An analysis of Indeed Job Postings: Salaries and Skills

Alex Brown and Asif Hasan

2022-12-09

Introduction

As data science students looking for co-op's and nearing the end of our degrees, we thought we would use our skills to gain some insight into the job market. Some questions we want answered are:

- Which sub-fields of Data Science pay the most?
- Can we predict the salary a job posting offers?
- Which data science skills are the most in demand right now and which should we learn?

To answer these questions we needed data. We chose to obtain our data by sampling job postings from indeed.com. We wrote scripts to perform searches for specific jobs on indeed and scrape their contents. We used the Beautiful Soup 4 and Pandas Python libraries to navigate and clean the data respectively. We performed ANOVA to answer the first question and fit a random forest regression model to answer the second. Finally we performed a keyword analysis to answer the third, making use of the Seaborn, matplotlib-venn and wordcloud libraries.

Scraping

Our web scraping process can be broken down into 2 steps:

1. Search for specific job posting URL's on Indeed
2. Visit each URL and scrape the page's contents

It was necessary to use the Selenium Python library which provides Python bindings to the Selenium web driver. This let's us control Google Chrome programmatically. This is because indeed.com has extensive checks to make sure you are visiting from a browser, and CURL and Requests will not work.

Getting Job URLS

Our first script "get_job_urls.py" was responsible for collecting the job urls. It simply uses a Selenium web driver instance to open Chrome and perform a search on indeed. We can select search parameters by modifying the arguments in the "indeed.com" URL. The script takes command line arguments for job location, search query and a number of pages to scrape. Specific html tags on the search results page correspond to the links in the job postings. We used Beautiful Soup to navigate the HTML tree and extract the URL's in these tags. We ran this script to search for American job postings. The reason for this is Canadian postings rarely had a salary specified. We performed three searches using "data scientist", "data analyst" and "data engineer" as our search queries. We collected a total of 620 URL's.

Scraping Page Contents

Once we had the URL's we again used Selenium to visit each URL. The script that did this was called "scrape_job_postings_html.py". We stored the entire html document of each page in separate files. BS4 was then used to extract the sections with the information we needed for further cleaning. This includes sections like title, salary and description. When finished we had a directory of pretty HTML files with only their relevant sections.

Cleaning

Data Extraction

Our next step in cleaning the data was extracting data from the each web page in our directory of HTML files (job postings). We have used the 'BeautifulSoup' library (allows to parse HTML/XML files) to extract the job title and job description section from the pretty HTML files. Then we used regular expressions, 're' library, to extract:

1. The salary values from the job description section include upper and lower bounds. For missing salary values on job postings, we filled the column with a value of zero so we can filter those postings when needed.
2. If the job is remote we looked for the keyword 'remote' in both the title and the job description section of job postings.
3. If the job is an internship we looked for the 'intern', 'internship', and 'co-op' keywords in both the title and the job description section of job postings.
4. The job type from the job description section, includes 'Full-time', 'Part-time', and 'Contract'; for missing values, we filled the column with the value 'Full-time (i)' where '(i)' indicates that the value was imputed; we made this decision because lots of job postings did not have a label for job type, although the job postings are referred to as 'Full-time' jobs. However, we ended up not using this column in our model analysis because it turned out not to be significant enough in our preliminary analyses of the data.

Lastly, we saved the dataframe containing the title column and five different columns that are extracted from looking at the title/job description section of each job posting in a CSV file.

Salary Cleaning

After we acquired the data, at first glance, we saw that the salary values needed to be cleaned. The salary values have two variations:

1. One such variation is $75,000. For this variation of salary values, we dropped the $-sign and converted the values to numeric.
2. And the other variation is $75.3K. For this variation of salary values, we dropped the $-sign and everything after the decimal and multiplied the value with 1000 (code is not implemented as in this description), basically rounding to nearest lower thousand. So, the given salary value, $75.3K, will convert to 75000. We made this choice to save some coding time.

After we finished the cleaning data phase, it was time for the aspect of the project we were looking forward to: feature engineering.

Feature Engineering

Till now, we lacked the right features to adequately answer our questions. However, when we extracted the whole title from job postings and saved it as a column, we had in mind to transform the titles into meaningful features for our model because the titles as they were had so much variability.

After carefully observing the data, specifically the job titles, we came up with two additional features that we wanted to add to our data. The two new features are:

1. Role; levels include 'Scientist', 'Analyst', 'ML Engineer', 'Research', and 'Unidentified'. To extract the values for this feature, we used regular expressions to search the job title for similar keywords as the levels themselves in a hierarchy that searches for the most common appearing word, such as 'scientist' at the very end.
2. Seniority; levels include 'Senior', 'Junior', and 'None'. To extract the values for this feature, we again used regular expressions to search the job title for similar keywords as the levels themselves.

At this point, we knew how we wanted to use the data for machine learning analyses—that is, if the statistical tests concluded the data was significant for analysis. So we went further and one-hot encoded the two new features and 'remote' column, which we intentionally encoded as a "Y" or "N" value during acquisition. To do this, for the 'remote' column, we used the pandas.DataFrame.apply function to convert the 'Y' and 'N' values to 1 and 0, respectively. And for the two new features columns, we used pandasDataFrame.apply function to obtain a Python list of lists of shape (n,levels of the respective feature), where n is the number of observations. Rows contain a single 1 that indicates the category while the rest of the values are 0's. For example, if a seniority observation returns [1,0,0], this means that the job posting is for a senior position. And for example, if a role observation returns [0,1,0,0,0], this means that the job posting is for an analyst position. Then we concatenated the matrix of binary values with the DataFrame. Then we finally put all the distinct columns into one dataframe and saved it as a CSV file.

The program called "feature_extraction_complete.py." does all this by taking a CSV file as an input argument and an output argument to write the results to a CSV file. It also produces 2 CSV files containing one-hot encoded features generated from role and seniority features.

It should be noted that features for the anova model were prepared using the "anova_preprocessing.py" script which follows a similar logic, but simpler implementation that only considers the role levels of Data Scientist, Data Analyst, Data Engineer and Unknown.Also to avoid repeating the same analysis twice, we averaged salary_low and salary_high to produce the salary_avg feature.

Salaries ANOVA

Some of the most common job titles in data science that we hear about are Data Analyst, Data Scientist and Data Engineer. We chose these because in our experience and in our job search they are the most quintessential titles, but obviously they are not an exhaustive list of data science job titles. To answer the question of whether or not these jobs earn us different salaries on average, we decided to perform ANOVA. Apart from job title we have other factors that might account for some of the variance in salaries. Whether or not a job is remote, is an internship and of course seniority all seem useful. These factors are is_remote, is_intern and seniority. After some preliminary analysis though we found that the seniority and is_remote factors reduce the group sizes by far too much, create unequal variance between groups and result in an extremely unbalanced design. Instead we will do a two-way ANOVA with factors role and is_intern.

To get a feel for our data, take a look at the table below:

```
##      salary_avg is_remote is_intern       role
## 0       45500.0         Y         1    analyst
## 1       70000.0         N         0    analyst
## 2      158000.0         Y         0  scientist
## 3      120000.0         N         0  scientist
## 4       84000.0         Y         0  scientist
## ..          ...       ...       ...        ...
## 607     83500.0         Y         0   engineer
## 608    135500.0         Y         0  scientist
## 610    117000.0         N         0   engineer
## 614    130000.0         Y         0  scientist
## 615     82500.0         N         0  scientist
##
## [452 rows x 4 columns]
```

Also the group counts:

```
##                       count
## role      is_intern
## analyst   0              60
##           1              22
## engineer  0              93
##           1              11
## scientist 0             213
##           1              53
```

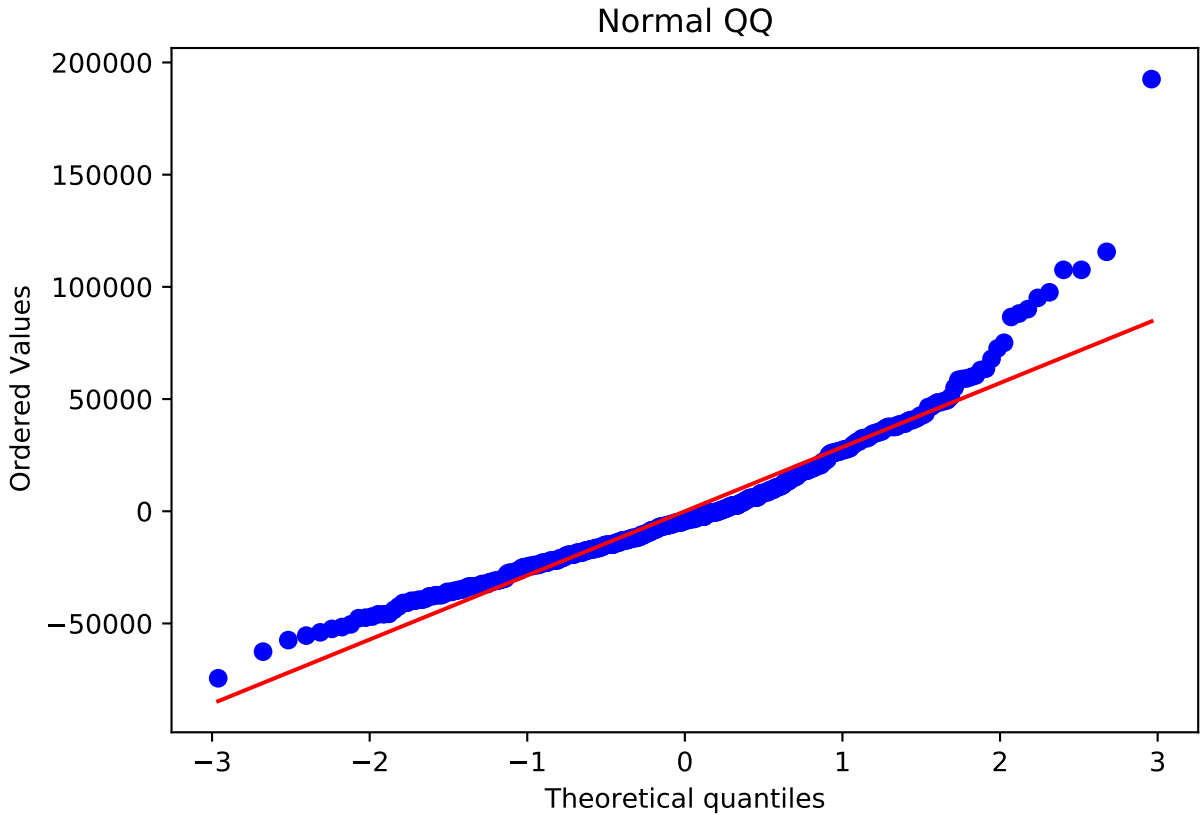
And group means:

```
##                         mean_salary
## role      is_intern
## analyst   0           94379.425000
##           1           54886.363636
## engineer  0          113961.446237
##           1           73545.454545
## scientist 0          117419.791080
##           1           58047.169811
```

Assumptions

Normality

Although ANOVA is said to be fairly robust to departures from normality, let's check for normality anyways. Our assumption is that residuals are normally distributed. In ANOVA, the residuals are the difference between values and their corresponding group's mean. Let's plot them against the Normal distribution on a qq plot. The probplot function of scipy.stats plots against the normal distribution by default and it's output is shown below.



Salaries look fairly normal throughout except for at the right tail. I'd say the assumption is met, moving on.

Equal Variances

Simulation studies have shown that unequal variances within groups affect the reliability of ANOVA results. Let's make sure the largest standard deviation is no more than twice the smallest (this is a good rule of thumb). Below is a table of standard deviations within groups.

```
## role       is_intern
## analyst    0            30618.179
##            1            19864.381
## engineer   0            25762.691
##            1            18336.377
## scientist  0            34091.064
##            1            20229.516
## Name: std, dtype: float64
```

```
## max/min=1.8592039155786446
```

The largest is less than two times the smallest which means we can consider the groups to have equal variance. Note there are more rigorous statistical tests for comparing variance, but for simplicity we went with this.

Independence

The independence assumption seems reasonable. Each observation refers to an independent job posting.

Looks like we're good to go. Let's perform ANOVA

Anova Model

Our null hypotheses:

- H0a: Mean salaries are equal between roles
- H0b: Mean salaries are equal between interns and non-interns
- H0ab: There is no interaction effect between role and is_intern with regard to salary

Our alternative hypotheses:

- H1a: Mean salaries are different between some roles
- H1b: Mean salaries are different between interns and non-interns
- H1ab: There is an interaction effect between role and is_intern with regard to salary

Two-way anova table

Note the anova was performed using type 3 sum of squares as we are considering interaction effects.

```
pg.anova(data=data,dv='salary_avg',between=['role','is_intern'],ss_type=3).set_index('Source').round(3)
```

```
##                              SS     DF            MS        F  p-unc    np2
## Source
## role               1.113111e+10    2.0  5.565555e+09    6.273  0.002  0.027
## is_intern          1.035486e+11    1.0  1.035486e+11  116.715  0.000  0.207
## role * is_intern   6.144753e+09    2.0  3.072376e+09    3.463  0.032  0.015
## Residual           3.956882e+11  446.0  8.871932e+08      NaN    NaN    NaN
```

Note all p-values are less than 0.05. This means we can reject our null hypotheses at 5% significance level and accept their alternatives.

Post-hoc analysis

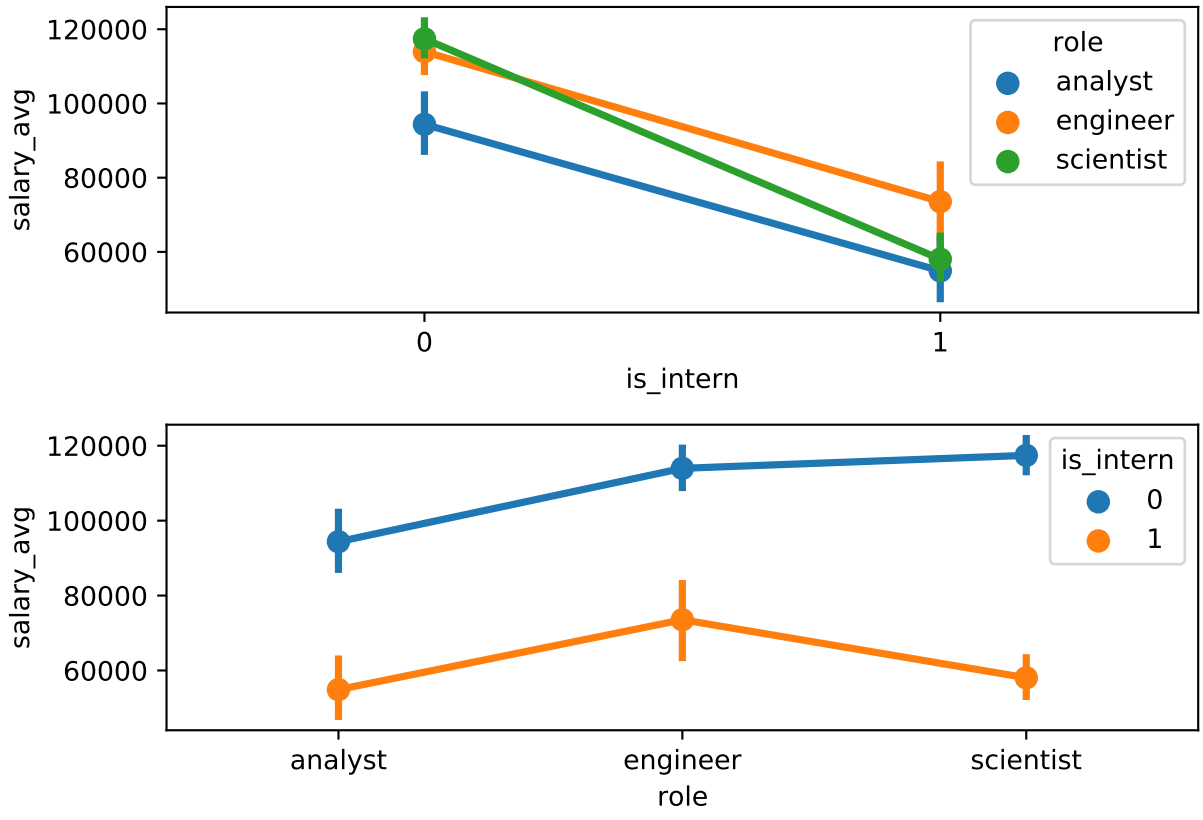
Rejecting our null hypotheses indicates that at least one pairwise difference between group means exists for each factor and interaction effect.Pairwise t-tests were performed between groups below.

```
##             Contrast is_intern          A          B  p-unc  p-corr
## 0         is_intern         -          1          0  0.000     NaN
## 1              role         -    analyst  scientist  0.000   0.000
## 2              role         -    analyst   engineer  0.000   0.000
## 3              role         -  scientist   engineer  0.265   0.265
## 4  is_intern * role         1    analyst  scientist  0.536   0.536
## 5  is_intern * role         1    analyst   engineer  0.014   0.028
## 6  is_intern * role         1  scientist   engineer  0.024   0.036
## 7  is_intern * role         0    analyst  scientist  0.000   0.000
## 8  is_intern * role         0    analyst   engineer  0.000   0.000
## 9  is_intern * role         0  scientist   engineer  0.331   0.397
```

Benjamoni-Hochberg adjustments were made to the p-values to control the False Discovery Rate. The first 4 rows correspond to ttests on the main effects. Of note here is that there is no significant difference between the mean salaries of data engineer and data scientist (p-corr=0.264). The other levels showed significant differences.

T-tests were also performed between levels of role within intern groups. We know internships earn less so we are focusing on whether different roles' internships earn differently.

Of note here, mean data analyst intern and data scientist intern salaries did not differ significantly (p-corr = 0.536). Neither do non-intern data scientist and data engineer salaries (p-corr = 0.397). Data scientist and data engineer intern mean salaries did differ however, highlighting the interaction effect. This interaction effect is visualized below:



Notice that data scientist jobs are more strongly affected by internship status.

Conclusions

Let's take a last look at the group means:

```
## role       is_intern
## analyst    1             54886.364
## scientist  1             58047.170
## engineer   1             73545.455
## analyst    0             94379.425
## engineer   0            113961.446
## scientist  0            117419.791
## Name: salary_avg, dtype: float64
```

How should a data science student considering his career path interpret our findings. Well he should note that analyst positions earn less on average than the other two. No significant difference was found between average data science and engineer salaries so he should go with whichever he prefers. That being said if he's applying for co-ops or internships, a data engineering position will earn him significantly more.

Salary Prediction Tool

We want to predict the salaries of job postings. This is useful for determining if a salary offer is fair, and for knowing what to expect from an unlabeled job posting. Below is a summary of our effort.

From our prior analysis, we knew which features to consider as predictors for the model. Furthermore, we decided to choose our models based on two important observations in our data. The first is that the sample from each group of 'role' has different sizes. This means the data is slightly imbalanced. And the second is that most features have binary values. Therefore, we only wanted to use decision trees or any ensemble variation of it to avoid overfitting or underfitting the data. Finally, we kept it simple and went with "Random Forest" and "Gradient Boosted Random Forest" as our models of choice.

Regression

For the regression models,

Response variable: Average_salary

Explanatory variables: Is_remote, Is_intern, Is_junior, Is_senior, Seniority_unknown, Is_scientist, Is_analyst, Is_ml_engineer, Is_research

Model 1:

Random Forest Regressor with 100 estimators and max depth = 10

Model training score: 0.492 (approx.)

Model validation score: 0.417 (approx.)

MAD validation: 18469.616 USD

Model 2:

Gradient Boosting Regressor with 100 estimators and max depth = 3

Model training score: 0.493 (approx.)

Model validation score: 0.414 (approx.)

Mean Absolute Error validation: 18629.970 USD

The top two scores of each model are R^2 values. The R^2 values, although not especially high, do indicate that a substantial portion of the variance in salaries is explained by our model. We achieved a mean absolute error of $18629 which isn't super precise. It is accurate enough though to be useful to identify abnormally low or high offers. There were some features like education and location that also could have been useful, but proved harder to extract.

We also fit a classification model for fun, using the same features but with salary and using role as our response variable.

Classification

For the classification models,

Response variable: Role

Explanatory variables: Salary_low, Salary_high, Is_remote, Is_intern, Is_junior, Is_senior, Seniority_unknown

Model 1:

Random Forest Classifier with 100 estimators and max depth = 7

Model training score: 0.723 (approx.)

Model validation score: 0.54 (approx.)

Model 2:

Gradient Boosting Classifier with 100 estimators

Model training score: 0.838 (approx.)

Model validation score: 0.513 (approx.)

The program called "ml_analysis.py." takes a CSV file, which include one-hot encoded features, as an input argument and prints the models that are being trained and the training and validation score for each of the four models. It also produces 4 CSV files containing the predictions of the 4 models trained (two regressors and two classifiers) in the program directory.

Skills Analysis

Skill Words

We wanted to get a feel for what skills jobs were asking for. To do this we needed to extract the skill words from a job posting. The vast majority of the time, when a skill is listed in a job posting its first letter is upper case. We decided to find all 1-3 word combinations of such words in the job description sections of the html documents. We used BS4 again to parse the html tree and pandas for the cleaning and manipulation. We filtered our a list of stop words. We put each unique instance of a skill in a job posting in a csv in a row with its job key. We joined to our other data (we were hoping to do further analysis). when we were done our data looked like this:

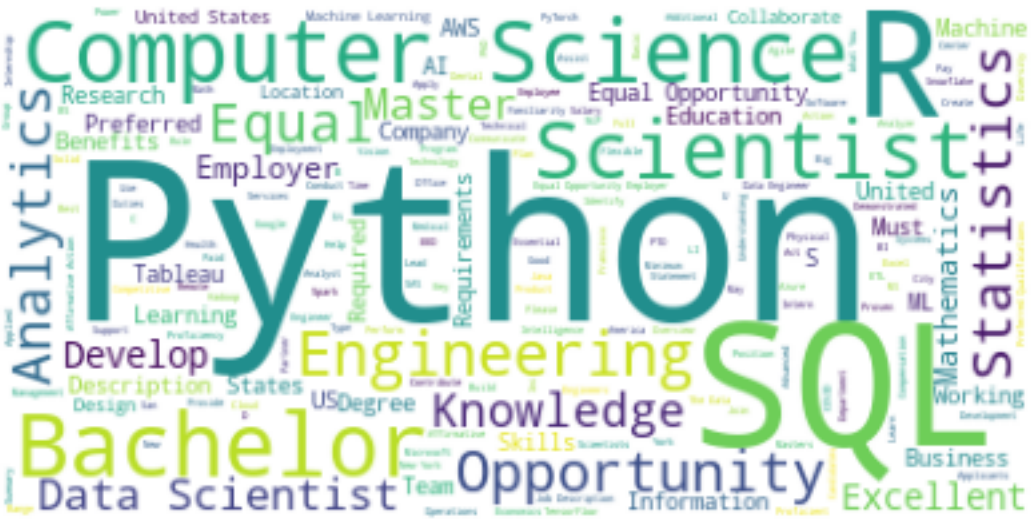
```
##                         role seniority  is_intern                      skill
## job_key
## 0011b268f48bc6c2    analyst   unknown          1                    Tenneco
## 0011b268f48bc6c2    analyst   unknown          1                  Motorparts
## 0011b268f48bc6c2    analyst   unknown          1                        Ride
## 0011b268f48bc6c2    analyst   unknown          1                 Performance
## 0011b268f48bc6c2    analyst   unknown          1                       Clean
## ...                     ...       ...        ...                         ...
## ffa00ac449c52622  scientist   unknown          1  Preferred Qualifications
## ffa00ac449c52622  scientist   unknown          1       Graduating December
## ffa00ac449c52622  scientist   unknown          1        Accommodations-AMS
## ffa00ac449c52622  scientist   unknown          1                Why Join Us
## ffa00ac449c52622  scientist   unknown          1   This Internship Program
##
## [48575 rows x 4 columns]
```

Each row corresponds to the presence of a skill in a job posting. job_key and skill are a candidate key for the table (no duplicates). Obviously some irrelevant terms were picked up, but grouping, counting and sorting we see that the data filtering is not so bad and our top words are indeed skills.

```
## skill
## Python                             419
## SQL                                328
## R                                  245
## Computer Science                   219
## Bachelor                           217
##                                   ...
## MAKING                               1
## MANAGING                             1
## MANAGING RESOURCES                   1
## MANAGING RESOURCES EFFECTIVELY       1
## LOCATION AND HOURS                   1
## Name: skill, Length: 14073, dtype: int64
```

We made a wordcloud using the library of the same name to visualize the relative frequencies of the skills.

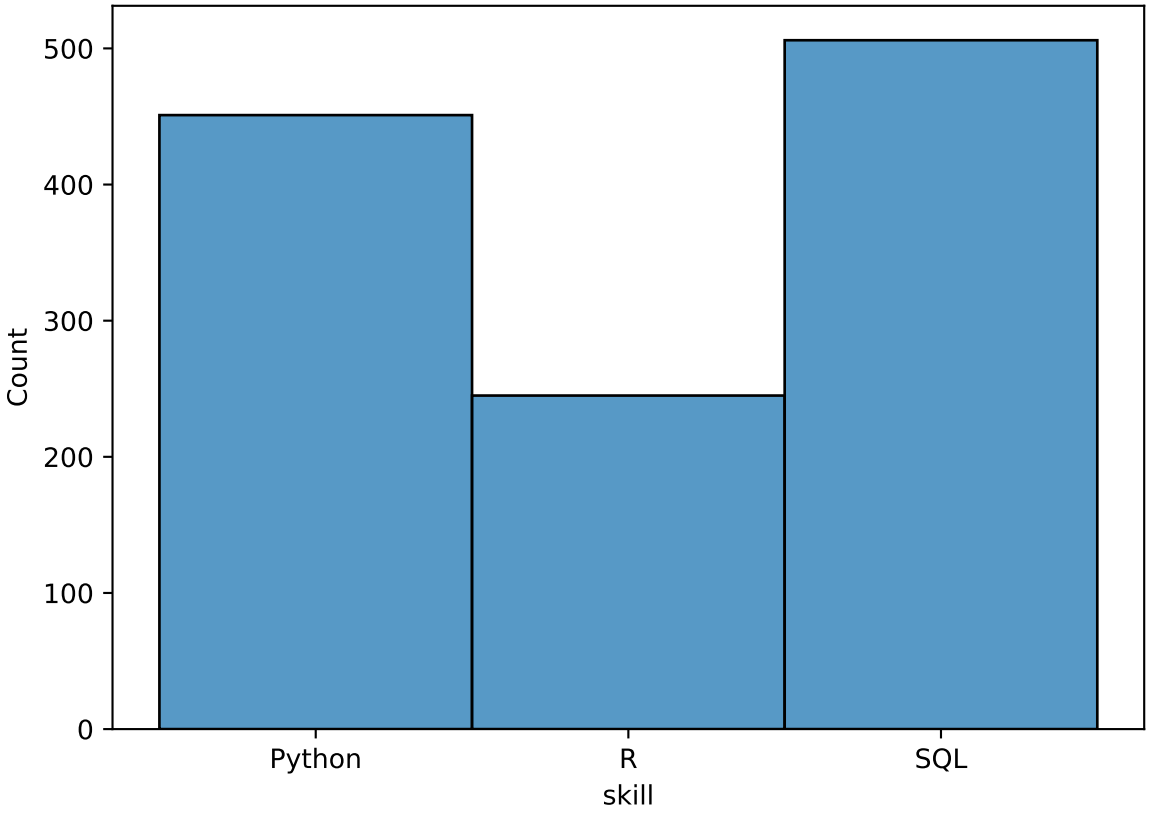
```
## (-0.5, 399.5, 199.5, -0.5)
```



Some notes before moving on. Bachelor is more common than Master. Statistics more common than Mathematics. Python R and SQL look like the top skills so let's investigate them further.

Python VS R VS SQL

Our top three skills are Python, SQL and R. R vs Python is a timeless debate in data science, and we were excited to provide data to help answer this question. We compare their demand below.



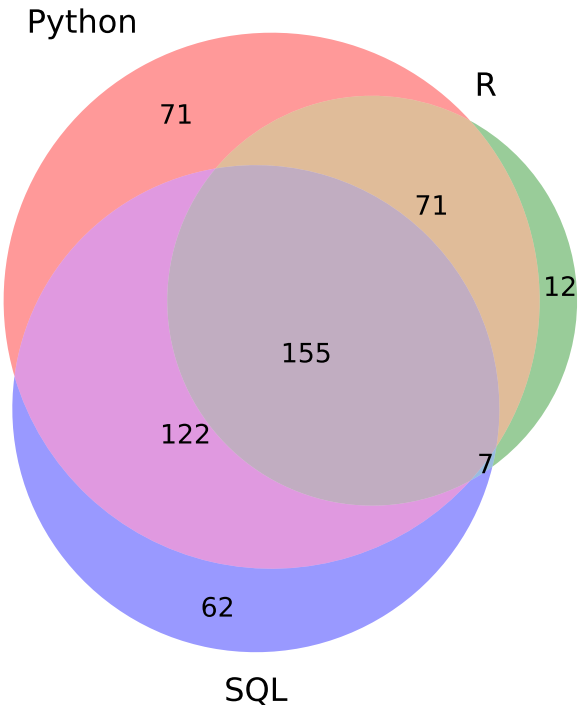

```
## R job proportion: 0.39579967689822293
```

```
## Python job proportion: 0.72859450726979
```

```
## SQL job proportion: 0.8174474959612278
```

Note the above proportions do not sum to 1. This is because of overlap. Many jobs ask for two or all three of the skills. It looks like python and SQL are the clear winners but this doesn't tell the whole story. Take a look at this Venn diagram:

```
## <matplotlib_venn._common.VennDiagram object at 0x0000000033D0C550>
```



We see here that a large portion of jobs demand all 3. If I were to draw a lesson from this it's that jobs often require all 3 of these skills and we should focus on learning which tasks we use each for.

Limitations

We would like to talk about how are project could have been improved upon.

More Data

Web scraping is tedious, and we had to scrape slowly so as not to overwhelm the indeed servers and get our IP's banned. Because we were anxious to start on our analyses, we chose to cut our data collection short. With more data, we could have had larger group sizes for the ANOVA and better more diverse features. For example role could have been split into further sub-categories.

More Questions

We had some more questions in mind when we started the project:

- Which roles demand which skills?
- How valuable is a skill to learn?
- Can we identify new(chronologically) skills as they start to appear in job postings.

- How does education impact salary?
- Which jobs require more education?

We chose to omit education because it was very difficult to extract. Often even if Master's or Bachelor's were listed they were not requirements. Worse still they were often listed together.

Sampling

Because we were relying on Indeed's search tool, our sampling is inherently biased. We can only see what they decide to show us. There could be thousands of bad listings that are being kept from us. Or, it could be that Indeed is missing quality listings from other sites.

Conclusions

Overall I'd say our project was a success. Our Anova analysis successfully identified differences between mean role salaries. The insight that Data Analyst's earn less and that Data Engineer co-op's pay more is practical to our job searches. It was also nice to see our prior beliefs about Data Analyst salaries confirmed. Our regression model achieved good accuracy, and we can use this to get a rough idea if the salaries we will be offered are decent. Finally skill analysis gave us a chance to use new visualization tools.