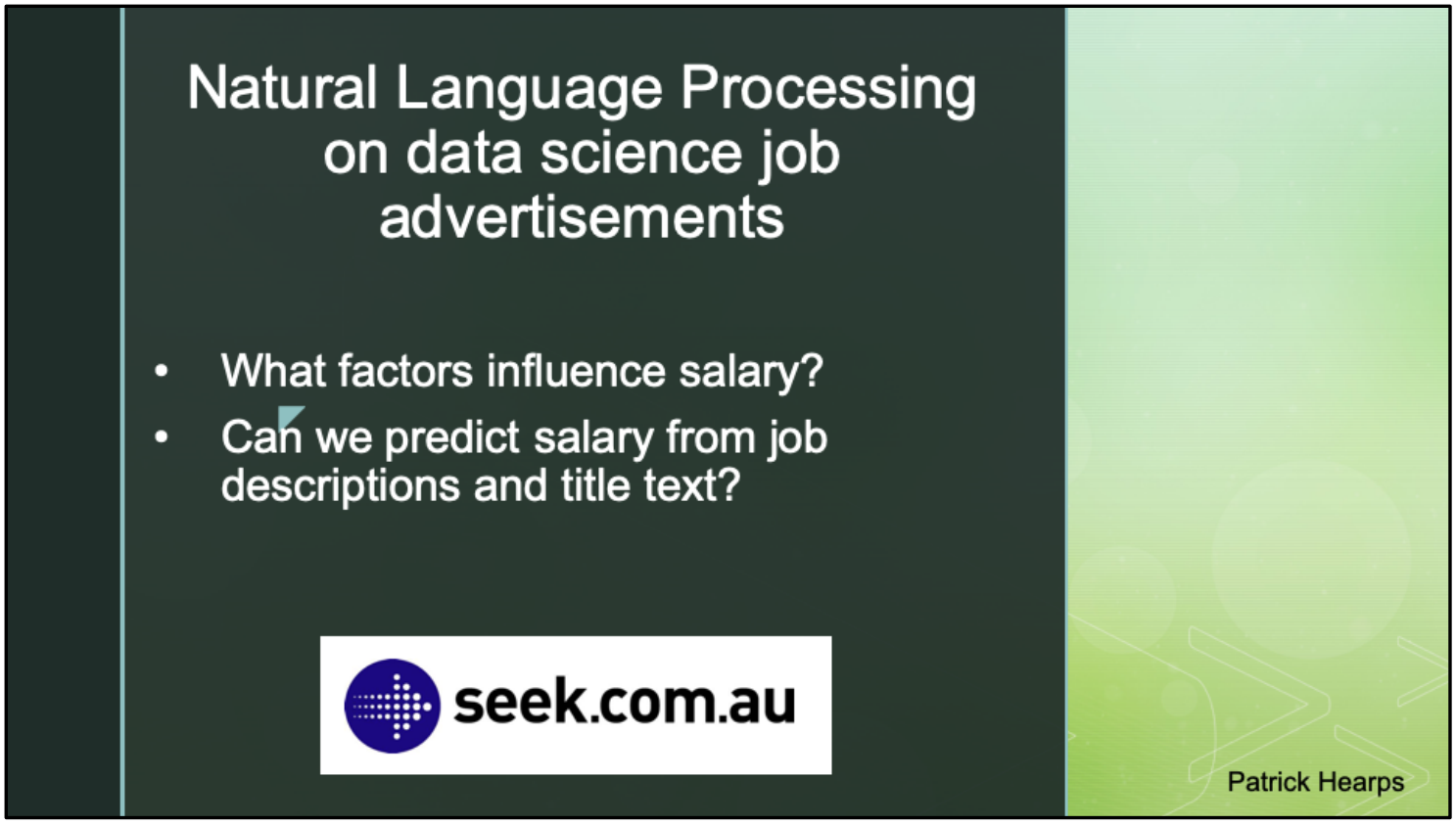
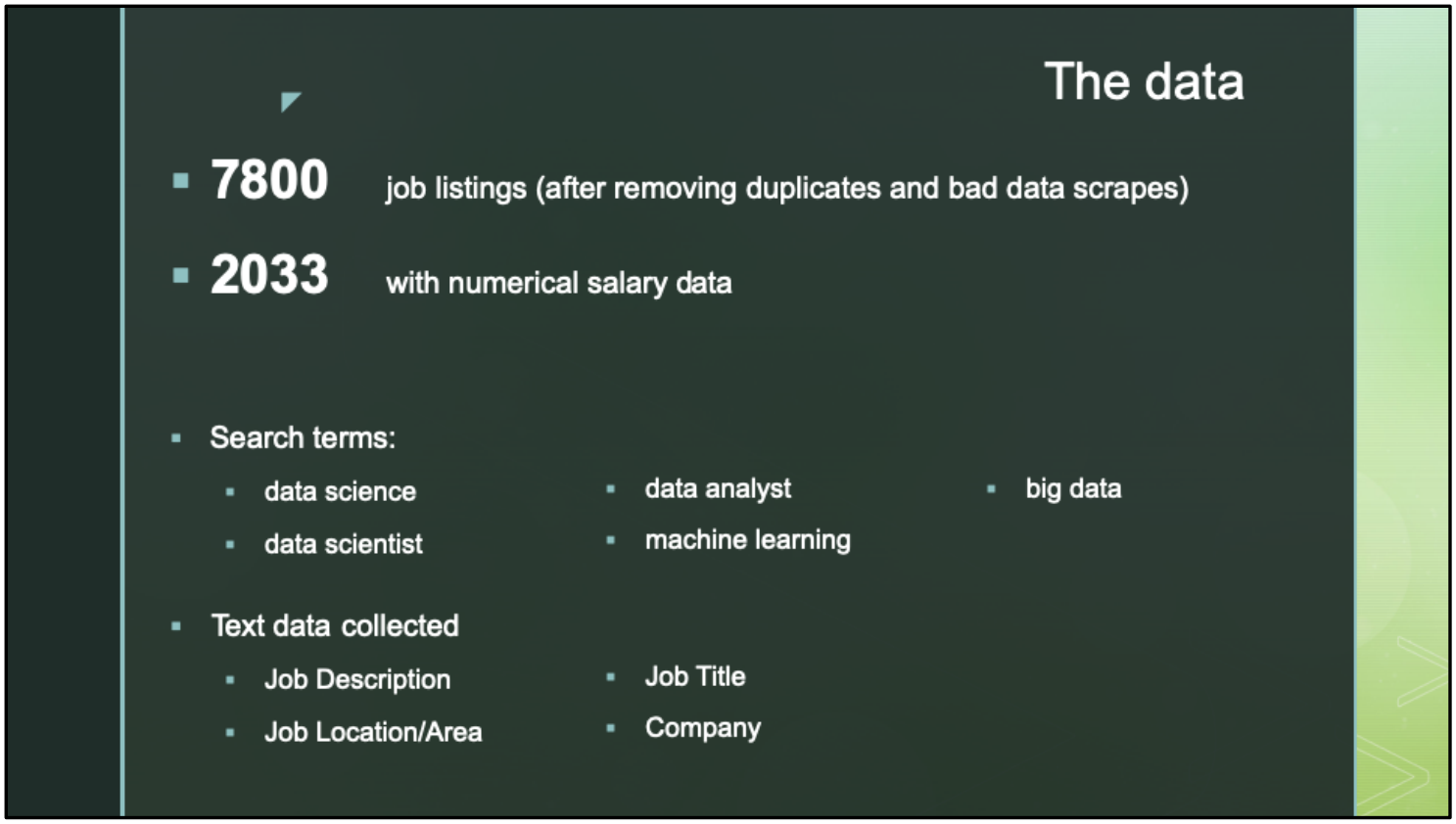


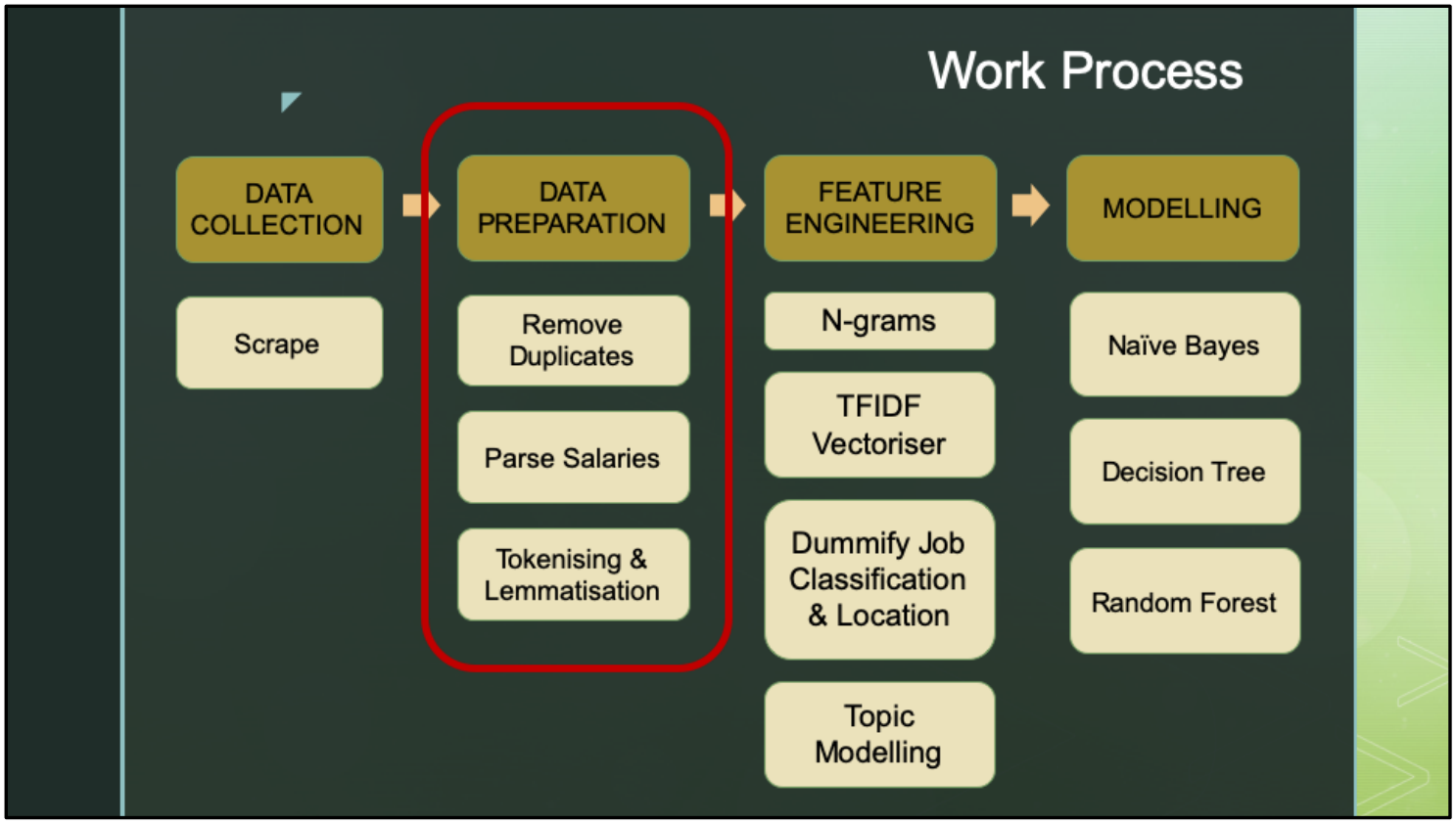
Project aims

Project not in any way affiliated with Seek.com.au, this was just the source of data by scraping publicly available web pages



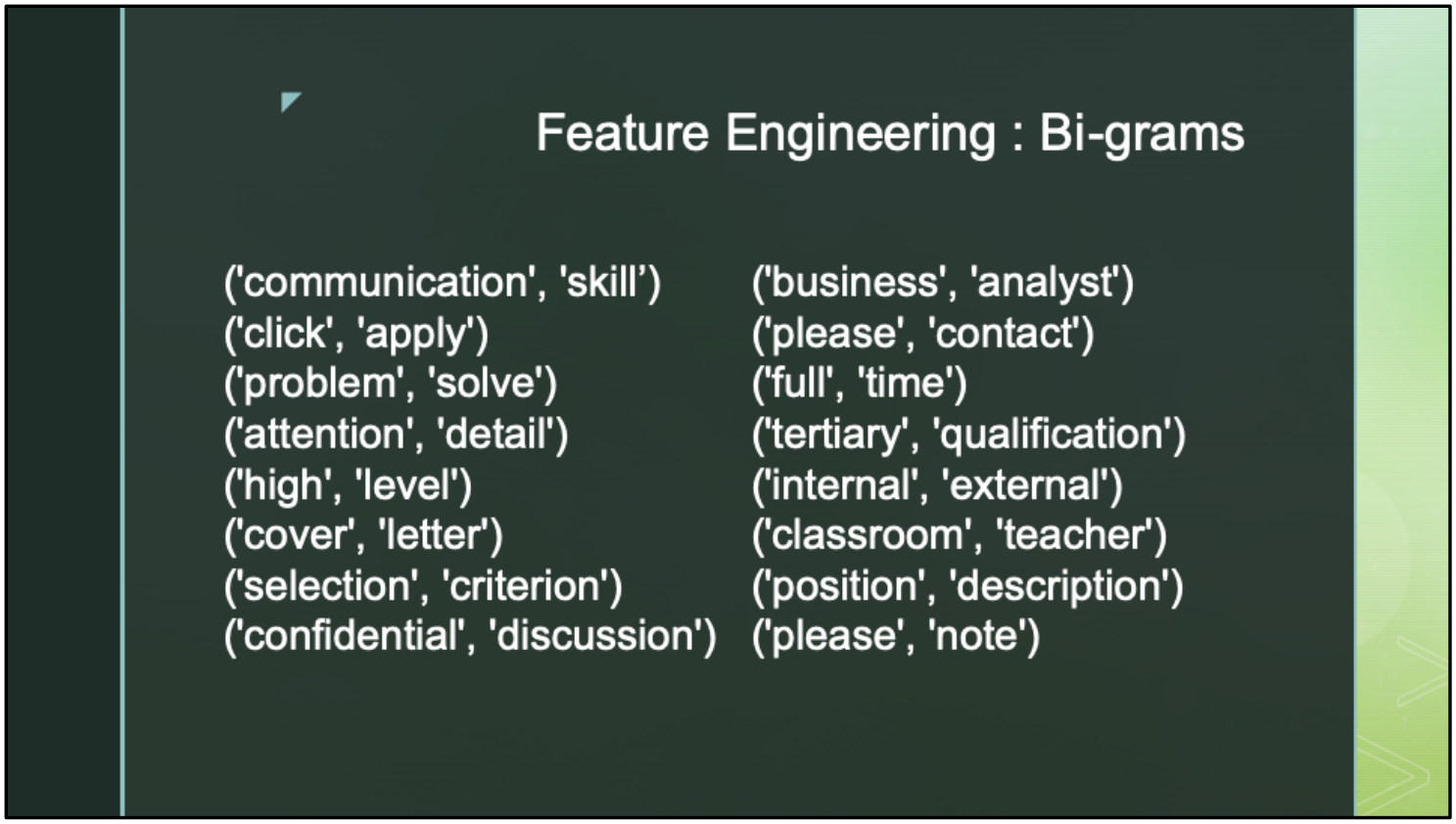
Overview of data scraped from Seek.com.au

Data was scraped with model runs on 3 separate dates over a period of about 3 weeks

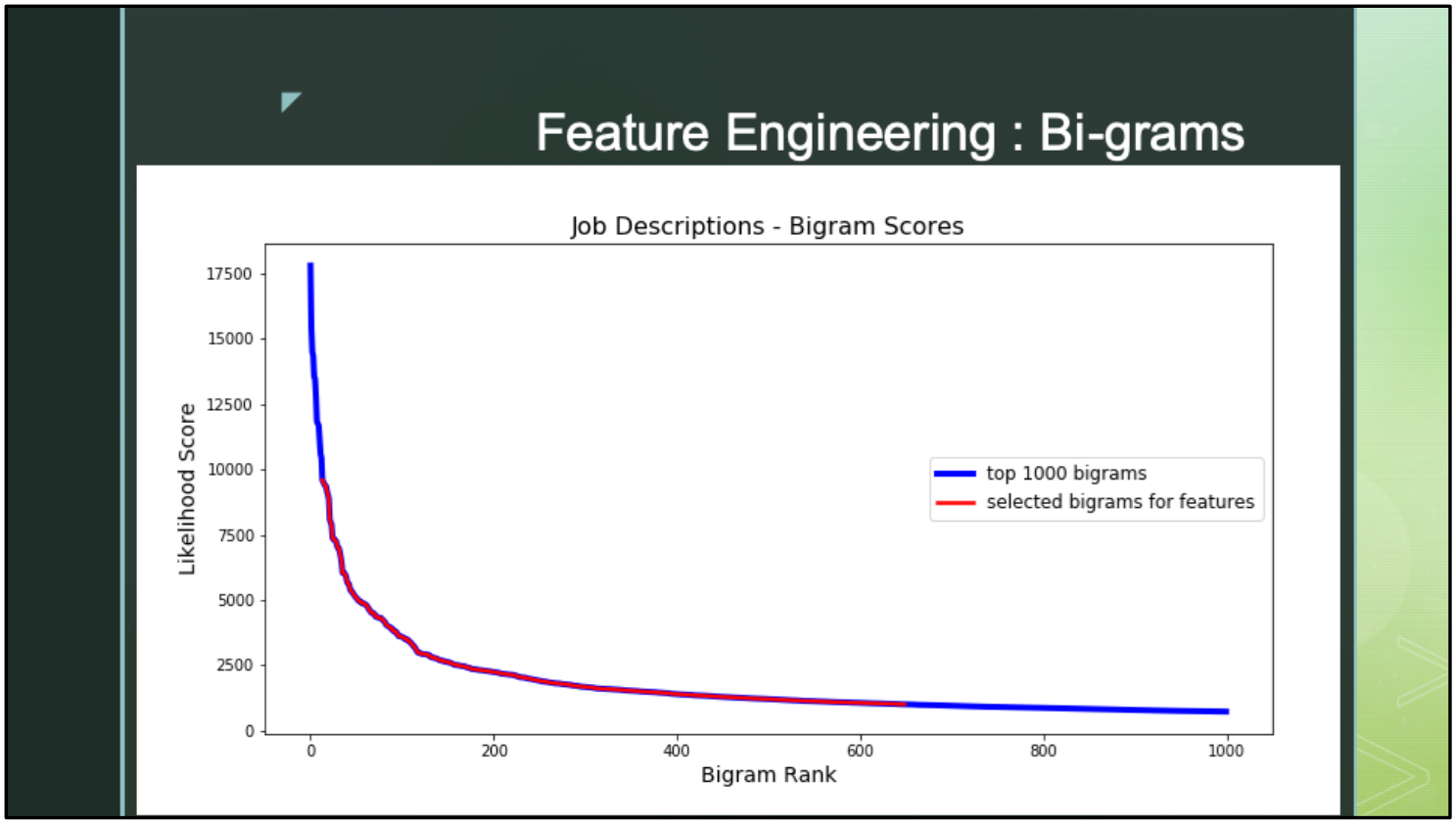


Outline of work process. Data preparation stage took by far the most amount of time. Process was not entirely linear, involved iteration, checking results and reworking at many steps.

The following slides highlight several major feature engineering steps taken, followed by modelling results.



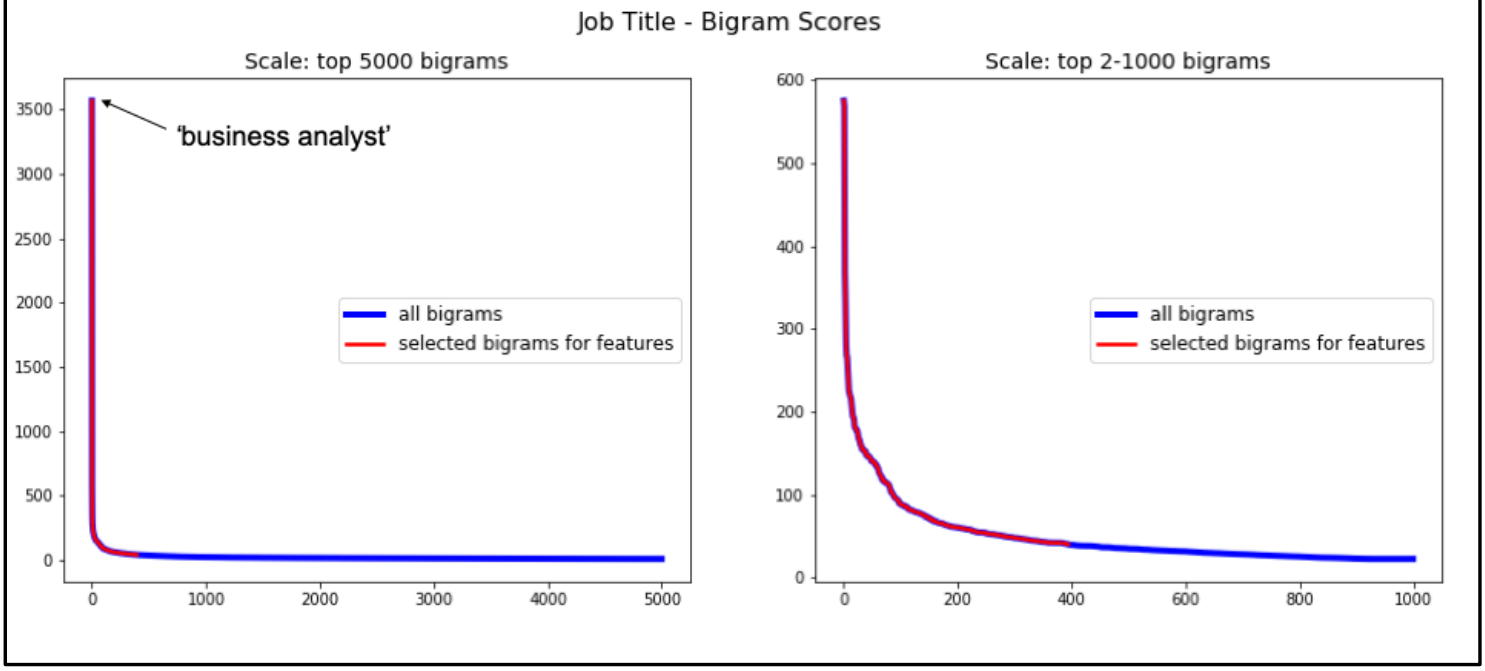
Choosing which bigrams to include as model features was a difficult call to make. These are the top 16 bigrams from job description text, which appear to be so generic as to not contain much meaningful information that would be a predictor of salary.



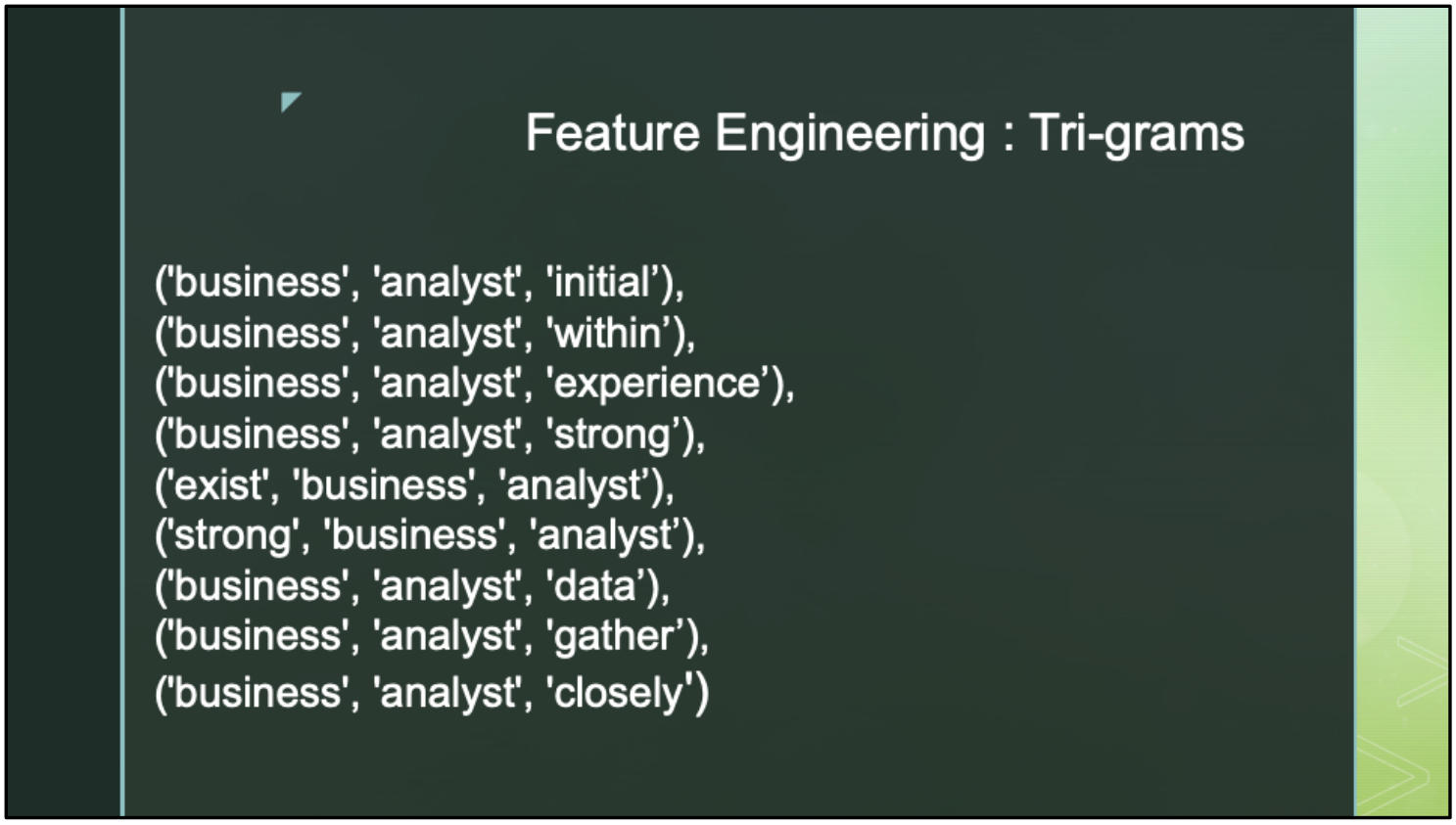
Graph shows Job Description bigrams ranked by likelihood score (frequency of occurrence in entire corpus). Made a judgement call to not include most frequent bigrams (or very infrequent) – red line shows the selected bigrams covering the 'elbow' of the graph. Ideally given more time could test impact of including more or less.

Chosen bigrams were find-replaced for each job description, which was then later fed into TFIDF vectorizer (term frequency-inverse document frequency) to identify most important bigrams weighted by frequency and uniqueness.

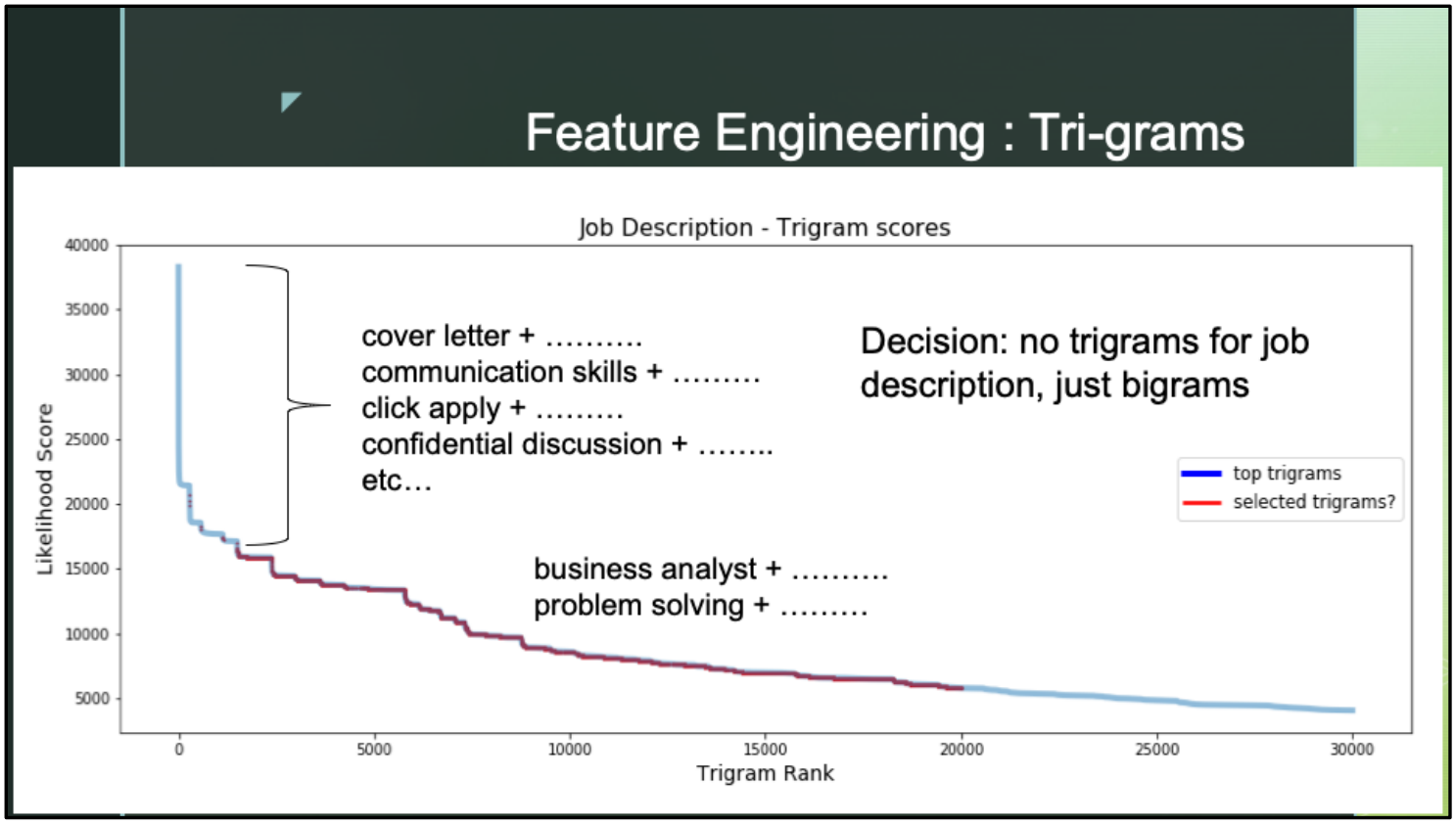
Feature Engineering : Bi-grams



Similar graphs to previous but data is bigrams from Job Title text (not Description). Number 1 ranked bigram is 'business analyst' at score 3568, next bigram occurs at 575, hence zoomed-in graph on right. Chose top 396 bigrams (those with scores > 40)

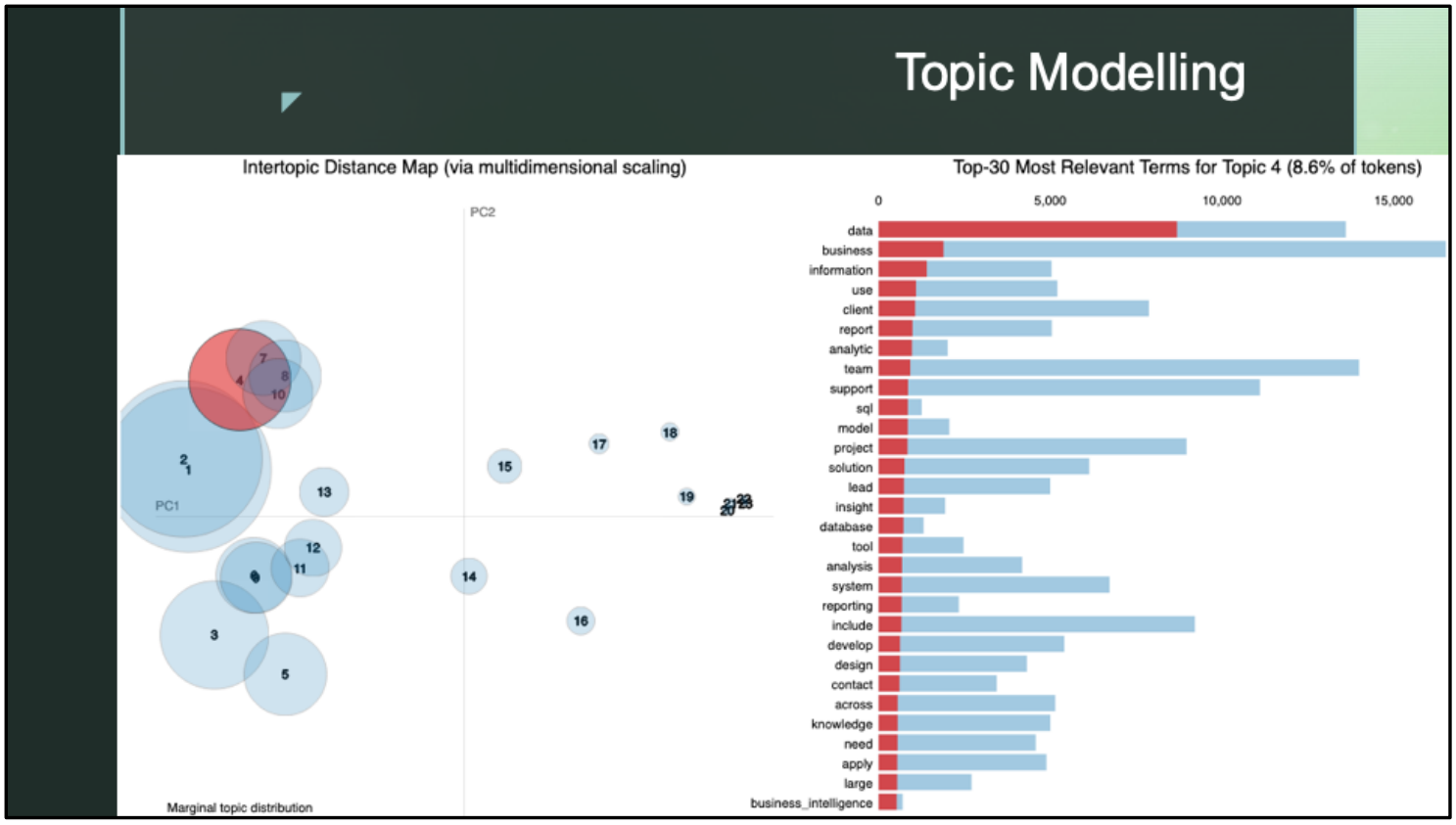


Trigrams for the job descriptions seemed less useful – many different combinations of over-represented bigrams (e.g. 'business analyst') with other various words.

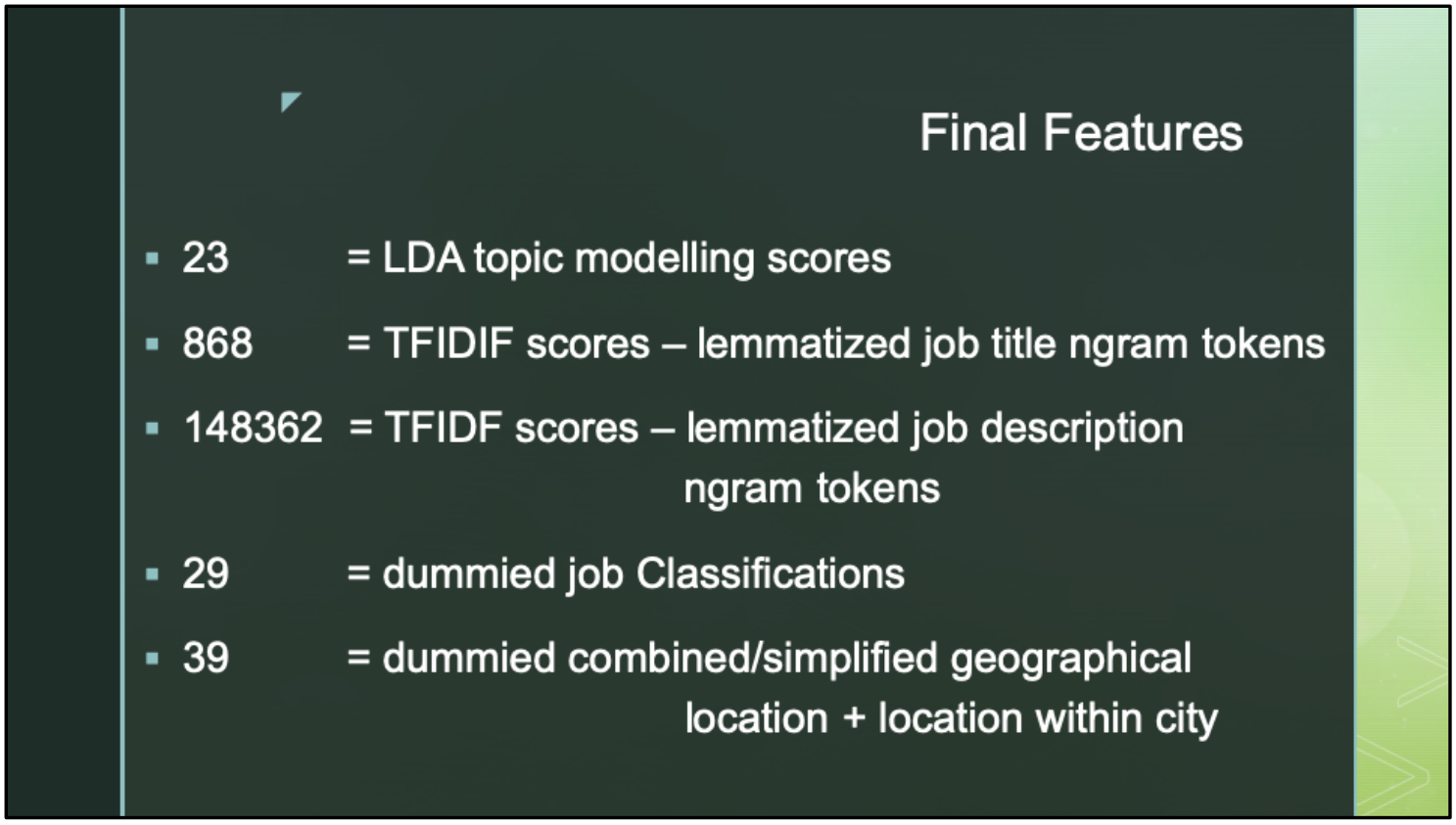


Looking closer at the ranked trigrams, the vast majority of the top 20,000 trigrams appear to be variations of one other word combined with the top half a dozen or so bigrams as shown here. Therefore it was decided not to include trigrams as for this text corpus they seemed to further spread out and dilute the meaning of bigrams that are already not very useful.

No actual empirical testing was done to check the impact of this – with more time I would like to but had to make a judgement call.

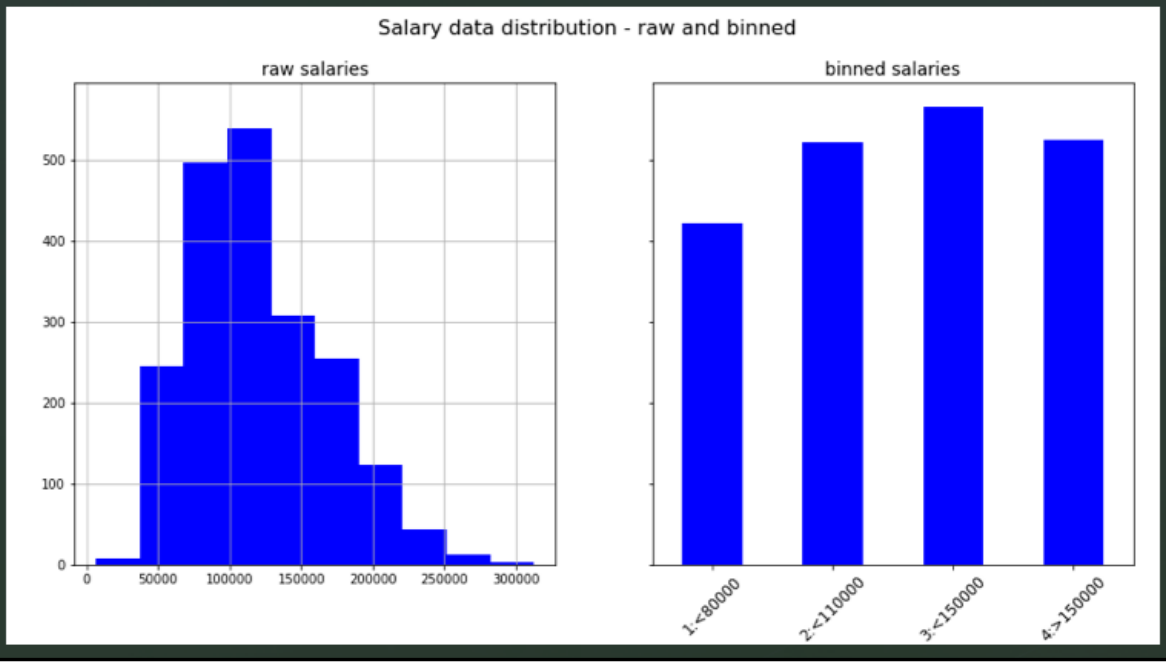


Latent Dirichlet Allocation (LDA) was use to identify clusters of words occurring together in the corpus to form topics. Results changed on each iteration of the model fitting to the data with varying results. Increasing the number of topics lead some larger topics splitting into two or more similar and slightly redundant topics, but also identified more smaller topics that had genuine differences – shown here by the smaller scattered topics numbered 14-23 on the right of the graph. Examples include mining, health/medical research, cybersecurity, and trade apprenticeship / machinery operator jobs which are useful to separate out, likely picked up from the ‘machine learning’ keyword search.

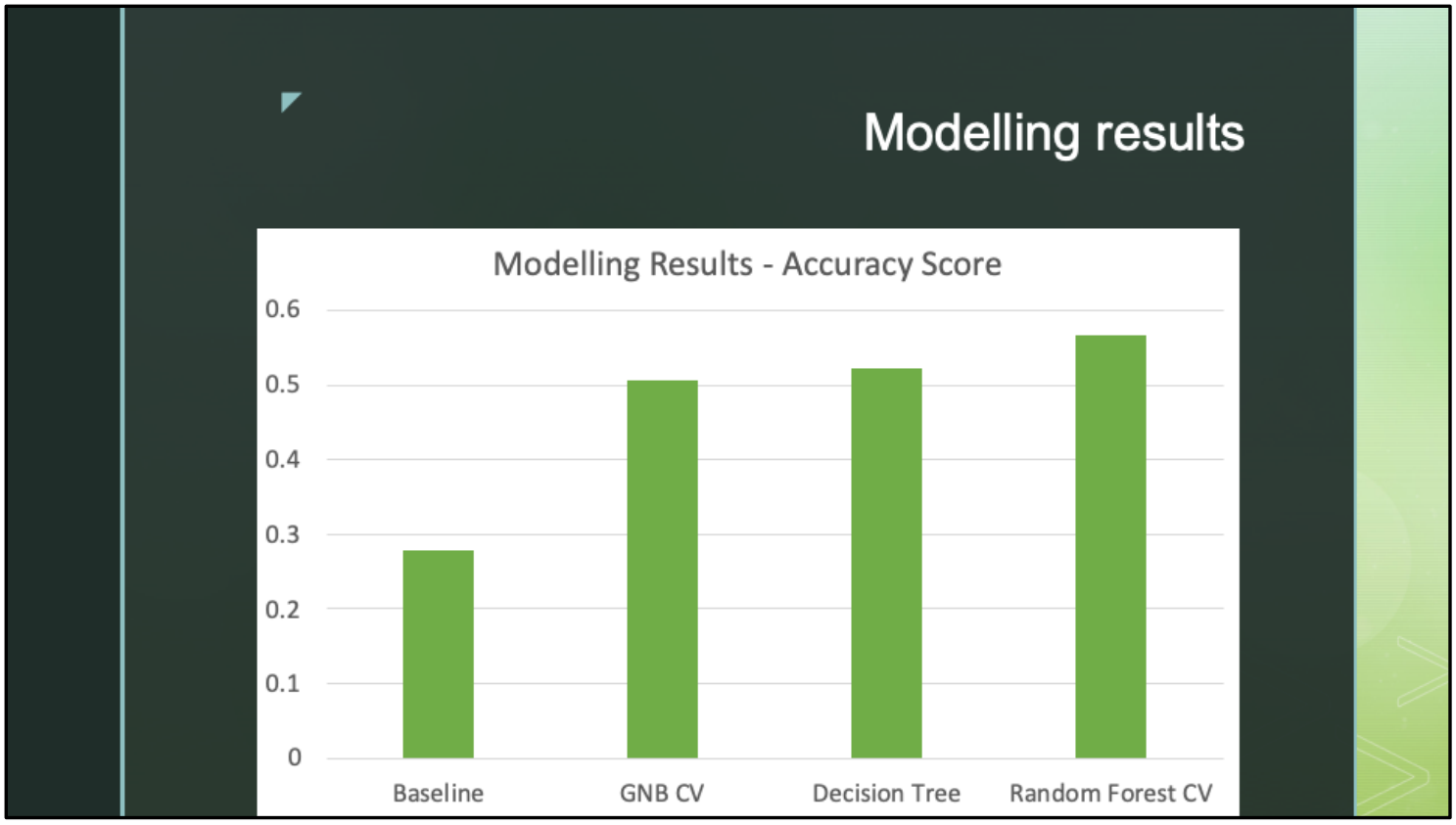


Fed all of these features into several machine learning models

Salary classification – target feature

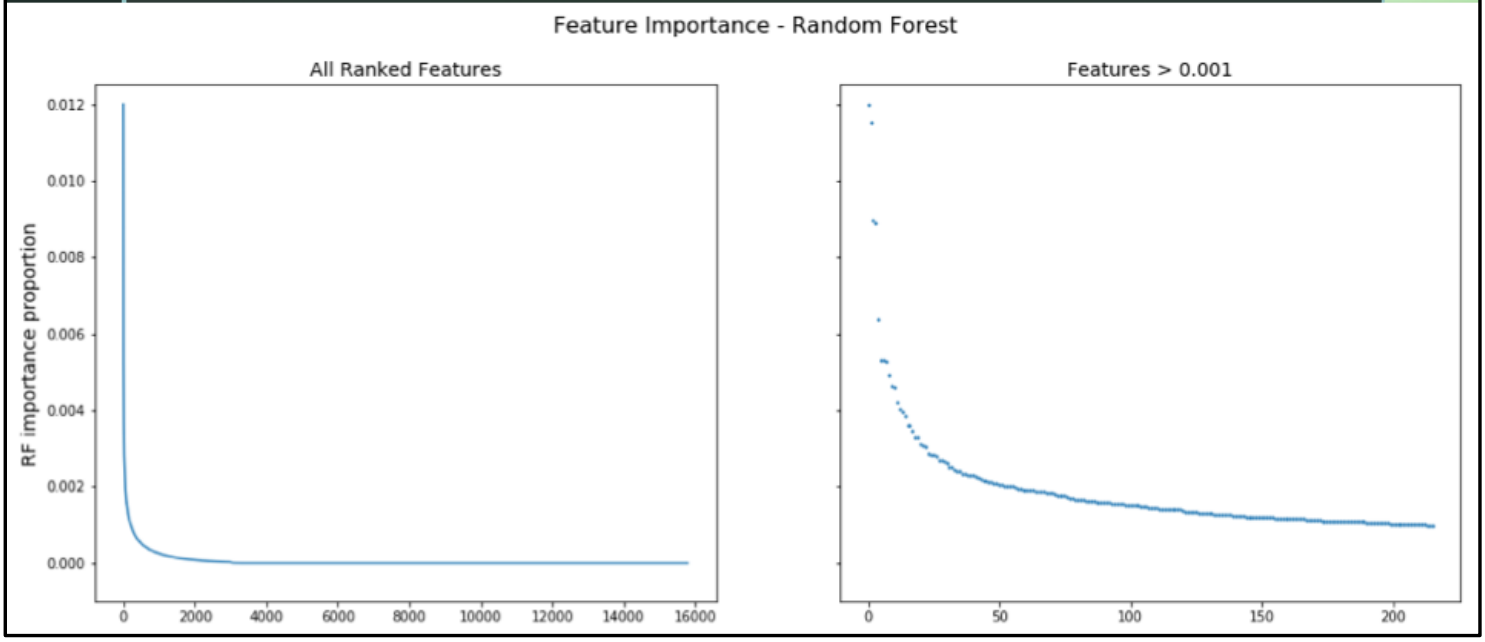


Salaries for jobs that had them posted were target variable for machine learning models. Left histogram is raw data, which for the purposes of predicting were binned into 4 categories for simplification.



Gaussian Naïve Bayes, Decision Tree and Random Forest were all applied to the dataset. Baseline comparison is highest occurring salary range of the 4 bins, at 28%. Highest accuracy model was Random Forest at 56%. Some gridsearching was done on Random Forest, which preferred a max depth = 20 with 40 individual estimators. Obviously this accuracy score is not particularly outstanding.

Modelling results



The Random Forest feature importance score ranks each feature by how relevant it was for the model's decision process. Out of the almost 150,000 features fed in, only the top few hundred were ranked as important by the model. A word cloud on the following slide gives an idea of the most important individual features

Feature Importance from Random Forest

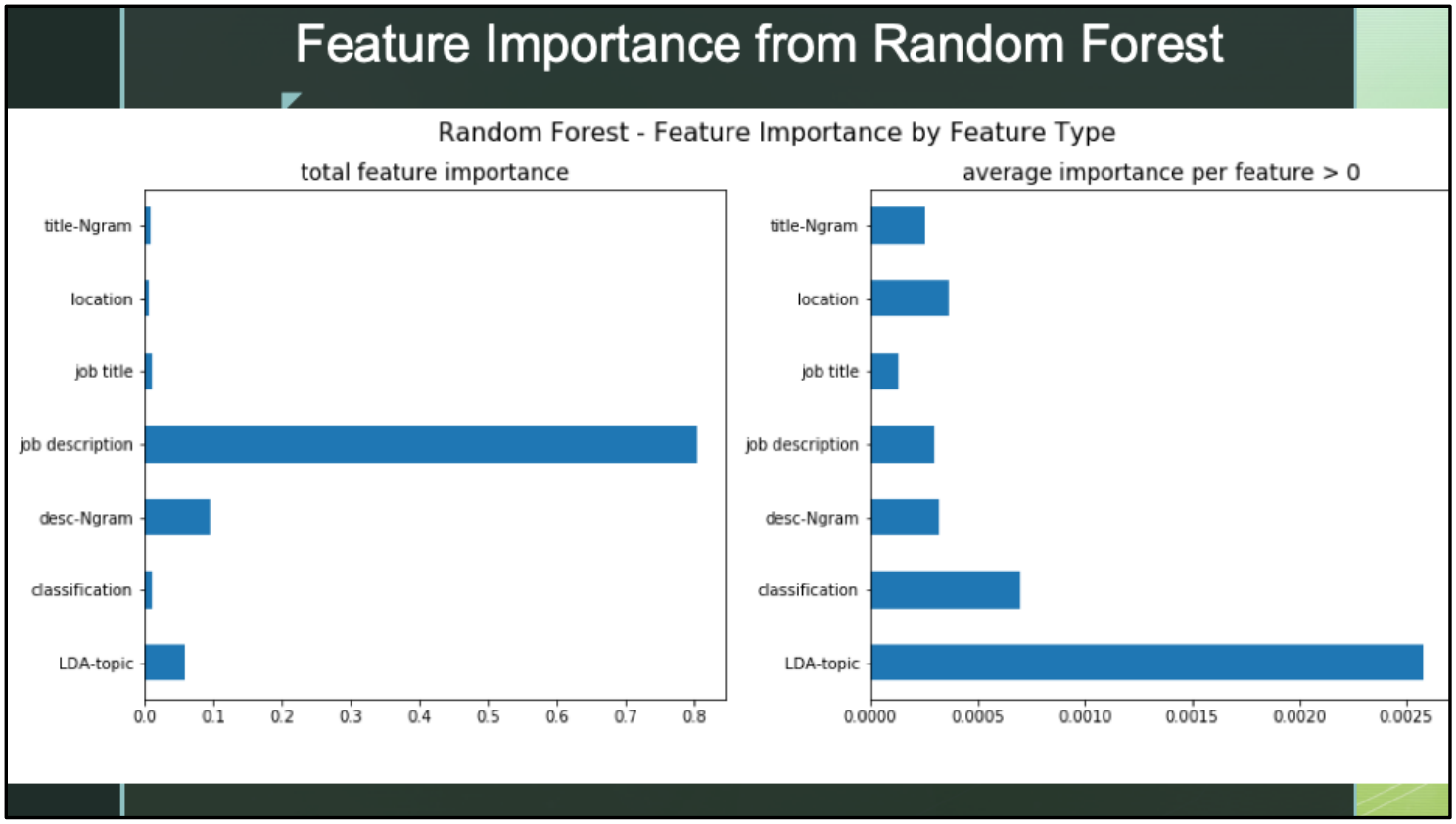
Prefix lda_..... refers to LDA topics, with words summarizing the topic meaning joined by hyphens, manually interpreted by myself

Prefix d_..... refers to individual ngrams (either single words or bigrams) from job descriptions, with the feature column ranking each document's TFIDF score for this ngram

Words joined by underscore (e.g. "attention_detail" are a bigram

Several words separated by spaces ("Information & Communication Technology") are job classifications as per Seek.com format

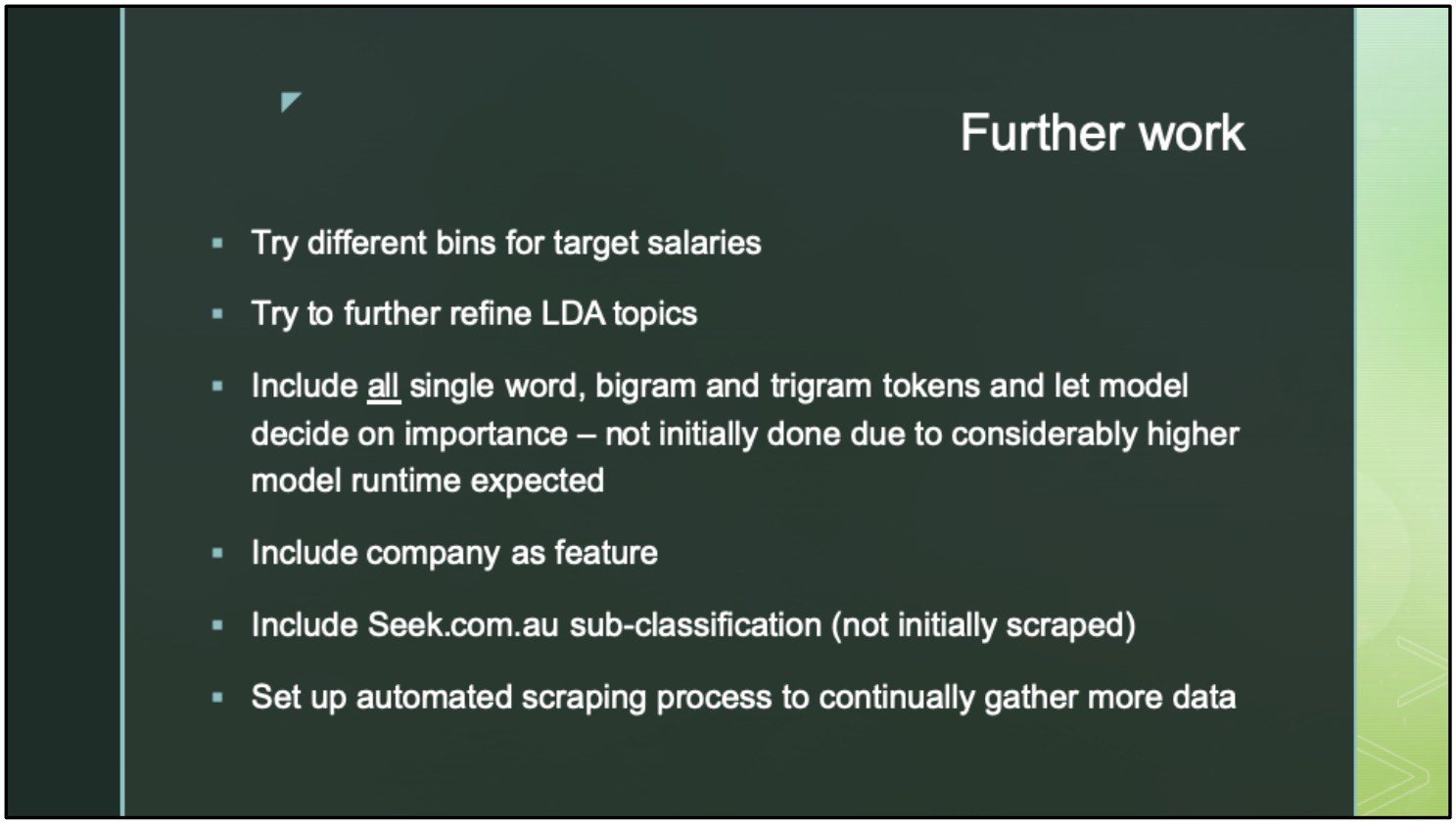
Prefix t_..... would refer to job title ngrams but these did not turn out to be significant in the model



The feature importances were further analysed by looking at how important each of the various classes of features that were fed to the model.
Note that 'job title' refers to single word tokens from job title text, whereas 'title-Ngram' refers to bigrams from the job title text, both features classes are TFIDF scores but for the purpose of this analysis were split into single and bigrams. Same for 'job description' vs 'desc-Ngram'

Graphs on left is raw sum of importance scores. However this is not so useful – most obviously the sheer number of job description single word tokens gives the impression that these are the most important, but this is not the case. More meaningful metric is the average score for each feature class – further filtered down to only divide the total score by the number of features in each class that actually had a score greater than zero.

This average score is in the right graph, and shows that the LDA topics were by far the most important features, followed by Seek.com job classification, location, with bigram and single word tokens among the least important features.



Ideas for further efforts that could be taken to improve model accuracy

LDA topics in particular are very interesting and promising, but through a number of iterations in the feature engineering process, it appears that the random nature of its initial seeding and separating topics commonly leads to a different mix of useful and less useful topics every time the LDA process is run. Would like to learn more about this process and ways to improve its output.