AI/ML Python Engineer Coding Assessment

Introduction

This assessment evaluates your skills in machine learning, software engineering, prompt engineering, and documentation. We're looking for practical problem-solving abilities and clean, production-ready code rather than perfect model performance.

Time Estimate: 4-6 hours
Experience Level: 2-3 years Python/ML experience

Dataset

You'll work with `cricket_dataset.csv` containing T20 cricket match data:

Column	Description
total_runs	Total runs scored by chasing team so far
wickets	Number of wickets fallen
target	Target runs to win
balls_left	Remaining balls in the match
won	1 if chasing team won, 0 if lost (target variable)

Task 1: Model Development & Analysis (Core ML Skills)

1.1 Exploratory Data Analysis

- Perform basic EDA and document key insights
- Identify any data quality issues and how you'd address them
- Create 2-3 meaningful visualizations

1.2 Model Training & Comparison

Train **at least 2 different algorithms** (e.g., Logistic Regression, Random Forest, XGBoost) and compare their performance. Document:

- Why you chose these specific algorithms
- Hyperparameter tuning approach (if any)
- Cross-validation strategy
- Model evaluation metrics and interpretation

1.3 Feature Engineering (Optional Bonus)

Create 1-2 new features that might improve model performance (e.g., required run rate, win probability based on historical data). Document your reasoning.

Task 2: Production API (Software Engineering)

Build a FastAPI application with the following requirements:

2.1 Core Endpoint

```python
POST /predict
Content-Type: multipart/form-data
Body: CSV file upload
```

Response Format:

```json
{
  "status": "success",
  "predictions_file": "/path/to/results.csv",
  "metadata": {
    "total_rows": 150,
    "filtered_rows": 89,
    "predictions_made": 89,
    "model_used": "random_forest_v1"
  }
}
```

2.2 Filtering Logic

Process only rows where:

- `balls_left < 60`
- `target > 120`

2.3 Additional Requirements

- Input validation with meaningful error messages
- Logging for debugging and monitoring
- Basic error handling (malformed CSV, missing columns, etc.)

- Model versioning consideration (how would you handle model updates?)

Task 3: Prompt Engineering Integration

3.1 Model Explanation Endpoint

Create an additional endpoint that uses an LLM (OpenAI, Anthropic, or local model) to generate human-readable explanations of predictions.

```
python

POST /explain/{prediction_id}
```

Example Response:

```
json

{
    "prediction": 1,
    "confidence": 0.78,
    "explanation": "Based on the current match situation, the chasing team has a good chance of winning because they r
}
```

3.2 Prompt Design

- Design and document your prompt template
- Show how you handle different scenarios (high confidence vs low confidence predictions)
- Include basic prompt optimization considerations

Task 4: Documentation & Testing

4.1 Documentation Requirements

- **README.md**: Setup instructions, API usage examples, model performance summary
- **Code documentation**: Docstrings for key functions
- **Model report**: 1-2 page summary of your approach, findings, and limitations

4.2 Testing (Choose 1)

Either:

- Write unit tests for at least 2 core functions, OR
- Create a simple integration test that validates the full prediction pipeline

Deliverables

1. Complete Python codebase (organized in logical modules/files)
2. Model artifacts (trained models, evaluation results)
3. Documentation (README, model report)
4. Test data results (processed `cricket_dataset_test.csv`)

Notes

- Use any Python ML libraries you're comfortable with
- For the LLM integration, you can use any provider or mock the responses if API access is limited
- Focus on demonstrating your thought process and engineering practices
- There's no single "correct" solution - we value your approach and reasoning

Submission

Please provide:

- Git repository with your solution
- Brief note on total time spent
- Any assumptions or limitations you'd like to highlight

Questions? Feel free to reach out via email for clarifications.