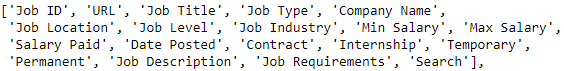
Project 4: Web Scraping Job Postings

In this document, I will explain the steps undertaken to retrieve information from a jobsite known as CareersFuture, cleaning and preparing of the data obtained, analysis and evaluation through the use of Machine Learning algorithms to determine the factors that affect Salary categories as well as Job Title categories.

# Web Scraping from CareersFuture

In order to scrape from CareersFuture, I utilized specific libraries in Python such as BeautifulSoup and Selenium to identify and process information from the page source. CareersFuture was picked as the source for the information scraping as it had many postings with salary information as compared to other jobsites such as Indeed. The search terms included “Data Scientist”, “Data Analyst”, “Business Analyst”, “Big Data” etc.

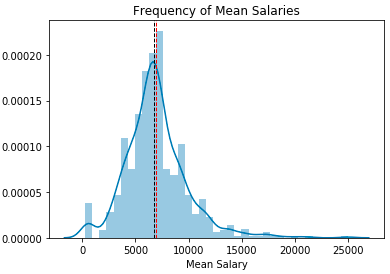
Some of the issues faced included slow loading of pages, irregular structures within the site which was resolved through adding wait times before continuing as well as introducing conditions to ensure the script knows how to handle situations. Through this process, of which I also pre-processed some information, resulted in 18 columns worth of information. My method of storing the scraped information was to combine all the information into a respective DataFrame before exporting it as individual CSVs. This was done on purpose for each search term in order to minimize the effect of data loss should an unforeseen error occur.

# Preparing/Cleaning of Data obtained

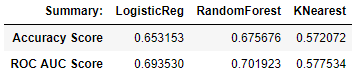
Having 8 different CSVs on hand with the same number of columns, I proceeded to concatenate the information together under one big DataFrame and resetting the index. Next step of the process was to remove all those rows with no salary information as there were some entries that had returned missing information such as Industry and Job Position. Following which, I removed the duplicates of certain entries through ensuring that the Job Title, Company Name and Job Type matched before discarding the duplicates, keeping only 1 entry.

Misleading salary information such as Annual Salary advertised as Monthly allowed me to adopt an approach whereby I set a threshold of 30k for Monthly Salary and set all those above as Annual Salary before changing them back down to Monthly through division by 12. As the salary information given by the website was a range, I decided to create a new aggregate column by obtaining the mean salary. An important process was to dummy code Job Level and Job Industries to ensure that they could be used as features for Machine Learning models at a later stage. At this point, I was confident that the data has been cleaned sufficiently to move on to the next stage.

# Identifying factors that affect Salary

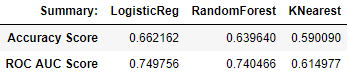
My approach was to frame the problem as a classification problem, of which I decided to classify Salary in terms of 2 categories. The first category was salaries either below or equals the median, and the other category was salaries above the median. The choice of median over mean was due to the presence of some extreme values which was under the threshold of 30k, which potentially could lead to class imbalance.

## Case 1: 1 single feature - Job Levels

As a simple case, I decided to use Job Levels to predict the salary. The reasoning behind was that different Job Levels are expected to get different salaries. With the Job Level feature, I decided to set up 3 different Machine Learning models to see which one worked best in predicting the categories using the features, namely Logistic Regression, Random Forest and K Nearest Neighbours. Baseline (selecting the majority class) was 0.544.

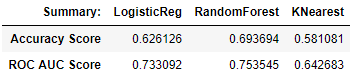
Using train\_test\_split with random state fixed for consistency, the 3 models fitted on train set and predicted on the test set. The eventual result was that Logistic Regression & Random Forest were similar in their Accuracy score while KNearest was the worst performing out of the three.

## Case 2: 2 features – Job Levels + Job Titles

Processing of Job Titles involved the use of NLP algorithms such CountVectorizer to encode the words into a bag similar to dummy coding. After splitting Job Titles into train and test sets, I then applied CountVectorizer to fit on train set and transformed test set to ensure that test set is not fitted. The next step was to concatenate with the previous features and ran the 3 models again with the new features included.

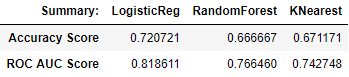
Logistic Regression and KNearest models improved in their accuracy score but Random Forest got worse scores. This was puzzling and resulted in me wanting to explore if Job Titles itself would be a better predictor on its own.

## Case 3: 1 feature - Job Title

As mentioned earlier, the weird results above prompted me to investigate if Job Title itself would be a better feature. By applying CountVectorizer again, the 3 models were constructed.

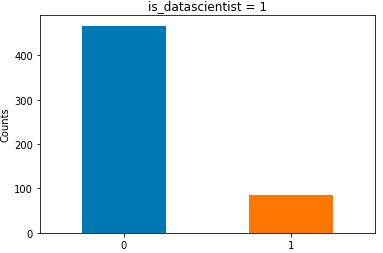
Surprisingly, the scores were better than the first case.

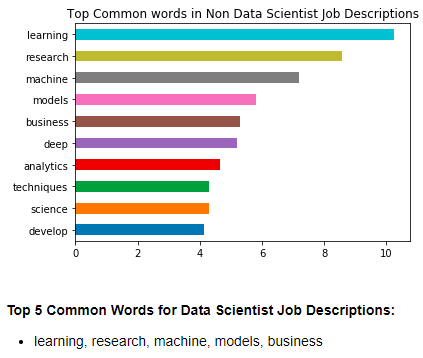
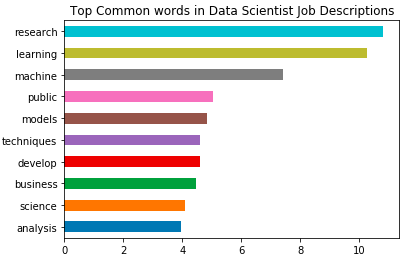
## Case 4: 3 features – Job Level + Job Title + Job Industry

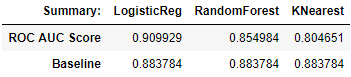
The final case I decided to attempt was to use 3 features and see if they as a whole would better predict Salary categories.

Ultimately, Logistic Regression model had the highest accuracy score out of the 4 cases.

# Factors that distinguish job category

Similarly to earlier question, I decided to approach this problem as a classification problem whereby job titles including “Scientist” as part of the title would be considered as a class, while the rest considered as another class. Baseline was identified as 0.883.

An additional step in the approach was finding out the most common words found in the Job Description between the 2 classes.

Following which, the same 3 types of Machine Learning models were constructed. Instead of Accuracy score, it was important to view ROC score due to the class imbalance present as majority of the postings were not Data Scientist.

In conclusion, Logistic Regression performed the best out of all 3, other models did not perform better than baseline.