Teaching Assistants Demand Prediction

Andrew Lau

BSc Computer Science

Submission Date: 26th April 2024

Supervisor: Sarah Sanders

*This report is submitted as part requirement for the BSc Degree in Computer Science at UCL. It is substantially the result of my own work except where explicitly indicated in the text.*

*The report may be freely copied and distributed provided the source is explicitly acknowledged.*

Abstract

The recruitment of Postgraduate Teaching Assistants (PGTAs) around University College London (UCL) Computer Science Department has been an ambiguous process with no clear metrics that can predict demand accurately. Despite the obvious answers of number of students, workload of the module and the difficulty of the syllabus, there is no linear relation between these metrics and the demand of PGTAs after plotting graphs to look at the results. Experimenting with AI models and different statistical tool has also proved to be inaccurate due to the unpredictability nature of modules. Hence, this project research deeper into what creates demand for PGTAs and how it can be better predicted for future references.

**Contents List**

1. Introduction
   1. Aims & Goals
   2. Deliverables
2. Background Information
3. Requirements and Analysis
4. Design and Implementation
5. Results Evaluation
6. Conclusion

**1 Introduction**

Problem Statement

At University College London (UCL) Computer Science Department, the recruitment of Postgraduate Teaching Assistants (PGTAs) is done manually through a recruitment process filled with ambiguity around the demand for PGTAs. For modules that are new, there isn’t a tried-and-tested method or official guideline on how to measure demand for PGTAs and it is usually done based on a review of the module content and intuition. This often leads to inaccurate estimation of workforce needed and inefficient allocation of resources within the department. The research is initially done around features that directly affect the demand of PGTAs such as number of students enrolled, coursework to exam ratio, module difficulty, etc. However, due to inconsistent datasets and unpredictability, further research is done in the field of AI prediction models and Natural Language Processing to analyse the module content and structure in more detail.

**1.1 Aims & Goals**

The project aims to better understand the demand PGTAs through analytical tools that drives data-driven decisions in the recruitment process. The analytical tool hopes to bring new insights into PGTA recruitment to realise its potential for ensuring that resources are better allocated to improve the learning experiences of students at UCL. Here are the aims and goals that should be achieved by the end of the project:

* Learn about the problems behind PGTA recruitment.
* Determine the factors that affects the demand of PGTA by researching into the recruitment process.
* Process data from module information and past records to create a combined dataset for training machine learning models.
* Accurately predict the necessary number of PGTAs to hire for a new and unfamiliar module.

**1.2 Deliverables**

The project submission would include the source files a documented and functional Dash app including:

1. Dashboards and analytical tools built using Plotly, including different types of graphs that provide a visual representation of the dataset:
   1. PGTAs Requested vs PGTAs Recruited Graph
      * Provides insights into which modules have inaccurate estimation of PGTA demand and its scale
   2. Students Enrolled vs PGTAs Recruited Graph
      * To determine whether the number of students enrolled affect the number of PGTAs recruited for the academic year 2022-2023 (only data for one year can be found)
   3. Exam-Coursework Ratio vs PGTAs Recruited Graph
      * To determine whether the weight of coursework and exams affect the number of PGTAs recruited
   4. Module Delivery Code vs PGTAs Recruited Graph
      * To determine whether module delivery code (modules taken by students in different years of study. In other words, difficulty of the module) affect the number of PGTAs recruited
2. Machine learning models drawn from Scikit-learn libraries.
   1. Several different machine learning models has been trained and tested with different variants. The one that was chosen as the final prediction model is a ridge regression model with cross-validation. A more detailed analysis can be found at section \_\_\_

**2 Background Information**

**2.1 Background Research**

The project kicked-off with a brief of the problem statement given by my project supervisor, explaining the problem with ambiguity around the hiring of PGTAs.

**2.1.1 Technology Stack**

Selecting the technology stack for a project can be dependent on specific requirements of the project, such as the complexity of data, real-time processing needs, interactivity level, and the intended user’s technical proficiency. My supervisor will be the intender user after completion of this project and hence, the technology stack chosen must suit her technical proficiency and preferences. Hence, the main requirements of the technology stack include ease of use, Python based and freely available. A few different platforms that meet the main requirements was considered for the implementation of this project. Their strengths and weaknesses in different areas of the project are considered carefully to enable a successful implementation of this project. Here are the platforms considered and their evaluations:

Grafana

* Best for real-time monitoring and observability
* Specialises in complex time-series data visualisation
* Offers extensive customisation options through a wide range of plugins

Streamlit

* Ideal for quickly creating data apps, reports, and dashboards
* More suitable with static data with smaller datasets
* Easy to set up, configure and deploy with minimal infrastructure requirements
* Limited in terms of real-time capabilities and complex interactive features

Plotly (Dash)

* Focuses more on data visualisation
* Has better graphing tools
* Better suited for advanced interactivity

Jupyter Notebooks

* Interactive computing environment that supports live code, equations, visualizations, and narrative text.
* Highly popular in the data science community for exploratory data analysis and prototyping.
* Integrates well with Python libraries like Matplotlib, Seaborn, Plotly for visualization.

In conclusion, Streamlit is chosen for its quick and easy setup along with Dash for its extensive data visualisation tools to enable easy plotting of graphs. Jupyter Notebooks are used for its support in data analysis and suitability in training machine learning models. It also works and integrates well with Plotly.

**2.1.2 Data Sources**

* Write about how data are given, and where it has been collected, the ethics behind retaining data and accessing them.
* The data are sourced by my supervisor in the format of excel sheets. The data are collected and recorded manually when students signed up to be a teaching assistant.
* The data provided did not contain any sensitive/personal information and hence, the ethical report is not needed to address the ethical conducts within this project

1. **Analysis and Implementation**

This section discusses …

**3.1 Data Processing**

The provided dataset required extensive preprocessing due to the presence of missing values and data in inconsistent formats. The dataProcessing.py file includes a suite of functions crucial for data preparation, enabling analysis and modeling. Each function plays a role in transforming raw data into a structured format conducive to analytics and predictive modeling, ensuring the usability of the dataset. Here's a detailed look at these functions:

1. **no\_data\_modules (df, col1, col2)**

This function identifies modules with 'No data found' entries in specified columns. It returns a list of module codes where either of the two specified columns contains 'No data found'. It is used for listing modules with incomplete data in the PGTAs recruited and requested columns for a selected year.

1. **create\_combined\_variables\_df (df\_moduleAssessmentData, df\_capVsActualStudents, df\_requestedVsRecruited)**

Combines data from multiple DataFrames (df\_moduleAssessmentData, df\_capVsActualStudents, df\_requestedVsRecruited) to create a single DataFrame. This new DataFrame includes Module Code, Number of Students, PGTAs Recruited, Exam:Coursework Ratio, and Delivery Code. It filters modules based on their presence in all source DataFrames and sorts them by Exam:Coursework Ratio. It is used in plotting the graphs of each individual variables (Module Code, Number of Students, Exam:Coursework Ratio, and Delivery Code) against PGTAs recruited to gain insights on how each variables affect the number of PGTAs recruited.

1. **create\_coursework\_exam\_ratio\_column (df)**

This function adds a new column 'Exam:Coursework Ratio' to the DataFrame by merging and calculating the weights of exam and coursework from different columns. It then drops duplicates and handles missing values. This column is essential to create the ‘Exam:Coursework Ratio vs PGTAs Recruited Graph’

1. **split\_coursework\_exam\_ratio\_column (df)**

Splits the 'Exam:Coursework Ratio' column into two separate columns ('Exam Weight' and 'Coursework Weight') and removes the original ratio column. This is processed so that each of the coursework weights and exam weights are used as features for training the machine learning model.

1. **handle\_missing\_data (df, columns)**

Substitutes 'No data found' entries with 0 in specified columns and converts them to numeric format. This ensures data uniformity and prevents computational errors during analysis.

1. **handle\_nan\_data (df)**

Fills NaN (not a number) entries with zero, addressing the issue of missing data. This also ensures data uniformity and prevents computational errors during analysis.

1. **column\_sum (df, column)**

Calculates and returns the sum of values in a specified column of the DataFrame, providing aggregated data of total students, PGTAs hired and requested for each year.

1. **difference\_calculation (df, selected\_year)**

Computes the difference between requested and recruited PGTAs for a given year and adds this as a new column in the DataFrame. This feature is pivotal in evaluating recruitment effectiveness and identifying gaps in estimation of PGTA demand for each year.

1. **set\_color (df)**

Assigns a color based on the difference between requested and recruited PGTAs, with red colour indicating the demand was higher than expected while green colour indicates demand lower than expected, allowing easy data interpretation.

1. **load\_data (df)**

Prepares features and target variables for modeling. It one-hot encodes the 'Delivery Code' column, merges this with the rest of the DataFrame, and returns the feature matrix (X) and target vector (y). This is fundamental for feeding data into predictive models.

**3.2 Graph Analysis**

1. **PGTAs Requested vs PGTAs Recruited Graph**

This graph aims to provide insights into which modules have inaccurate estimation of PGTA demand and its scale.

**Analysis:** The final graph examines the relationship between module delivery codes and the number of PGTAs recruited. Variability across delivery codes could suggest that the nature of the module, potentially its complexity or the year of study it's catered to, might influence TA recruitment. However, the data presents a diverse pattern, suggesting a more intricate interplay of factors beyond just the delivery code.

1. **Students Enrolled vs PGTAs Recruited Graph**

This graph aims to determine whether the number of students enrolled affect the number of PGTAs recruited for the academic year 2022-2023 (only data for one year can be found)

**Analysis:** The final graph examines the relationship between module delivery codes and the number of PGTAs recruited. Variability across delivery codes could suggest that the nature of the module, potentially its complexity or the year of study it's catered to, might influence TA recruitment. However, the data presents a diverse pattern, suggesting a more intricate interplay of factors beyond just the delivery code.

1. **Exam-Coursework Ratio vs PGTAs Recruited Graph**

This graph aims to determine whether the weight of coursework and exams affect the number of PGTAs recruited.

**Analysis:** The final graph examines the relationship between module delivery codes and the number of PGTAs recruited. Variability across delivery codes could suggest that the nature of the module, potentially its complexity or the year of study it's catered to, might influence TA recruitment. However, the data presents a diverse pattern, suggesting a more intricate interplay of factors beyond just the delivery code.

1. **Module Delivery Code vs PGTAs Recruited Graph**

This graph aims to determine whether module delivery code (modules taken by students in different years of study. In other words, difficulty of the module) affect the number of PGTAs recruited.

**Analysis:** The final graph examines the relationship between module delivery codes and the number of PGTAs recruited. Variability across delivery codes could suggest that the nature of the module, potentially its complexity or the year of study it's catered to, might influence TA recruitment. However, the data presents a diverse pattern, suggesting a more intricate interplay of factors beyond just the delivery code.

**3.2.1 Findings and Evaluations**

* Across all visualizations, it is evident that predicting TA demand is multifaceted, with no single variable providing a complete understanding.
* The disparities in requested versus recruited PGTAs highlight the need for refined predictive models that can accommodate a range of influencing factors.
* The lack of a strong correlation in some graphs suggests that further qualitative data, such as module content analysis or instructor feedback, might be necessary to fully understand TA recruitment needs.
* These analyses underscore the complexity of academic resource planning and the potential for data-driven approaches to enhance decision-making processes.

**3.2.2 Conclusion**

The visualizations provide valuable insights into the factors influencing TA recruitment. However, they also highlight the complexities inherent in academic administration and the potential benefits of adopting comprehensive analytic strategies to improve resource allocation efficiency.

**3.3 Machine Learning Models**

**3.3.1 Linear Regression and Ridge Regression**

**Linear Regression** is a statistical method that models the relationship between a scalar dependent variable y and one or more independent variables (or 'predictors') X. The core idea is to obtain a linear equation that predicts y as accurately as possible from X. It is based on the ordinary least squares (OLS) estimator and assumes a linear relationship between the variables.

**Ridge Regression** is an extension of linear regression where the loss function is modified to minimize the complexity of the model. This is achieved by adding a penalty term to the OLS loss function, which is proportional to the square of the magnitude of the coefficients. Ridge Regression aims to prevent overfitting by penalizing large coefficients.

The linear\_regression and ridge\_regression functions defined in the script are designed to instantiate the respective regression models from the scikit-learn library.

**linear\_regression(X, y):** This function configures a RandomForestRegressor from the scikit-learn library with a fixed number of estimators and a set random state for reproducibility.

**ridge\_regression(X, y):** It sets up a Ridge regression model with a regularization strength alpha and a specified random state.

**train\_model(X, y, model\_type):** Depending on the model type ('ridge' or 'linear'), this function trains the respective regression model on the dataset.

**save\_model(model, filename):** This utility saves the trained model to the disk, allowing for later retrieval and inference.

Each function accepts the features X and the target variable y and returns a model object configured with a predetermined number of estimators (n\_estimators=100) and a random state (random\_state=42) to ensure reproducibility. The script proceeds to load and preprocess the dataset using various functions from the data\_processing module. After preparing the data, it trains the specified model, performs K-Fold cross-validation to estimate the model's performance, and saves the model. The save\_model function saves the trained model as a .pkl file to the specified filename, ensuring the model's persistence for future predictions. The function train\_model determines which regression model to train based on the model\_type parameter, which can be either 'ridge' or 'linear'.

**Results and Evaluation**

The performance of the two models was evaluated using a 5-fold cross-validation approach, with RMSE as the performance metric. The summarized results are as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model Type | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Mean RMSE | Standard Deviation |
| Ridge | 116.26 | 83.91 | 94.65 | 89.18 | 96.46 | 96.09 | 11.00 |
| Linear | 123.72 | 88.99 | 100.68 | 103.74 | 91.19 | 101.66 | 12.34 |

**Conclusion**

The Ridge Regression model outperformed the Linear Regression model with a lower mean RMSE, indicating better average performance across all folds. The standard deviation of the RMSE scores, which reflects the variability in model performance across different data splits, was also lower for the Ridge model, suggesting more consistent performance.

* 1. **Generalised Additive Model**

This section dives into another method of PGTA recruitment predictions by deploying a Generalized Additive Model (GAM). GAMs offer a more flexible approach to modeling complex, non-linear relationships in data compared to linear regression, which is more straightforward but less adaptable to non-linearity. The focus is on refining the dataset for predictive accuracy and assessing the model's performance using root mean square error (RMSE) as a metric. A GAM was constructed to predict PGTA requirements, considering factors such as student numbers and module delivery codes. The dataset was partitioned into training and testing subsets, maintaining a test size of 20%. The GAM is then fitted on the training data, leveraging spline functions to capture non-linear relationships.

**Results and Evaluation**

The analysis revealed a RMSE score of 131.68, higher than the mean RMSE of both linear and ridge regression models. This result indicates high inaccuracy of the model despite being trained on a non-linear dataset. In order to assess these results further, the next section compares GAMs and Linear Regression models.

**Generalized Additive Model (GAM) and Linear Regression Comparison**

The comparison between Generalized Additive Models (GAMs) and Linear Regression models is crucial in understanding the flexibility and applicability of these statistical techniques in various data analysis scenarios. Here's a detailed examination of their differences, along with evaluations:

|  |  |  |
| --- | --- | --- |
|  | GAM | Linear Regression |
| Flexibilty and Complexity | Offers the flexibility to model complex, non-linear relationships between predictors and the response variable.GAM manages complexity via smoothing functions and regularization | A structurally simpler model that assumes a linear relationship between these elements​​, which can be a limiting factor for accuracy if the true relationship is non-linear. |
| Regularization | Can inherently include regularization techniques to control model complexity | Requires external regularization methods like Ridge or Lasso |
| Sample Size Requirements | Often require larger sample sizes to capture non-linear trends effectively | Less demanding due to its simplicity​​ |
| Visualizations | Visualization of smooth functions in GAMs can be particularly helpful in understanding the relationships within the data | Difficult to be represented in graphs or other visual representations |

**Conclusion**

This research highlights the significance of conplex data processing and the effectiveness of GAMs in forecasting non-linear datasets. However, it is difficult to capture non-linear relationships and since the dataset it was trained on has only limited data, the desired accuracy cannot be achieved.

**3.5 Prediction Prompt**

The prediction prompt represents a critical component of the research project, bridging the gap between user input and sophisticated machine learning models. This section of the research paper documents the implementation of a user-friendly interface that allows for the input of relevant features to predict the number of Postgraduate Teaching Assistants (PGTAs) required. The system dynamically selects between linear and ridge regression models based on the user's preference, providing a seamless and interactive experience.

Implementation Details:

Model Selection:

Users can switch between linear regression and ridge regression models by altering the model\_type variable.

The selected model is loaded from the models directory using the load\_model function.

User Interface:

A web-based interface is implemented using Dash, a Python framework for building analytical web applications.

The interface includes input fields for 'Number of Students', 'Exam Weight', 'Coursework Weight', and 'Delivery Code'.

A 'Predict' button initiates the prediction process.

Data Preparation:

User inputs are captured and formatted into a DataFrame that matches the model's expected input structure.

The 'Delivery Code' is transformed into dummy variables to align with the model’s training data.

Missing columns in the input data (not present during model training) are filled with zeros to maintain consistency.

Prediction Process:

The formatted input data is passed to the machine learning model for prediction.

The model outputs the predicted number of PGTAs required, which is displayed to the user.

Error Handling and Validation:

The system includes mechanisms to handle incorrect or incomplete input, ensuring robustness.

Input validation ensures data integrity and relevance for accurate predictions.

Evaluation and User Experience:

The prediction prompt has been designed with an emphasis on usability and accessibility, catering to users with varying levels of technical expertise.

The interface is intuitive, guiding the user through the input process with clear placeholders and an easy-to-navigate layout.

The system’s flexibility in model selection allows users to experiment with different predictive algorithms, enhancing the educational and exploratory aspects of the project.

Conclusion:

The prediction prompt serves as an effective tool for demonstrating the practical application of machine learning models in educational administration. Its user-centric design and integration of advanced predictive algorithms exemplify the project’s commitment to innovative and accessible data-driven solutions. Future enhancements may include the incorporation of real-time data updates and further customization options to cater to diverse user preferences and emerging analytical requirements.

**4 Results Evaluation**

**4.1 Cross-validation (CV)**

* A model usually creates a prediction function from the same set of data, which might lead to a bias towards that dataset. This might produce inaccuracies when prompted with un-seen data (a situation called overfitting).
* Cross-validation solves this problem by using the K-fold strategy. The dataset is split into k smaller datasets and the model is trained and tested k times on k-1 folds and 1-fold respectively. E.g. a CV with 5 folds running on a dataset of 100 data will produce 20 data per fold.
* Cross-validation is computationally done using the cross\_val\_score helper function on the dataset by splitting the data, fitting a model, and computing the score for k number of consecutive times (with different splits each time). The Root Mean Squared Error (RMSE) is used as a performance metric.
* The results of each of the selected number of folds are documented below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K - folds | Linear Regression | | Ridge Regression | |
| Mean RSME | Standard Deviation | Mean RSME | Standard Deviation |
| 5 | 101.66 | 12.34 | 96.09 | 11.00 |
| 10 | 101.00 | 24.29 | 93.96 | 22.67 |
| 15 | 99.06 | 25.88 | 92.84 | 28.69 |
| 20 | 99.10 | 30.38 | 92.24 | 30.91 |

* Low Mean RMSE: This indicates that the model, on average, has a low prediction error, directly relates to the model's predictive power.
* Low Standard Deviation: This suggests that the model's performance is consistent across different subsets of the dataset, crucial for ensuring that the model is reliable and not just performing well on specific types of data.

**Conclusion**

Considering the trade-offs between accuracy and consistency, a model with a balanced mean RMSE and standard deviation is chosen for the context of predicting PGTAs to hire. Hence, the optimal model is the ridge regression model with 5-fold cross-validation.

**5 Conclusion**

**6 References**

1. Analytics Vidhya. (2023). Understanding Generalized Additive Models (GAMs): A Comprehensive Guide. Retrieved from <https://www.analyticsvidhya.com/blog/2023/09/understanding-generalized-additive-models-gams-a-comprehensive-guide/>. 24th January 2024