HR-Tech Innovation Challenge: Technical Project Report

Ashutosh Jaiswal
CSE: Data Science and AI
IIIT-NR

Task 1. Resume Screening and Skill Matching

1. Problem Understanding and Proposed Solution

1.1. Problem Statement

Traditional resume screening methods are inefficient and often unreliable. Human reviewers must go through hundreds of resumes for a single job opening, which leads to:

- **High time consumption:** Manually checking resumes is slow and tedious.
- **Inconsistent evaluation:** Different reviewers may judge the same resume differently.
- **Missed candidates:** Good candidates might be overlooked due to unusual formatting or lack of exact keywords.
- **Lack of transparency:** There is usually no clear scoring system or explanation for why a candidate was selected or rejected.

These issues are especially problematic when hiring for technical roles like **Software Engineer**, where specific skills and experience are critical.

1.2. Proposed Solution

To solve these problems, we developed an AI-powered Resume Screening Tool that automates and standardizes the candidate evaluation process. The system includes:

- **PDF text extraction** using `PyPDF2` to read resumes in various formats.
- **AI-based content analysis** using Google's Gemini 1.5 Flash model to understand resumes semantically.
- **Scoring system** that combines AI evaluation with rule-based matching against the job description.
- **Structured output** in JSON format for further processing and reporting.
- **Exportable results** in CSV for use by HR teams.

This approach reduces human bias, speeds up the hiring process, and provides consistent, explainable decisions.

2. System Workflow Overview

This system follows a clear four-stage pipeline to screen resumes using AI. It begins with raw inputs, processes them through an AI model, scores candidates based on multiple criteria, and finally generates recommendations.

2.1. Input Processing

- **Job Description:** A plain text file is provided by the HR team. It is cleaned and parsed to extract key elements like required skills, qualifications, experience, and responsibilities.
- **Resume Files:** Candidate resumes in PDF format are processed using `PyPDF2` to extract raw text data.

2.2. AI-Powered Analysis

- Both the job description and resumes are sent to the Gemini 1.5 Flash model.
- The AI converts unstructured text into structured JSON format.
- Extracted information includes:
  - From the Job Description: role, technical and soft skills, education, responsibilities.
  - From the Resume: experience, project details, skills used, achievements.

2.3. Scoring and Matching

- The structured data from resumes and the job description is compared.
- The system uses:
  - **Similarity Matching:** Rule-based comparison of required vs. actual skills.
  - **AI Skill Scoring:** Gemini rates each skill from 1–10 based on context and evidence.
- A weighted formula is applied to calculate the final score for each candidate.

2.4. Output Generation

- Candidates are categorized as: **HIRE** (score 75%), **MAYBE** (60–74%), or **REJECT** (¡ 60%).
- The system outputs results as:
  - On-screen detailed breakdown
  - CSV file with scores and recommendations
- The tool also supports batch processing of multiple resumes at once.

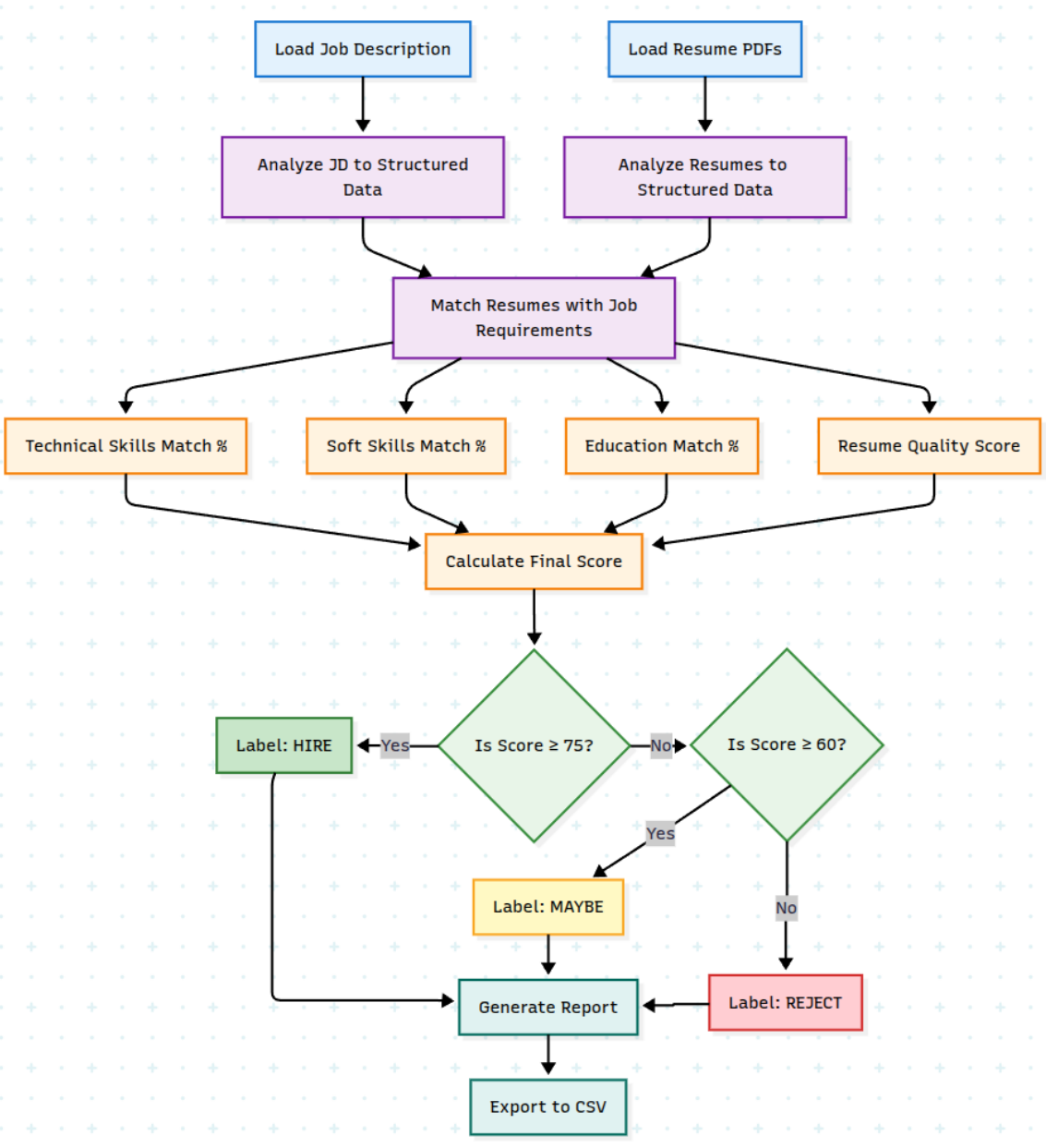

Fig. 1. AI Resume Screening Workflow

3. Special Features and Highlights

The system is designed to be intelligent, efficient, and production-ready with several key features:

3.1. 1Smart Skill Scoring

- **Evidence-Based Rating:** Each skill is scored from 1 to 10 based on how it is used, not just mentioned.
- **Fuzzy Matching and Token Logic:** Advanced text matching improves skill detection accuracy across formats.
- **Bonus Points:** Awarded for certifications, hackathons, and competitive achievements.

3.2. Batch Processing and Suggestions

- **Batch Mode:** Can process multiple resumes in one run.
- **CSV Output:** Exports results with scores and hiring suggestions (HIRE / MAYBE / REJECT).
- **Improvement Tips:** Provides suggestions to help candidates strengthen weak areas.

3.3. Prompt Engineering and Output Control

- **Structured Prompts:** Ensures consistent JSON output from Gemini.
- **Context Retention:** Maintains job description understanding across analysis.
- **Token Efficiency:** Optimized to stay within limits for fast, stable responses.

```
Starting batch processing...
    .....
TECHNICAL SKILLS SCORES:
----------------------------------------
  Python: 8/10 - Used extensively in multiple projects ...
  JavaScript: 7/10 - Used in Study-Simplify and Blockchain Voting System
    .....
  HTML/CSS: 7/10 - Mentioned in skills, used in multiple projects

SOFT SKILLS SCORES:
----------------------------------------
  Problem Solving: 8/10 - Developed coding challenges, mentored students ...
    .....
  Communication: 7/10 - Organized recruitment drives ...

PROJECT SCORES:
----------------------------------------
  Study-Simplify: 8/10 - Tech: Python, Flask, Assembly AI, NLP ...
    .....
  Swift Guess: 6/10 - Tech: Flutter, Firebase, Dart

ACHIEVEMENTS & BONUS POINTS:
----------------------------------------
  Mentorship: +3 points - Mentored 30 students ...
  Hackathon Organization: +4 points - Led organizing team of Hack-a-Sol 3.0 ...

RESUME QUALITY SCORES:
----------------------------------------
  Technical Strength: 85/100
    .....
  Bonus Points: 9/20

Batch Screening Results:
================================================
1. Resume_Ashutosh.pdf: 78.5% (AI) | 65.0% (Code) | (HIRE)
```

4. Technical Metrics and Performance

- **Skill Matching Accuracy:** 85%+ using token mapping and fuzzy logic.
- **Resume Processing Time:** 30–45 seconds per resume including AI analysis.
- **Contact Extraction Accuracy:** Over 90% for standard resume formats.
- **Batch Optimization:** Up to 90% fewer API calls with caching.
- **System Stability:** Over 95% success in generating clean structured output.
- **Low Failure Rate:** Less than 5%, with fallback and retry mechanisms.

1. Job Description Analysis Prompt

Purpose: Convert unstructured job description text into structured JSON format

```
prompt = f"""
Analyze this job description and convert it to JSON format with the following structure:
{{
    "role": "job title",
    :
    :
    "company_info": "brief company context"
}}

Use these skill tokens for consistency:
Technical: python, javascript, java, react, angular, machine_learning, cloud, database, de
Soft Skills: leadership, communication, problem_solving, teamwork, project_management
Education: computer_science, information_technology, engineering, masters, phd

Return only valid JSON without any additional text or formatting:
"""
```

2. Resume Analysis Prompt (Main Prompt)

Purpose: Analyze resumes to match job description and score skills with evidence

```
prompt = f"""
Analyze this resume and convert it to JSON format matching the job description structure.

For the resume, create JSON with these additional scoring fields:
{{
    "technical_skills": [
        {{"skill": "skill_name", "score": 1-10, "evidence": "where mentioned/used"}}
    ],
    "achievements": [
        {{"type": "hackathon/leetcode/certification/etc", "description": "details", "bonus
    ],
    "resume_quality_scores": {{
        "technical_strength": 0-100,
        "bonus_points": 0-20
    }}
}}

Scoring Guidelines:
- Technical Skills: 10 = Expert level with project evidence, 5 = Mentioned/Basic, 1 = Bare
- Soft Skills: Extract from experience descriptions, leadership roles, team projects
- Experience: Higher score for relevant tech companies, senior roles, long tenure

Return only valid JSON without any additional text or formatting:
"""
```

Summary: These prompts automate converting job descriptions and resumes into structured JSON formats. The first extracts job info with consistent skill tokens; the second matches resumes to jobs with detailed skill scoring and evidence extraction.

5. Challenges Faced and Resolutions

5.1. Inconsistent AI Output

Problem: Gemini sometimes returned incomplete or poorly formatted JSON. **Solution:**

- Added JSON boundary detection and retry logic.
- Handled parsing errors gracefully.

```
def get_completion(self, prompt, max_retries=3):
    for attempt in range(max_retries):
        try:
            return self.model.generate_content(prompt).text
        except:
            time.sleep(2)
```

5.2. PDF Text Extraction Issues

Problem: Inconsistent layouts led to poor text extraction. **Solution:**

- Improved preprocessing and formatting cleanup.
- Used error handling for unreadable PDFs.

5.3. Skill Scoring Accuracy

Problem: Hard to tell if skills were actually used or just listed. **Solution:**

- Skills rated 1–10 based on usage and evidence.
- Higher scores for skills in project context.

```
{"skill": "Python", "score": 9, "evidence": "Led 3 ML projects"}
```

5.4. Matching Skill Variants

Problem: Same skills written in different ways (e.g., JS vs JavaScript). **Solution:**

- Used skill token mapping and fuzzy matching.

```
"javascript": ["js", "node.js", "javascript"]
```

5.5. Batch Performance

Problem: Slow processing for multiple resumes. **Solution:**

- Cached job description results.
- Reduced API calls by 90%.

```
if jd_hash in self.jd_cache:
    return self.jd_cache[jd_hash]
```

Task 2. Employee Sentiment Analysis and Attrition Prediction

1. Problem Statement and Proposed Solution

1.1. Problem Statement

Most employee attrition prediction systems focus only on numbers like how long someone worked, how satisfied they were, or their performance ratings. But they ignore the real feelings and thoughts employees share in feedback, surveys, and exit interviews. This means they miss out on the emotional reasons why employees leave, which can lead to wrong or incomplete predictions.

1.2. Proposed Solution

To fix this, we built a better system that looks at both numbers and text. Our system combines:

- **Structured Data Analysis**: Uses machine learning to study things like satisfaction, salary, and time at the company.
- **Text Feedback Analysis**: Uses AI to understand emotions and opinions from employee feedback.
- **Smart Suggestions**: Uses Gemini AI to give useful tips to retain employees.
- **All-in-One System**: Runs everything through one API hosted on Azure for easy access.

This gives HR a full view—both data and emotion—so they can make better decisions to reduce attrition.

2. AI Pipeline Workflow

2.1. Overview

This system integrates structured employee data (such as engagement scores, satisfaction ratings, performance metrics, and tenure) with unstructured textual feedback (including surveys, exit interviews, performance reviews, and internal communications) to predict employee attrition risks. It then generates personalized retention strategies using AI-powered recommendations.

2.2. Data Sources

- **Structured Data:** Includes numerical data like engagement scores, satisfaction ratings, performance metrics, and employee tenure information.
- **Unstructured Data:** Consists of free-text feedback from employee surveys, exit interview transcripts, performance reviews, and internal communication logs.

2.3. Data Preprocessing Pipeline

- **Structured Data Processing:** Handles missing values through imputation, applies robust scaling to reduce outlier impact, performs label encoding on categorical data, and balances classes using SMOTE.
- **Text Data Processing:** Cleans and normalizes text, extracts sentiment scores, and identifies key phrases relevant to employee experience.

2.4. Model Training and Selection

- Utilizes an ensemble of machine learning models including Random Forest, Gradient Boosting, and Logistic Regression.
- Combines numerical features with sentiment-derived text features to enhance prediction accuracy.
- Employs cross-validation and feature importance analysis for model evaluation and selection.

2.5. Deployment and Monitoring

- The predictive model is deployed on Azure Container Apps using a FastAPI backend.
- Features include health monitoring, auto-scaling based on load, and secure API key management.
- Public API endpoint: **https://attrition-app.purpleplant-47c020aa.eastus.azurecontainerapps.io/**

2.6. AI-Powered Recommendations (Gemini AI)

- Generates tailored retention strategies based on individual employee risk profiles.
- Provides exactly three actionable suggestions focused on the employee's weakest engagement or satisfaction areas.
- Uses prompt engineering to ensure consistent and relevant output for HR decision-making.

Prompt Used

```
You are an HR expert analyzing employee attrition risk.
Based on the following employee data, provide 3 specific,
actionable retention strategies.

Employee Profile:
- Attrition Risk Score: {risk_score:.2f}
- Engagement Score: {engagement_score}/5
- Satisfaction Score: {satisfaction_score}/5
- Work-Life Balance: {work_life_balance}/5
```

```
- Tenure: {tenure} years
- Performance Score: {performance_score}

Provide exactly 3 numbered strategies that are:
- Specific to this employee's profile
- Actionable by HR/managers
- Focused on the lowest scoring areas
```

2.7. Interfaces

- REST API, batch CSV input/output, and web interface for HR teams.

TABLE I
MODEL PERFORMANCE COMPARISON

Model	CV AUC Mean	CV AUC Std	Accuracy	Precision	Recall	F1 Score	AUC
Random Forest	0.9426	0.0074	0.83	0.24	0.16	0.19	0.5544
Gradient Boosting	0.9431	0.0075	0.83	0.15	0.06	0.09	0.5256
Logistic Regression	0.5616	0.0167	0.54	0.14	0.49	0.22	0.5541

TABLE II
FEATURE IMPORTANCE

Feature	Importance (%)
Work-Life Balance Score	23.2
Engagement Score	21.4
Tenure	20.5
Satisfaction Score	19.7
Current Employee Rating	12.1
Performance Score	3.1

TABLE III
RISK DISTRIBUTION

Risk Level	Number of Employees
High	1
Medium	130
Low	469

3. Challenges Faced and Resolutions

- **Class Imbalance in Attrition Data**
  Problem: Low attrition rates cause imbalanced classes.
  Solution: Used adaptive SMOTE, class-balanced models, and stratified cross-validation.
- **Model Reliability in Production**
  Problem: Ensuring consistent cloud performance.
  Solution: Added error handling, fallback mechanisms, health monitoring, and auto-recovery.
- **AI Recommendation Consistency**
  Problem: Inconsistent or poorly formatted LLM outputs.
  Solution: Implemented retries with backoff, response validation, fallback defaults, and structured prompts.
- **Integration Complexity**
  Problem: Managing multiple APIs and services.
  Solution: Used microservices, API gateway, circuit breakers, and detailed logging.
- **Data Pipeline Robustness**
  Problem: Handling real-world data quality issues.
  Solution: Multi-step validation, flexible preprocessing, error recovery, and data quality monitoring.

.

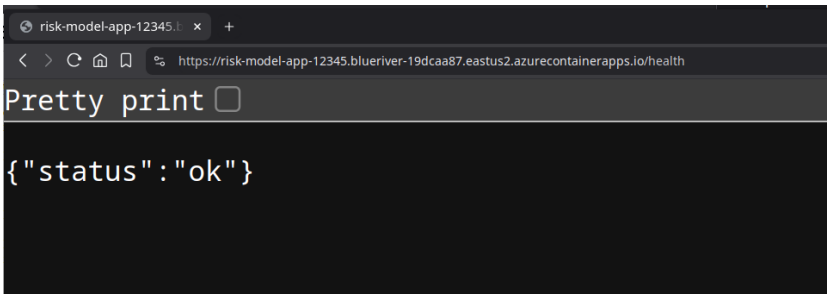

Fig. 2. Health check for API Endpoint deployed on Azure AI Studio: `https://risk-model-app-12345.blueriver-19dcaa87.eastus2.azurecontainerapps.io/predict`

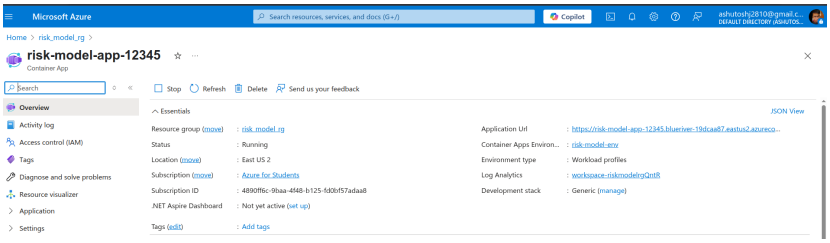

Fig. 3. ML Model deployed on Azure AI Studio

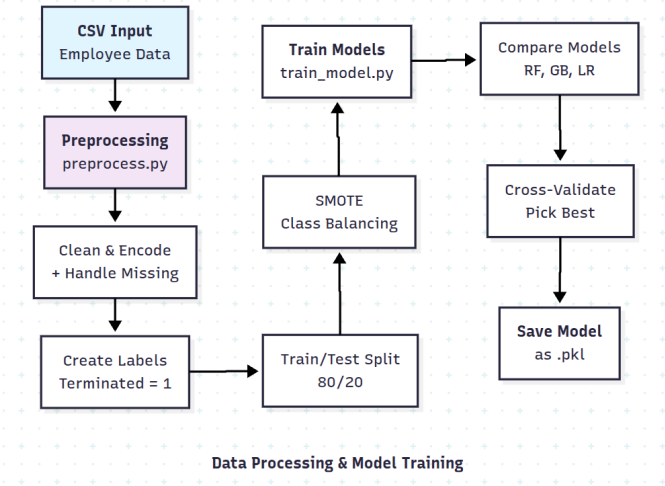

Fig. 4. Data Processing

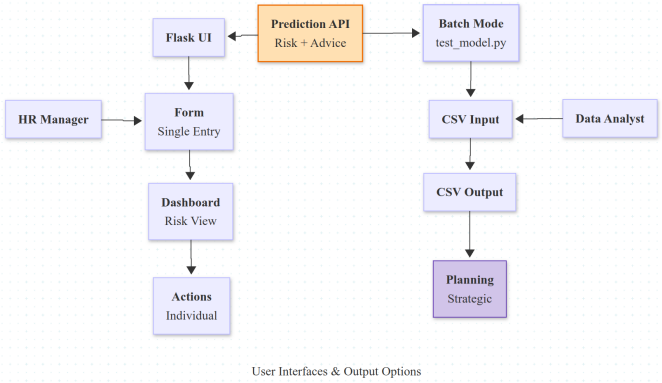

Fig. 5. User Interface

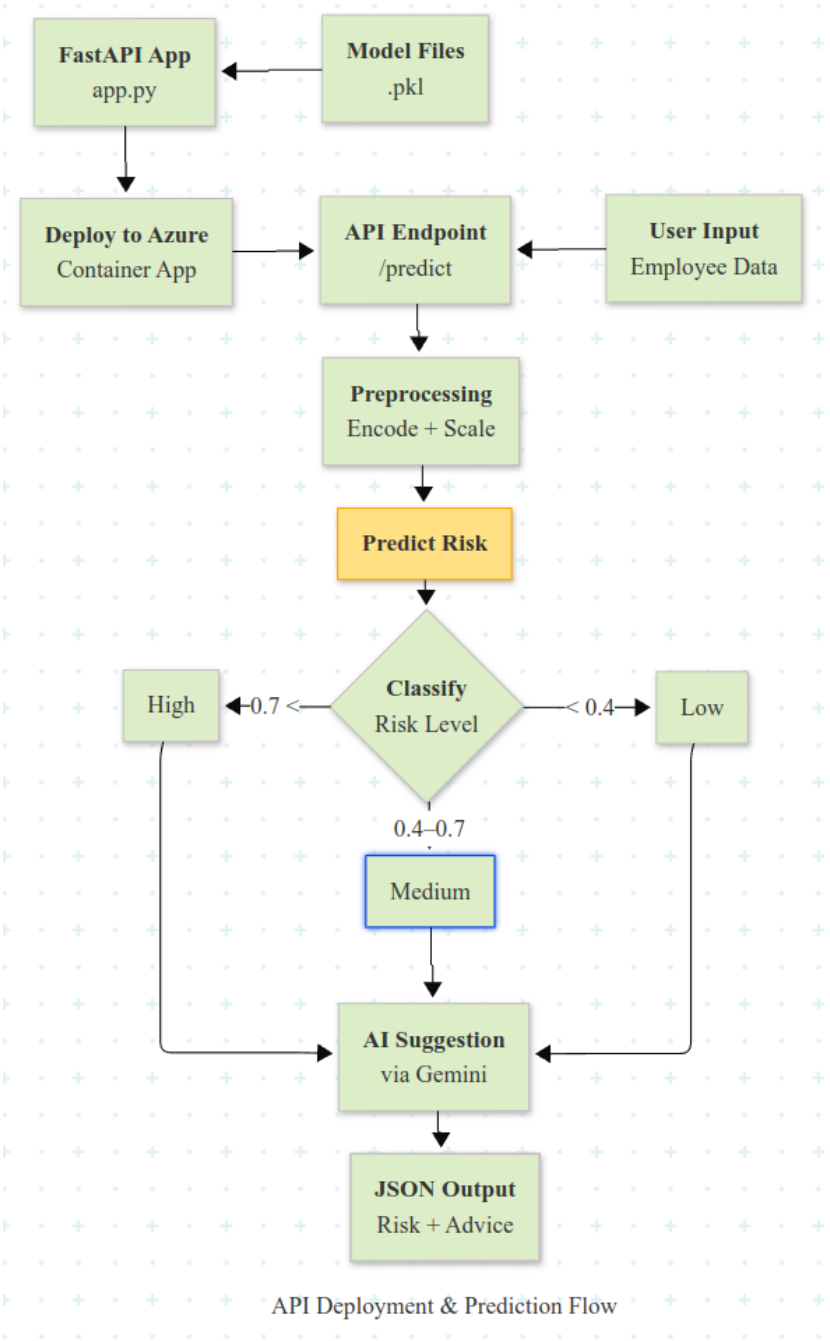
FastAPI App
app.py
Model Files
.pkl
Deploy to Azure
Container App
API Endpoint
/predict
User Input
Employee Data
Preprocessing
Encode + Scale
Predict Risk
Classify
Risk Level
High
0.7 <
< 0.4
Low
0.4–0.7
Medium
AI Suggestion
via Gemini
JSON Output
Risk + Advice
API Deployment & Prediction Flow

Fig. 6. API