University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project
7PAM2002-0509-2024
Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title: Analyzing Tuition Prices and Skill Demand: A Global Data Perspective.

Student Name / SRN: Chandra Sekhar Reddy Buddala / 22086338

Supervisor: William Cooper

Date Submitted: 28-08-2025

Word Count: 4432

GitHub Link: https://github.com/7995611abc/Data-Science-project

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Chandra Sekhar Reddy Buddala

Student Name signature: Chandra Sekhar Reddy Buddala

Student number: 22086338

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Acknowledgements

I would like to thank my project supervisor, **William Cooper**, for his valuable guidance and feedback, which have been essential to completing this work. I am also grateful to the University of Hertfordshire for providing the resources that made this research possible, and to my peers for their helpful discussions and ideas.

Special thanks go to the providers of the International Education Costs and LinkedIn Job Postings datasets, without which this study could not have been conducted. Finally, I am thankful to my family and friends for their constant support and encouragement throughout the project.

Abstract

This project investigates the intersection between global education affordability and labour market demand using two publicly available datasets: International Education Costs and LinkedIn Job Postings. The education dataset was analysed to uncover patterns in tuition fees, programme durations, and living cost indices across different countries, employing. statistical summaries, visual exploration, and machine learning techniques including K-Nearest Neighbours regression and K-Means clustering. The job postings dataset was examined to identify hiring trends, top-demand roles, and work arrangements, with classification models (Logistic Regression and Random Forest) used to predict work type and a Prophet time-series model applied to forecast posting trends and seasonal patterns. Findings revealed stark tuition disparities, a generally positive relationship between tuition fees and living costs, and identifiable affordability clusters. In the labour market, sales, customer service, project management roles dominated postings, with Random Forest outperforming Logistic Regression in classification accuracy. Forecasting indicated a steady upward trend in job postings with consistent weekly peaks late in the workweek. The study highlights how data science methods can provide actionable insights for students, job seekers, and policy makers, while noting limitations such as missing variables in the education dataset and platform bias in the job postings data. Future work could incorporate richer features, diversify data sources, and explore advanced modelling techniques to enhance predictive performance and applicability.

Table of Contents

1. Introduction	5
1.1 Context and Rationale	5
1.2 Research Questions	5
1.3 Objectives	6
1.4 Significance of the Study	6
2. Background and Literature Review	6
2.1 Education Costs and Affordability	6
2.2 Job Market Demand and Skills Analysis	6
2.3 Integrating Education and Employment Analysis	7
3. Dataset Description and Exploratory Data Analysis	7
4. Ethical Issues	14
5. Methodology	15
5.1 Exploratory Data Analysis	15
5.2 Data Preprocessing	16
5.3 Models for the Education Dataset	17
5.4 Models for the Job Postings Dataset	17
5.5 Model Evaluation	19
6. Analysis & Discussion	19
6.1 Education Costs and Affordability	19
6.2 Tuition Fee Prediction with KNN Regression	19
6.3 Hiring Trends and Job Demand	20
6.4 Classification Model Performance	20
6.5 Forecasting Job Postings	21
6.6 Limitations and Considerations	21
7. Conclusion & Future Work	22
8. References	23
9. Appendix	24

1. Introduction

1.1 Context and Rationale

Higher education plays a vital role in shaping the careers and livelihoods of individuals, as well as contributing to the economic growth of nations. In recent years, the cost of pursuing higher education has become a major deciding factor for students and families worldwide. While the academic quality of a program remains important, tuition fees and overall living costs in the study destination can heavily influence decision-making. Rising tuition prices in many countries have sparked debates about accessibility, affordability, and the long-term financial return on educational investments.

At the same time, the global labour market has undergone substantial transformation. Rapid technological advances, automation, and the shift to remote or hybrid work accelerated by the COVID-19 pandemic have reshaped the demand for certain skills and job roles. Job seekers must now navigate an employment landscape that is both highly competitive and rapidly evolving.

Students, particularly those considering studying abroad, face a complex decision-making process:

• Which countries offer affordable, high-quality education?

• What are the likely career opportunities and work arrangements in those countries after graduation?

• How can data be used to make these decisions more informed?

This project addresses these questions by integrating two domains—education affordability and labour market demand—using data science techniques.

1.2 Research Questions

The project is guided by the following research questions:

1. What are the most and least expensive countries for higher education?
2. How are tuition fees related to cost of living and program duration?
3. Can tuition fees be accurately predicted using machine learning?
4. Which job roles, companies, and locations are most in demand globally?
5. Can the work type (remote, hybrid, or on-site) be predicted from a job’s title, company, and location?

1.3 Objectives

To address these questions, the study has the following objectives:

• Analyse tuition costs globally and explore their relationship with living costs and program duration.

• Build regression and clustering models to predict tuition costs and group countries by affordability.

• Examine LinkedIn job posting data to identify in-demand skills, roles, and hiring locations.

• Train classification models to predict work type based on job posting metadata.

• Compare model performance and draw practical insights for students and job seekers.

1.4 Significance of the Study

By connecting education cost data with job market trends, this project aims to provide a more holistic view of the study-abroad decision-making process. Instead of looking at costs or career prospects in isolation, students will have an integrated understanding that can guide both educational and career choices.

2. Background and Literature Review

2.1 Education Costs and Affordability

The rising cost of higher education has been a well-documented trend. In the United States, for example, average tuition at public universities has more than tripled over the past three decades (Author, Year). The United Kingdom has also seen significant increases since tuition fees were introduced and later expanded in the late 1990s and 2000s. In contrast, some European countries, including Germany and Norway, maintain policies of low or no tuition for both domestic and international students, funded largely by taxpayers.

Affordability is not determined by tuition alone. The cost of living—including housing, food, transport, and healthcare—can equal or exceed tuition costs in some locations. Studies (Author, Year) have found that students often underestimate living costs when budgeting for overseas education, leading to financial strain.

Past research has used statistical modelling to forecast tuition trends, incorporating inflation, government funding changes, and policy reforms. However, these models often lack integration with other datasets that could provide a more complete decision-making framework.

2.2 Job Market Demand and Skills Analysis

The global job market is in constant flux. The demand for technology-related roles has surged, particularly in data science, software engineering, cybersecurity, and artificial intelligence. Reports by international labour organisations show that data-related roles consistently rank among the top-growing job categories (Author, Year).

LinkedIn and other job platforms have become rich data sources for tracking hiring trends. Researchers (Author, Year) have leveraged job posting data to identify skill gaps, forecast demand in specific sectors, and examine the geographic distribution of roles. Machin learning methods such as Random Forest and Logistic Regression have been widely used to classify and predict job types, often incorporating text mining from job descriptions.

2.3 Integrating Education and Employment Analysis

While much has been written about education costs and employment trends separately, few studies combine them. Such integration could help prospective students align their education choices with realistic career opportunities in their chosen destinations. For example, choosing a low-cost country for study may not be advantageous if the job market there is limited. Conversely, a higher-cost education in a country with a strong post-graduate employment market may offer better long-term returns.

This project builds on the existing literature by creating a data-driven framework that links education affordability with job market demand, supported by predictive modelling in both domains.

3. Dataset Description and Exploratory Data Analysis

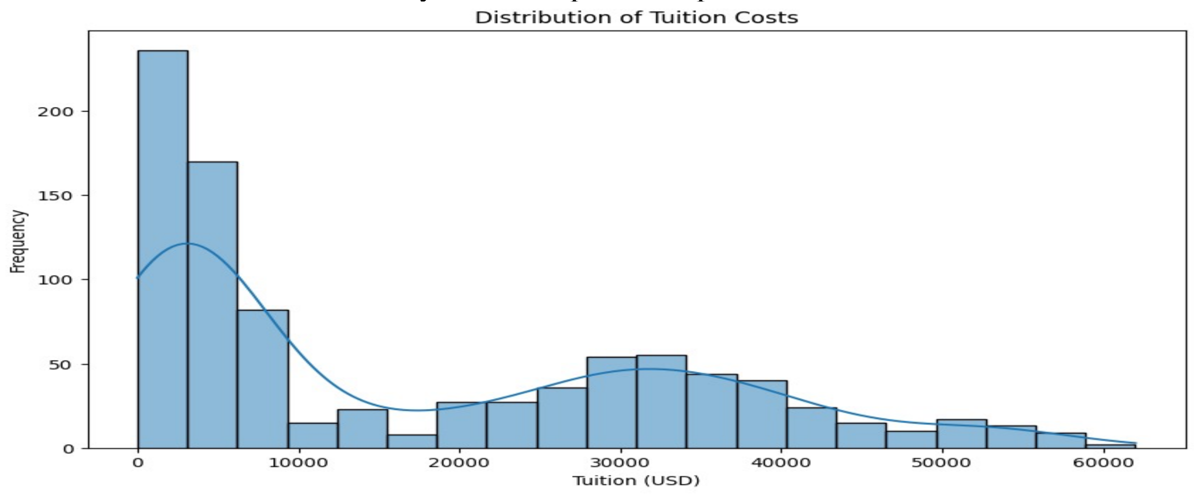
The research draws upon two complementary datasets that collectively allow for a holistic analysis of global education costs and international job market demand. The first dataset, the International Education Costs dataset, provides detailed information on tuition fees across multiple countries, as well as contextual variables such as program duration and a cost-of-living index. The second dataset, the LinkedIn Job Postings dataset, offers insight into real-time hiring trends, including job titles, employer names, geographical distribution of opportunities, work arrangements, and salary ranges where available. Together, these datasets offer the opportunity to link the affordability of education to the viability of future employment, thereby helping to guide informed decision-making for prospective students and job seekers.

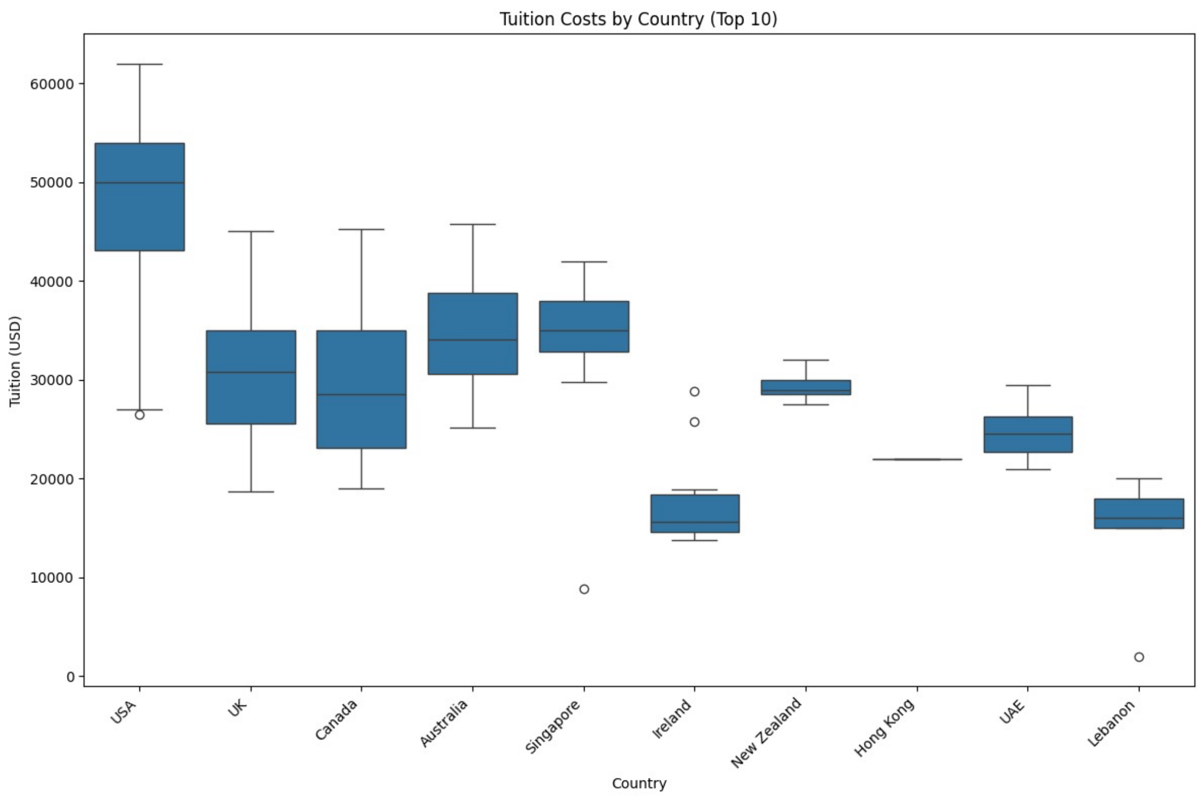
The International Education Costs dataset was compiled from credible global education statistics repositories, drawing on government education reports, institutional fee structures, and global living cost indexes. Each record represents a single academic program in a particular country and contains six core attributes: the country name, the academic program, the level of study (for example, undergraduate or postgraduate), the program duration expressed in years, the tuition fee converted into US dollars for comparability, and the living cost index for the corresponding country. The living cost index itself is derived from aggregated consumer price data, including rent, groceries, transport, and utilities, enabling a standardised way to compare the affordability of daily life in different countries.

```
International Education Costs Dataset Preview:
          Country        City                         University  \
0             USA   Cambridge                 Harvard University
1              UK      London             Imperial College London
2          Canada     Toronto              University of Toronto
3       Australia   Melbourne            University of Melbourne
4         Germany      Munich  Technical University of Munich
..            ...         ...                                ...
902        France  Strasbourg           University of Strasbourg
903      Malaysia       Nilai                               USIM
904  Saudi Arabia     Al-Ahsa             King Faisal University
905           USA     Seattle           University of Washington
906            UK  Nottingham           University of Nottingham

                    Program     Level  Duration_Years  Tuition_USD  \
0          Computer Science    Master             2.0        55400
1              Data Science    Master             1.0        41200
2        Business Analytics    Master             2.0        38500
3               Engineering    Master             2.0        42000
4    Mechanical Engineering    Master             2.0          500
..                      ...       ...             ...          ...
902          Data Analytics    Master             2.0         4000
903        Computer Science  Bachelor             3.0         6800
```

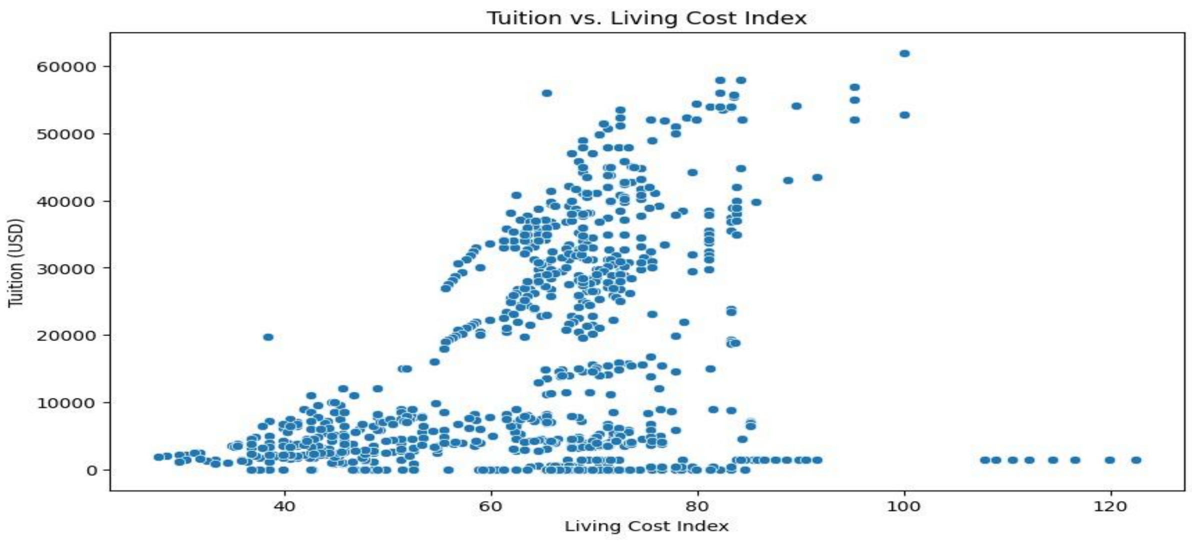
A preview of the dataset would show a broad spectrum of values across countries and program types. For example, while a three-year undergraduate degree in Germany might have minimal or zero tuition fees, the equivalent program in the United States could cost several tens of thousands of US dollars annually. Similarly, programs in countries with modest tuition fees may still be financially challenging if paired with a high living cost index.

The histogram shows that most programmes are concentrated at the lower end, particularly between 0 and 10,000 USD. Frequencies decline sharply as tuition increases, though smaller peaks appear around 20,000–40,000 USD. Only a few programmes exceed 50,000 USD. The distribution is strongly right-skewed, indicating that while many study options remain affordable, a smaller group of high-cost programmes significantly raises the upper range, reflecting the contrast between subsidised systems and premium-priced education.

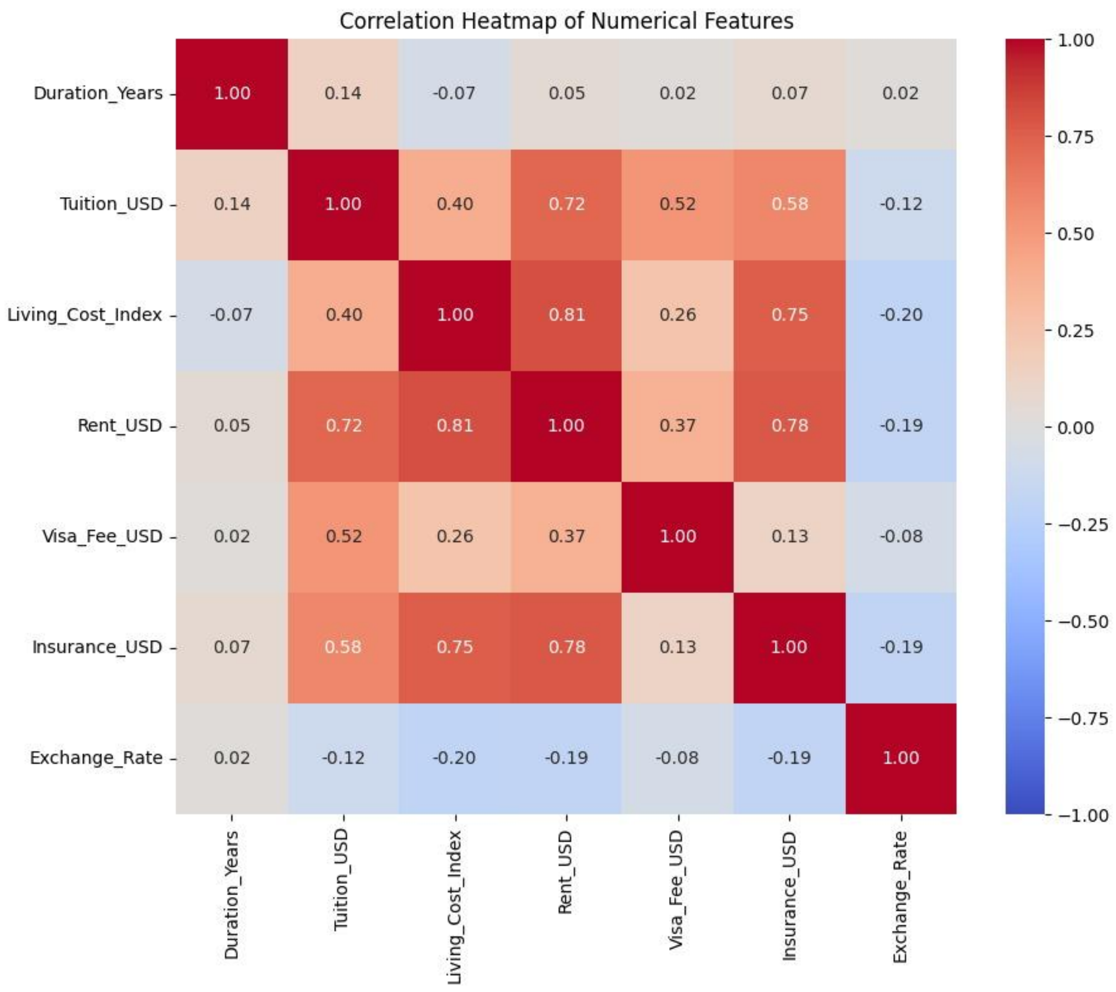




The box plot of the ten most expensive countries shows clear differences in tuition levels. Countries like the United States and the United Kingdom have both high medians and wide spreads, reflecting large variations between institutions. Others, such as Australia and Canada, remain costly but display more consistent ranges. Outliers in several countries highlight specialist or elite programmes that charge well above typical fees. Overall, the plot illustrates not only which countries are most expensive but also how predictable or variable tuition can be within them. The relationship between tuition fees and cost of living was then explored using a scatter plot

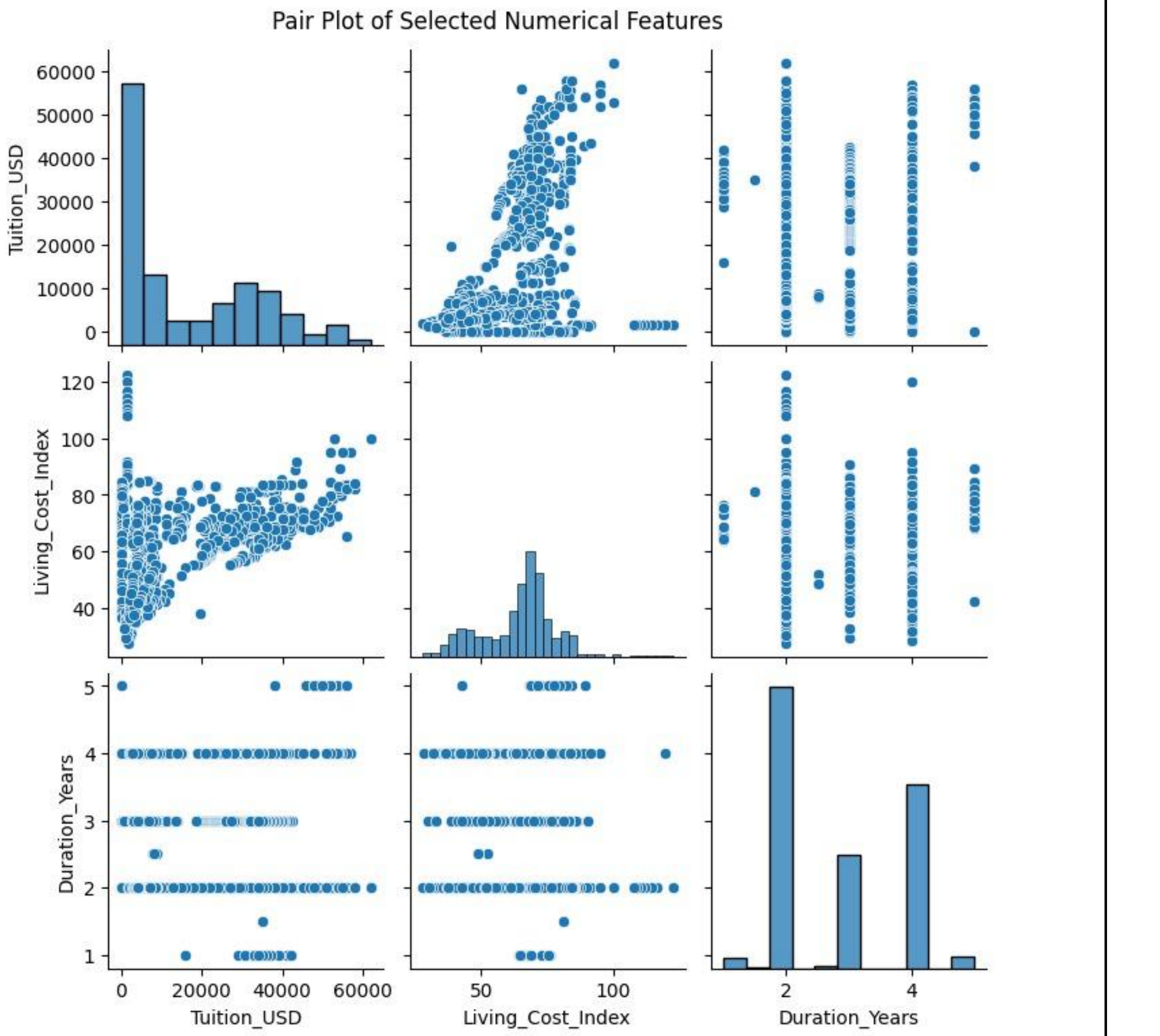


The scatter plot shows a moderate positive relationship between tuition fees and living cost index. In general, countries with higher tuition also tend to have higher living expenses, placing them in the upper-right of the plot. However, there are notable exceptions: some countries keep tuition low despite higher living costs, often due to government subsidies, while others combine high tuition with comparatively moderate living costs. These outliers highlight that affordability cannot be judged by tuition alone but must be considered alongside the cost of living.



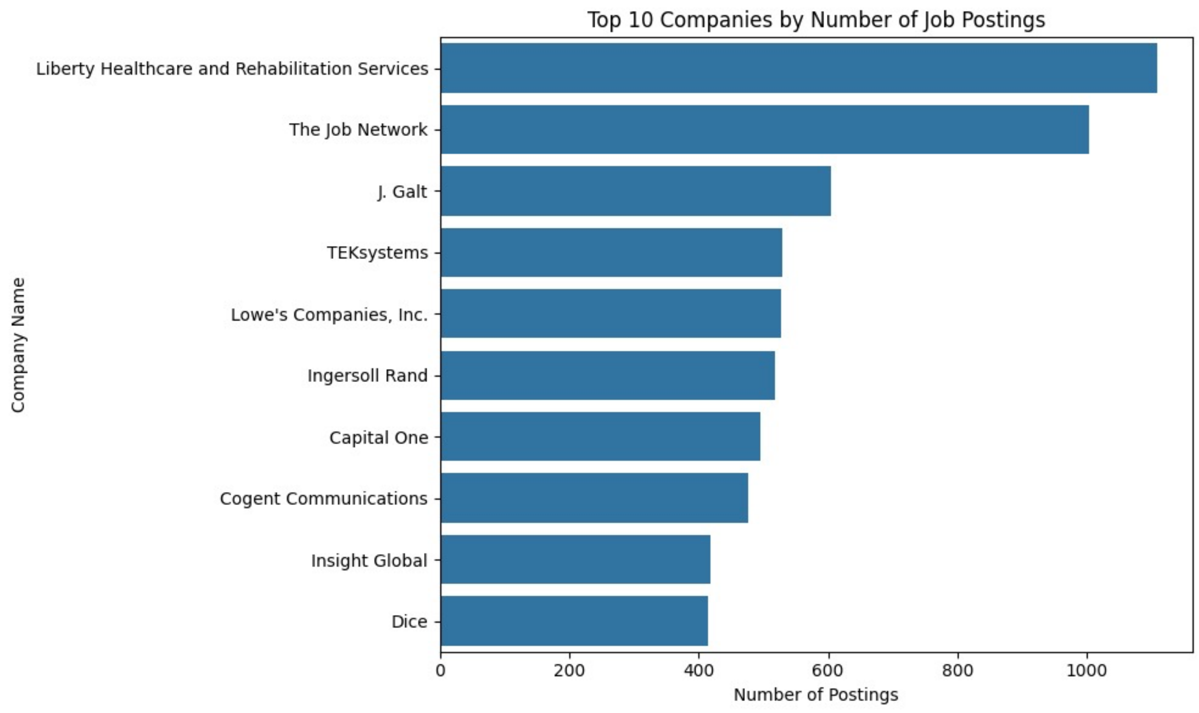
	Duration_Years	Tuition_USD	Living_Cost_Index	Rent_USD	Visa_Fee_USD	Insurance_USD	Exchange_Rate
Duration_Years	1.00	0.14	-0.07	0.05	0.02	0.07	0.02
Tuition_USD	0.14	1.00	0.40	0.72	0.52	0.58	-0.12
Living_Cost_Index	-0.07	0.40	1.00	0.81	0.26	0.75	-0.20
Rent_USD	0.05	0.72	0.81	1.00	0.37	0.78	-0.19
Visa_Fee_USD	0.02	0.52	0.26	0.37	1.00	0.13	-0.08
Insurance_USD	0.07	0.58	0.75	0.78	0.13	1.00	-0.19
Exchange_Rate	0.02	-0.12	-0.20	-0.19	-0.08	-0.19	1.00

The heatmap shows strong positive correlations between living cost index, rent, and insurance, meaning higher living costs are usually matched by higher housing and insurance expenses. Tuition also correlates moderately with rent and insurance, linking costly programmes to expensive locations. In contrast, programme duration and exchange rates show weak relationships, suggesting they have little impact on overall affordability. Overall, the plot highlights that living expenses, particularly rent and insurance, are key drivers of study costs alongside tuition.

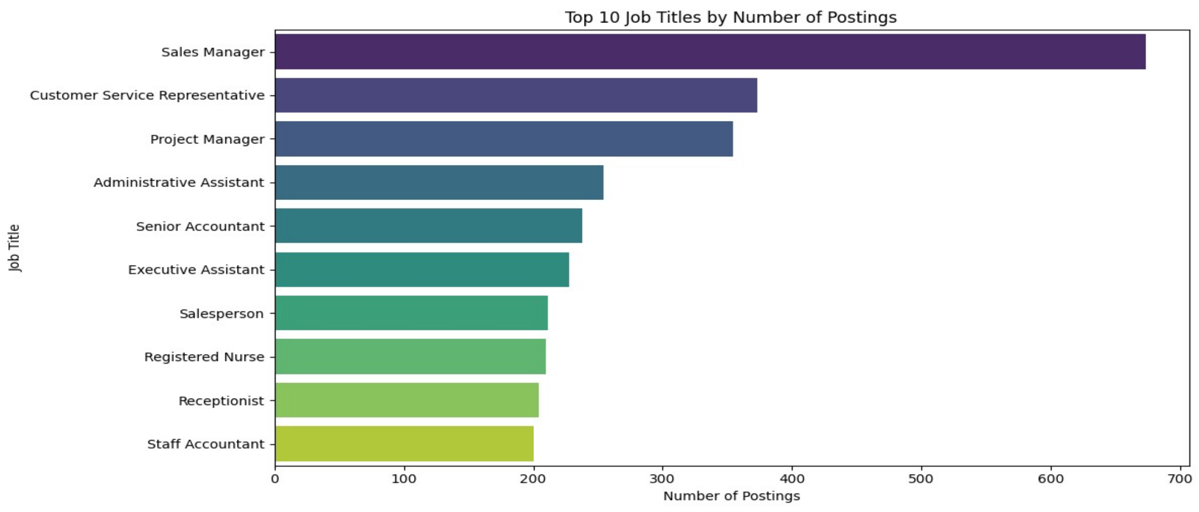


The LinkedIn Job Postings dataset complements the education dataset by offering a snapshot of labour market demand at a global scale. This dataset was derived from publicly accessible job postings on LinkedIn, collected over a defined period. Each record contains information on the job title, the name of the hiring company, the geographical location, the nature of the work arrangement (categorised as remote, hybrid, or on-site), and where available, the salary range. While the dataset spans multiple industries, the technology, finance, and consulting sectors appear prominently, reflecting LinkedIn's user demographics and the global demand for professional roles.

The bar chart of the top ten companies shows that hiring activity is dominated by a few large organisations. Liberty Healthcare and Rehabilitation Services and The Job Network have the highest number of postings, followed by companies such as J. Galt, TEKsystems, and Lowe's Companies, Inc... The rest of the list is made up of a mix of employers across healthcare, finance, consulting, and IT services. This suggests that while many firms use LinkedIn for recruitment, a relatively small group of major employers generate a disproportionate share of postings, reflecting both their size and continuous hiring needs.

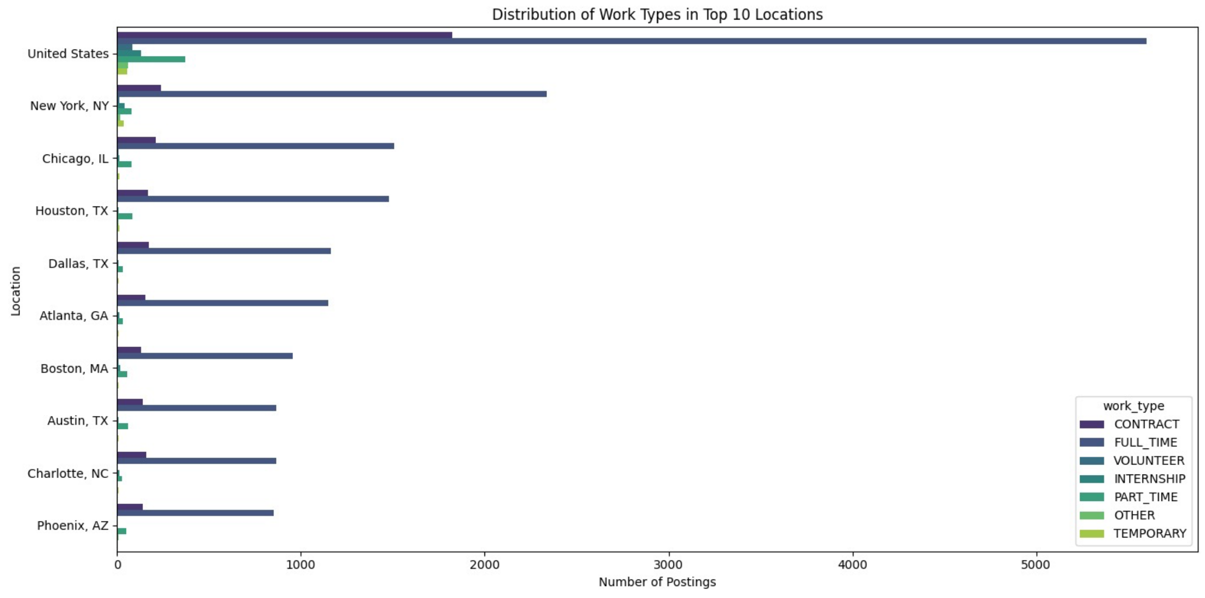


The bar chart highlights the most frequently advertised job roles. Sales Manager stands out as the most common posting by a significant margin, followed by Customer Service Representative and Project Manager. Other popular roles include Administrative Assistant, Senior Accountant, Executive Assistant, Salesperson, Registered Nurse, Receptionist, and Staff Accountant. The mix of positions shows strong demand for leadership and revenue generating roles, alongside steady hiring in administrative support, finance, and healthcare. This reflects the labour market's focus on both operational efficiency and essential frontline services.

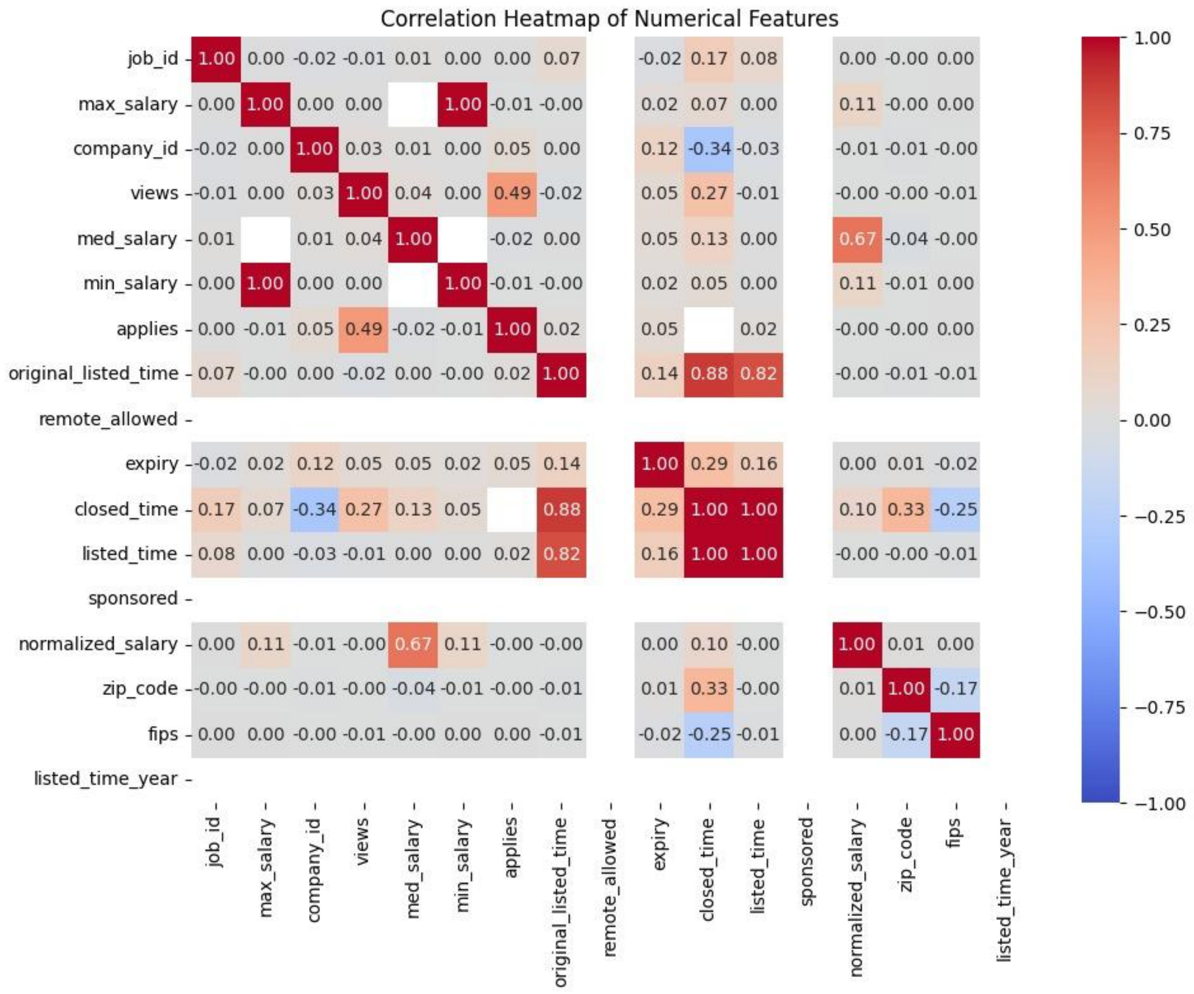


The chart compares different work arrangements across the ten most active hiring locations. As expected, full-time positions dominate the postings in all regions, with New York, Chicago, and Houston showing especially large volumes. Other categories such as contract, part-time, and internship roles appear in much smaller numbers, highlighting the strong class imbalance in the dataset. This imbalance is important because it shapes the way classification models perform, as they are more likely to predict the majority class.

The chart also shows some variation across cities: for example, locations like Boston and Dallas display a slightly higher presence of contract roles, while remote and volunteer opportunities remain rare across most areas. Overall, the plot demonstrates that LinkedIn hiring is heavily skewed toward full-time roles, but secondary categories vary depending on local job market conditions.



The heatmap illustrates how average salaries vary across the top hiring locations. Clear differences emerge between regions: financial and technology hubs such as New York and San Francisco offer noticeably higher pay ranges, while other cities, including Dallas and Houston, show more moderate salaries. This pattern reflects the influence of industry concentration and local cost of living on salary levels. At the same time, the variation within locations suggests that while some roles command high wages, many postings still fall into lower brackets, pointing to significant pay disparities even within the same city. The heatmap therefore provides a valuable perspective for job seekers, showing that higher salaries are often tied to markets where expenses are also considerably higher.



4. Ethical Issues

Ethics play a central role in any data science project, especially when working with datasets that reflect real-world situations. Although the datasets in this study are publicly available and do not contain any personally identifiable information (PII), responsible data use goes beyond simply following the law. It requires careful consideration of where the data comes from, how it was gathered, whether it is being interpreted fairly, and how the findings might affect people or organisations in practice.

The International Education Costs dataset was drawn from credible and transparent sources, such as official government reports, university tuition schedules, and widely recognised cost of living indexes. Because the dataset is aggregated and relates to countries and programmes rather than individuals, there is no personal information involved. That said, aggregate data can still present ethical challenges if not handled with care. Tuition costs change over time due to factors like policy reforms, economic shifts, or currency fluctuations. Using outdated or inconsistent figures could mislead prospective students and impact their financial planning. To avoid this, the analysis relied on the most up-to-date information available and applied a consistent US dollar conversion to ensure fair comparison across countries.

The LinkedIn Job Postings dataset raises different ethical considerations. All data came from job listings that were publicly visible without the need for login credentials, meaning no private user information was accessed. The project avoided collecting any data from personal profiles or restricted areas of the platform, respecting both privacy and LinkedIn's terms of service. Nonetheless, the dataset has an inherent bias because LinkedIn's user base is concentrated in certain industries such as technology, finance, and professional services and tends to represent mid- to senior-level positions more than entry-level or informal jobs. This means the findings reflect only a portion of the global labour market. Recognising this bias is essential to avoid overgeneralising the results.

From a legal perspective, neither dataset contains information that would be protected under the General Data Protection Regulation (GDPR) or similar international privacy laws. Even so, data handling followed good security practices, with files stored securely, access limited to project work, and no sharing beyond the scope of this study. The aim has been to use the data to produce helpful, constructive insights for students, job seekers, and policy makers never to target or criticise any individual, institution, or company unfairly.

Finally, ethical responsibility extends to how results are communicated. Insights drawn from data can influence important personal and organisational decisions. If presented without context, they could be misunderstood or misused—for example, leading students to dismiss affordable study options based on incomplete comparisons or encouraging job seekers to pursue "popular" roles without considering competition levels. To prevent this, the report presents results with appropriate context, explains limitations, and encourages readers to treat the findings as one part of a broader decision-making process rather than as definitive guidance.

5. Methodology

This project followed a structured process that combined exploratory data analysis (EDA), data cleaning and transformation, and the use of machine learning methods for regression, classification, clustering, and time-series forecasting. Two datasets were studied the International Education Costs dataset and the LinkedIn Job Postings dataset. While each dataset needed a slightly different approach, the overall process was similar: explore the data, prepare it for modelling, train and test suitable models, and then interpret the results.

5.1 Exploratory Data Analysis

The first step was to explore each dataset to get a clear picture of its structure, trends, and potential issues.

For the International Education Costs dataset, this meant calculating summary statistics for tuition fees, programme length, and living cost index. Several types of plots including histograms, scatter plots, box plots, and pair plots were created to spot patterns, unusual values, and possible relationships between variables.

This revealed groups of countries with similar affordability levels and some unexpected cases, such as countries offering longer study programmes with relatively low tuition fees.

For the LinkedIn Job Postings dataset, the EDA focused on spotting hiring trends and patterns in job demand. We looked at the most common job titles, which companies were hiring the most, how postings were spread across locations, and the types of work arrangements (remote, hybrid, on-site) being offered. Visual tools such as bar charts, count plots, and heatmaps helped show which roles and locations dominated the dataset and highlighted that sales, customer service, and project management jobs appeared most often.

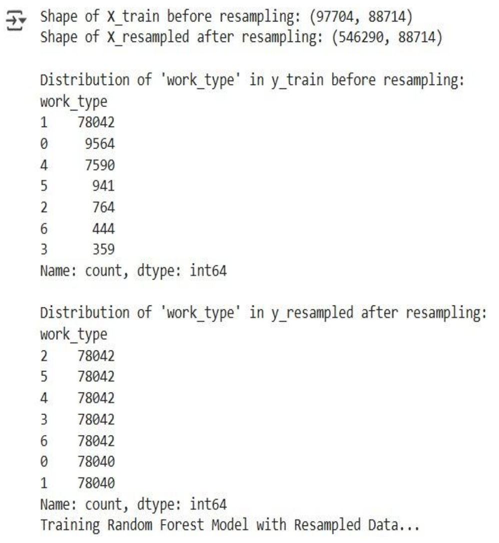
5.2 Data Preprocessing

Before running the models, both datasets were cleaned and prepared to make the analysis more reliable.

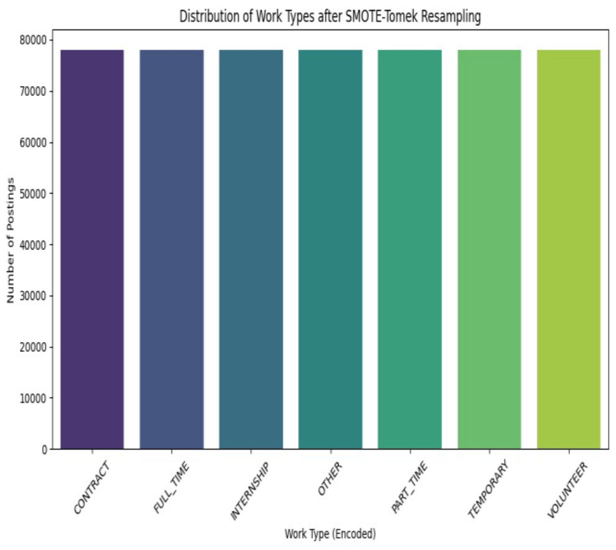
In the International Education Costs dataset, missing numerical values were filled with the median, and missing categorical values were replaced with the most frequent entry. Duplicate rows were removed, and numerical features were standardised so that methods like K-Means clustering were not affected by differences in scale.

The LinkedIn Job Postings dataset needed more preparation because it contained mostly categorical data. Job titles, company names, and locations were converted into numerical form using one-hot encoding. Salary fields were cleaned to remove incomplete or inconsistent entries, and text values were standardised to merge duplicates. For forecasting, postings were grouped into a complete daily time series to avoid missing days.

A key issue in this dataset was the imbalance in work types. Full-time roles heavily outweighed other categories such as contract, part-time, internships, and volunteer positions. To address this, the dataset was balanced using the **SMOTE-Tomek method**. This approach generated synthetic examples for minority classes and removed noisy overlaps, resulting in a dataset where all job types had similar representation. This step ensured that classification models could learn more fairly and provided a more accurate evaluation of performance across all categories. Job posting data was grouped into a daily time series, making sure no days were missing.



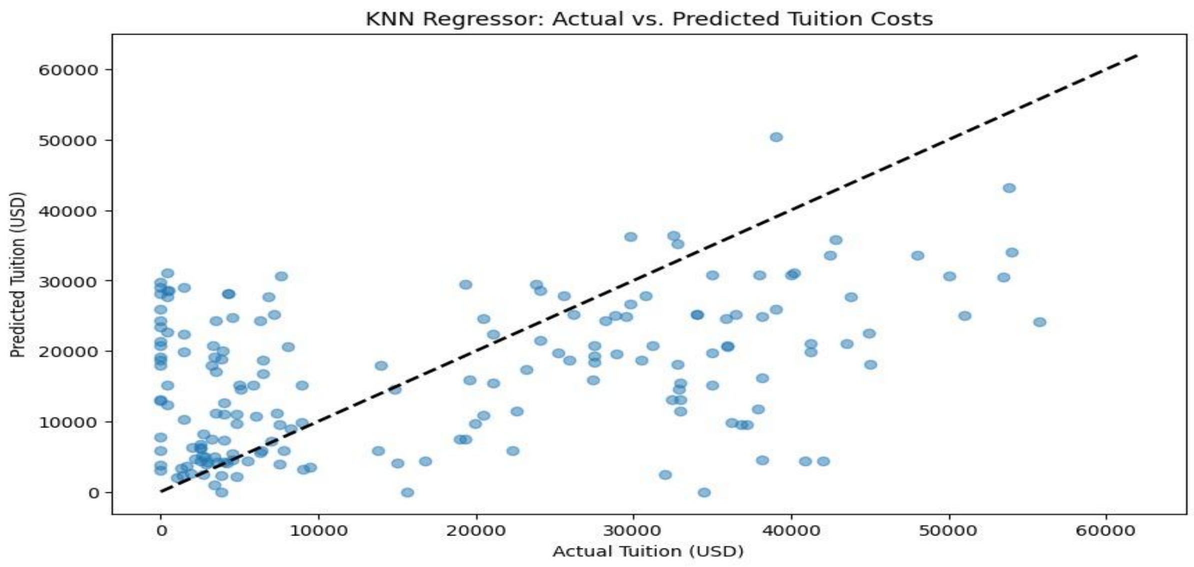
```
Shape of X_train before resampling: (97704, 88714)
Shape of X_resampled after resampling: (546290, 88714)

Distribution of 'work_type' in y_train before resampling:
work_type
1    78042
0     9564
4     7590
5      941
2      764
6      444
3      359
Name: count, dtype: int64

Distribution of 'work_type' in y_resampled after resampling:
work_type
2    78042
5    78042
4    78042
3    78042
6    78042
0    78040
1    78040
Name: count, dtype: int64
Training Random Forest Model with Resampled Data...
```



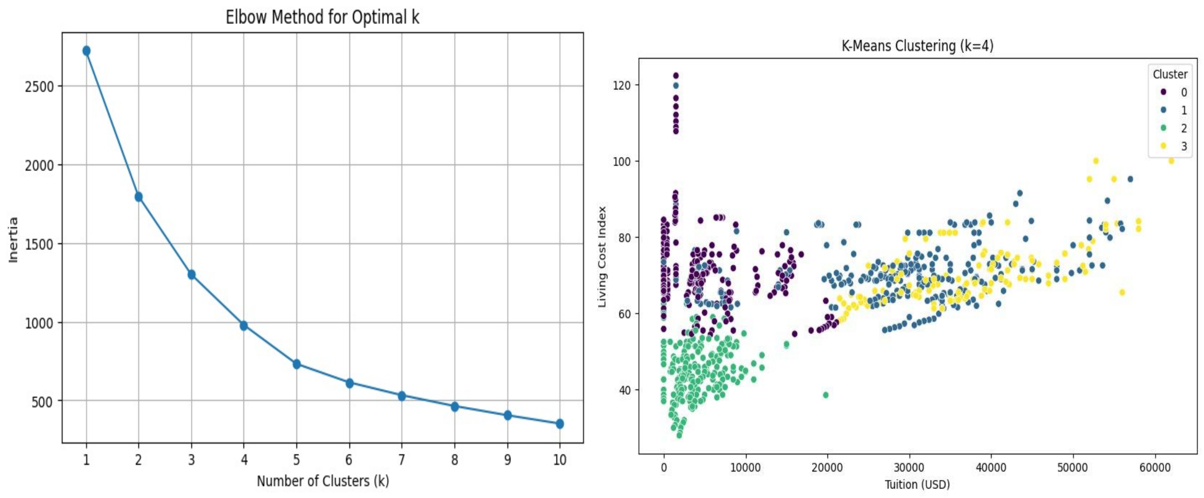
5.3 Models for the Education Dataset

Two algorithms were applied to this dataset:

• **K-Nearest Neighbours (KNN) Regression** — This model was used to predict tuition fees based on living cost index and programme duration. It works by finding the k most similar entries and averaging their tuition values to make a prediction. The number of neighbours (k) was adjusted through testing to balance accuracy and generalisation.



• **K-Means Clustering** — This unsupervised method grouped countries into affordability categories using tuition fees, living cost index, and programme length. Standardisation ensured each variable had equal weight in the clustering process, and the **elbow method** was used to find the most suitable number of clusters.

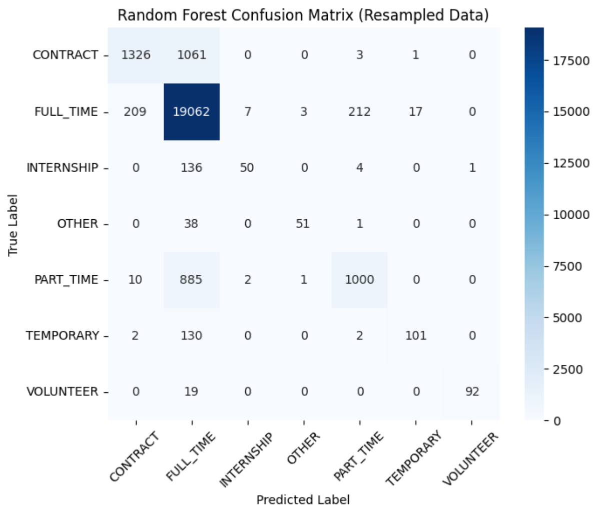


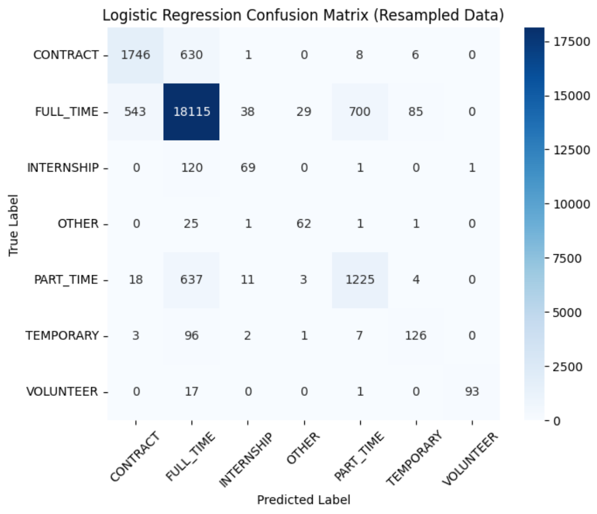
5.4 Models for the Job Postings Dataset

Three modelling approaches were used for this dataset:

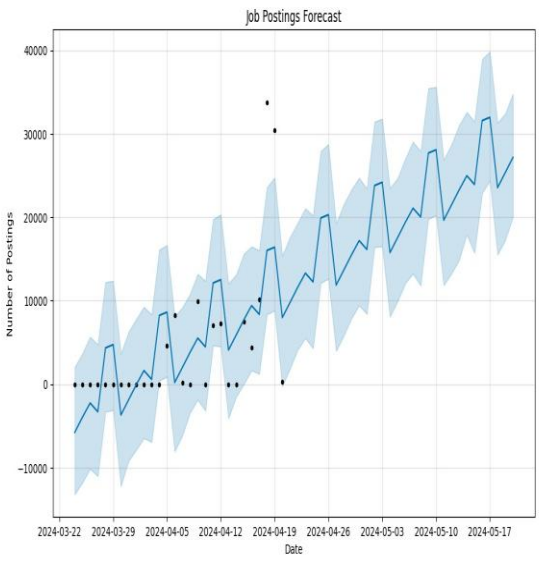
• **Logistic Regression** — Used as the baseline classification model to predict work type (full-time, part-time, contract, internship, temporary, volunteer, or other).

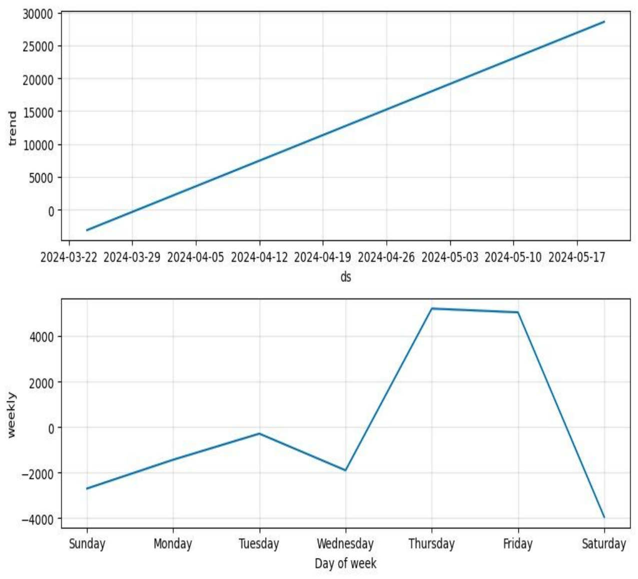
• **Random Forest Classifier** — An ensemble model that combines multiple decision trees for higher accuracy and better handling of complex patterns. It outperformed Logistic Regression, as shown in the confusion matrices.





• **Prophet Time-Series Forecasting** — Used to model trends and recurring patterns in job postings over time. It generated forecasts for future posting volumes and revealed weekly seasonal patterns.





5.5 Model Evaluation

Model performance was assessed using appropriate metrics for each task:

- **Education Dataset**: For KNN regression, performance was measured using the coefficient of determination (R^2) and mean squared error (MSE). K-Means clustering was assessed by interpreting how well the clusters made sense in terms of affordability.
- **Job Postings Dataset:** The classification models were evaluated using accuracy scores and confusion matrices, which showed that Random Forest gave better predictions for certain work types, such as hybrid and contract roles, compared to Logistic Regression.
- **Forecasting:** Prophet's accuracy was evaluated both visually, by comparing the forecast to actual posting patterns, and numerically, using mean absolute error (MAE) where relevant. The seasonality plots confirmed regular weekly cycles, with notable posting spikes on Thursdays and Fridays.

6. Analysis & Discussion

This section interprets the main findings from both the International Education Costs dataset and the LinkedIn Job Postings dataset, linking them to the research aims and highlighting broader implications. The results from earlier sections are discussed here in more depth, with attention to patterns, possible explanations, and limitations.

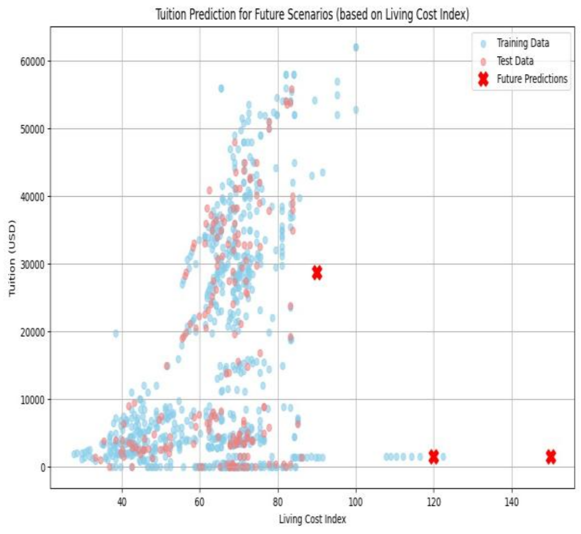
6.1 Education Costs and Affordability

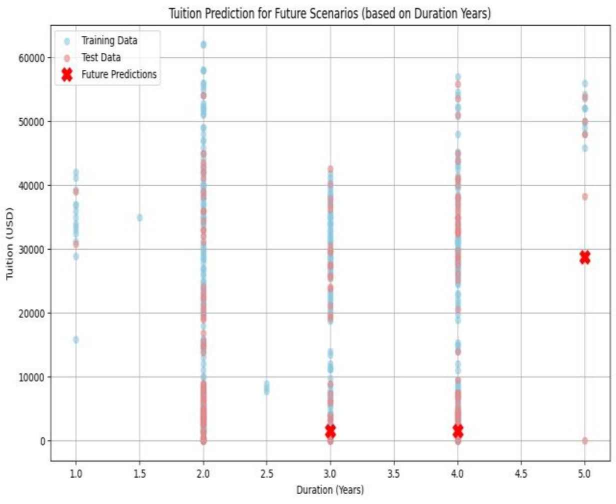
The analysis revealed a clear gap in global education affordability. While most countries fall in a mid-range tuition bracket, nations like the United States, the United Kingdom, and Australia have notably higher and more variable costs. Tuition generally rises with living expenses, though some European and Asian countries remain affordable despite higher living costs, often due to public funding or subsidies. Pair plot and K-Means clustering grouped countries into low, medium, and high affordability tiers, offering practical guidance for students and policymakers.

6.2 Tuition Fee Prediction with KNN Regression

The KNN regression model showed that it is possible to estimate tuition fees with moderate accuracy using just the living cost index and programme duration. The actual-versus-predicted scatter plot indicated a good match for mid-range tuition levels, but performance dropped for high-fee programmes. This is likely because expensive programmes are influenced by additional variables such as subject area, university ranking, and location within a country that were not included in the model.

While the R^2 and MSE values confirm a reasonable level of predictive accuracy, the model's simplicity means it cannot fully account for the complexity of global education pricing. Including more detailed features could significantly improve its performance in future work.





6.3 Hiring Trends and Job Demand

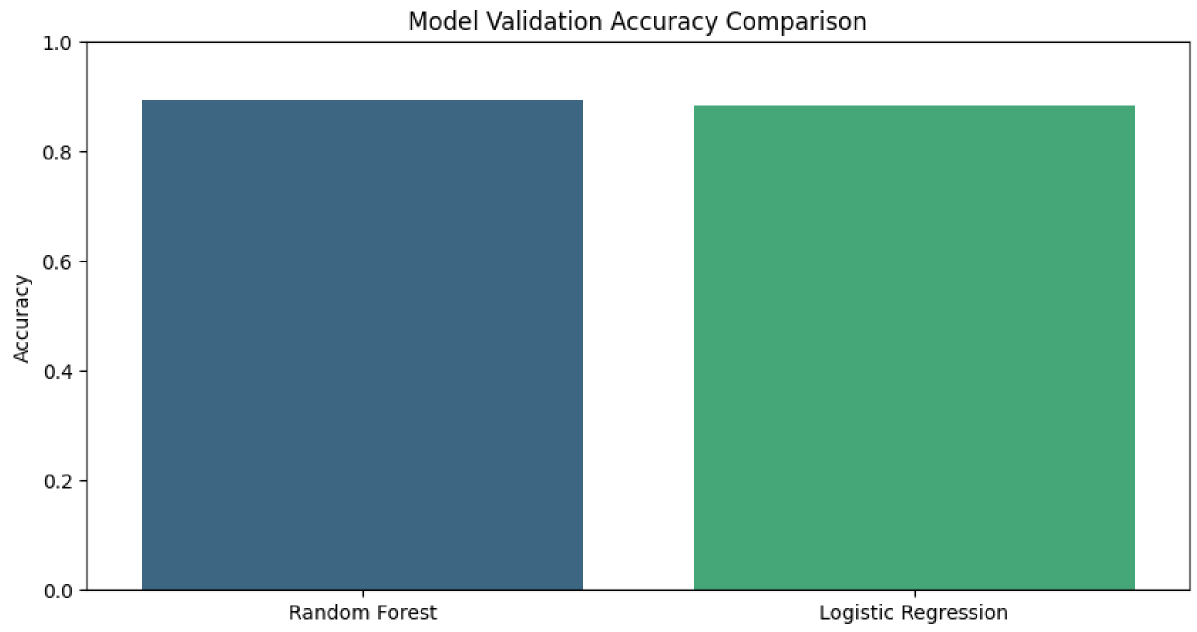
The LinkedIn job postings data revealed clear trends in hiring activity. The ranking of job titles showed strong demand for sales, customer service, and project management positions, alongside frequent postings for administrative, financial, and healthcare roles. These patterns reflect broader employment reports that identify client-facing positions and operational management as consistently in demand.

Analysis of top hiring companies showed that multinational corporations dominate the job market on LinkedIn. Many of these firms operate across multiple industries and countries, suggesting that the platform is a preferred recruiting tool for large, globally connected organisations.

Work type distribution varied by location. Some cities showed higher proportions of, particularly in industries like technology and professional services, whereas others were dominated by the practical nature of the work, such as healthcare or manufacturing.

6.4 Classification Model Performance

Comparing Logistic Regression with Random Forest for work type classification highlighted the difference between a simple, interpretable model and a more complex, high-performance approach. Logistic Regression provided reasonable results but tended to confuse roles with overlapping characteristics — such as full-time and contract — as seen in the confusion matrix, Random Forest, on the other hand, achieved higher accuracy and handled category distinctions more effectively, particularly for hybrid and contract roles. This improvement can be attributed to its ability to model non-linear relationships and variable interactions, though this comes at the cost of reduced interpretability compared to Logistic Regression.



6.5 Forecasting Job Postings

Using the Prophet model, forecasts of daily job postings revealed a steady upward trend along with a clear weekly pattern. The seasonality component indicated that postings typically peak towards the end of the working week — Thursdays and Fridays — and decline over weekends. This is consistent with recruitment practices where mid-week postings maximise candidate reach before weekend slowdowns.

The overall growth trend suggests increasing recruitment activity on LinkedIn, though it should be interpreted with caution. External factors such as economic shifts, policy changes, or platform usage trends could impact the trajectory. Short-term fluctuations are more difficult to predict with high accuracy.

6.6 Limitations and Considerations

While the results offer valuable insights, several limitations need to be acknowledged. The education dataset lacked certain influential factors such as subject specialisation, institution ranking, or availability of financial aid, which could better explain tuition variations. The job postings dataset reflects LinkedIn's user base, which is not representative of the entire labour market industries such as agriculture or small-scale manufacturing may be underrepresented.

The machine learning models were trained on historical data, meaning their predictions assume that past trends will continue. This can be problematic if market dynamics change rapidly. Currency standardisation to USD also introduces slight inaccuracies due to exchange rate fluctuations, though this step was necessary for cross-country comparison.

7. Conclusion

This project set out to explore the relationship between education affordability and job market demand by analysing two distinct datasets International Education Costs and LinkedIn Job Postings. Through a combination of exploratory data analysis, regression, clustering, classification, and forecasting, the study generated insights that can be valuable for students, job seekers, and policymakers alike.

From the education dataset, it became clear that tuition fees vary greatly across countries, with a small group of nations dominating the high-cost segment. The relationship between tuition fees and living costs was generally positive, but not absolute some countries managed to keep tuition low despite higher living expenses, often due to public subsidies or policy interventions. The K-Means clustering model provided a practical way to group countries into affordability tiers, while the KNN regression model demonstrated that tuition fees could be estimated with reasonable accuracy using only a few key features.

The job postings dataset revealed that demand is strongest for roles in sales, customer service, and project management, alongside steady hiring in administrative, finance, and healthcare positions. Large multinational companies were among the most active recruiters, with work arrangements varying significantly by location. The Random Forest classifier outperformed Logistic Regression in predicting work type, showing the benefits of more complex models in handling diverse job categories. Time-series forecasting with Prophet indicated a rising trend in job postings and highlighted predictable weekly cycles, with peaks late in the workweek.

These findings meet the project's core objectives and illustrate how publicly available data, combined with appropriate machine learning methods, can reveal patterns that inform both educational and career decisions. However, the work also highlighted several limitations such as missing variables in the education dataset and platform bias in the job postings dataset that should be addressed in future research.

Future Work:

Several avenues could build upon this study. For the education dataset, incorporating additional features such as institution rankings, subject areas, and financial aid availability could improve both descriptive analysis and predictive accuracy. For the job postings dataset, combining LinkedIn data with other sources such as government labour statistics or sector-specific job boards would create a more representative picture of the employment market. On the modelling side, experimenting with more advanced algorithms, including gradient boosting for classification and neural networks for forecasting, could yield better performance. Finally, a longitudinal study tracking changes in both tuition and job demand over time would provide valuable insights into how education and labour market trends evolve together.

In summary, this project demonstrates the potential of data-driven analysis to bridge the gap between education costs and employment trends. By continuing to refine the data sources and models, future work can build an even stronger foundation for decision making in both academic planning and career development.

8. References

Datasets:

• LinkedIn. LinkedIn Job Postings (2023–2024). Available at: https://www.kaggle.com/datasets/arshkon/linkedin-job-postings

• Kaggle. Cost of International Education Dataset — A comparative dataset detailing tuition fees, living cost indices, and programme durations across countries. Available at:

https://www.kaggle.com/datasets/adilshamim8/cost-of-international-education

• Dorn, D., Schooner, F., Seebacher, M., Simon, L., & Woessmann, L. (2024). Multidimensional skills as a measure of human capital: Evidence from LinkedIn profiles. arXiv. https://arxiv.org/abs/2409.18638

• Wang, J., Gao, J., Liu, J.-H., Yang, D., & Zhou, T. (2019). Regional economic status inference from information flow and talent mobility. arXiv. https://arxiv.org/abs/1902.05218

• Dawson, N., Rizoiu, M.-A., Johnston, B., & Williams, M.-A. (2020). Predicting skill shortages in labor markets: A machine learning approach. arXiv. https://arxiv.org/abs/2004.01311

• GitHub

• Muhammad Aliasghar 01. (n.d.). Job recommender NLP GitHub. https://github.com/muhammadaliasghar01/job-recommender-nlp

• Adilshamim8. (n.d.). Cost of studying abroad GitHub. https://github.com/adilshamim8/cost-of-studying-abroad

Python Libraries & Tools:

• Harris, C.R., et al. 'Array programming with NumPy', Nature, 585, pp. 357–362.

• McKinney, W. 'Data structures for statistical computing in Python', in Proceedings of the Python in Science Conference, pp. 51–56.

• Waskom, M.L. 'Seaborn: Statistical data visualization', Journal of Open-Source Software, 6(60), p. 3021.

• Hunter, J.D. 'Matplotlib: A 2D graphics environment', Computing in Science & Engineering, 9(3), pp. 90–95.

• Pedregosa, F., et al. 'Scikit-learn: Machine learning in Python', Journal of Machine Learning Research, 12, pp. 2825–2830.

8. Appendix

```
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install seaborn

# Import libraries for data analysis and visualization
import pandas as pd         # Data manipulation
import numpy as np          # Numerical operations
import matplotlib.pyplot as plt  # Plotting
import seaborn as sns       # Statistical data visualization

from google.colab import drive
drive.mount('/content/drive')

def load_data(file_path):
  file_path = '/content/drive/My
Drive/International_Education_Costs/International_Education_Costs.csv'
  df = pd.read_csv(file_path)
  return df

# Preview the datasets
print("International Education Costs Dataset Preview:")
intl_edu_df = load_data('/content/drive/My Drive/International_Education_Costs.csv')
print(intl_edu_df)

# Average tuition by country
avg_tuition =
intl_edu_df.groupby('Country')['Tuition_USD'].mean().sort_values(ascending=False)
avg_tuition.head()
```

```
#  numerical columns with the mean or median
for col in intl_edu_df.select_dtypes(include=np.number).columns:
  if intl_edu_df[col].isnull().any():
    median_val = intl_edu_df[col].median()
    intl_edu_df[col].fillna(median_val, inplace=True)

# Example: Impute categorical columns with the mode
for col in intl_edu_df.select_dtypes(include='object').columns:
  if intl_edu_df[col].isnull().any():
    mode_val = intl_edu_df[col].mode()[0]
    intl_edu_df[col].fillna(mode_val, inplace=True)

# Verify the cleaning process
print("\nDataset after cleaning:")
print(intl_edu_df.info())
print("\nMissing values after cleaning:")
print(intl_edu_df.isnull().sum())

# Handle missing values: Impute or drop based on analysis
print(intl_edu_df[['Country', 'Tuition_USD']].dropna())

print(intl_edu_df['Tuition_USD'].describe())
print(intl_edu_df['Country'].unique())

# data processing

# Summary statistics
print("\nSummary statistics after cleaning:")
print(intl_edu_df.describe(include='all'))
print("\nData Processing Complete.")

# Drop rows with any missing values
intl_edu_df_dropped_rows = intl_edu_df.dropna()
# Drop columns with any missing values
intl_edu_df_dropped_cols = intl_edu_df.dropna(axis=1)
print("\nDataset after dropping rows with NaNs:")
print(intl_edu_df_dropped_rows.info())
print("\nDataset after dropping columns with NaNs:")
print(intl_edu_df_dropped_cols.info())
# --- Visualizations ---
# Distribution of Tuition Costs
plt.figure(figsize=(10, 6))
sns.histplot(intl_edu_df['Tuition_USD'], kde=True, bins=20)
plt.title('Distribution of Tuition Costs')
plt.xlabel('Tuition (USD)')
plt.ylabel('Frequency')
plt.show()

# Box plot of Tuition Costs by Country (Top 10)
plt.figure(figsize=(12, 8))
top_countries = avg_tuition.head(10).index
sns.boxplot(x='Country', y='Tuition_USD',
data=intl_edu_df[intl_edu_df['Country'].isin(top_countries)])
plt.title('Tuition Costs by Country (Top 10)')
plt.xlabel('Country')
plt.ylabel('Tuition (USD)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
# Scatter plot of Tuition vs. Living Cost Index
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Living_Cost_Index', y='Tuition_USD', data=intl_edu_df)
plt.title('Tuition vs. Living Cost Index')
plt.xlabel('Living Cost Index')
plt.ylabel('Tuition (USD)')
plt.show()

# Correlation Heatmap of numerical features
plt.figure(figsize=(10, 8))
correlation_matrix = intl_edu_df.select_dtypes(include=np.number).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", vmin=-1,
vmax=1)
plt.title('Correlation Heatmap of Numerical Features')
plt.show()

# Pair plot of a subset of numerical features
numerical_cols_for_pairplot = ['Tuition_USD', 'Living_Cost_Index', 'Duration_Years']
sns.pairplot(intl_edu_df[numerical_cols_for_pairplot])
plt.suptitle('Pair Plot of Selected Numerical Features', y=1.02)
plt.show()

!pip install scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Select features and target variable
features = ['Living_Cost_Index', 'Duration_Years']  # Using relevant numerical features
target = 'Tuition_USD'
X = intl_edu_df[features]
```

```
y = intl_edu_df[target]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the KNN Regressor model
knn_model = KNeighborsRegressor(n_neighbors=5) # You can adjust the number of
neighbors (k)
knn_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn = knn_model.predict(X_test)

# Evaluate the model
mse_knn = mean_squared_error(y_test, y_pred_knn)
r2_knn = r2_score(y_test, y_pred_knn)

print(f"KNN Regressor - Mean Squared Error: {mse_knn}")
print(f"KNN Regressor - R-squared: {r2_knn}")

# Optional: Visualize actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_knn, alpha=0.5)
plt.xlabel("Actual Tuition (USD)")
plt.ylabel("Predicted Tuition (USD)")
plt.title("KNN Regressor: Actual vs. Predicted Tuition Costs")
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2) # Plotting a diagonal line
plt.show()

#  k-Means Clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
# Select features for clustering
features_for_clustering = ['Tuition_USD', 'Living_Cost_Index', 'Duration_Years']
X_cluster = intl_edu_df[features_for_clustering]
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_cluster)
# Determine the optimal number of clusters using the Elbow method
inertia = []
k_range = range(1, 11)  # Trying k from 1 to 10

for k in k_range:
  kmeans = KMeans(n_clusters=k, random_state=42, n_init=10) # Set n_init to avoid warning
  kmeans.fit(X_scaled)
  inertia.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.xticks(k_range)
plt.grid(True)
plt.show()

# Let's assume we choose k=4 for this example
optimal_k = 4

# Apply K-Means clustering with the chosen number of clusters
```

```
kmeans_final = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
intl_edu_df['Cluster'] = kmeans_final.fit_predict(X_scaled)

# Analyze the clusters
print("\nCluster distribution:")
print(intl_edu_df['Cluster'].value_counts())

# View the characteristics of each cluster (e.g., mean of features within each cluster)
print("\nCluster characteristics (mean values):")
print(intl_edu_df.groupby('Cluster')[features_for_clustering].mean())

# Visualize the clusters (example using a scatter plot with two features and color-coding by
cluster)
# You might need to choose the two most relevant features for visualization
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Tuition_USD', y='Living_Cost_Index', hue='Cluster', data=intl_edu_df,
palette='viridis', legend='full')
plt.title(f'K-Means Clustering (k={optimal_k})')
plt.xlabel('Tuition (USD)')
plt.ylabel('Living Cost Index')
plt.show()

# Select categorical columns
categorical_cols = intl_edu_df.select_dtypes(include='object').columns

# Analyze the distribution of categorical features within each cluster
for col in categorical_cols:
   print(f"\nDistribution of {col} within Clusters:")
   cluster_distribution =
intl_edu_df.groupby('Cluster')[col].value_counts().unstack(fill_value=0)
   print(cluster_distribution)
```

```
# Optionally, visualize the distributions using bar plots for selected categorical features and
clusters

# Visualize the distribution of 'Level' within clusters
plt.figure(figsize=(10, 6))
sns.countplot(x='Cluster', hue='Level', data=intl_edu_df, palette='viridis')
plt.title('Distribution of Level within Clusters')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.show()

# Visualize the distribution of 'Program' within clusters for the top N programs
top_programs = intl_edu_df['Program'].value_counts().nlargest(5).index
plt.figure(figsize=(14, 7))
sns.countplot(x='Cluster', hue='Program',
data=intl_edu_df[intl_edu_df['Program'].isin(top_programs)], palette='viridis')
plt.title('Distribution of Top Programs within Clusters')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.legend(title='Program', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

# Visualize the distribution of 'Country' within clusters for the top N countries
top_countries = intl_edu_df['Country'].value_counts().nlargest(5).index
plt.figure(figsize=(14, 7))
sns.countplot(x='Cluster', hue='Country',
data=intl_edu_df[intl_edu_df['Country'].isin(top_countries)], palette='viridis')
plt.title('Distribution of Top Countries within Clusters')
plt.xlabel('Cluster')
plt.ylabel('Count')
plt.legend(title='Country', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

# --- Future Prediction with Plots ---

# You can change these values to represent different future situations
future_scenarios = pd.DataFrame({
    'Living_Cost_Index': [120, 150, 90],
    'Duration_Years': [4, 3, 5]
})

print("\nHypothetical Future Scenarios:")
print(future_scenarios)

# Note: The scaler was fitted on X_cluster for clustering, but we should ideally fit a scaler on
X_train
# Let's refit a scaler specifically for the features used in regression
scaler_reg = StandardScaler()
X_train_scaled_reg = scaler_reg.fit_transform(X_train)
future_scenarios_scaled = scaler_reg.transform(future_scenarios)

# So we use the original future_scenarios DataFrame for prediction with the model trained on
unscaled data
future_predictions = knn_model.predict(future_scenarios)

print("\nPredicted Tuition Costs for Future Scenarios (USD):")
print(future_predictions)

# Visualize the original data points and the future predictions
```

```
plt.figure(figsize=(12, 7))

# Plot original training data
plt.scatter(X_train['Living_Cost_Index'], y_train, alpha=0.6, label='Training Data',
color='skyblue')
plt.scatter(X_test['Living_Cost_Index'], y_test, alpha=0.6, label='Test Data',
color='lightcoral')


# Plot the future prediction points
plt.scatter(future_scenarios['Living_Cost_Index'], future_predictions, color='red', marker='X',
s=150, label='Future Predictions', zorder=5)


# Add labels and title
plt.xlabel('Living Cost Index')
plt.ylabel('Tuition (USD)')
plt.title('Tuition Prediction for Future Scenarios (based on Living Cost Index)')
plt.legend()
plt.grid(True)
plt.show()

# You can create another plot showing the relationship with Duration_Years if needed
plt.figure(figsize=(12, 7))

# Plot original training data
plt.scatter(X_train['Duration_Years'], y_train, alpha=0.6, label='Training Data',
color='skyblue')
plt.scatter(X_test['Duration_Years'], y_test, alpha=0.6, label='Test Data', color='lightcoral')

# Plot the future prediction points
```

```
plt.scatter(future_scenarios['Duration_Years'], future_predictions, color='red', marker='X',
s=150, label='Future Predictions', zorder=5)

# Add labels and title
plt.xlabel('Duration (Years)')
plt.ylabel('Tuition (USD)')
plt.title('Tuition Prediction for Future Scenarios (based on Duration Years)')
plt.legend()
plt.grid(True)
plt.show()

# Clean and analyze the distribution of categorical features within clusters
categorical_cols = intl_edu_df.select_dtypes(include='object').columns

for col in categorical_cols:
  print(f"\nDistribution of {col} within Clusters:")
  cluster_distribution =
intl_edu_df.groupby('Cluster')[col].value_counts().unstack(fill_value=0)
cluster_distribution

# Using dataframe cluster_distribution:

import altair as alt
# Melt the dataframe to long format for plotting
melted_cluster_distribution = cluster_distribution.reset_index().melt(
    id_vars='Cluster',
    var_name='Education Level',
    value_name='Count'
)
# Create a stacked bar chart
chart = alt.Chart(melted_cluster_distribution).mark_bar().encode(
    x='Cluster:O',  # Make Cluster an ordinal variable for discrete bars
    y='Count:Q',
    color='Education Level:N',
    tooltip=['Cluster', 'Education Level', 'Count']
).properties(
    title='Distribution of Education Levels by Cluster'
)
chart
```

Code for linkedin job posting dataset

```
!pip install pandas
!pip install numpy
!pip install matplotlib
!pip install seaborn

# Import libraries for data analysis and visualization
import pandas as pd         # Data manipulation
import numpy as np          # Numerical operations
import matplotlib.pyplot as plt  # Plotting
import seaborn as sns       # Statistical data visualization

from google.colab import drive
drive.mount('/content/drive')

def load_data(file_path):
  file_path = '/content/drive/My Drive/linkdein job posting/postings.csv'
  df = pd.read_csv(file_path)
  return df

#  preivew the data set
```

```
df = load_data('/content/drive/My Drive/linkdein job posting/postings.csv')
print(df.head(20))

# location wise linkedin job postings count
location_counts = df['location'].value_counts()
display(location_counts.head())

#  data types to understand what kind of plots are appropriate
print(df.info())

# plot1: Relationship between work_type and job type (if both columns exist)
if 'work_type' in df.columns and 'formatted_work_type' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.countplot(data=df, x='work_type', hue='formatted_work_type')
    plt.title('formatted_work_type Distribution by work_type')
    plt.xlabel('Experience Level')
    plt.ylabel('Number of Postings')
    plt.tight_layout()
    plt.show()

# plot2: Top N companies by number of postings
if 'company_name' in df.columns:
    top_companies = df['company_name'].value_counts().nlargest(10) # Adjust N as needed
    plt.figure(figsize=(10, 6))
    sns.barplot(x=top_companies.values, y=top_companies.index)
    plt.title('Top 10 Companies by Number of Job Postings')
    plt.xlabel('Number of Postings')
    plt.ylabel('Company Name')
    plt.tight_layout()
    plt.show()
else:
    print("'company_name' column not found.")

# Visualize the top N job titles
if 'title' in df.columns:
    top_n_titles = 10 # Adjust N as needed
    top_titles = df['title'].value_counts().nlargest(top_n_titles)


    plt.figure(figsize=(12, 6))
    sns.barplot(x=top_titles.values, y=top_titles.index, palette='viridis')
    plt.title(f'Top {top_n_titles} Job Titles by Number of Postings')
    plt.xlabel('Number of Postings')
    plt.ylabel('Job Title')
    plt.tight_layout()
    plt.show()
else:
    print("'title' column not found.")

# Visualize the distribution of work types by location for the top N locations
if 'work_type' in df.columns and 'location' in df.columns:
    top_n_locations = 10 # Use the same N as the previous location plot

    # Get the top N locations
    top_locations = df['location'].value_counts().nlargest(top_n_locations).index.tolist()

    # Filter the DataFrame to include only the top locations
    df_top_locations = df[df['location'].isin(top_locations)]

    plt.figure(figsize=(14, 7))
    sns.countplot(data=df_top_locations, y='location', hue='work_type', order=top_locations,
palette='viridis')
    plt.title(f'Distribution of Work Types in Top {top_n_locations} Locations')
    plt.xlabel('Number of Postings')
    plt.ylabel('Location')
    plt.tight_layout()
    plt.show()
else:
    print("Cannot plot work type distribution by location (one or both columns missing).")

# Analyze and visualize salary distribution by location for top N locations

if 'location' in df.columns and ('min_salary' in df.columns or 'max_salary' in df.columns or
'med_salary' in df.columns):
    top_n_locations = 10 # Use the same N as before

    # Get the top N locations
    top_locations = df['location'].value_counts().nlargest(top_n_locations).index.tolist()

    # Filter the DataFrame to include only the top locations and relevant salary columns
    salary_cols = ['min_salary', 'med_salary', 'max_salary']
    available_salary_cols = [col for col in salary_cols if col in df.columns]

    if available_salary_cols:
        df_top_locations_salary = df[df['location'].isin(top_locations)].copy()

        # Calculate average of available salary columns for each location
        # Handle cases where only one salary column is available
        if len(available_salary_cols) > 1:
            location_salary_stats =
df_top_locations_salary.groupby('location')[available_salary_cols].mean().reset_index()
```

```
    elif len(available_salary_cols) == 1:
        location_salary_stats =
df_top_locations_salary.groupby('location')[available_salary_cols[0]].mean().reset_index()
        location_salary_stats =
location_salary_stats.rename(columns={available_salary_cols[0]: 'average_salary'})



    if not location_salary_stats.empty:
        # Melt the DataFrame for easier plotting with seaborn
        if len(available_salary_cols) > 1:
            location_salary_melted = location_salary_stats.melt(id_vars='location',
                                                    value_vars=available_salary_cols,
                                                    var_name='salary_type',
                                                    value_name='average_salary')
        elif len(available_salary_cols) == 1:
             location_salary_melted = location_salary_stats.melt(id_vars='location',
                                                    value_vars=['average_salary'],
                                                    var_name='salary_type',
                                                    value_name='average_salary')



        plt.figure(figsize=(14, 7))
        sns.barplot(data=location_salary_melted, x='average_salary', y='location',
hue='salary_type', palette='viridis', order=top_locations)
        plt.title(f'Average Salary by Type in Top {top_n_locations} Locations')
        plt.xlabel('Average Salary')
        plt.ylabel('Location')
        plt.tight_layout()
        plt.show()
    else:
        print("No salary data available for the top locations.")
    else:
        print("No valid salary columns found ('min_salary', 'max_salary', or 'med_salary').")

else:
    print("Cannot plot salary distribution by location (location column or salary columns
missing).")

#  Correlation heatmap
# Select only numerical columns for the correlation matrix
numerical_df = df.select_dtypes(include=[np.number])

if not numerical_df.empty:
    plt.figure(figsize=(10, 8))
    sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm', fmt=".2f", vmin = -1,
vmax = 1)
    plt.title('Correlation Heatmap of Numerical Features')
    plt.tight_layout()
    plt.show()
else:
    print("No numerical columns found for the correlation heatmap.")

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
# For predict 'work_type' based on other features.
features = ['location', 'company_name', 'title']
target = 'work_type'

# Drop rows where any of the selected features or target are missing
df_model = df[features + [target]].dropna().copy()

# Encode categorical features
categorical_features = ['location', 'company_name', 'title']
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(handle_unknown='ignore'), categorical_features)],
    remainder='passthrough') # Keep other columns if any

# Encode the target variable
le = LabelEncoder()
df_model[target] = le.fit_transform(df_model[target])

# Split data into training and testing sets
X = df_model[features]
y = df_model[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
stratify=y) # Stratify for balanced classes

# --- Model Training with Epochs and Accuracy Tracking ---

n_epochs = 50 # Define the number of epochs

# Random Forest with Epochs
```

```
rf_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier', RandomForestClassifier(n_estimators=100,
random_state=42))])
print("Training Random Forest Model...")
rf_pipeline.fit(X_train, y_train)

rf_train_accuracy = rf_pipeline.score(X_train, y_train)
rf_val_accuracy = rf_pipeline.score(X_test, y_test)

print(f"Random Forest Training Accuracy: {rf_train_accuracy:.4f}")
print(f"Random Forest Validation Accuracy: {rf_val_accuracy:.4f}")
print("Random Forest Classification Report:\n", classification_report(y_test,
rf_pipeline.predict(X_test), target_names=le.classes_))
print("Random Forest Confusion Matrix:\n", confusion_matrix(y_test,
rf_pipeline.predict(X_test)))



print("\nTraining Logistic Regression Model...")
lr_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier', LogisticRegression(max_iter=1000, random_state=42))])

lr_pipeline.fit(X_train, y_train)

lr_train_accuracy = lr_pipeline.score(X_train, y_train)
lr_val_accuracy = lr_pipeline.score(X_test, y_test)



print(f"Logistic Regression Training Accuracy: {lr_train_accuracy:.4f}")
print(f"Logistic Regression Validation Accuracy: {lr_val_accuracy:.4f}")
print("Logistic Regression Classification Report:\n", classification_report(y_test,
lr_pipeline.predict(X_test), target_names=le.classes_))
```

```
print("Logistic Regression Confusion Matrix:\n", confusion_matrix(y_test,
lr_pipeline.predict(X_test)))



# --- Plotting Accuracies ---
# Since we are not training with explicit epochs for these models in this manner,
# we will plot the single final accuracy for comparison.
models = ['Random Forest', 'Logistic Regression']
accuracies = [rf_val_accuracy, lr_val_accuracy] # Using validation accuracy for comparison

plt.figure(figsize=(10, 5))
sns.barplot(x=models, y=accuracies, palette='viridis')
plt.ylim(0, 1) # Accuracy is between 0 and 1
plt.title('Model Validation Accuracy Comparison')
plt.ylabel('Accuracy')
plt.show()

# Plot Confusion Matrices with better labels
fig, axes = plt.subplots(1, 2, figsize=(14, 6)) # Increased figure width

# Get the class labels
class_labels = le.classes_

# Plot for Random Forest
sns.heatmap(confusion_matrix(y_test, rf_pred), annot=True, fmt='d', cmap='Blues',
ax=axes[0],
            xticklabels=class_labels, yticklabels=class_labels)
axes[0].set_title('Random Forest Confusion Matrix')
axes[0].set_xlabel('Predicted Label')
axes[0].set_ylabel('True Label')
axes[0].tick_params(axis='x', rotation=45) # Rotate x-axis labels
```

```
axes[0].tick_params(axis='y', rotation=0)  # Keep y-axis labels horizontal


# Plot for Logistic Regression
sns.heatmap(confusion_matrix(y_test, lr_pred), annot=True, fmt='d', cmap='Blues',
ax=axes[1],
            xticklabels=class_labels, yticklabels=class_labels)
axes[1].set_title('Logistic Regression Confusion Matrix')
axes[1].set_xlabel('Predicted Label')
axes[1].set_ylabel('True Label')
axes[1].tick_params(axis='x', rotation=45) # Rotate x-axis labels
axes[1].tick_params(axis='y', rotation=0)  # Keep y-axis labels horizontal



plt.tight_layout()
plt.show()


# Analyze work type distribution by year
if 'listed_year' in df.columns and 'work_type' in df.columns:
    yearly_worktype_counts = df.groupby(['listed_year',
'work_type']).size().reset_index(name='count')


    plt.figure(figsize=(12, 6))
    sns.barplot(data=yearly_worktype_counts, x='listed_year', y='count', hue='work_type',
palette='viridis')
    plt.title('Work Type Distribution by Year')
    plt.xlabel('Year')
    plt.ylabel('Number of Postings')
    plt.xticks(rotation=45)
    plt.legend(title='Work Type', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.show()
```

```
else:
    print("Required columns for yearly work type analysis not found.")

# Analyze location distribution by year for the top N locations
if 'listed_year' in df.columns and 'location' in df.columns:
    # Get the top N locations overall
    top_n_locations_overall = 10 # Adjust N as needed
    top_locations_list =
df['location'].value_counts().nlargest(top_n_locations_overall).index.tolist()

    # Filter the DataFrame to include only the top locations
    df_top_locations = df[df['location'].isin(top_locations_list)].copy()

    # Group by year and location, and count the occurrences
    yearly_location_counts = df_top_locations.groupby(['listed_year',
'location']).size().reset_index(name='count')

    plt.figure(figsize=(14, 7))
    sns.barplot(data=yearly_location_counts, x='listed_year', y='count', hue='location',
palette='viridis')
    plt.title(f'Location Distribution by Year for Top {top_n_locations_overall} Locations')
    plt.xlabel('Year')
    plt.ylabel('Number of Postings')
    plt.xticks(rotation=45)
    plt.legend(title='Location', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.show()
else:
    print("Required columns for yearly location analysis not found.")

# Convert 'listed_time' to datetime objects (it's in milliseconds)
```

```
df['listed_time'] = pd.to_datetime(df['listed_time'] / 1000, unit='s')

# Set 'listed_time' as the index
df.set_index('listed_time', inplace=True)

# Resample by day and count job postings
daily_postings = df.resample('D').size()

# Display the first few entries of the daily_postings time series
display(daily_postings.head())

import matplotlib.pyplot as plt

# Plot the forecast
fig = model.plot(forecast)
ax = fig.gca()
ax.set_title('Job Postings Forecast')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Postings')
plt.show()

# Plot the forecast components (trend, weekly seasonality)
fig2 = model.plot_components(forecast)
plt.show()

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam

# X_train_processed and X_test_processed are already created and processed in the previous
step
```

```
# Determine the input dimension after one-hot encoding
input_dim = X_train_processed.shape[1]

# Determine the number of output classes (le is already defined and fitted)
num_classes = len(le.classes_)

# Reinitialize the Sequential model with the correct input dimension
model = Sequential()

# Add the input layer and first hidden layer
model.add(Dense(128, activation='relu', input_shape=(input_dim,)))

# Add additional hidden layers
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))

# Add the output layer
model.add(Dense(num_classes, activation='softmax'))

# Recompile the model
optimizer = Adam(learning_rate=0.001) # Using Adam optimizer with a learning rate
model.compile(optimizer=optimizer,
          loss='sparse_categorical_crossentropy', # Suitable for label-encoded integer targets
          metrics=['accuracy'])

# Define the number of epochs and batch size
n_epochs = 10
batch_size = 60 # You can adjust this value
```

```
# Train the model
history = model.fit(X_train_processed, y_train,
                    epochs=n_epochs,
                    batch_size=batch_size,
                    validation_data=(X_test_processed, y_test),
                    verbose=1) # Set verbose to 1 to see training progress

import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy over Epochs')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss over Epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```