling of varied inputs/errors.
- **Technical Depth & Sophistication:**
    - **Data Handling:** Schema design, preprocessing, encoding of complex data (geometry, GD&T), multi-tenant storage, security.
    - **AI/ML Modeling:** Model choice, training rigor, evaluation depth, feature engineering.
    - **Uncertainty & Continuous Learning:** Soundness of uncertainty methods; coherence of continuous learning strategy.
- **Architectural Design:** Demonstrated consideration for scalability, maintainability, modularity, security principles, and suitability for a conceptual multi-user environment.
- **Code Quality & Documentation:** Clarity, organization, and overall quality.

Important Design Considerations for this Assessment

While not expecting a deployable system, this assessment focuses on your architectural thinking and approach to key design principles vital for robust AI systems. Your prototype should demonstrate your understanding of building for reliability, scalability, and maintainability. This includes evidencing consideration for security throughout your design (data/model protection, secure practices), thoughtful error handling, and how you might approach logging, configuration, and multi-tenant

operations in a fuller implementation. The goal is to effectively showcase your skills and design philosophy.

We look forward to reviewing your solution. Good luck!

odens