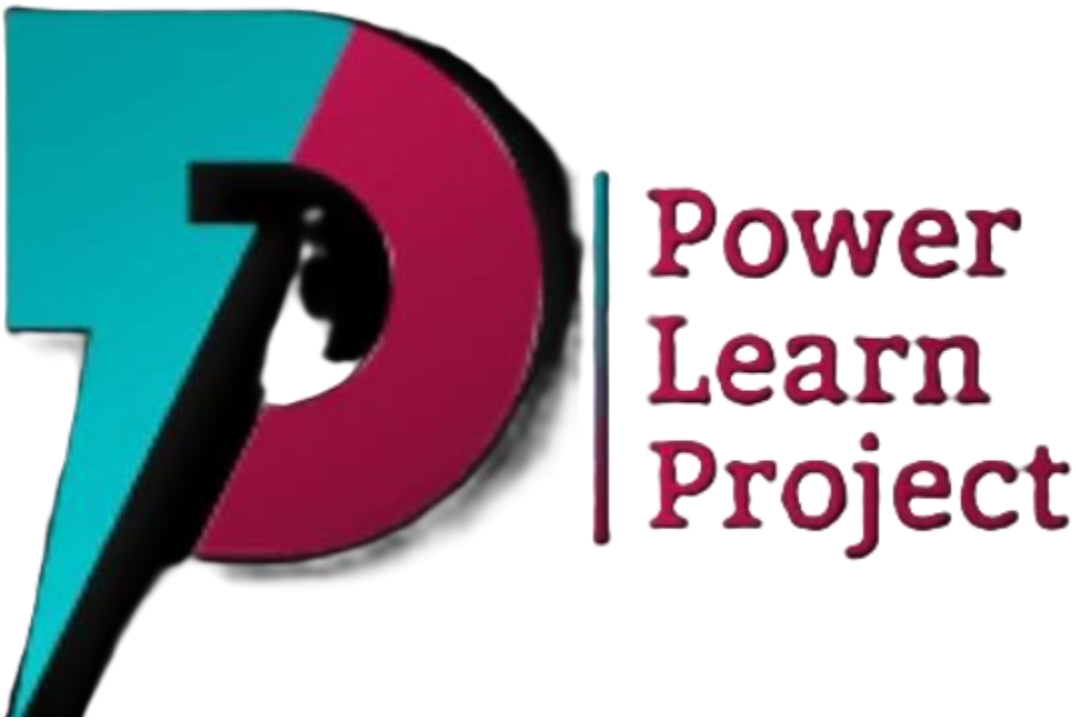

POWER LEARN PROJECT

SCHOOL OF SOFTWARE ENGINEERING SHOLARSHIP

CSC - AI FOR SOFTWARE ENGINEERING

Prepared

BY

BENEDICT OOKO

BSC

SUPERVISOR:

AI Engineer. Evans Mutuku

NOVEMBER 2025

1.(a). Problem Definition (6 points)

Hiring Trend Analysis for Predicting Future Skill Demand

1.(b). List 3 objectives and 2 stakeholders.

i. List 3 objectives

a. To understanding what skills, roles, and qualifications employers are actively seeking in the present job market using job postings and workforce data.

b. To extract and classify skill requirements from job descriptions using NLP techniques.

c. To develop an AI-powered predictive model that forecasts future skill demand based on historical and real-time hiring trends.

ii. List 2 stakeholders

a. Employers

b. Job Seekers

1. (c). Propose 1 Key Performance Indicator (KPI) to measure success.

a. Prediction Accuracy of Future Skill Demand (%).

2. Data Collection & Preprocessing (8 points)

2. (a). Identify 2 data sources for your problem.

a. Online Job Portals – LinkedIn API

b. Open datasets (Kaggle job market datasets)

2. (b). Explain 1 potential bias in the data.

Job Posting Bias (Industry or Role Overrepresentation)

If the dataset has **more job postings from certain industries or popular job portals**, the AI model may learn trends that favor those sectors while ignoring others.

Example:
If most job postings come from the **tech industry**, the model may overestimate demand for tech-related skills (e.g., Python, cloud computing) and underestimate skills needed in underrepresented sectors like healthcare, education, or manufacturing.

2. (c) Outline 3 preprocessing steps (e.g., handling missing data, normalization).

Handling missing or In-complete data - Some job postings may have missing fields such as required skills, salary, or job location. Removing entries with too many missing values or Add missing values (e.g., fill location using company info).

Cleaning and Normalization for Job Descriptions - apply NLP preprocessing such as: Lowercasing

3. Model Development (8 points)

3. a) Choose a model (e.g., Random Forest, Neural Network) and justify your choice.

Gradient Boosted Trees (LightGBM / XGBoost)

Why: excellent performance on mixed tabular and engineered text features, fast to train, robust to irrelevant features, handles missing values well, and offers feature-importance explanations (useful for stakeholders).

3. b) Describe how you would split data into training/validation/test sets.

Because this problem depends on time, random split is avoided to prevent temporal leakage.

Use a time-aware split:

Training: earliest 70% of the time range.

Validation: next 15% (used for hyper-parameter tuning and early stopping).

Test: final 15% (held out for the unbiased assessment of forecasting performance).

3. c) Name 2 hyper-parameters you would tune and why.

learning_rate (aka eta) it controls the step size at each boosting iteration.

Reason: lower values often improve generalization but require more trees; tuning balances bias vs variance and affects convergence speed and final accuracy. Use with early stopping.

num_leaves (LightGBM) / max_depth (XGBoost) it controls model complexity (size of individual trees)

Reason: affects model capacity (too large) overfitting to past trends (dangerous for forecasting); under-fitting and missed non-linear relationships between features and skill demand.

4. Evaluation & Deployment (8 points)

4. (a) Select 2 evaluation metrics and explain their relevance.

Mean Absolute Error (MAE): measures the average difference between the predicted skill demand and the actual observed demand.

Relevance:

- It is easy to interpret (in real units of demand).

- Less sensitive to large outliers than RMSE.

- Helps determine how accurately the model forecasts future hiring trends.

F1-Score: F1-score is the harmonic mean of precision and recall.

Relevance:

- Useful when classes are imbalanced (e.g., few skills become "high-demand").

- Ensures the model not only identifies high-demand skills, but does so accurately without too many false positives or false negatives.

- Balances both precision and recall for more reliable decision-making.

4. (b) What is concept drift? How would you monitor it post-deployment?

Concept drift refers to the phenomenon where the statistical properties of the target variable in a predictive model change over time, leading to decreased model accuracy

Track model performance over time using metrics like MAE or F1-score. A consistent decline indicates drift.

Monitor data distributions (skill frequencies, job-posting trends) and compare them with training data.

Use drift detection algorithms such as DDM (Drift Detection Method) or ADWIN.

Set alerts when key features (e.g., emerging skills) show unusual changes

4. (c) Describe 1 technical challenge during deployment (e.g., scalability).

Scalability of Real-time Predictions

When deployed, the system may need to process large volumes of incoming job postings or trend data in real time.
Why this is challenging:

- NLP feature extraction (e.g., TF-IDF or embeddings) can be computationally heavy.
- Spikes in traffic from API calls or data ingestion may overwhelm the system.

Result:
Slow response times, failed requests, or delayed trend insights.

Possible solution:
Use distributed systems like Docker and Kubernetes, implement **batch processing**, or use **message queues** (Kafka) to scale efficiently.