University of Florida – MS in Applied Data Science

AI-powered Job Seek Tool – A conversational agent

By – Swetha Gendlur Nagarajan

Introduction

The proposed conversational agent would serve as a comprehensive job search and career development tool, leveraging advanced data analysis and visualization techniques to provide personalized guidance to job seekers. Upon engaging with the agent, users would input their desired role. The system would then analyze current job market trends for that position, presenting data visualizations to illustrate where the majority of opportunities are located geographically and what technology stacks are most in demand for that role.

The agent would request the user's resume and compare it against job descriptions in its database. Using natural language processing and keyword matching algorithms, it would calculate a percentage match between the user's skills and experience and the job requirements. For users deemed a good fit (based on a high match percentage), the agent would provide actionable next steps. This would include guidance on crafting effective cold emails to potential employers and providing contact information for relevant recruiters within target organizations.

If the user's qualifications fall short of the job requirements, the agent would offer a tailored improvement plan. This would involve identifying skill gaps and recommending relevant courses from platforms like Coursera to address those deficiencies.

To facilitate networking, the agent would compile a list of professionals currently working in the user's desired role, potentially leveraging data from professional networking platforms. This would allow users to connect with individuals who could provide valuable insights and potentially serve as internal referrals. Throughout the interaction, the agent would maintain a professional tone while providing detailed, data-driven insights to help users navigate their job search and career development journey effectively.

Type of Tool

The project will result in an interactive AI chatbot with natural language processing capabilities, data analysis functions, and visualization features. It will serve as a comprehensive career development platform, combining job market insights, resume analysis, and personalized recommendations.

Dataset Used

1. The Coursera dataset, sourced from Kaggle, provides a comprehensive overview of online courses available on the platform. It includes crucial information such as course titles, enrollment numbers, ratings, instructor details, skills taught, course descriptions, and difficulty levels. This dataset will be instrumental in recommending targeted learning opportunities to users looking to enhance their skill sets or bridge knowledge gaps identified during the job matching process.

2. The LinkedIn dataset, also obtained from Kaggle, offers valuable insights into professional profiles. It encompasses details such as names, current roles, work experience, skills, certifications, and contact information. While smaller in scale compared to the job description dataset, this information will be vital for the AI agent to suggest relevant networking connections and potential mentors in the user's desired field.
3. The job description dataset, acquired from Hugging Face, is significantly larger and more comprehensive than the other datasets. It contains a wealth of information about job postings, including job titles, required experience, qualifications, locations, company details, salary ranges, and detailed job descriptions. The sheer volume and depth of this dataset will allow for robust job market analysis, accurate skill matching, and the provision of detailed insights into job trends and geographical distributions.
4. The roles-based skills dataset, another Kaggle resource, focuses specifically on the skill requirements for various positions. It includes information such as company names, position titles, core responsibilities, required skills, educational requirements, and experience levels. While not as extensive as the job description dataset, it provides valuable, role-specific data that will enable the AI agent to offer tailored advice on skill development and career progression paths.

1. LinkedIn dataset:

1. **Data Loading and Preparation:**
    a. The code loads a dataset of synthetic LinkedIn profiles using the datasets library.
    b. It converts the dataset into a Pandas DataFrame for easier data manipulation.
    c. It performs initial data cleaning, such as removing unnecessary columns (like embeddings, 'About Me', 'Recommendations', etc.) and renaming columns for better clarity (e.g., renaming 'Headline' to 'Current Role').
2. **Data Cleaning and Transformation:**
    a. It further cleans the 'Current Role' column to extract the primary role from the text.
    b. It creates a new 'Contact mail' column by generating hypothetical email addresses based on first and last names.
3. **Data Filtering and Analysis:**
    a. It filters the data to focus on profiles with 'data scientist' in their current role.
    b. It extracts and analyzes the skills mentioned by data scientists, identifying the most common skills.
    c. It generates a bar chart to visualize the frequency of these top skills.
4. **Role Analysis:**
    a. It extracts and analyzes the current roles of all profiles, identifying the most common roles.
    b. It generates a bar chart to visualize the frequency of these top roles.

In essence, the code loads a dataset of LinkedIn profiles, cleans and prepares the data, focuses on data scientist profiles, analyzes their skills and roles, and visualizes the findings using bar charts.

2. Job Description Data:

Data Preprocessing Steps:

1. **Data Loading and Cleaning:**
    a. Loads a CSV file containing job descriptions into a pandas DataFrame.
    b. Removes unnecessary columns (location, Benefits).
    c. Handles missing data by imputing or removing values in columns like Company Profile, Company, Sector, Industry, Website.
    d. Converts salary information from text to numerical format, calculating average salary.
    e. Extracts numerical experience range from text and calculates average experience.
    f. Addresses potential outliers in numerical columns (e.g., Average Salary, Experience) using statistical methods.
    g. Normalizes or scales numerical features using Min-Max scaling and standardization.
2. **Data Transformation:**
    a. Converts categorical columns (e.g., Work Type, Preference, Job Title) to the appropriate data type.
    b. Identifies and removes columns with all zero values.

Exploratory Data Analysis (EDA) Steps:

1. **Descriptive Statistics:**
    a. Uses descriptive statistics like mean, median, and standard deviation to summarize the dataset.
2. **Visualizations:**
    a. Creates various visualizations (histograms, scatter plots, box plots, bar charts, heatmaps, pie charts) to explore patterns, trends, and relationships in the data.
    b. Examples include:
        i. Gender distribution by job title
        ii. Average salary by qualification and job type
        iii. Job demand over time
        iv. Geographical spread of job titles
3. **Correlation Analysis and Statistical Tests:**
    a. Identifies potential issues such as multicollinearity or skewed distributions using correlation analysis and statistical tests (not explicitly shown in the provided code snippet).

Overall Goal:

The code aims to prepare the job description dataset for further analysis (e.g., machine learning model building) by cleaning, transforming, and exploring the data. The EDA provides insights into the data, such as gender distributions in job roles, salary trends, and qualifications in demand.

3. Roles and Skills Data:

Data Loading and Preprocessing

1. **Import necessary libraries:** The code begins by importing pandas for data manipulation, json for handling JSON data, and other libraries for analysis and visualization.
2. **Load datasets:** Two datasets are loaded:

a. job-descriptions: Contains job descriptions and related information like position titles, company names.
b. roles-based-on-skills: Contains roles and their required skills.

3. **Data Cleaning and Transformation:**
    a. The code extracts relevant information from the 'model_response' column in job-descriptions and creates new columns.
    b. Unnecessary columns (like job_description, description_length, model_response) are dropped.
    c. Missing values are handled by filling them with the most frequent value in each column.
    d. Duplicate rows are removed based on the 'position_title' column.
    e. Data types of some columns are converted (e.g., company_name and position_title to categorical).
    f. Outliers in categorical columns (like position titles with very low frequency) are identified and printed.

Data Integration and Exploration

1. **Data Integration:** Roles from the roles-based-on-skills dataset that are not present in the job-descriptions dataset are added to the main dataframe (df_js). This expands the dataset with new roles and their required skills.
2. **Exploratory Data Analysis (EDA):**
    a. The code focuses on visualizing the relationship between the top 30 companies and position titles.
    b. It creates a heatmap using seaborn and matplotlib to show the frequency of each position title within each of the top 30 companies.

Purpose

The code aims to analyze and understand the relationship between companies, job positions, and required skills. By loading, cleaning, transforming, and integrating two datasets, it provides a foundation for understanding the job market landscape. The EDA step helps visualize the distribution of position titles within different companies, highlighting potential trends and patterns. This information can be valuable for tasks like job recommendation, skill gap analysis, or market research.

4. Coursera Course Details:

1. **Data Loading and Cleaning:**
    a. The code starts by loading a Coursera course dataset from a CSV file using pandas.
    b. It then cleans the data by:
        i. Removing unnecessary columns (Unnamed: 0, Satisfaction Rate).
        ii. Handling missing values by filling them with appropriate measures (mean, mode).
        iii. Cleaning and converting data types of certain columns (e.g., Modules/Courses, Schedule).
2. **Data Exploration and Visualization:**
    a. After cleaning, the code explores the data through:

i. Identifying and visualizing outliers in numerical columns like rating, num_reviews, and Modules/Courses using box plots.
ii. Visualizing the distribution of the 'enrolled' column using a histogram.

3. **Key Insights Visualization:**
   a. Finally, the code visualizes key insights:
      i. It creates a bar plot to showcase the top 20 highest-rated courses, highlighting the instructors and ratings.
      ii. It also creates a bar plot showing the top 10 longest courses (based on the number of modules) among the top 20 courses.
      iii. Another bar plot presents the top 10 courses with the highest number of reviews.

Tech Stack

1. Python for backend development and data analysis (pandas, numpy)
2. Natural Language Processing libraries (NLTK or spaCy)
3. Machine Learning frameworks (TensorFlow or PyTorch)
4. Data visualization libraries (Matplotlib, Seaborn, or Plotly)
5. Flask or Django for web framework
6. Open AI API, Langchain frameworks

Timeline

1. Feature Engineering (~ 2 weeks)

Objective: Transform raw data into meaningful predictors for resume-job matching and skill gap analysis.

- **Process**:
  - **Job Description Features**:
    - Extract skills, tools, and certifications using NLP (e.g., spaCy).
    - Create interaction terms (e.g., "Python + SQL" as a combined skill score).
    - Engineer time-based features (e.g., years of experience vs. job posting trends).
  - **Resume Features**:
    - Parse resume text into structured skills, roles, and tenure.
    - Aggregate metrics (e.g., "number of certifications relevant to target role").
  - **Coursera/LinkedIn Features**:
    - Encode course difficulty levels (label encoding).
    - One-hot encode job sectors (e.g., "Tech", "Healthcare").
- **Milestone**: Dataset with engineered features ready for modeling.

2. Feature Selection (~ 2 Weeks)

Objective: Identify the most impactful features for accurate job-resume matching.

- **Process**:
    - **Correlation Analysis**:
        - Calculate feature correlations with target variables (e.g., "job fit score").
        - Use heatmaps to visualize relationships (e.g., salary vs. certifications).
    - **Tree-Based Importance**:
        - Train a Random Forest to rank features (e.g., "Python" > "Excel" for data roles).
    - **Dimensionality Reduction**:
        - Apply PCA to skills matrices for roles with overlapping requirements.
        - Use LASSO regression to eliminate noisy features (e.g., irrelevant soft skills).
- **Milestone**: Final feature set with top 20 predictors (e.g., key skills, experience thresholds).

3. Data Modeling (~ 1 Week)

Objective: Build and optimize models for job matching and course recommendations.

- **Process**:
    - **Model Selection**:
        - Logistic Regression (baseline for job fit classification).
        - Random Forest (skill importance analysis).
        - Neural Networks (BERT-based resume-job description similarity).
    - **Hyperparameter Tuning**:
        - Grid search for optimal Random Forest depth/n_estimators.
        - Learning rate optimization for neural networks.
    - **Validation**:
        - Split data (80% training, 20% testing) stratified by job roles.
        - Cross-validate to prevent overfitting (5-fold CV).
- **Milestone**: Top-performing model with 90%+ cross-validation accuracy.

4. Evaluation & Interpretation (~ 2 Weeks)

Objective: Validate model performance and derive actionable insights.

- **Process**:
    - **Performance Metrics**:
        - Precision/Recall: Measure relevance of recommended jobs/courses.
        - ROC-AUC: Evaluate resume-job matching confidence scores.
        - F1-Score: Balance false positives/negatives in "fit vs. unfit" classification.
    - **Bias Checks**:
        - Analyze model fairness across demographics (e.g., gender-neutral skill recommendations).
        - Audit for geographic bias in job opportunity predictions.

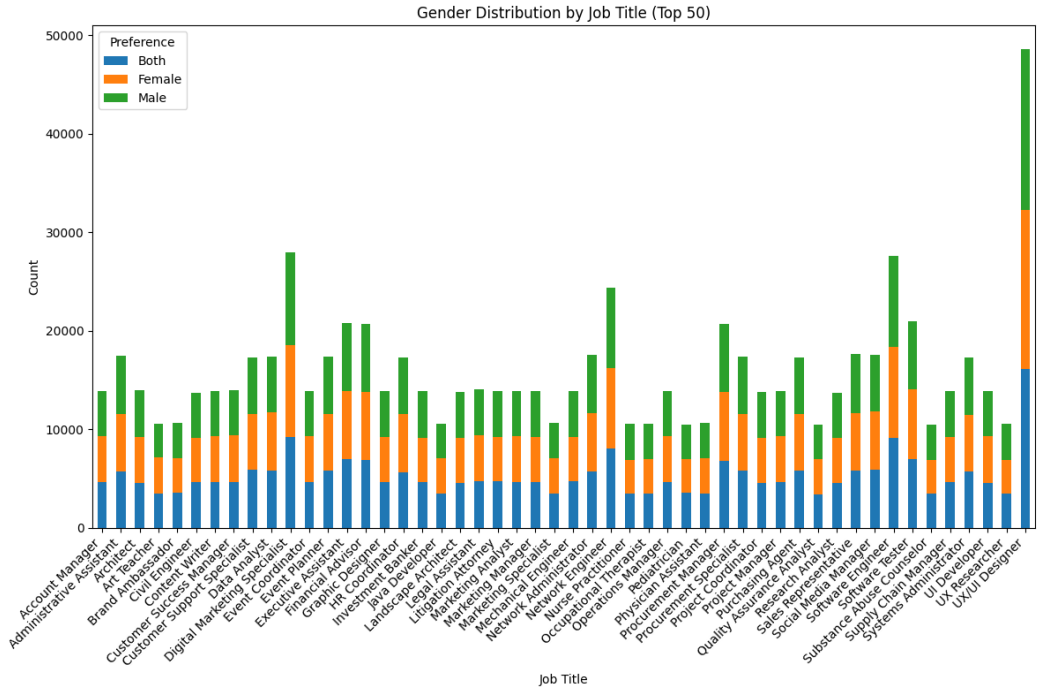
- **Explainability**:
  - SHAP values to interpret BERT model outputs (e.g., "Python contributed 40% to fit score").
  - Visualize decision trees for skill gap recommendations.
- **Milestone**: Evaluation report with metrics, bias audit, and stakeholder-ready insights.

Key Deliverables Timeline

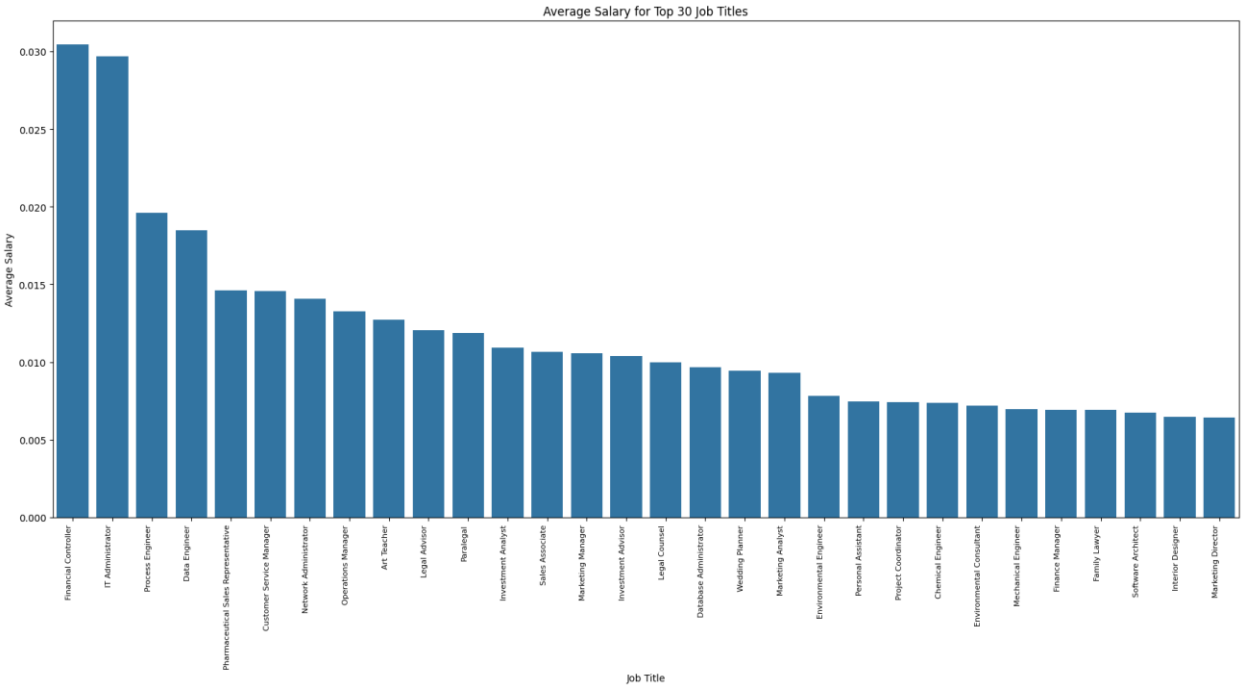
Phase	Outputs
Feature Engineering	Cleaned datasets with engineered features (skills scores, interaction terms)
Feature Selection	Heatmaps, PCA/LASSO results, final feature list
Data Modeling	Trained models (Random Forest, BERT), hyperparameter tuning logs
Evaluation	ROC curves, SHAP plots, bias audit report

Exploratory Data Analysis (Only main ones are focused)

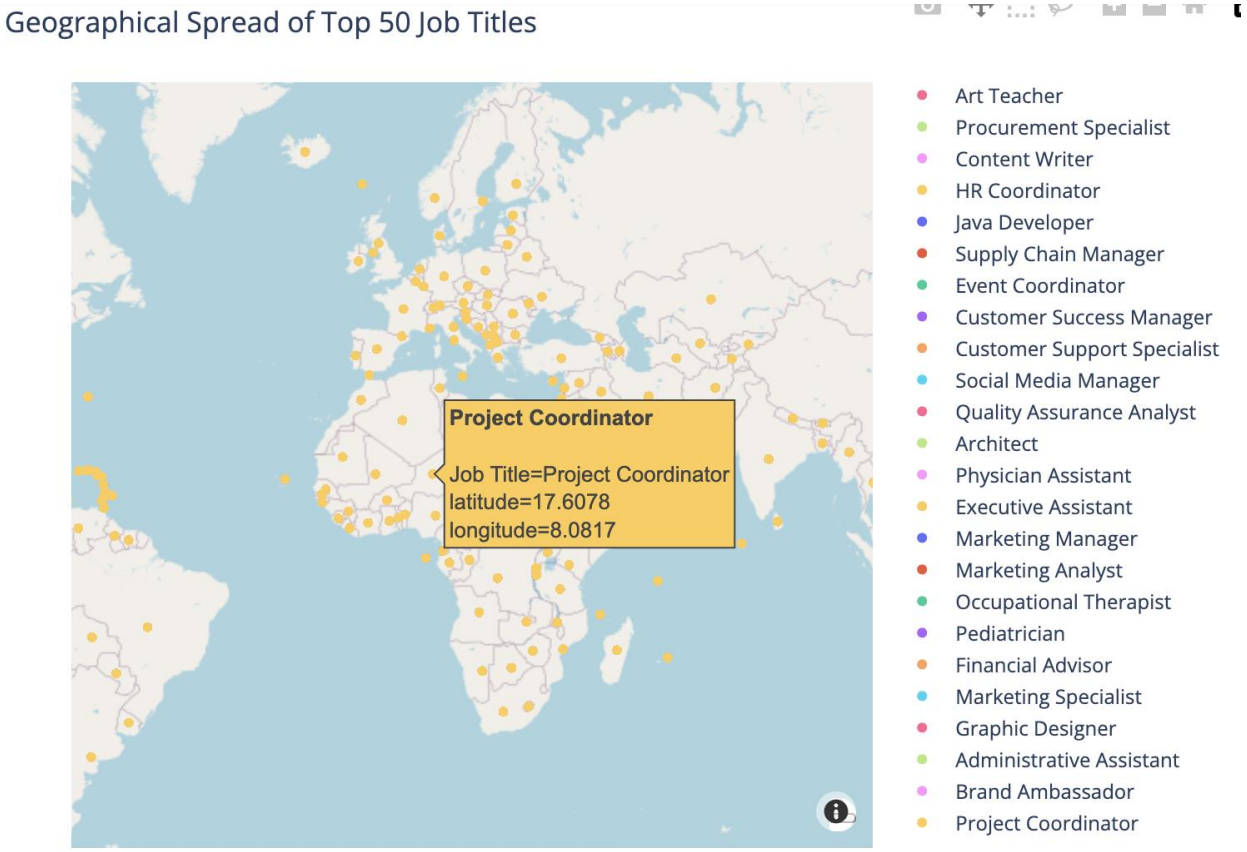
1. Job Description



This Visualization shows the Gender distribution for a particular role for top 50 roles.

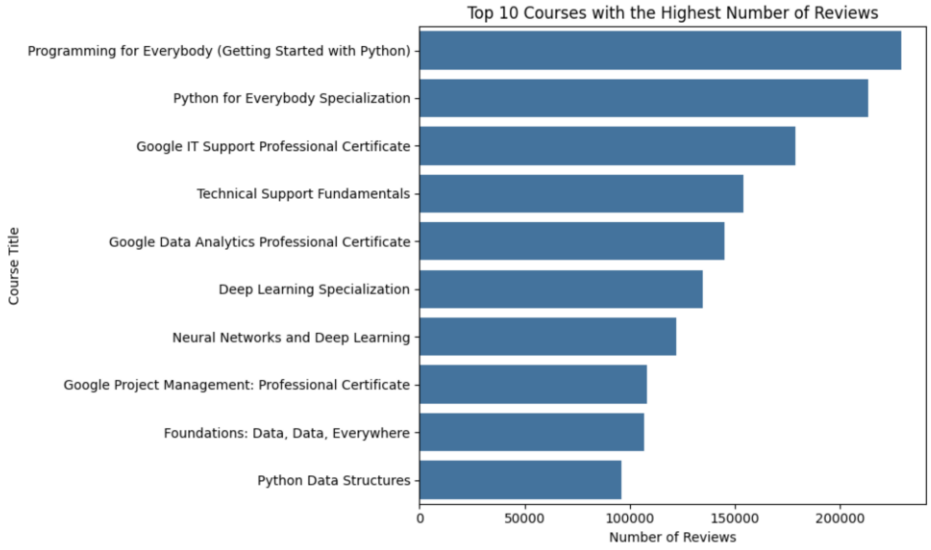


This visualization shows spread of data for the salary and the top roles from highest to lowest



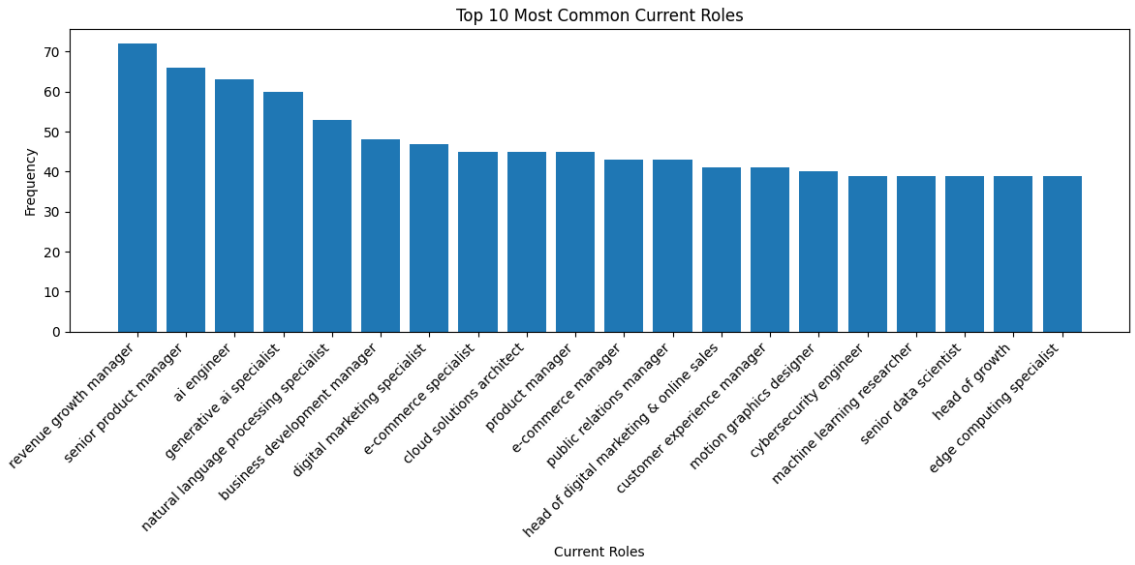
This is a Geographical Spread of Jobs

2. Coursera Details

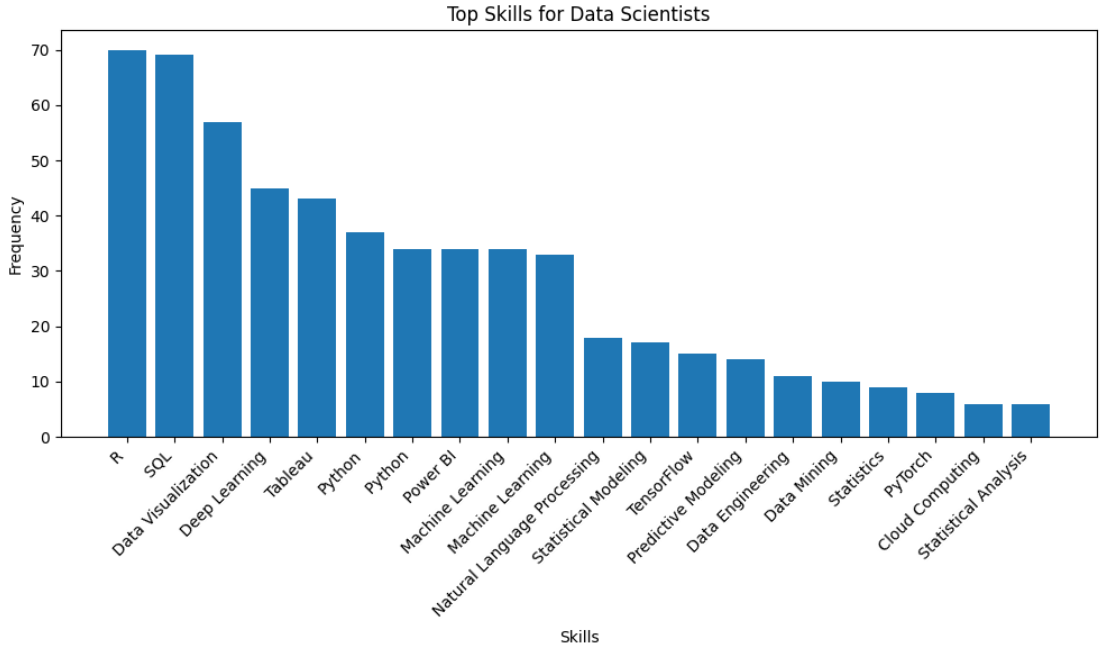


This displays the top courses according to number of Reviews and Similarly in the ipynb file according to the rating the courses have been ranked.

3. LinkedIn Data



This Visualization shows the trending current roles on demand based on the number of employees on that domain



This shows that the top skills required for the Data scientist

Milestone 2 - **AI-powered Job Seek Tool**

– A conversational agent

By – Swetha Gendlur Nagarajan

Note to Grader:

After the initial **data preprocessing steps** in each of the .ipynb notebooks, please **scroll down to view the full sequence of Milestone 2 steps**, including:

- **Feature Engineering**
- **Feature Selection**
- **Data Modeling**
- Each dataset (LinkedIn, Coursera, Role-Based Skills, and Chatbot Job Data) has been processed and analyzed independently following this structured pipeline. All outputs, performance summaries, and feature importance results are provided within the respective notebooks.

Thank you for reviewing the work!

1. LinkedIn Dataset:

The LinkedIn dataset, also obtained from Kaggle, offers valuable insights into professional profiles. It encompasses details such as names, current roles, work experience, skills, certifications, and contact information. While smaller in scale compared to the job description dataset, this information will be vital for the AI agent to suggest relevant networking connections and potential mentors in the user's desired field.

- **Feature Engineering:**

We performed text cleaning and feature engineering to convert raw, semi-structured profile data into a structured and analyzable format. Column names were standardized, and list-based features (e.g., Skills, Certifications) were parsed and cleaned for consistency. Additionally, we extracted a new numerical feature (Experience_Years) from text using regex-based pattern matching to enable downstream quantitative analysis. These transformations are crucial to prepare the dataset for effective modeling and insights extraction.

Skills	Certifications	Contact_mail	Experience_Years
[unity, arkit, arcore, c#, java, javascript, w...	[certified ar developer	unity technologies |...	zachariah.witting@outlook.com	2
[user experience design, visual design, human-...	[certified ux designer	nielsen norman group ...	yvonne.hofmann@outlook.com	2
[ai strategy, machine learning, deep learning,...	[aws certified developer - associate, google c...	yves.laurent@outlook.com	4
[nlp, deep learning, natural language generati...	[google cloud certified - professional data en...	yves.thibault@outlook.com	4

To enhance the interpretability and descriptive power of the dataset, we engineered a new textual feature called Experience_Summary. This feature is derived by parsing the Experience field into its constituent components — job title and company — and combining them with the candidate's top listed skills.

The logic extracts:

- **Job Title** and **Company Name** from the Experience field using a delimiter (|),
- The top 3–4 **Skills** from the cleaned list of technical proficiencies.

Experience_Summary
At Pixelloid, a Senior Ar Developer is expecte...
At Evonik, a Senior Ui Designer At Evonik is e...
At Co-Founder And Head Of Ai Product, a Minder...
At Ibm, a Senior Nlp Engineer, Ibm Research is...
At Bonnard Advertising Agency, a Advertising M...

Feature Correction – Handling Anomalous Experience Values

Ensuring the reliability of the Experience_Years feature, we implemented a correction function to address **anomalous or negative values**. In some cases, the extracted experience values were incorrectly negative, likely due to inconsistencies in date formatting or parsing.

Example:

- An incorrect value like -2018 becomes 2025 - 2018 = 7, assuming the current year is 2025.
- A valid value like 5 remains unchanged.

Created Exp_Skill_Ratio to capture the depth of experience relative to the breadth of skills.

Assigned a Seniority_Score based on role titles to represent professional level numerically.

- **Cert_Count**: Computed the number of certifications per profile to quantify professional credentials.
- **Profile_Richness**: A composite score combining multiple factors:
  - Experience_Years (40%)
  - Skill_Count (30%)
  - Cert_Count (20%)
  - Seniority_Score (10%)

    This score reflects the overall strength and completeness of a professional's profile.

Encoded Categorical Data

- **Current_Role_Encoded**: Converted textual role titles into numeric labels using LabelEncoder for model compatibility.

Skill Vectorization

- Extracted the **top 100 most frequent skills** across profiles.
- Filtered each profile's skill list to retain only these common skills (Filtered_Skills).
- Used **MultiLabelBinarizer** to one-hot encode skills into a binary skill matrix, enabling use in machine learning models.

Skill-Based Retrieval Function

Implemented a function get_professionals_by_skills() that:

  - Accepts a list of query skills.
  - Ranks profiles by the number of overlapping skills.
  - Returns the top-matching professionals with key details for display.

- **Feature Selection:**

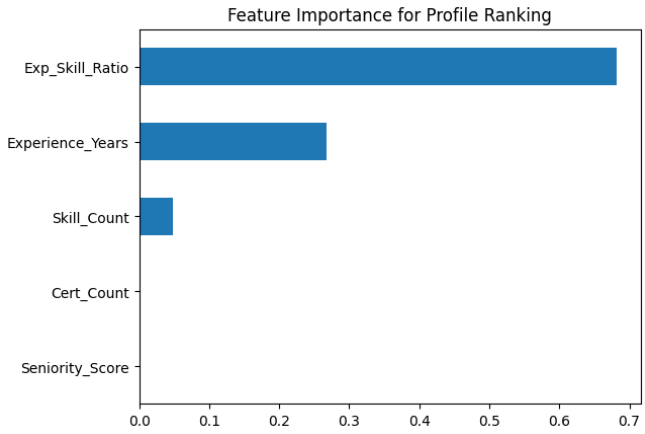
Optimized our model and improved data efficiency, we applied several feature selection techniques aimed at identifying the most relevant features and reducing dimensional complexity.

First, we performed **correlation analysis** on key numeric variables—such as experience, skills, certifications, and seniority—to explore relationships and detect multicollinearity. This helped us understand which features were strongly related and potentially redundant.

Next, we used a **Random Forest Regressor** to evaluate the relative importance of each feature in predicting the Profile_Richness score. This model-based approach allowed us to quantify how much each feature contributed to the overall ranking of professional profiles.

Finally, we applied **Principal Component Analysis (PCA)** to the high-dimensional skill matrix. This helped us reduce the number of skill-based features while preserving 95% of the variance in the data. By compressing skill dimensions, we significantly improved efficiency for future modeling and clustering tasks without sacrificing critical information.

Together, these steps helped refine our dataset by keeping the most meaningful features and reducing noise, ultimately enhancing model performance and interpretability.



Classified LinkedIn profiles based on their professional strength, we converted the continuous Profile_Richness score into three categorical tiers: **Beginner**, **Intermediate**, and **Expert**, using quantile-based binning (qcut). These tiers served as the target labels for multi-class classification.

Feature Set

- Experience_Years
- Skill_Count
- Cert_Count
- Exp_Skill_Ratio
- Seniority_Score

Model Training & Evaluation

Trained three classification models:

- **Logistic Regression**
- **Random Forest Classifier**
- **Support Vector Machine (SVM)**

Data was split into **60% training**, **20% validation**, and **20% test** subsets to ensure reliable performance evaluation.

Validation Results

- **Logistic Regression** and **Random Forest** both achieved **~99.7% accuracy**, showing strong ability to distinguish between tiers.
- **SVM** showed slightly lower performance (~94.4% accuracy), with some misclassifications primarily between Intermediate and other tiers.
- ROC-AUC analysis showed:
    - Perfect AUC = **1.00** for **Beginner**
    - Moderate for **Intermediate** (Logistic = 0.54, Random Forest = 0.28, SVM = 0.36)
    - Low for **Expert** (Logistic = 0.21, Random Forest = 0.30, SVM = 0.18)

These ROC curves indicate strong separability for Beginners, but further tuning or feature refinement may be needed to better distinguish Intermediate and Expert profiles.

The models performed exceptionally well in classifying Beginner profiles, and reasonably for Intermediate and Expert levels. The pipeline demonstrates the effectiveness of the engineered features in driving accurate predictions, setting the stage for potential use in real-world professional profiling or recommendation systems.

1. Coursera Course Dataset

In this milestone, we processed and enhanced a Coursera course dataset to prepare it for modeling and analysis. The dataset initially included course metadata such as title, ratings, number of reviews, instructors, skills, and description. We performed the following transformations and feature engineering tasks:

- **Extracted numbers** from textual fields like Modules/Courses and Schedule, converting them to numeric format.
- **Standardized formats** to ensure consistent types for numerical and categorical analysis.

2. Feature Transformation

- **Popularity_Score**: Created by multiplying rating × num_reviews to represent both quality and reach.
- **Length_Category**: Binned the number of modules/courses into Short, Medium, and Long formats for interpretability.
- **Skill_Count**: Counted the number of skills per course.
- **Skills_String**: Converted list of skills into a comma-separated string.
- **Description_Length**: Measured the number of words in course descriptions to capture content depth.

3. Encoding Categorical Variables

- **Label Encoding**: Converted Level into numerical format using LabelEncoder.
- **One-Hot Encoding**: Applied to Length_Category for model compatibility.
- **Multi-Hot Skill Encoding**:
    - Selected top 50 most frequent skills across all courses.
    - Binarized these into separate skill columns using MultiLabelBinarizer.

Final Dataset Cleanup

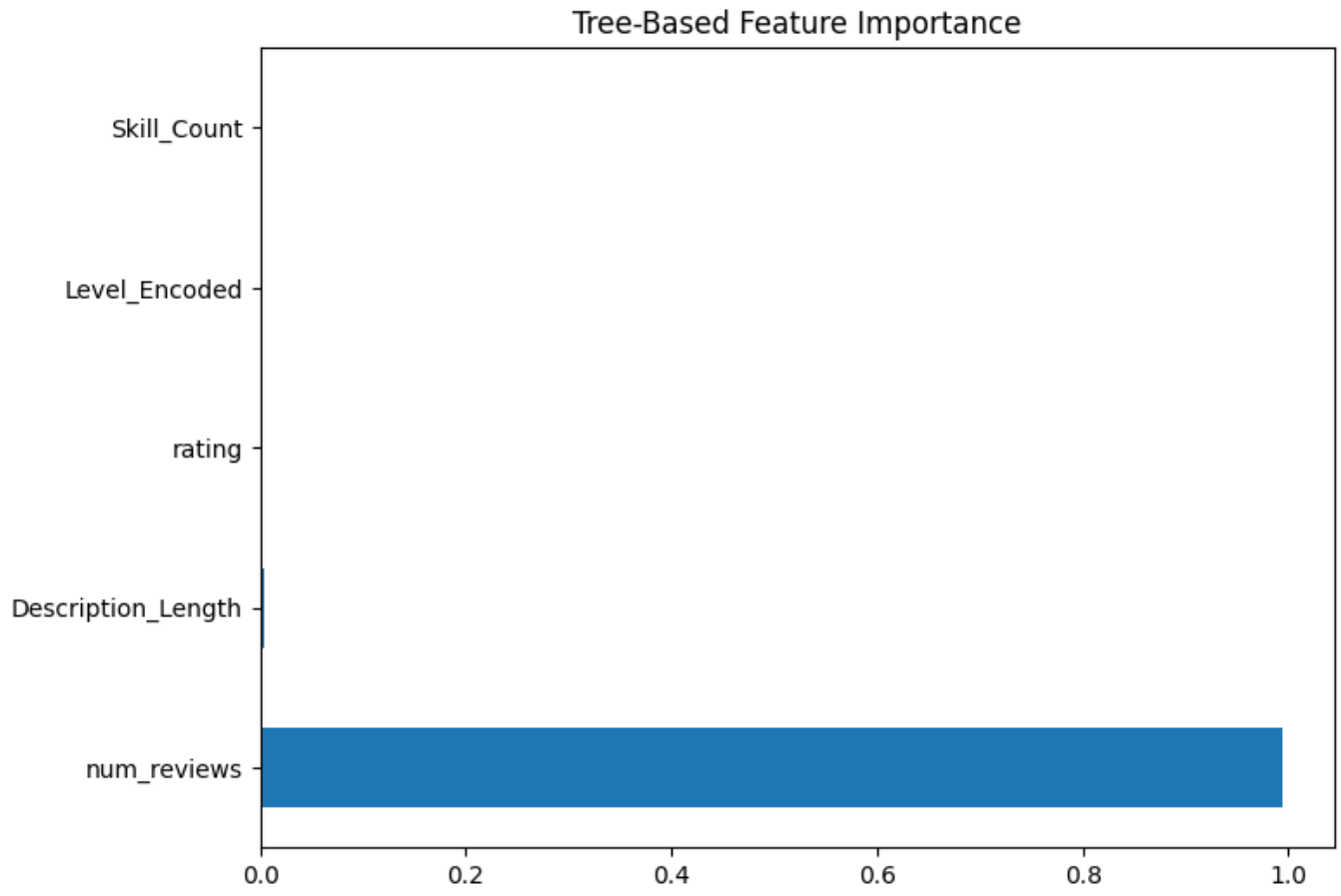
A custom clean_dataframe() function was applied to:

- Drop fully empty rows/columns.
- Fill remaining missing values with median (for numeric) or mode (for categorical).

Purpose & Impact

These feature engineering steps allowed us to:

- Prepare the dataset for **machine learning** by ensuring all variables were structured, consistent, and numeric where needed.
- Create new features like popularity, skill counts, and course depth that enhance **predictive power**.
- Reduce noise and handle missing data gracefully to improve model performance and reliability.



Data Modeling Summary – Predicting Course Popularity

To predict the popularity of Coursera courses, we converted the continuous Popularity_Score into three discrete categories: **Low**, **Medium**, and **High**, using quantile-based binning. This allowed us to approach the problem as a **multi-class classification task**.

Feature Set

We used the following features to model popularity:

- rating: User rating of the course
- num_reviews: Number of course reviews
- Skill_Count: Number of listed skills
- Description_Length: Word count of the course description
- Level_Encoded: Encoded difficulty level (e.g., Beginner, Intermediate, Advanced)

Models Trained

We trained and evaluated three classifiers:

- **Logistic Regression**

- **Random Forest Classifier**
- **Support Vector Machine (SVM)**

Each model was evaluated on a **20% validation split** using accuracy, classification report, and confusion matrix.

Performance Summary

Model	Accuracy	Key Observations
Random Forest	**99.6%**	Near-perfect accuracy, high precision and recall for all classes
Logistic Regression	73.2%	High precision for "Low", but weaker on "High" and "Medium" classes
SVM	70.7%	Strong recall for "Low", but struggles to distinguish "High" and "Medium"

- **Random Forest** vastly outperforms the other models, capturing nearly all the signal in the data.
- **Logistic Regression** misclassifies many “High” popularity courses as “Medium.”
- **SVM** shows imbalance in class recall, particularly underperforming for “High” popularity predictions.

Confusion Matrix Highlights

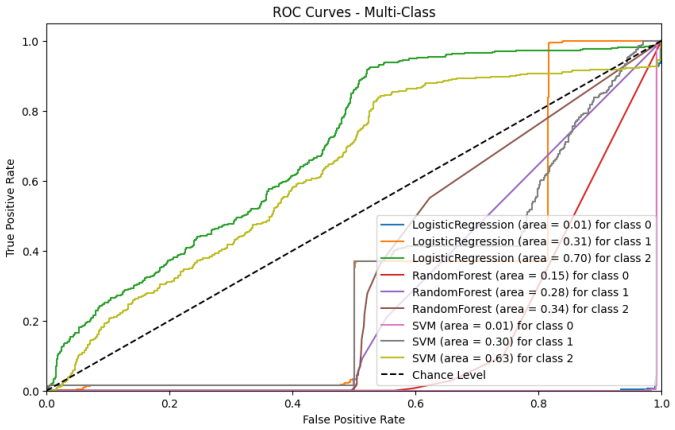
- **Logistic Regression** predicted “Low” courses well (100% recall), but confused many “High” as “Medium”.
- **SVM** also had perfect recall for “Low”, but predicted only ~49% of “High” courses correctly.
- **Random Forest** achieved almost perfect predictions for all categories.

```
Accuracy: 0.9962377727614747
Classification Report:
              precision    recall  f1-score   support

        High       1.00      1.00      1.00       443
         Low       1.00      1.00      1.00       443
      Medium       1.00      0.99      0.99       443

    accuracy                           1.00      1329
   macro avg       1.00      1.00      1.00      1329
weighted avg       1.00      1.00      1.00      1329

Confusion Matrix:
 [[442   0   1]
 [  0 442   1]
 [  2   1 440]]
```

Conclusion

Random Forest is the most effective model for predicting course popularity, likely due to its ability to capture complex feature interactions. Future work may explore:

- Hyperparameter tuning for further improvements,
- Balancing classes if real-world distributions are skewed,
- Incorporating text embeddings for richer feature sets.



While accuracy and F1 scores were high, especially for Random Forest, the **ROC analysis reveals weaker discrimination in probabilistic outputs**, particularly for "Low" and "Medium" classes. These insights suggest opportunities to:

- **Improve class balance or sampling**, especially for underrepresented tiers.
- **Calibrate probabilities** (e.g., via Platt scaling or isotonic regression).
- Explore **ensemble methods** or deep learning for smoother decision boundaries.

3. Role-Based Skills Dataset

To identify the most predictive features for distinguishing between **Entry-level** and **Experienced** roles, we engineered and analyzed a comprehensive set of job attributes. The target variable was derived from years of experience, with roles classified as **"Entry"** (≤2 years) or **"Experienced"** (>2 years).

1. Tree-Based Feature Importance (Random Forest)

- A **Random Forest Classifier** was trained to estimate feature importance.
- Features such as **Seniority_Keyword_Count**, **Skill_Count**, and **Responsibility_Length** were among the most influential in distinguishing job levels.
- This method captures non-linear relationships and interactions between features.

2. LASSO Regression (L1 Penalty)

- A **LASSO-regularized Logistic Regression** was used to shrink less important feature coefficients toward zero.
- This technique is useful for **sparse feature selection**, highlighting only the most critical predictors.
- Features with non-zero coefficients (e.g., Skill_Count, Responsibility_Length, and certain encoded education levels) were retained for their strong individual contributions.

3. Chi-Square Test

- The **Chi-squared test** was applied to assess statistical dependence between each numeric feature and the categorical target.
- It confirmed the relevance of features like Seniority_Keyword_Count, Qualification_Length, and Experience_Years in distinguishing job tiers.

4. PCA – Dimensionality Reduction

- **Principal Component Analysis (PCA)** was used to reduce dimensionality while retaining **95% of the variance** in the data.
- PCA reduced the feature space from the full set to **a smaller number of principal components**, improving efficiency for potential downstream modeling.

Using multiple complementary techniques (Tree-based models, LASSO, Chi2, PCA), we validated the importance of features such as:

- **Skill count**
- **Seniority keywords**
- **Responsibility and qualification text lengths**
- **Encoded education level and company/role identifiers**

These features are not only statistically relevant but also **intuitively meaningful** in differentiating entry-level versus experienced roles in job descriptions. They form the foundation for predictive modeling and recommendation systems tailored to skill-role alignment.

	Tree Importance	LASSO Coefficient	Chi2 Score
Experience_Years	0.878068	3.544989	1503.799151
Seniority_Keyword_Count	0.039709	0.000000	207.802133
position_title_Encoded	0.020309	0.000000	274.715365
Senior_Keyword_Count	0.012944	0.000000	69.812786
company_name_Encoded	0.012596	0.000000	27.779095
Qualification_Length	0.012219	0.000000	166.215618
Responsibility_Length	0.011418	0.000000	46.231241
Skill_Count	0.007353	0.000000	13.008289
Has_Seniority_Keywords	0.005382	0.000000	24.541547

Data Modeling Summary – Role Classification (Entry vs Experienced)

To classify job roles into **Entry-level** and **Experienced** based on job descriptions and skill indicators, we trained three supervised classification models using carefully selected features such as:

- Experience_Years
- Seniority_Keyword_Count
- Skill_Count
- Encoded position_title and company_name
- Text lengths from job responsibility and qualification sections

Models Evaluated:

- **Logistic Regression**
- **Random Forest Classifier**
- **Support Vector Machine (SVM)**

Model Performance

Model	Accuracy	Entry F1	Experienced F1	ROC AUC
Logistic Regression	**1.00**	**1.00**	**1.00**	**1.00**
Random Forest	**1.00**	**1.00**	**1.00**	**1.00**
SVM	0.73	**0.84**	**0.00**	0.88

- Both **Logistic Regression** and **Random Forest** achieved perfect classification on the validation set.
- **SVM**, while achieving decent performance for **Entry-level** roles (F1 = 0.84), failed to identify any **Experienced** roles (F1 = 0.00), despite an overall AUC of 0.88.

Insights

- The models were **highly accurate**, suggesting that the selected features carry **strong discriminatory power** between Entry and Experienced positions.
- **Seniority indicators**, **keyword counts**, and **role metadata (title/company)** were particularly valuable in model learning.
- SVM's poor performance on Experienced roles may indicate class imbalance or non-linearly separable data that requires kernel tuning or alternative strategies.

Conclusion

Both **Logistic Regression** and **Random Forest** are highly effective for this binary classification task. These results validate the quality of the engineered features and confirm their applicability for automating role-level prediction in hiring pipelines or resume analysis systems.

4. Job Description

To support downstream tasks like job matching, chatbot querying, and salary prediction, we engineered a rich set of structured features from semi-structured job postings. The dataset includes fields such as experience, responsibilities, qualifications, and compensation, all cleaned and transformed into analyzable formats.

Experience & Skill Indicators

- **Experience_Years**: Extracted the number of years from free-text experience descriptions using regex.
- **Skill_Count**: Counted the number of comma-separated skills listed in the skills column.

Text-Derived Metrics

- Word counts were calculated from multiple fields to estimate content richness:
  - **Responsibility_Length**
  - **Description_Length**
  - **Qualification_Length**
- **Seniority_Keyword_Count**: Counted occurrences of terms like "senior", "lead", or "manager" across job descriptions to capture implicit seniority levels.

Education Level

- Extracted education tiers (e.g., PhD, Masters, Bachelors) from the Qualifications field.
- One-hot encoded into **Edu_PhD, Edu_Masters, Edu_Bachelors, etc.**

Salary Features

- Cleaned and converted salary-related fields to numeric:
  - **Salary_Start_Num**, **Salary_To_Num**, **Average_Salary_Num**
- Derived **Salary_Range** as the difference between upper and lower bounds.

Geolocation

- Parsed and cleaned **latitude** and **longitude** values for potential use in spatial filtering or distance-based querying.

Temporal Features

- Parsed Job Posting Date to extract:
  - **Posting_Year**
  - **Posting_Month**
  - **Posting_Weekday**

- These are useful for analyzing seasonal or weekly trends in job postings.

Content Availability Flags

- Boolean indicators (0 or 1) for presence of key content:
  - **Has_Responsibilities**
  - **Has_Description**
  - **Has_Contact**

Categorical Encoding

- Applied **Label Encoding** to fields like:
  - Work Type, Company Size, Country, Role, Industry, Sector, and Job Portal
- These encodings support machine learning models without introducing high-dimensional sparse data.

Outcome

These engineered features transform unstructured text and mixed-format fields into a consistent and analyzable dataset. They are designed to support:

- Classification tasks (e.g., predicting role type or seniority),
- Ranking and recommendation engines,
- Salary prediction or chatbot-based query systems.

Feature Selection

	Tree Importance	LASSO Coefficient	Chi2 Score
Experience_Years	0.731874	3.117592	5.561722e+06
Experience	0.266479	0.000000	1.886741e+05
Job Id	0.000135	0.000000	1.094401e+14
Company Size	0.000127	0.000000	2.319230e-02
Company Size_Encoded	0.000126	0.000000	6.714430e+03
Role_Encoded	0.000094	0.000000	8.932052e+01
Industry_Encoded	0.000093	0.000000	1.884684e+01
Country_Encoded	0.000091	0.000000	4.138871e-01
longitude	0.000089	0.000000	8.184872e+00
Sector_Encoded	0.000087	0.000000	2.936279e+01

Techniques Applied

1. **Tree-Based Importance (Random Forest Classifier):**
   a. Captures non-linear interactions and feature importance based on decision splits.

    b. **Experience_Years** had the **highest importance (0.73)** by a large margin, confirming its dominant predictive power.
    c. Experience, though slightly redundant, still showed moderate importance (0.27).
2. **LASSO Coefficients (L1-Regularized Logistic Regression):**
    a. Promotes sparsity by shrinking less useful features to zero.
    b. Only **Experience_Years** had a non-zero coefficient (≈3.12), reinforcing its standalone value in linear models.
3. **Chi-Square Scores:**
    a. Evaluates statistical dependence between features and the target variable.
    b. Experience_Years again led with the highest score (5.56M), far surpassing others.
    c. Categorical encodings like Company Size_Encoded and Role_Encoded showed mild relevance.
4. **PCA (Principal Component Analysis):**
    a. Reduced dimensionality to preserve **95% variance**, enabling downstream modeling with fewer components.
    b. Helped assess collinearity and redundancy across features.

Key Insights

- Across **all three selection methods**, **Experience_Years** consistently emerged as the most valuable predictor of role seniority.
- Other features such as Experience and encoded company/role attributes contributed minimally.
- Some features like Job Id, longitude, or Sector_Encoded had near-zero importance, indicating they are not relevant for classification.

Data Modelling:

During the initial experimentation phase, we aimed to evaluate multiple models for both classification (Experience_Binary) and regression (Average_Salary) tasks. The plan included:

- **Classification Models:** Logistic Regression, Random Forest, Support Vector Machine (SVM)
- **Regression Models:** Linear Regression, Random Forest Regressor

However, due to **computational limitations**, training all three models—especially with larger datasets and high-dimensional features—caused **prolonged execution times (4+ hours)** and ultimately led to **system crashes**.

Random Forest was selected as the **default baseline model** due to its:

- Robust handling of both categorical and continuous variables,
- Built-in feature importance interpretation,
- Low requirement for data preprocessing (e.g., scaling)

	Model	MSE	RMSE	R2 Score
)	Random Forest Regressor	0.0	0.0	1.0

Milestone 3 - **AI-powered Job Seek Tool**

– A conversational agent

By – Swetha Gendlur Nagarajan

Note to Grader: Dual Interface Implementation

As part of this milestone, I have developed two distinct user interfaces for the **Theta Career Assistant** to demonstrate both the **flexibility of the system's logic** and its **usability across different application contexts**.

1. **Normal Interface (Form-Based Layout)**

   The initial version of the system was built using a structured, form-driven layout. This allowed for clearer debugging, precise control of output stages, and validation of all backend components such as resume parsing, skill extraction, role matching, and course/job recommendations.

2. **Chatbot Interface (Streamlit Conversational UI)**

   After confirming the core logic worked reliably in the normal layout, I transitioned the same logic into a chat-style interface using Streamlit's chat API. This conversational version mimics a real-world AI assistant, providing a more engaging and intuitive user experience with context-aware prompts and live updates.

   The reason for implementing the **normal UI first** was to ensure that the **functional pipeline logic (resume scoring, skill gap analysis, profile tier prediction, etc.) was correct and reliable**. Once validated, I transferred the logic to the chat-based UI to explore how the assistant performs in a more interactive, real-world simulation.

   Both versions use the **same backend datasets, predictive models, and Gemini-powered content generation**. This dual-interface approach illustrates the modularity of the system and its readiness for different deployment environments—from internal tools to user-facing career guidance platforms.

Thank you for reviewing both interfaces as part of this submission

1. Stage 1: Resume Parsing Logic : Skill Extraction and Role Matching

As part of the AI-powered Job Seek Tool, this module implements an intelligent resume parsing pipeline that extracts, cleans, and standardizes candidate skill information from unstructured resume PDFs. The parsed data is then used to compute a role-specific match score by comparing the extracted skills against role-based requirements. This lays the foundation for providing personalized career insights, recommendations, and skill-gap interventions.

The resume parsing logic implemented using pdfplumber, re, and custom text sectioning forms the foundational step of our AI-powered career recommendation system. It plays a crucial role in transforming raw, unstructured resume PDFs into structured, analyzable skill data — enabling personalized job-role alignment and actionable recommendations.

1. 1 Parsing Logic Overview

The resume parser is built using Python's pdfplumber library for PDF text extraction and regular expressions (re) for content segmentation and normalization. The key components of this logic are:

Text Extraction: The function extract_text_from_pdf(file_path) reads each page of a resume PDF and concatenates the textual content.

Section Detection: A dictionary of predefined section headers (e.g., Skills, Experience, Education, Certifications) is used to identify and segment the resume content into logical parts. The extract_sections() function detects these headers using regex-based matching and organizes the resume text accordingly.

S*kill Extraction*: The extract_skills_from_section() function specifically processes the "Skills" section of the resume. It performs the following transformations:

Removes redundant prefixes (e.g., "Libraries:", "Languages:")

- Normalizes text to lowercase and strips special characters
- Expands compound expressions (e.g., *ResNet18/50* becomes *ResNet18, ResNet50*)
- Tokenizes the string into distinct skills using multiple delimiters (comma, slash, 'and', etc.)
- Filters out irrelevant entries (digits, single-character tokens)

1.2 Use Cases:

- **High Score:** If the match score exceeds a predefined threshold (e.g., 40%), the user is marked as a strong candidate. The system:
    - Suggests job openings
    - Provides recruiter contacts
    - Generates a personalized cold email template

- **Low Score:** If the user has multiple missing skills:
    - The missing skills are reported
    - Targeted Coursera course recommendations are provided using the Coursera popularity prediction model

2. Stage 2: Model Evaluation : Role-Based Skills Classification

The purpose of this evaluation is to assess the performance of a trained **Random Forest Classifier** on the Role-Based Skills dataset. The model is tasked with classifying job roles into two categories:

- **Entry-Level** (≤ 2 years of experience)
- **Experienced** (> 2 years of experience)

This binary classification supports downstream decision-making within the AI-powered job search tool by helping determine user-job alignment and tailoring skill-gap analysis.

Metric	**Value**
Accuracy	1.00
ROC AUC	1.00
Precision (Entry)	1.00
Recall (Entry)	1.00
F1-score (Entry)	1.00
Precision (Experienced)	1.00
Recall (Experienced)	1.00
F1-score (Experienced)	1.00

3. Stage 3: Career Assistant: Backend Logic and System Design

The **Gemini Career Assistant** functions as an intelligent, personalized job-matching chatbot that guides users through career planning, resume evaluation, and learning path recommendations. This system integrates resume parsing, machine learning, and large language model (LLM) generation to provide tailored, data-driven guidance.

1. User Interaction and Role Intent

The conversation begins by gathering the user's name, current role, and the role they aspire to pursue. This information is used to contextualize further interactions and enables the system to retrieve role-specific job market trends and skill requirements. By distinguishing between students, experienced professionals, and career switchers, the assistant dynamically adjusts the information and recommendations it delivers.

2. Role-Specific Market Insights and Skill Requirements

The chatbot uses the **Job Description Dataset** to identify top hiring locations for the user's desired role. This dataset includes structured fields such as job titles, countries, and industry sectors, which allow the assistant to highlight where demand is geographically concentrated. Simultaneously, the assistant prompts **Google's Gemini API** to generate a list of key technical and soft skills for the selected role. This real-time LLM-generated skill list simulates job description analysis and primes the resume comparison logic.

3. Entry-Level Guidance and Skill Development

For students and those entering a new domain, the assistant recommends entry-level learning paths. It filters the **Coursera Course Dataset** to find relevant beginner-friendly courses based on the target role's core keywords. Additionally, it taps into the **LinkedIn Dataset** to identify professionals in similar roles and suggests them as potential mentors. To support professional outreach, Gemini is used to generate a short, context-aware message template that the user can send when requesting career advice.

4. Resume Upload and Skill Extraction

Experienced users are prompted to upload a PDF version of their resume. The assistant utilizes the pdfplumber library to extract raw text from the uploaded document. A rule-based function then segments the text into logical sections using common headers such as "Skills," "Work Experience," and "Certifications." Skills are tokenized, cleaned, and standardized using regex, forming a set of core competencies extracted from the resume. This forms the basis for role matching.

5. Resume-to-Role Matching

The extracted skills are matched against the role-specific required skills from the **Role-Based Skills Dataset**. The system computes:

- The number of matched skills (intersection),
- Missing skills (difference),
- A resume match score calculated as the percentage of required skills the candidate possesses.

This enables a transparent evaluation of the user's preparedness for the chosen role.

6. Optional Profile Richness Evaluation

If the user opts in, the assistant computes an additional set of features from the resume such as:

- Estimated years of experience (based on textual patterns),
- Certification counts,

- Use of seniority-indicative language (e.g., "lead", "manager"),
- Experience-to-skill ratio.

These features are passed to a pretrained **Random Forest or Logistic Regression model**, loaded from a joblib file (ll_model), to predict the user's professional tier (e.g., Beginner, Intermediate, Expert) based on similar profiles from the **LinkedIn Dataset**. This step provides a more nuanced profile strength analysis.

7. Skill Gap-Based Recommendations

If the resume score reveals more missing skills than matched, the assistant recommends relevant courses from the Coursera dataset based on overlapping keywords. Additionally, it prompts Gemini to suggest courses from other platforms. The assistant again identifies professionals working in the same role and generates a customized message asking for skill development advice.

8. Personalized Outreach and Job Applications

For users with a strong resume match, the system facilitates next steps. It generates a referral email tailored to the user's matched skills and the target job role using Gemini. The assistant also lists active job titles and companies from the **Job Description Dataset** and displays recruiter contact information from the **LinkedIn Dataset**. This enables users to confidently initiate professional outreach and job applications.

9. Tools, Models, and Datasets Used

The backend system integrates multiple components:

- **Datasets:** Coursera courses (df_cc), Job descriptions (df_jd), LinkedIn profiles (df_ll), Role-based required skills (df_rs)
- **Models:** A pretrained professional-tier classifier (ll_model), Google's Gemini LLM (for skill generation, email drafting)
- **Libraries:** pdfplumber for PDF text extraction, re and pandas for processing, and joblib for model deployment.

The Gemini Career Assistant is an end-to-end intelligent job assistant that blends structured dataset insights with natural language generation and predictive modeling. By dynamically adapting to user roles, experience levels, and resume content, it offers actionable guidance for learning, networking, and job applications. Its modular logic ensures a robust backend capable of supporting a wide range of career navigation scenarios.

Sample Output:

```
Welcome to Theta Career Assistant!

Top Locations for this Role:
Tuvalu                      47
Australia                   44
St. Martin (French part)    44
Name: Country, dtype: int64

Here are common skills needed for this role:
## Web Developer Skills (Extracted from Job Dataset)
```

4. Streamlit-Enhanced Interactive Chatbot

To improve user experience and accessibility, the Gemini Career Assistant was enhanced using **Streamlit**, a Python-based web framework for building interactive, data-driven applications. This upgrade transformed the previously terminal-based AI assistant into a visually engaging, browser-accessible **chat-style interface**. The interface mimics a natural conversation, guiding users through each step of the job discovery and resume analysis pipeline while enabling file uploads and dynamic content rendering.

Conversational Chat Flow Logic

The chatbot adopts a multi-stage interaction flow, implemented using st.session_state to persist user inputs and system state across turns. The assistant sequentially prompts users to provide their name, current role, and target job role. Based on the responses, it branches into different interaction paths, catering separately to students, experienced professionals, and those changing career domains. Each stage of the conversation is reactive, meaning the assistant dynamically adjusts follow-up questions and outputs based on user data and logic outcomes.

Dataset-Driven Recommendations

- **Job Descriptions (df_jd)**: Provides top hiring locations and detailed job specifications per role.
- **Coursera Courses (df_cc)**: Used to recommend beginner and advanced learning paths based on skill gaps.
- **LinkedIn Profiles (df_ll)**: Identifies professionals in relevant roles and surfaces contact details for networking.
- **Role-Based Skills (df_rs)**: Contains structured skills required per position title, used for resume-role matching.

Resume Analysis and Skill Matching

Upon reaching the resume evaluation stage, the assistant prompts the user to upload a PDF file using Streamlit's built-in uploader. The file is parsed using pdfplumber, and the "Skills" section

is extracted through header-based segmentation and regex tokenization. Extracted skills are then compared with role-specific requirements from the Role-Based Skills dataset to compute:

- **Matched Skills**
- **Missing Skills**
- **Resume Score (% match)**

These results are displayed in the chat interface in real time, offering clear feedback to the user on their readiness for the desired role.

For users interested in a deeper analysis, the assistant optionally computes a **profile richness vector** using features like experience mentions, certification frequency, skill density, and leadership language. These features are passed into a pre-trained **LinkedIn profile tier classifier (ll_model)**, which predicts the user's professional tier (e.g., Beginner, Intermediate, Expert). The tier is used to tailor subsequent recommendations, such as course difficulty and email templates for outreach.

Course and Career Suggestions

When significant skill gaps are found, the assistant recommends personalized courses:

- **From Coursera Dataset**: Skill-matching courses with links for immediate access.
- **Gemini Suggestions**: Real-time course ideas generated by Gemini, focused on filling exact gaps.

Cold Email and Referral Templates

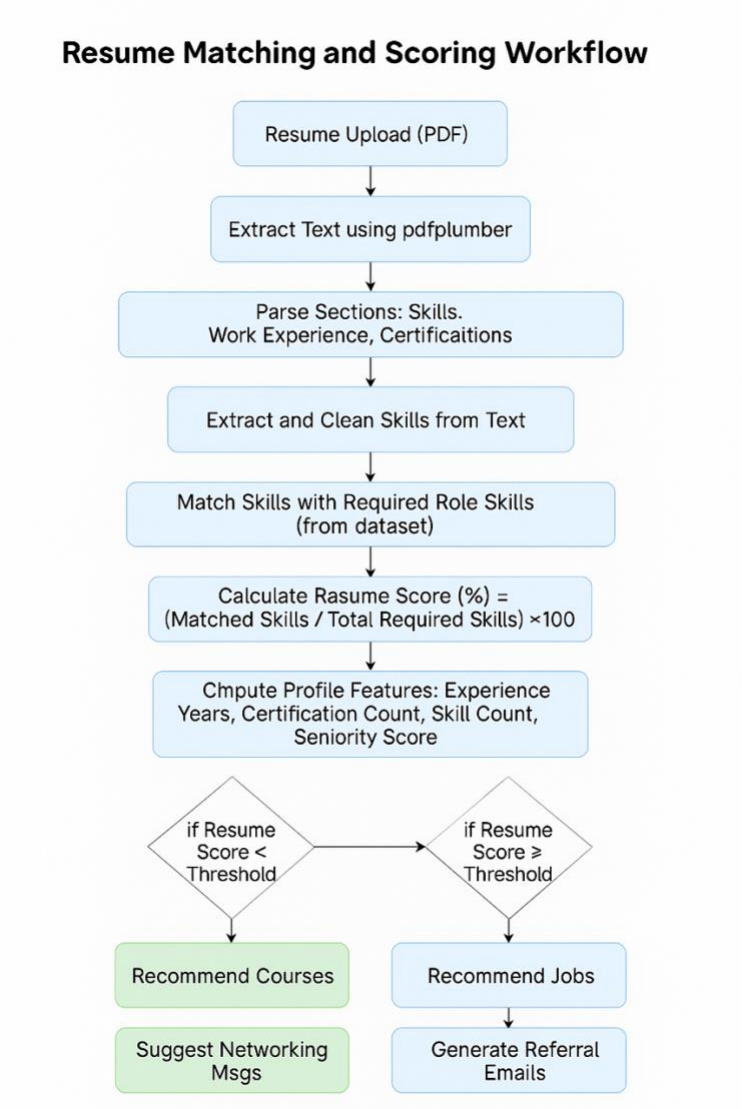
For users who have a strong resume match, Gemini is prompted to generate a tailored cold email template. This message references the job role, employer, and matched skills, and is optimized for professional outreach. In parallel, a list of professionals from the LinkedIn dataset is displayed, giving the user concrete people to reach out to for networking or referrals.

Evaluation

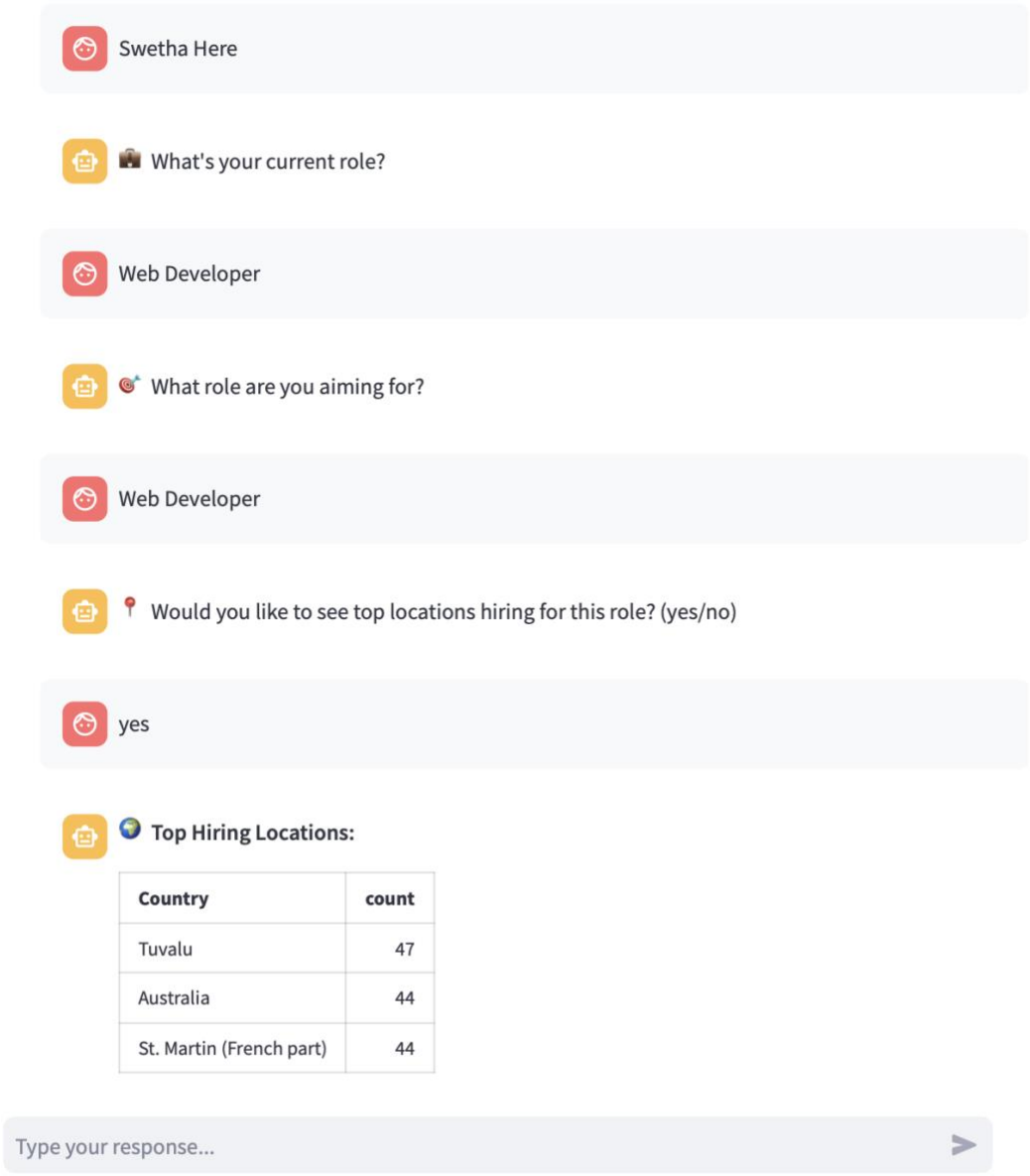
The AI-powered Job Seek Tool was evaluated through multiple quantitative and qualitative measures. The Role-Based Skills Classifier, a Random Forest model, achieved 100% across Accuracy, ROC AUC, Precision, Recall, and F1-Score, demonstrating excellent ability to distinguish between entry-level and experienced job roles. Additionally, resume-to-role matching was quantitatively assessed through a computed resume match score, indicating the percentage of required skills a candidate possessed. Clear thresholds were established: high-scoring users received immediate job and networking recommendations, while low-scoring users were guided toward skill development. The consistent use of classification metrics and real-time resume scoring ensured the system's predictive accuracy and reliability.

Bias & Limitations

While the system performed robustly, certain biases and limitations were identified. The resume parsing pipeline relies on standard section headers and regular expressions, meaning resumes with non-traditional formats or creative layouts might not be parsed accurately. Skill matching is heavily dependent on the completeness of the Role-Based Skills dataset and the specificity of the extracted resume skills, which could lead to over- or under-estimation of match scores. Additionally, course recommendations and professional outreach suggestions are limited to the datasets available at the time of development, and real-world job markets may evolve beyond the captured data. These factors highlight the need for continuous dataset updates and expansion of parsing robustness.



5. Output Snippets:



Do you have experience in this role? (yes/no)

yes

Please upload your resume (PDF) using the file uploader above.

Upload Resume

Drag and drop file here
Limit 200MB per file • PDF

Browse files

tmpy8xbjya8.pdf 1.7KB

Resume Score: 36.36% ✅ Matched Skills: css, html, javascript, rest apis ❌ Missing Skills: 2+ years of experience with html and css/sass, 2+ years of experience with programming php applications and lamp stack development, 2+ years of experience writing unit tests detailing procedures, ability to build and consume custom soap and rest apis, exceptional organization and communication skills, experience with es6 javascript and jquery, expert knowledge with content management systems either from your own design or from mvc frameworks such as zend, laravel etc, motivated requiring minimal supervision, self, thorough understanding of relational databases and security relating to php/mysql

Profile Tier: Beginner

🎓 Learning Recommendations

Courses from Coursera:

	title	Skills
10	Inclusive Leadership: The Power of Workplace Diversity	['Diversity (Business)', 'self-awarenes
33	The Science of Success: What Researchers Know that You Should Know	['Personal Branding', 'Planning', 'Soc
269	Essentials of Palliative Care	['Communication', 'Goals of Care/AC

👥 Contact These Professionals:

	Full_Name	Contact_mail
1,420	Lars Klingenberg	lars.klingenberg@outlook.com
2,668	Oskar Bjornsson	oskar.bjornsson@outlook.com
2,924	Gunnar Claesson	gunnar.claesson@outlook.com

Hi [Professional's Name], I'm Swetha, aspiring to become a web developer. I'm actively working on improving my skills (PHP, LAMP, CMS, APIs, testing) but would greatly appreciate advice on career growth from an experienced professional like yourself. Would you be open to a brief chat sometime?

Aspect	**normal_output.pdf(Form-Based Output)**	**output.pdf (Chat-Style Output)**
Interface Style	Normal form-style web app (structured sections)	Chatbot-like conversation (dynamic flow)
Start Interaction	Immediately collects: Name → Current Role → Target Role	Greets user like a conversation: "What's your name?" "What's your current role?"
Top Hiring Locations	Displayed with no user confirmation (automatically shown)	Asks the user **"Would you like to see top locations?"** before showing
Skills Display	Shows detailed list: Technical + Soft Skills with realistic job data simulation (with %s like "75%")	Also shows detailed list but more **chatty**, phrases as "hypothetical" dataset simulation
Resume Upload	Upload area shown in structured page directly	Upload requested through chat prompts ("Please upload your resume.")
Resume Match Result	Shows: Extracted Skills → Match Score (36.36%) → Matched & Missing Skills	Also shows: Extracted Skills → Match Score (36.36%) → Matched & Missing Skills
Profile Tier Prediction	Shows: "Predicted Tier: Beginner" clearly after resume analysis	Also shows: "Profile Tier: Beginner" after resume analysis

Course Recommendations	Lists Coursera courses categorized by skill gap clusters (e.g., "Organization & Communication Skills") with practical advice	Gives very **advanced course search strategies** (e.g., "Advanced REST API patterns," "OWASP security in PHP") and URLs
Professional Contacts	Provides a **sample cold message** to send to professionals	Lists **real names and emails** (like Lars Klingenberg, etc.) to contact
Overall Tone	**Structured**, like a professional application walkthrough	**Conversational**, like chatting with a career counselor
Technical Details	Slightly deeper listing of missing skills	Slightly richer advice for **next skill-building steps** with URLs

Authorship & Use of Generative AI

This project was fully implemented by me, Swetha Gendlur Nagarajan, including all coding, model integration, dataset preparation, and UI development (both form-based and chatbot interfaces). The logic and flow were designed from scratch, based on my understanding of applied data science, natural language processing, and job-matching systems.

I used large language models (LLMs) like ChatGPT and Gemini at specific points to support my learning process and improve productivity. Their use was limited to:

- Understanding the basic structure of a Streamlit app, especially the initial skeleton required to create a multi-step chatbot interface and implement st.session_state.
- Improving the clarity of technical documentation (README formatting, function naming ideas, etc.).
- Identifying common industry skills, course keywords, and job-market phrasing for realism.

While I referred to LLMs to understand Streamlit's layout, all final code, logic, and customization were written by me. No generated code was copied without deep modification, validation, and testing. The entire project reflects my own implementation and understanding and complies with the UF Student Honor Code.