**Report: Predicting First-Hour Job Applicants Using Regression Techniques**

**Executive Summary**

For my midterm project, I created a prediction model using four types of regression techniques in RapidMiner: Linear, Ridge, Lasso, and Support Vector Regression (SVR). The goal of this model was to predict how many applicants apply to a job post on LinkedIn within the first hour based on time-related variables such as posting time, day of the week, and categorized time blocks. In the end, the Linear Regression model returned the best combination of R² and RMSE on the test data with an R² of 0.869 and RMSE of 12.829. However, Ridge Regression also performed competitively, suggesting strong reliability for generalization.

**Business Understanding**

When thinking about relevant problems, I noticed a pattern while applying for jobs—some job postings on LinkedIn had a huge number of applicants within minutes, while others barely had any. This made me wonder: "Does the time and day of posting affect how quickly people apply?" This led to the core question: "Can we predict the number of applicants within the first hour based on the job's posting time and day?" This is a real-world problem for recruiters and companies aiming to maximize visibility. With this in mind, I decided to apply machine learning to discover the optimal time to post jobs for higher applicant engagement.

**Data Understanding**

To ensure that no posting was missed during data collection, I set a recurring reminder every hour (Figure – 1) on my phone to manually log new job posts and track how many applicants each post received at the 1-hour, 2-hour, and 2+ hour marks. This helped maintain the consistency and accuracy of the data by minimizing forgetfulness and standardizing the observation intervals.

The data collection was conducted directly through LinkedIn’s job search platform using filters to target only relevant posts. I specifically applied filters for the keyword **"Business Analyst"**, selected **"Entry Level"** under experience, and filtered posts from the **last 24 hours (Figure -2)**. The base URL used for this filtered search was:

https://www.linkedin.com/jobs/search/?currentJobId=4207410557&distance=25.0&f\_E=2&f\_TPR=r86400&geoId=103644278&keywords=business%20analyst&origin=JOB\_SEARCH\_PAGE\_JOB\_FILTER

However, to focus solely on posts that were added within the **last hour**, I modified the r86400 value (which represents 24 hours in seconds) to r3600 (representing 1 hour).

https://www.linkedin.com/jobs/search/?currentJobId=4207410557&distance=25.0&f\_E=2&f\_TPR=r3600&geoId=103644278&keywords=business%20analyst&origin=JOB\_SEARCH\_PAGE\_JOB\_FILTER

This small change allowed me to efficiently gather only the most recently posted jobs, improving both the relevance and precision of the data. I manually collected job posting data for entry-level Business Analyst roles on LinkedIn over several days (Figure – 3). I created an Excel sheet to log all relevant information: company name, job title, location, date and time of posting, and the number of applicants within 1 hour, 2 hours, and after 2 hours. I also created a new variable, "Time Blocks," to categorize the posting time into segments like early morning, mid-day, evening, and late night. Days of the week were converted into numbers (e.g., 1 = Sunday, 2 = Monday, ..., 7 = Saturday) for easier processing later in RapidMiner. The final dataset had fewer than 95 observations.

A correlation matrix revealed strong relationships. For example, there was a 0.920 correlation between applicants in the first hour and within two hours, while Time Blocks had a -0.529 correlation with applicants in the first hour, suggesting fewer applications during late hours.

**Data Preparation**

Once I had all the data structured in Excel, I prepared it for modeling. I first converted all categorical values (like Time Blocks and Day of the Week) into numeric format. Then, I imported the dataset into RapidMiner. In RapidMiner, I began by selecting the key attributes relevant to prediction. Using the "Select Attributes" operator, I chose Time Track, Day of Week, and Time Blocks as predictors, and "Applicants within 1 hour" as the target. I used the "Set Role" operator to designate this target variable. After that, I added a "Cross Validation" operator to ensure robust model evaluation and reduce overfitting. Each model was wrapped inside the validation loop using "Apply Model" and "Performance" operators, which enabled me to measure metrics like R, R², and RMSE for both training and test sets.

**Modeling**

Once the data was cleaned and structured, I proceeded to model building using four regression algorithms. The goal was to identify which method best predicted the number of applicants within the first hour of a job posting. I started with Linear Regression, a basic model that assumes a linear relationship between the variables. Then, I tested Ridge Regression, which adds a penalty term to the loss function to handle multicollinearity and prevent overfitting. I also used Lasso Regression, which penalizes coefficients more aggressively and can shrink some to zero, effectively selecting features. Finally, I tried Support Vector Regression (SVR), which attempts to fit the data within a specified error margin (epsilon) and is more sensitive to non-linear relationships.

Since the dataset was small, I implemented Cross Validation in RapidMiner to ensure fair evaluation. Every model followed the same pipeline of partitioning, training, testing, and scoring. I used "Apply Model" to generate predictions and "Performance" to compute R, R², and RMSE metrics.

**Interpretation**

* **Linear Regression** had the best overall balance of high R² and low RMSE on the test set. (Figure – 4)
* **Ridge Regression** is a strong alternative with slightly lower R² but more consistent regularization, making it a robust choice, especially for small datasets with collinearity. It had the lowest RMSE on training data and second-lowest on test data, showing it generalizes well. (Figure – 5)
* **SVR** produced a good R² but had the highest RMSE, which means while the fit looked good statistically, the error margins were larger than other models.
* **Lasso Regression** was useful for its ability to reduce less important variables, but it didn’t outperform Linear or Ridge in this case 9 (Figure – 6 & 7).

**Deployment / Conclusion**

Based on the model insights, the best time to post a job to maximize early applicants is early morning to mid-day, while evening and late-night postings result in lower applicant numbers. This predictive model offers a simple yet impactful way for recruiters to strategically plan their job postings.

Although Linear Regression gave the highest R² with a low RMSE, Ridge Regression showed consistent performance across both training and test data and slightly better RMSE on training, making it a reliable model for generalization, especially if new features are added later.

**Recommendations**

* Use Linear Regression when interpretability and speed are the priority.
* Use Ridge Regression when you anticipate more variables or want to guard against overfitting.
* Scale this model for other roles, industries, or platforms.
* Experiment with ensemble methods like Random Forests or gradient boosting, or even deep learning models if more data is available in the future.

Appendix

Figure – 1 Figure – 2

A screenshot of a phone

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Figure – 3

A table of numbers and names

AI-generated content may be incorrect.

Figure – 4

A diagram of a project

AI-generated content may be incorrect.

Figure – 5 A screenshot of a computer

AI-generated content may be incorrect.

Figure – 6 Figure – 7

A screenshot of a table

AI-generated content may be incorrect. A screenshot of a data sheet

AI-generated content may be incorrect.