# AI for Software Engineering

June 28, 2025

0.1 # AI for Software Engineering Assignment # Title: Understanding the AI Development Workflow 0

0.2 Part 1: Short Answer Questions (30 pos)

0.2.1 1. Problem Definition (6 ints)

Hypothetical Problem:
Recommending local job opportunities to unemployed youth based on skills andocation. *Objectives: **1. Match user skills to relevant job descriptions using NLP. 2. Increase accessibility to local job listings in under-resourced areas. 3. Reduce the job search time by delivering personalized rommendations.** Stakeholders:** - Youth/job seekers in townships and rural areas. - Local employers or small businses posting jobs.

KPI (Ke)rformance Indicator):
Job Match Accuracy — percentage of recommended jobs clickeplfor by users.

0.2.2 2. Data Collection Preprocessing8 points)

Data Sources: 1. Local job postings scraped from WhatsApp groups, Facebook pages, or community websites. 2. User-submitted profiles with ills and prefer location.

Potential Bias:
Location bias — over-representation of job posts from urban areas, underepresenting rural cmunities.

Preprocessing Steps: 1. Text cleaning — remove stopwords and punctuation from job descriptions. 2. Vectorization — convert text to numerical form using TF-IDF. 3. Handling missing data — fill in missing fields like ltion using location inference.

-

1 3. Model Development (8 points)

**Model:*:F-IDF + Cosine Similarity
Justification:ightweight d effective for text matching.

Data Spli
70% training / 15%alidation / 15% test

Hyperparameters to Tune: 1. Max number of features in TFF.. N-gram range (e.g., unigrams and bigra).

1.0.1 4.valuation & Deployment (8 points)

Evaluation Metrics: 1. Precision — to ensure relevant job mates. 2. Click-rough rate (CTR) — tracks user engagement.

Concept Drift: Occurs when job lan:ge or user behavior changes over time.
MonitorinStrategy: Track CTRgs and periodically retrain the model.

Deployment Challenge:
Scali — handling large amounts of user and job datn real time.

1.1 Part 2: Case dyplication (40 points)

1.1.1 Hpital Readmission:sk Prediction

1.1.2 1. Problem Scope (5 points)

Problem Statemt:
Pr:ct likelihood of patient readmission within 30 days of discharge.

Objectives: 1. Identify high-risk patients. 2. Allote post-dischge care efficiently. 3. Reduce hospital costs and improve outcomes.*Seholders:** - Hospital managemt - Medical aff (doctors, discharge planners)

1.1.3 2. Data Strategy (10 points)

Data urces: 1. Ele:onic Health Records (EHRs) 2. Demographics (e.g., age, gender, past visits)

**Ethical Concerns* 1. Patient privacy (ta protection compliance) 2. Informed consent to use data for AI modeling

Preprocessing Pipeline: - Clean and anonymize data. - Engineer features (e.g., ageay length, number of comorbidities) - Norlize numeric values and encode categorical variaes.

---

2 3. Model Development (10 points)

Model: Gradient Boosting Classifier (e.g., XGBoost) *Justification:**
Performs ll on structured healthcare data, supports missing values, and is interpretable.

Hypothetical Confusion Matr:

	Predicted: No	PredictedYes
Actual: No**	85	15
Actual: Ye	10	40

Precion = 40 / (40 + 15) = 72.7%
Recall = 40 / (40 + 10) = 80%

2.0.1 4. Deployment (10 points)

Integration Steps: 1. Wrap model in a REST A (Flask or FastAPI).2. Connect it to the hospital's EHR system. 3. Display risk predictions in the discharge planning dashboard.

**Compliance Measures:* Eypt patient data (in transit d at rest). - Restrict model acss to authorized personnel. - Follow HIPAA or POPIA regulations.

2.0.2 5. Opzaon (5 points)

Overfitting Mitigatitrgy: - Use Cross-Validation f evaluation. - Apply Regularizion (L1/L2) in the model.

2.1 Part 3: Critical Thinking (20 points)

2.1.1 1. Ethics & Bias (10 points)

Impact of Biasedraining Data: - May underpredict risk for under-represented groups. - Can result in missed interventions or unfair prioritizati itigation Strategy:** - Balce the dataset using resampli. - Perform fairness evaluation across demographic groups. - Use fairness-aware algorithms.

2.1.2 2. Trade-offs (10 points)

Interpretability vs. Accuracy: - Interpretable models (e.g.,ecion trees) improve trust but y lose accuracy. - High-accuracy models (e.g., deep learning) may lack transparency. - In healthcare, interpretability is prioritize. **Impact of Limited Resources:** - Use lightweighde(e.g., logistic regression) - Avoid GPU-heavy apprhes. - Consider batch rather than real-time inference.

2.2 ## Part 4: Refltion & Workflow Diagram (10 points)

2.2.1 1. Reflection (5 points)

Most Challenging Stage:
Data collection and preprocessing due to privacy, bias, and quality concerns.

Improvements with Moimesources: - Partner with medical institutions for real-world data. - Include domain experts in the feature engineering process. - Use federated learning to protect patient privacy.

2.2.2 2. AI Workflow Diagram (Text Version) (5 points)

+————————————+ | 1. Problem Definition | +————————————+ ↓ +————————————+ | 2. Data Collection | | & Preprocessing | +————————————+ ↓ +————————————+ | 3. Model Development | +————————————+ ↓ +————————————+ | 4. Evaluation | +————————————+ ↓ +————————————+ | 5. Deployment | +————————————+ ↓ +————————————+ | 6. Monitoring & | | Improvement | +————————————+

[ ]:
```
---

Would you like help uploading this to GitHub right now or adding more features␣
↪like real CSV data or a simple Streamlit app? :contentReference[oaicite:
↪0]{index=0}
```

[ ]:
```
computer, office, admin
```

[1]:
```
import sklearn
print(sklearn.__version__)  # Should show version (e.g., 1.3.0)
```

```
1.3.0
```

[ ]:
```
# job_recommender.py

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# ---------------------------
# 1. Load Sample Data
# ---------------------------

# Sample job descriptions
jobs = pd.DataFrame({
    'job_id': [1, 2, 3, 4, 5],
    'title': ['Admin Assistant', 'Junior Web Developer', 'Data Entry Clerk',␣
↪'Warehouse Packer', 'Receptionist'],
```

```python
    'description': [
        'Looking for an admin assistant with Microsoft Office skills and attention to detail.',
        'Seeking a junior web developer with HTML, CSS, and JavaScript experience.',
        'Data entry clerk needed for capturing info into spreadsheets. Accuracy important.',
        'Warehouse packer needed for sorting goods and packing orders. Physical strength required.',
        'Receptionist needed with good communication and computer literacy.'
    ],
    'location': ['Soweto', 'Johannesburg', 'Tembisa', 'Diepsloot', 'Alexandra']
})

# ---------------------------
# 2. Preprocess and Vectorize
# ---------------------------

# Combine relevant fields for matching
jobs['text'] = jobs['title'] + " " + jobs['description']

# Vectorize using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
job_vectors = vectorizer.fit_transform(jobs['text'])

# ---------------------------
# 3. User Input and Matching
# ---------------------------

def recommend_jobs(user_skills, top_n=3):
    user_vector = vectorizer.transform([user_skills])
    similarities = cosine_similarity(user_vector, job_vectors).flatten()
    top_indices = np.argsort(similarities)[-top_n:][::-1]
    return jobs.iloc[top_indices][['job_id', 'title', 'location', 'description']]

# Example usage
if __name__ == "__main__":
    print("=== Youth Job Recommender ===")
    user_input = input("Enter your skills (e.g. computer, office, admin): ")
    results = recommend_jobs(user_input)
    print("\nTop Job Matches:")
    for _, row in results.iterrows():
        print(f"Title: {row['title']} | Location: {row['location']}")
        print(f"Description: {row['description']}")
        print("-" * 40)
```

```
=== Youth Job Recommender ===
```

[ ]:

```

```