Job salary prediction

PRESENTED BY HRISHIKESH “RICHIE” TELANG

Overview

- The job salary prediction system essentially aims to predict the salary of any UK job ad based on its contents.
- It also aims to compare how addition of text(job description) into the models increases the prediction power.
- I also want to find out it is likely that there are meaningful relationships between the salaries of jobs in a similar geographical area.
- I am using **Text Analytics** to perform preprocessing of the job description, and **Regression** to quantify the salary range.

Dataset Overview

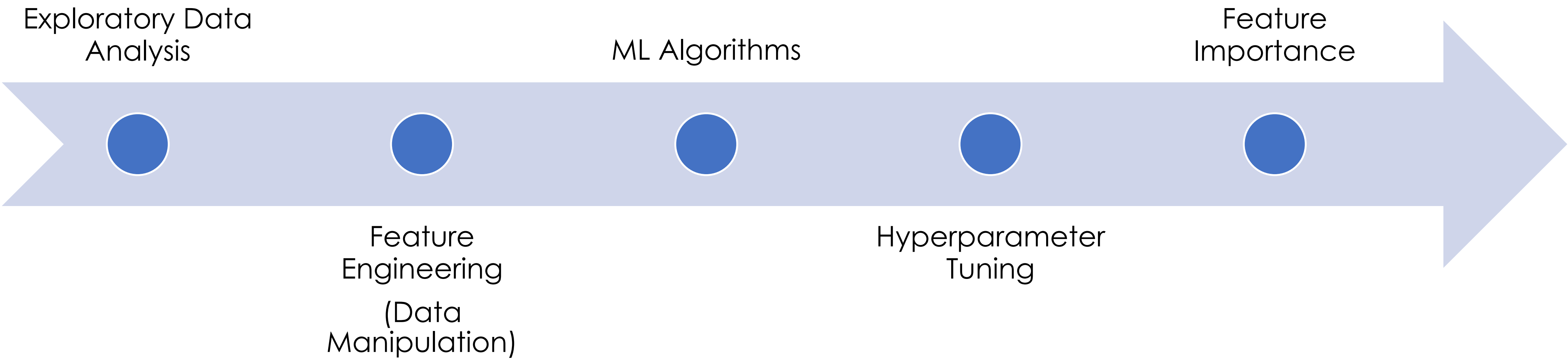
- I got this dataset from the Adzuna Job board dataset which was extracted on Kaggle for the competition.
- Adzuna is a new, innovative search engine for job, property and car ads based in the UK, but expanding rapidly internationally.
- The training dataset is of 60,000 rows and testing dataset of 20,000 rows.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59998 entries, 0 to 59997
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Title               59997 non-null  object
 1   FullDescription     59998 non-null  object
 2   LocationRaw         59998 non-null  object
 3   LocationNormalized  59998 non-null  object
 4   ContractType        16071 non-null  object
 5   ContractTime        44191 non-null  object
 6   Company             52108 non-null  object
 7   Category            59998 non-null  object
 8   SalaryRaw           59998 non-null  object
 9   SalaryNormalized    59998 non-null  int64
 10  SourceName          59998 non-null  object
dtypes: int64(1), object(10)
memory usage: 5.0+ MB
```

Research Questions

- What is the salary of a particular profession based on a location in the UK?
- What is the median salary distribution across regions in the UK?
- What are the top 10 parts of speech in the job descriptions? How frequently do they appear?
- Which parts of speech have the highest frequency?
- How do these numbers change if you exclude stopwords?

My Project Approach



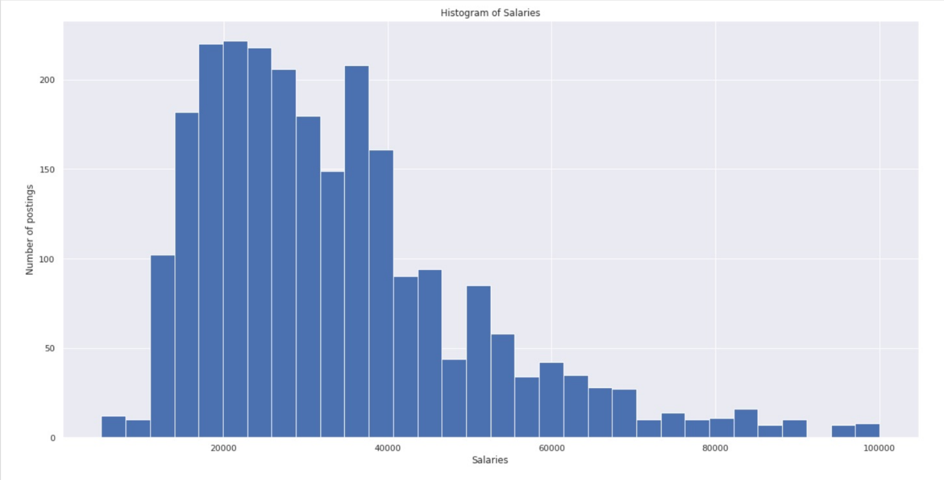
Exploratory Data Analysis

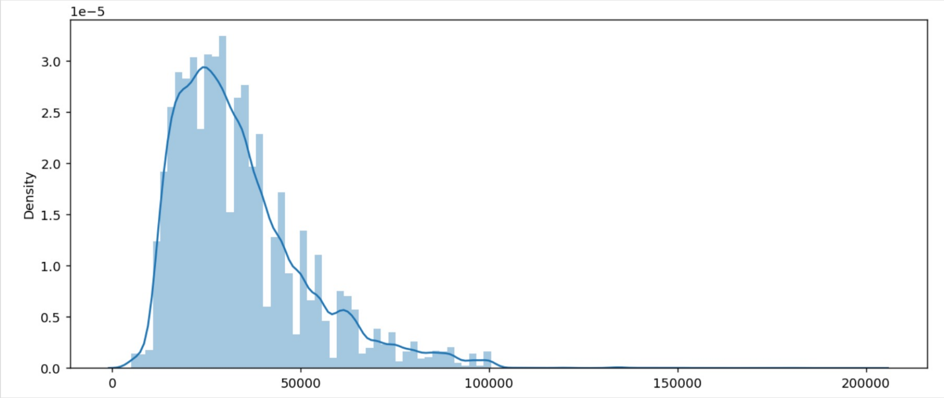
1. EXPLORATORY DATA ANALYSIS FOR **REGRESSION**
2. EXPLORATORY DATA ANALYSIS FOR **CLASSIFICATION**

Exploratory Data Analysis for Regression

I. Salary Distribution

- Left-skewed
- Jobs are on the lower end of the job salary spectrum.
- Ranges majorly concentrated in the median range of £50,000 or less





I. Salary Analysis

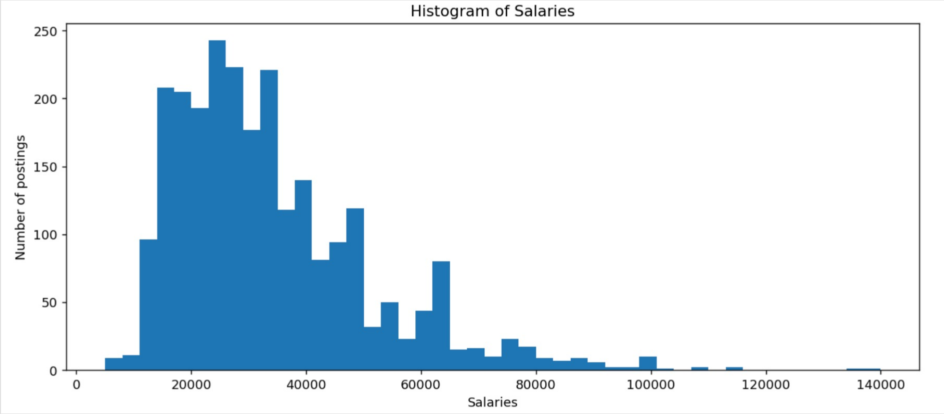
- Performed Random Sampling of 2,500 rows if the representation is the same
- Turns out that indeed it is!

```
# Randomly selecting 2500 rows to train the classifier
import random
random.seed(1)
indices = list(data.index.values)
random_2500 = random.sample(indices,2500)

# Subsetting the train data based on the random indices
sample = data.loc[random_2500].reset_index()
```

```
plt.figure(figsize=(12,5), dpi=130)
plt.hist(sample['SalaryNormalized'], bins='auto')
plt.xlabel('Salaries')
plt.ylabel('Number of postings')
plt.title('Histogram of Salaries')
```

```
Text(0.5, 1.0, 'Histogram of Salaries')
```



2. Title

- The top 5 Titles in the dataset were:

a) **Business Development Manager**

b) **Project Manager**

c) **Management Accountant**

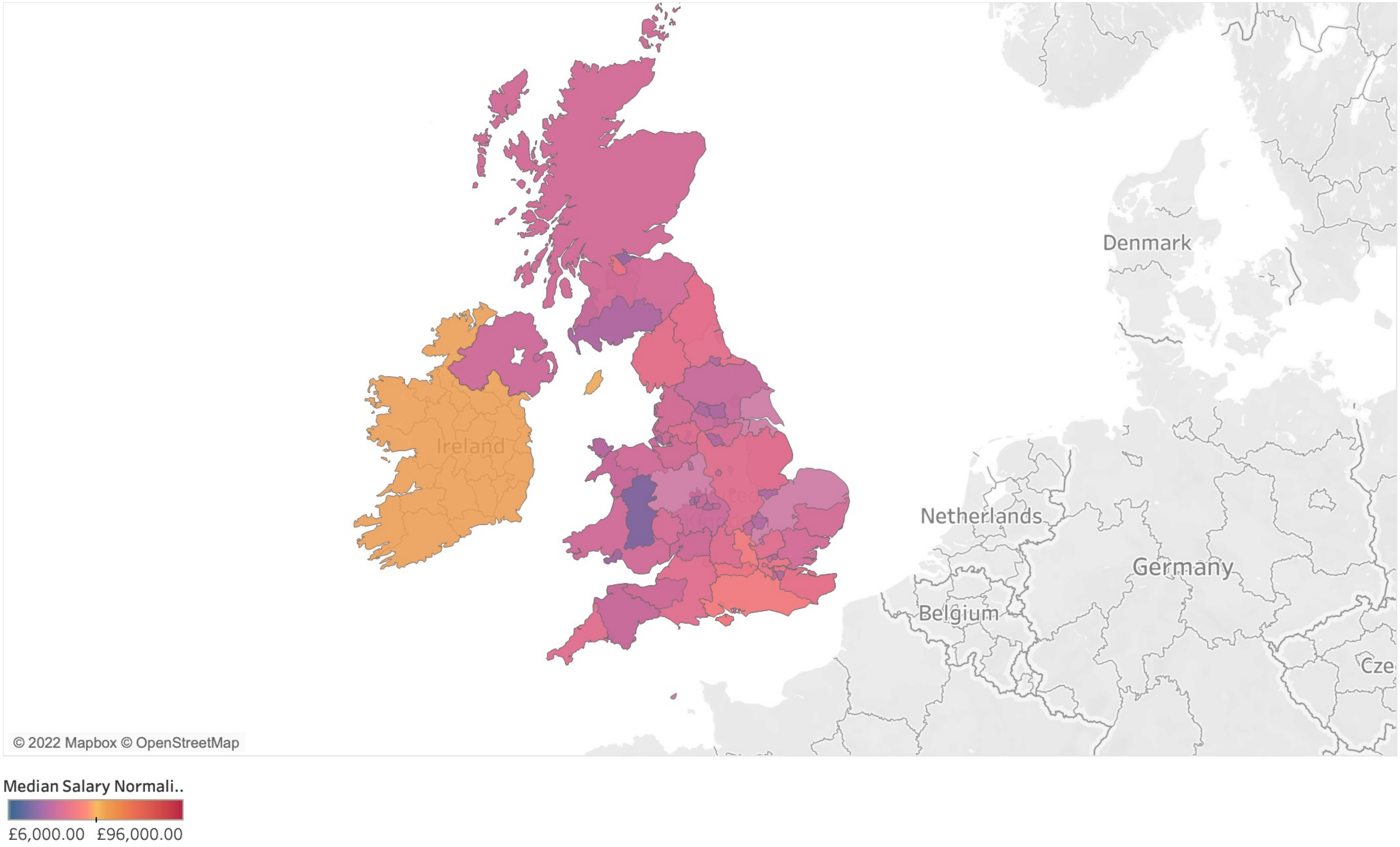
d) **Sales Executive**

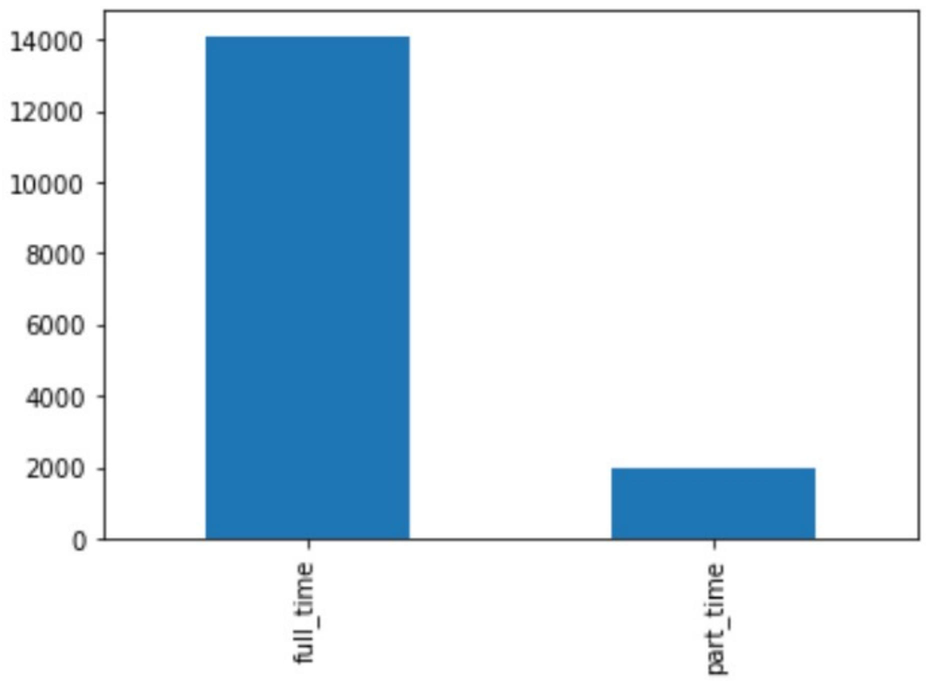
e) **Cleaner**

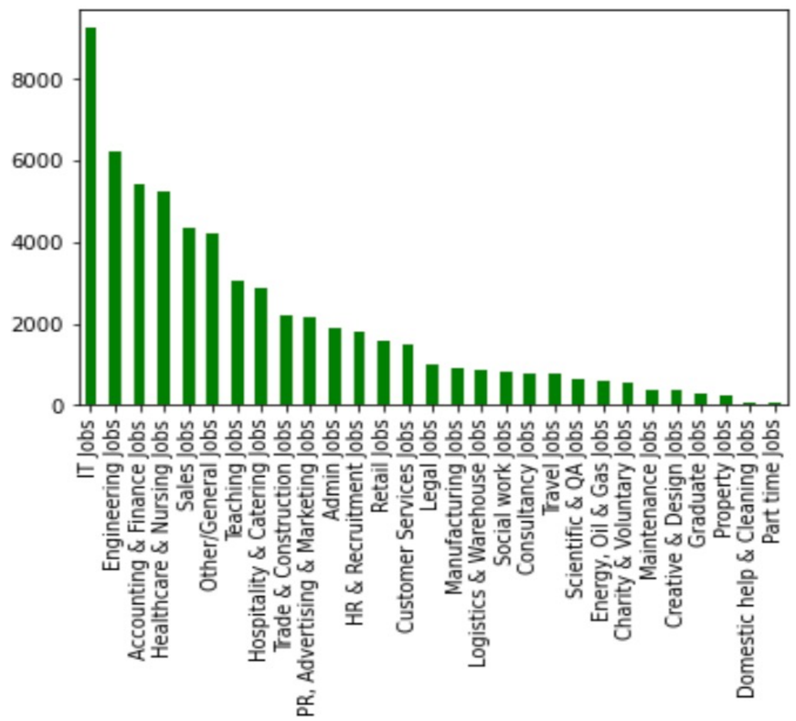
```
data["Title"].value_counts()[:50]
```

```
business development manager        230
project manager                     189
management accountant               175
sales executive                     144
cleaner                             139
mechanical design engineer          120
assistant manager                   119
administrator                       118
account manager                     117
recruitment consultant              110
finance manager                     109
credit controller                   109
accounts assistant                  102
financial controller                 92
sales manager                        89
```

3. Median Salary Distribution across Locations in the UK







4. Contract Type and Job Categories

Feature Engineering

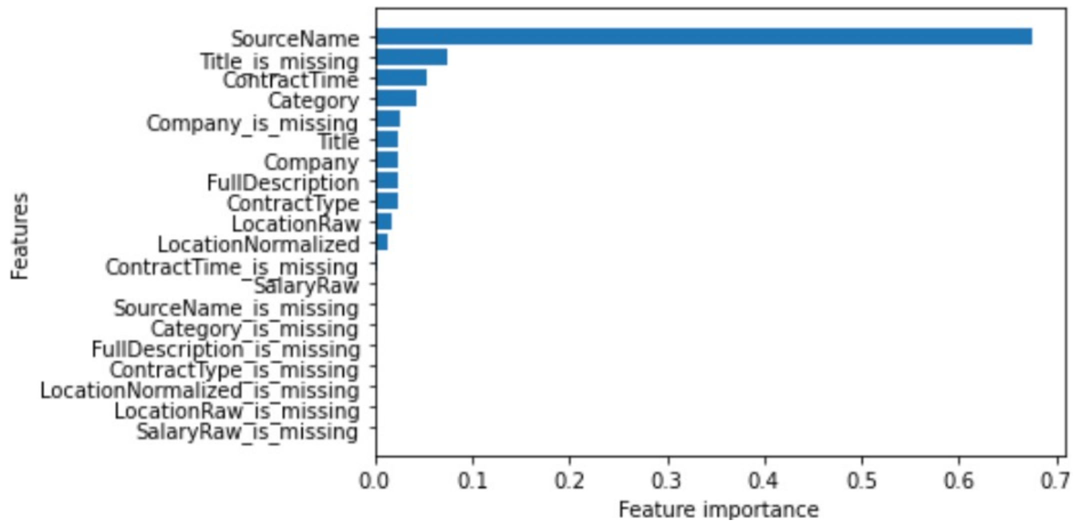
- I dropped “id” and “NUM_RAND”
- Then, I checked for categorical and numerical variable
- Later, I converted categorical (string values) into numerical values.
- To fill missing values, I added label_is_missing as a Boolean value
- I converted categories into numbers and added +1 to it

Modeling

	RF Regressor	SVM Regressor	Lasso Regularization	DT Regression
MSE	7048.91	17608.69	16078.42	9707.56
R-Score	0.8373	-0.01478	0.1539	0.6915
Adjusted R-Score	0.8371	-0.0164	0.1525	0.6910

Tree Importance

- Used Testing set and performed the same data manipulation operations
- Trained and tested the model



Exploratory Data Analysis for Classification

Steps:

- Step 1: Preprocessing – Tokenization, lemmatization, Stopword removal, POS
- Step 2: Perform ML Analysis **with** and **without** the job description

Exploratory Data Analysis (for job descriptions)

While looking at the data, some of the values are masked as *** and they turn out to be of no value for us in the analysis. In addition to that, there are a few data cleaning steps that I have performed as follows:

- Remove website links from the data
- Remove punctuations
- Removing numbers

```
# To obtain the full width of a cell in a dataframe
pd.set_option('display.max_colwidth', -1)
desc = data.loc[1,'FullDescription']

# Creating a list of words from all the job descriptions in train_df1 data
all_desc = []
for i in range(0,data.shape[0]):
    desc = data.loc[i,'FullDescription']
    desc1 = desc.lower()
    # Removing numbers, *** and www links from the data
    desc2 = re.sub('[0-9]+\S+|\s\d+\s|\w+[0-9]+|\w+[\*]+.*|\s[\*]+\s|www\.[^\s]+','',desc1)
    # Removing punctuation
    for p in punctuation:
        desc2 = desc2.replace(p,'')
    all_desc.append(desc2)
```

```
# Creating word tokens for all the descriptions
final_list = []
for desc in all_desc:
    word_list = word_tokenize(desc)
    final_list.extend(word_list)
```

POS Identification before and after Stopword Removal

```
# 3. Tagging parts of speech
pos_tagged = nltk.pos_tag(final_list)

# 4. Identifying the most common parts of speech
tag_fd = nltk.FreqDist(tag for (word, tag) in pos_tagged)
tag_fd.most_common()[:10]
```

```
[('NN', 3328920),
 ('JJ', 1501159),
 ('IN', 1377394),
 ('DT', 1093017),
 ('NNS', 1082790),
 ('VB', 702013),
 ('CC', 656927),
 ('VBG', 488373),
 ('TO', 432236),
 ('VBP', 329925)]
```

```
# Excluding stopwords from the analysis
list_wo_stopwords = []
for w in final_list:
    if w not in stop_words:
        list_wo_stopwords.append(w)

# 3. Tagging parts of speech
pos_tagged_wo_sw = nltk.pos_tag(list_wo_stopwords)

# 4. Identifying the most common parts of speech
tag_fd_wo_sw = nltk.FreqDist(tag for (word, tag) in pos_tagged_wo_sw)
tag_fd_wo_sw.most_common()[:10]
```

```
[('NN', 3226148),
 ('JJ', 1553326),
 ('NNS', 1069511),
 ('VBG', 550203),
 ('VBP', 368583),
 ('RB', 284362),
 ('VBD', 169385),
 ('VBN', 163728),
 ('VB', 163504),
 ('IN', 125710)]
```

Frequency of POS (WordCloud)

	words	freq
24	experience	92336
42	role	68976
115	team	66979
77	client	63068
110	work	62569
403	business	59998
5	service	53074
164	skill	51527
202	working	47984
492	manager	46180

Machine Learning Algorithms

Model **without** job description:

Target variable will be the **SalaryNormalized** column

- I created labels for the SalaryNormalized column on the basis of its percentile: if greater than 0.75 it is 1 else 0.
- **Creating a proxy variable for location** Since creating dummy variable will bloat the dataset, we will create a proxy variable for the location by taking the cities with high cost of living under one group(1) and the others in a separate group(0).

Creating dummy variables for all the columns except job description.

Machine Learning Algorithms

Model **without** text variables:

Machine Learning Classifier used: Bernoulli Naïve Bayes Classifier

Accuracy: 76.27%

Machine Learning Algorithms

Model **with** job description:

- I ran only Multinomial NB Classifier by performing these steps:
  - Without removing stopwords from the data
  - After removing stopwords from the data
  - After lemmatizing the data
- This will help us understand the effect of each step on the accuracy of the result
- Performance Metric to use: Accuracy

Machine Learning Algorithms

Without removing stopwords from the data	Confusion matrix: [[10898 2552] [1172 3378]] Accuracy using MultinomialNB: 0.7931111111111111
After removing stopwords from the data	Confusion matrix: [[10878 2572] [1124 3426]] Accuracy using MultinomialNB: 0.7946666666666666
After lemmatizing the data	Confusion matrix: [[10899 2551] [1170 3380]] Accuracy using MultinomialNB: 0.7932777777777777

Future Scope

- Tensorflow aka Deep Learning or Stacking Ensembles.
- Usage of Unigrams, bi-grams and N-gram models
- We can use SVM algorithm to predict the salary based on the text data.
- **Sentiment Analysis Results** can also be incorporated as a feature.

Thank you!