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:**

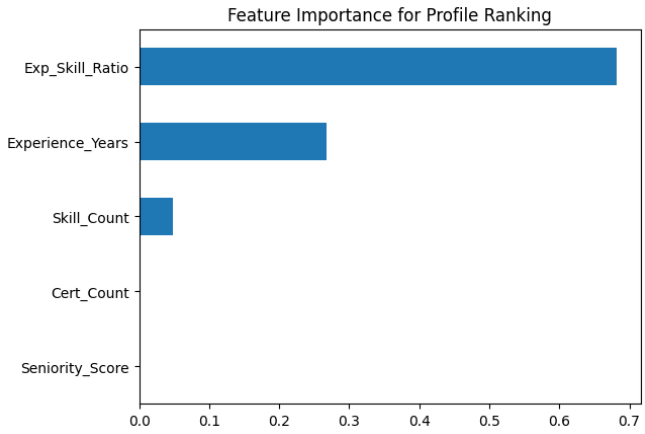
Optimized our model and improved data efficiency, we applied several feature selection techniques aimed at identifying the most relevant features and reducing dimensional complexity.

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

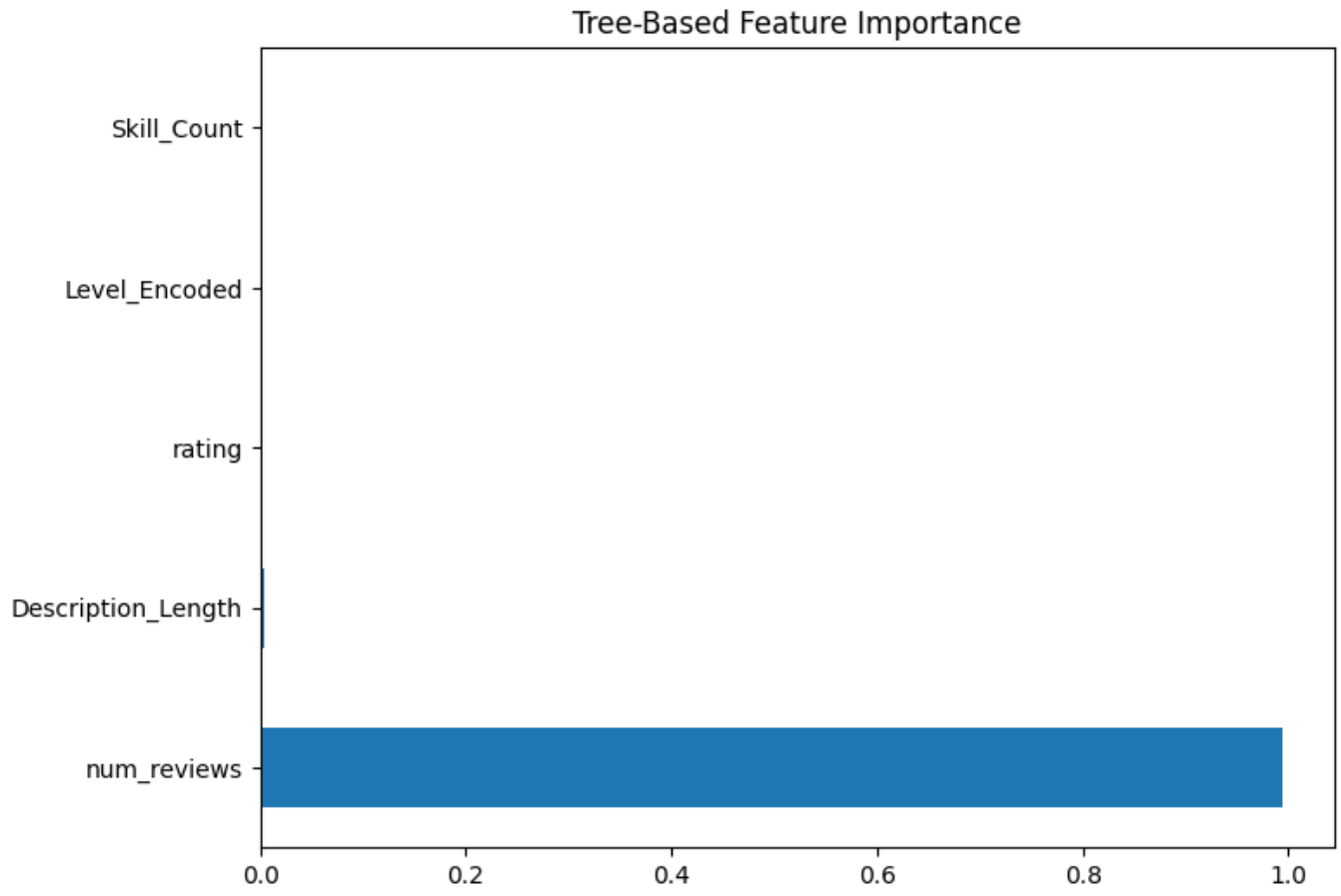
A custom clean_dataframe() function was applied to:

- 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

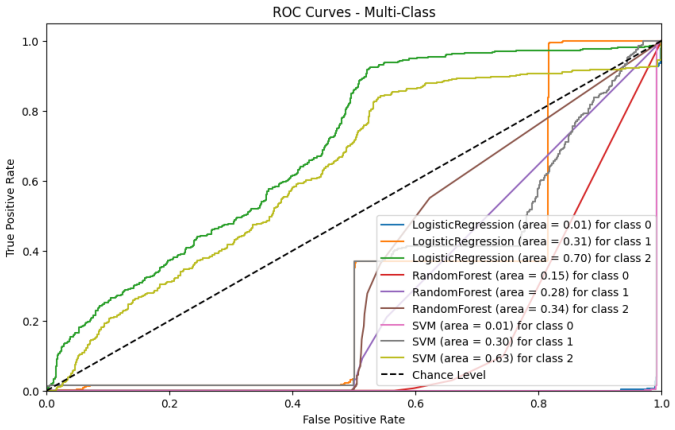
- **Logistic Regression** predicted “Low” courses well (100% recall), but confused many “High” as “Medium”.
- **SVM** also had perfect recall for “Low”, but predicted only ~49% of “High” courses correctly.
- **Random Forest** achieved almost perfect predictions for all categories.

```
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