Executive Summary by Elke Hansen

Webscraping JOB POSTINGS

# OVERVIEW

This mission of this project is to scrape data from a job posting website and use it to create several different mathematical models to be able to predict salary and job titles of data related roles. Using analysis of the words used within each listing as well as any other information that might be of use, the aim is select the most important features that inform each target. The main steps involved were Site Selection, Web Scraping, Data Cleaning, Salary Prediction and Job Title Classification.

# Site Selection

In choosing which website to scrape, a couple of different aspects were taken into account. The main concern was to find a website that would have enough salary information to be able to use it to predict salary, another concern was to make sure it was a site that could have each piece scraped with relative ease. As I could not find any websites that had consistent salary information, I decided to go with Seek.com.au for two main reasons. One being that the sheer volume of ads on Seek outweigh similar Australian job sites and the other, most important, reason being that Seek has a career insights section with salary estimates min, mode and max.

# WEB SCRAPING

After selecting Seek as my preferred website, the first step was to look at one page of around 20 job ads. From these, I found the different pieces of code that I needed. I found that some of the pieces of information were relatively easy to pull out while others were a little more complicated as they were clustered with multiple pieces of information and had to be selected individually. I had to go through a lot of trial and error until I had code to pull out all the information I wanted for these first 20 pages, I then used this code to go through a further 4,300 posts. The final scraping process code ran for ~ 3 hours.

# DATA CLEANING

The key steps in cleaning this particular data was just to go through each column to check it was in a format that I needed. I then had to figure out how to impute the salary as only just over 400 of the 4,363 job scraped had usable salary information. For this I decided to take the salary estimators that I had pulled out and compare the mean and the mode of these to the real salaries that I did actually have. I selected the mean of the estimators to be the better of the two because it was evenly distributed, like the actual salaries and the overall mean was also almost exactly the same as the actual salaries. The biggest issue I face with using this mean of estimated salary is that I don’t know exactly what information Seek has used to calculate it. From what I can tell on the website, it is some kind of combination of City and Job Title which potentially could introduce some bias into the data.

# PREDICTING SALARY

Using natural language processing, namely a count vectorizer, as the main analytic tool, I tried several different approaches to the problem of predicting salary:

### Linear Models: I started with trialing various regression models before I quickly realized they would not get very good results, none could get an accuracy of 50%

### Bucketing Salary: Once I decided to bucket the salary into 5 categories, I started getting better results off the Role Description column alone. The best of these models produced an accuracy of 56.8%, up from a baseline of 36.1%

### Combining All Text: After still not being happy with the scores I was getting, I decided to combine all the text columns together to be analysed as one block of text. The best two of these models were a Logistic Regression (62%) and a Linear Support Vector Classifier (63%).

### Feature Reduction: I decided to take the two best working models and see if I could improve them using feature reduction techniques such as Truncated SVD and Select K Best using a chi squared distribution. Neither of these methods improved the score.

I chose the Linear SVC as my final model and once I used it to create a list of the top twenty words used to predict each of the 5 classifications, it was easy to a pattern within these and they made sense as far as my domain knowledge is concerned.

# JOB TITLE CLASSIFICATION

For the task of classifying job titles, I took a bit of an unorthodox approach. I started by using a Linear Dirichlet Allocation to see if it would find topics within the role descriptions without being told specifically what it was looking for. After looking at the interactive visualization, I could see that there were some definite themes within these allocated topics. Once I had this model working, I used it to assign these labels to each of the roles. I then used these labels as the target to try to predict using the words within the job title, industry and sub industry columns. Again, the best model I got working was a logistic regression using a basic count vectorizer and a TFIDF transformer. This produced an accuracy of 63% above a baseline of 32.3% so does not perform too badly. Once I had selected this model, I compared the 5 top job title features to the top ten role description features for each of the 5 topics. While some of the topics do make sense, I’m not convinced that this was the right approach to take on this problem.