Job Salary Prediction

Hrishikesh Mahesh Telang

SUID: 889489533

Syracuse School of Information Studies (iSchool), Syracuse University

IST 707: *Applied Machine Learning*

Dr. Stephen Wallace

May 10, 2022

Problem Statement and Definition:

In today's day and age of so many jobs present on websites such as LinkedIn, Glassdoor, AngelList, etc., it is usually nice to know what the salary could be based on its contents. Every job posting contains a unique id, title, the job description, the location, the contract type (whether it is full-time or part-time), the company, and the salary associated with it. Sometimes, whenever a job applicant wishes to understand the median salary based on similar job titles in the market, they would have to go through the hassle of checking every job description and comparing the salaries with similar titles in the same job portal to determine their salary. Not only is this time-consuming but also unrealistic and less akin to today's technological growth happening in our society. In this project, I have created a prediction system that shall provide whether the job falls into the higher or lower category of items. However, I have also helped quantify the job title's job salary. I have also mapped which geographical locations have higher median salaries than the rest of the UK. I have used regression to quantify the salary and classification to perform binary classification of higher and lower salary ranges to achieve this. However, for the classification system, I have used two types of target variables: one that uses only the job description as the target variable and the other that uses all the rest of the variables except the job description. Thus, it is also mandatory to do some application text mining preprocessing skills for this project.

Research Question:

Every machine learning or a data science project needs to have an end goal: *what is it exactly that we are trying to predict, or what insights are we supposed to gain from our analyses?* Some of our most pertinent questions are as follows:

a) What is the salary of a particular profession based on location in the UK?
b) What is the median salary distribution across every county in the UK?
c) What are the top 10 parts of speech in the job descriptions? How frequently do they appear?
d) Which parts of speech have the highest frequency? (WordCloud Analysis)
e) How do the parts of speech change if you exclude stopwords?
f) How does the accuracy of the machine learning model vary with the inclusion and exclusion of stop words and lemmatization?

Dataset Description:

a) The dataset has been obtained from the Adzuna Job board, which was extracted on Kaggle for a coding competition. [1]
b) Adzuna is a new, innovative search engine for job, property, and car ads in the UK. [1]
c) The training dataset originally contained 195,814 rows. The testing and validation set contained 48,954 rows; however, for more efficient and faster analysis, I randomly subsampled the dataset to procure only 60,000 rows for the training set and the testing and validation set dataset of 20,000 rows. [1]

The dataset description is as follows:

The primary dataset consists of many rows representing individual job ads and a series of fields about each job ad. These fields are as follows:

a) **Id**: A unique identifier for each job ad. [1]
b) **Title:** It is a free text field supplied to us by the job advertiser as the title of the job ad. Normally, this is a summary of the job title or role. [1]

c) **FullDescription**: The full text of the job ad as provided by the job advertiser. Some columns appear as ***s; because the dataset was already stripped of values from the description to ensure that no salary information appears within the descriptions. [1]
d) **LocationRaw:** The free text location provided by the job advertiser. [1]
e) **LocationNormalized:** Adzuna's normalized location from within our location tree, interpreted by us based on the raw location. Note that the normalizer was not perfect. [1]
f) **ContractType:** full_time or part_time, interpreted by Adzuna from the description or a specific additional field received from the advertiser. [1]
g) **ContractTime:** permanent or contract, interpreted by Adzuna from the description or a specific additional field received from the advertiser. [1]
h) **Company:** the employer's name supplied by the job advertiser. [1]
i) **Category:** The 30 standard job categories this ad fits into. [1]
j) **SalaryRaw:** the freetext salary field we received in the job advert from the advertiser. [1]
k) **SalaryNormalised:** the annualized salary interpreted by Adzuna from the raw salary. Note that this is always a single value based on the midpoint of any range found in the raw salary. This is the target value that I plan to predict. [1]
l) **SourceName:** the name of the website or advertiser from whom the job advert was received. [1]

Workflow of the system:
My workflow is divided into two types of problems: Regression and Classification. Since I have incorporated NLP Principles in the Classification Problem, the workflow shall differ slightly from the regression problem:

a) Regression:
Generally, every ML prediction problem goes through the following standard steps as mentioned in the following diagram:

i. **Exploratory Data Analysis:** Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypotheses, and check assumptions with the help of summary statistics and graphical representations. [2]
ii. **Feature Engineering (Data Manipulation):** Feature engineering uses domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling. The goal of feature engineering and selection is to improve the performance of machine learning (ML) algorithms [3].
iii. **ML Predictive Modeling:** After the dataset is ready for prediction, we use either one of the ML models (supervised, unsupervised, reinforced) in our system. Our problem is not only supervised, but we are also going through the "regression analysis" route. I have only used Random Forest, Decision Tree, Lasso, and SVM Regression in my approach.
iv. **Hyperparameter Tuning:** Hyperparameter Tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process. Each trial is a complete execution of a training application with values for chosen hyperparameters set within the specified limits. This process, once finished, will provide

the set of hyperparameter values that are best suited for the model to give optimal results. This process was only applied to RF Regression. [4]

v. **Feature Importance:** Feature Importance refers to techniques that calculate a score for all the input features for a given model—the scores represent the "importance" of each feature. A higher score means that the specific feature will significantly affect the model used to predict a particular variable. [5]

b) Classification:

i. Using only job description as the input variables

ii. Without using job description as the input variables

For both classifications, the following workflow was followed:

1. **Preprocessing type 1: Stemming, Tokenization, Lemmatization, and POS Tagging**:
   - **Stemming:** Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or the roots of words, known as a lemma.
   - **Tokenization:** Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either word, characters, or subwords. Hence, tokenization can be broadly classified into word, character, and subword (n-gram characters) tokenization. In my current approach, I have only performed word analysis.
   - **Lemmatization**: Lemmatization usually refers to doing things properly using a vocabulary and morphological analysis of words, usually aiming to remove inflectional endings only and return the base or dictionary form of a word, known as the lemma.
   - **POS Tagging**: Part-of-speech (POS) tagging is a popular Natural Language Processing process that categorizes words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context.

2. **Preprocessing type 2: Bag of Words (CountVectorizer) model:**
   A very common feature extraction procedures for sentences and documents is the bag-of-words approach (BOW). In this approach, the histogram of the words within the text, i.e. considering each word count is taken as a feature.

3. **Machine Learning Modeling:** As defined earlier, upon processing each of the sentences with the BOW model, I performed ML Classification using Multinomial and Bernoulli Naive Bayes Algorithms.

Descriptive Analytics:

a) EDA and Data Preprocessing (Data Preparation):

i. Regression

The dataset essentially contains 13 rows of which ID and NUM_RAND values were dropped. Except for SalaryNormalized column, most of the others are all categorical values. When we wanted to check the null values, we found out that ContractType, ContractTime and Company had 43927, 15807 and 7890 missing rows respectively. Later, I checked the values of SalaryNormalized only to find out that the mean range falls around £34,005. The dispersion value of the salary ranges is £17,577. Then I checked the histogram of salaries using hist() and kde plots in the following diagrams:

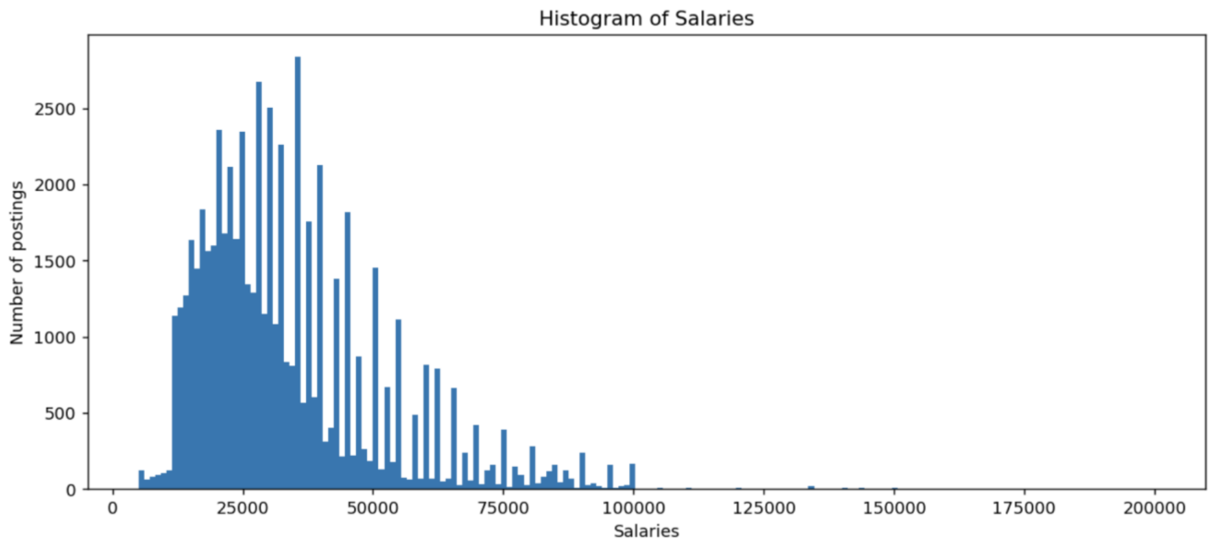


Fig 1: Histogram of SalaryNormalized

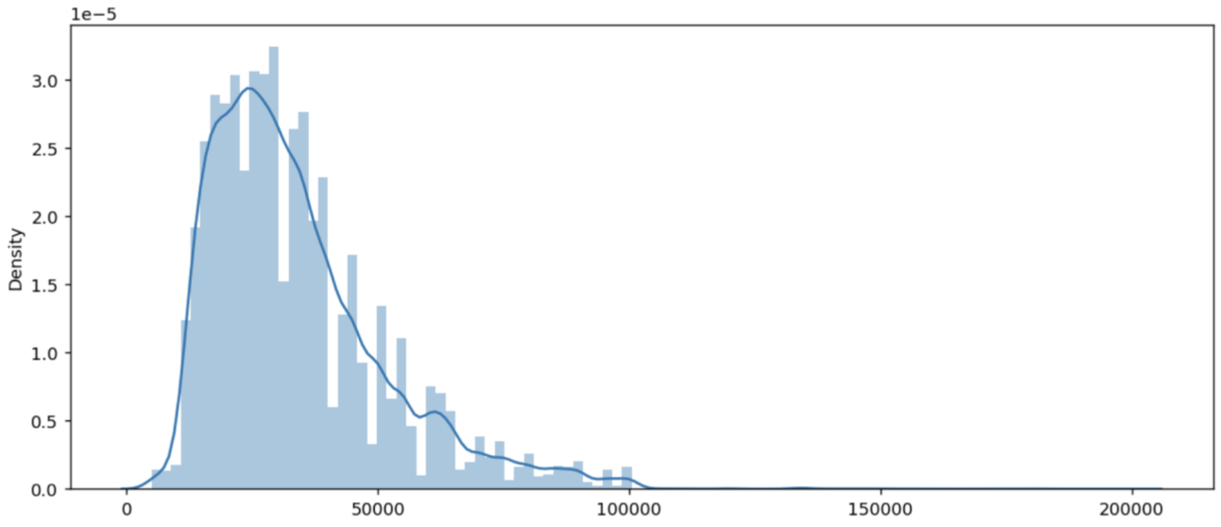


Fig 2: KDE plot of SalaryNormalized

We can observe that the salaries are skewed mostly towards the lower end (mostly < 50,000) showing that most of the jobs are on the lower end of the job salary spectrum. This distribution might be useful as we move further into the analysis, as this might help us detect if there is any bias in our final analysis. In order to testify whether the SalaryNormalized is really this left skewed, I took a random sample of 2500 rows and plotted its histogram in the following figure. It can be seen that the sampled distribution does not differ a lot from the previous two distributions which means that our distribution really has a skew.

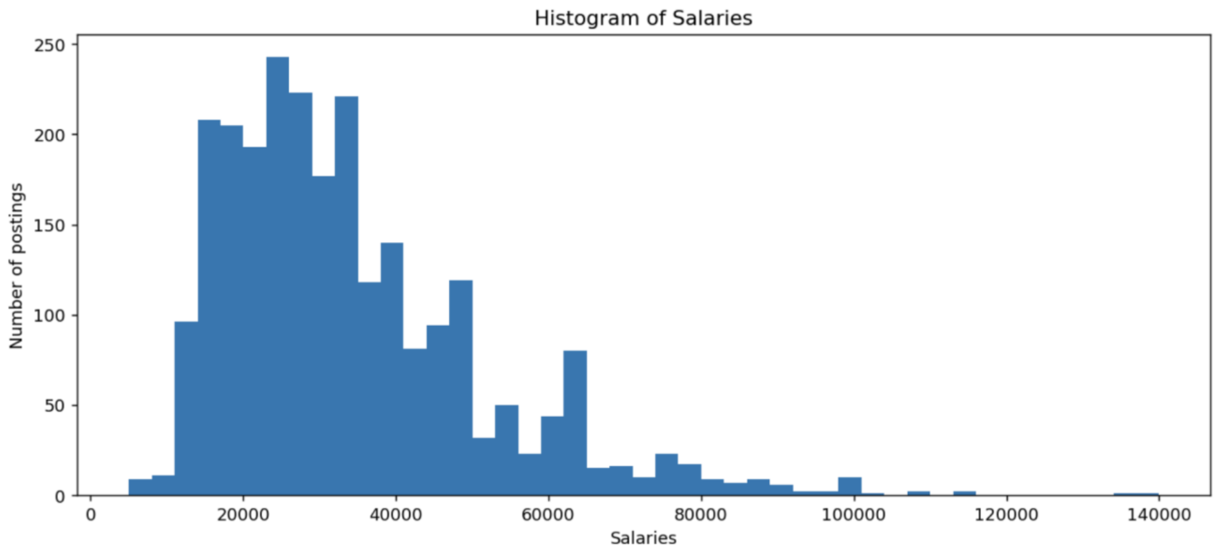


Fig 3: Sample distribution

I plotted maps using Tableau to see the distribution of median salaries across the UK. We can observe that Ireland has a higher median salary than the rest, but that is also because it contains only one column of a job posting which was located at Ireland. North-East England (£36,000), South-East England (£42,500), London (£38,500), etc. have higher median salary ranges than Falkirk (£19,796) and Powys (£15,184).

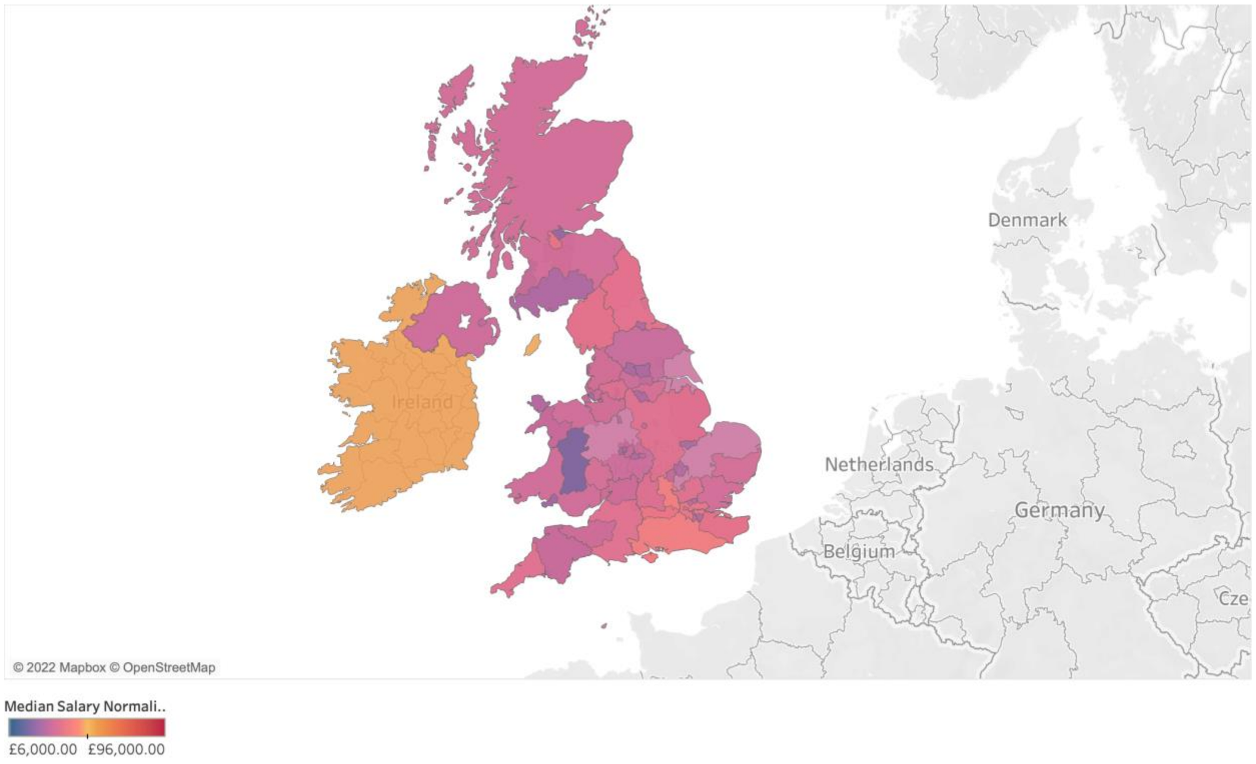


Fig 4: Median Salary Distribution across the UK & Ireland

Upon further analysis, we checked that LocationNormalized has 1761, with major job postings centered around London, Manchester, Leeds, Birmingham, Surrey and Bristol. We can see in Fig

4 and 5 that ContractType had higher Full Time Roles than Part Time roles, and more permanent jobs than contract roles, exactly as per the real-world scenario. Amongst the category of jobs, there are more number of IT and Engineering jobs than other types of jobs in the market. This is followed by Accounting and Finance Jobs as well as Healthcare jobs.

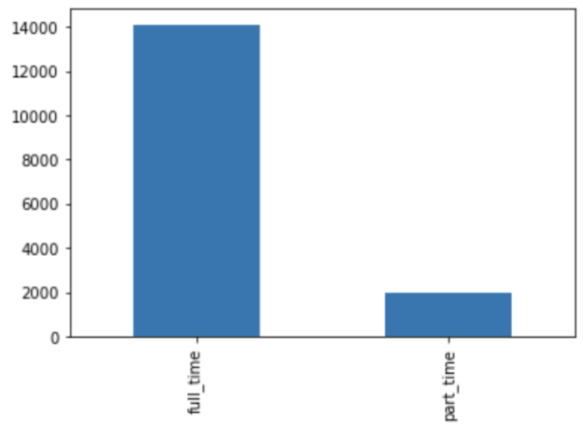


Fig 5: ContractType

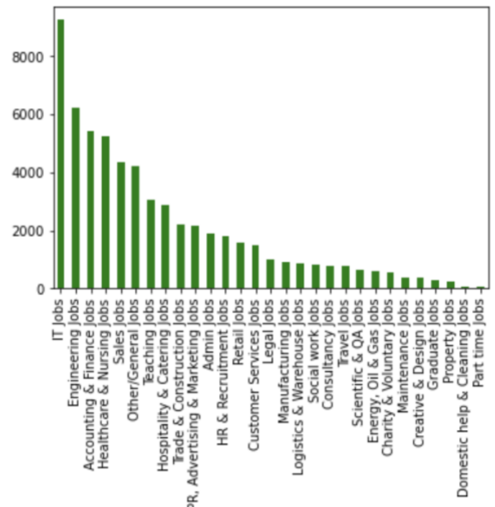
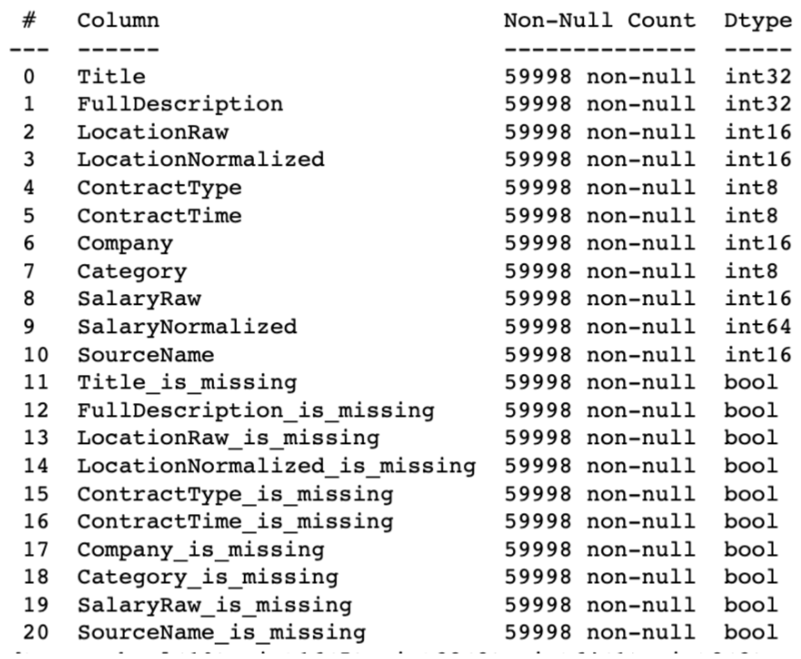


Fig 6: Category

Data Preprocessing: I first tried checking which values were string values and which were numeric. Then I converted all the string values into categorical ones, added a binary column that would raise a Boolean value of '1' if the label have a missing value else '0', and converted all the categories into numbers and added +1 to it, since all categories begin with a 0. The final data set is as follows:

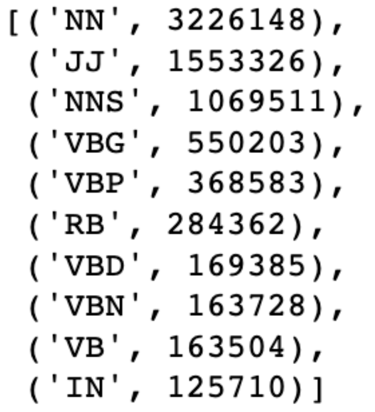


```
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Title                         59998 non-null  int32
 1   FullDescription               59998 non-null  int32
 2   LocationRaw                   59998 non-null  int16
 3   LocationNormalized            59998 non-null  int16
 4   ContractType                  59998 non-null  int8
 5   ContractTime                  59998 non-null  int8
 6   Company                       59998 non-null  int16
 7   Category                      59998 non-null  int8
 8   SalaryRaw                     59998 non-null  int16
 9   SalaryNormalized              59998 non-null  int64
 10  SourceName                    59998 non-null  int16
 11  Title_is_missing              59998 non-null  bool
 12  FullDescription_is_missing    59998 non-null  bool
 13  LocationRaw_is_missing        59998 non-null  bool
 14  LocationNormalized_is_missing 59998 non-null  bool
 15  ContractType_is_missing       59998 non-null  bool
 16  ContractTime_is_missing       59998 non-null  bool
 17  Company_is_missing            59998 non-null  bool
 18  Category_is_missing           59998 non-null  bool
 19  SalaryRaw_is_missing          59998 non-null  bool
 20  SourceName_is_missing         59998 non-null  bool
```

Fig 7: Input Dataset for ML Prediction

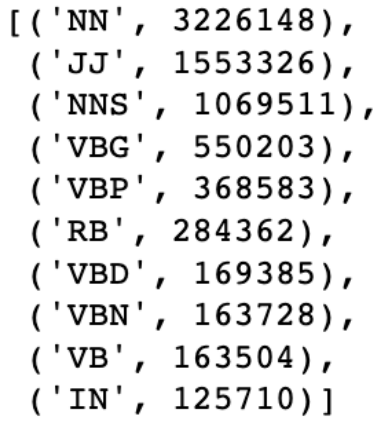
ii. **Text Mining:**

After having imported all the packages, including the stopwords, I acquired the entire FullDescription column and removed all the punctuations and the special characters and symbols using regular expression functions. Then, I performed tokenization and used parts of speech tagging to identify the words. Note that this operation included all the stopwords. The POS tagging is done using the Penn Treebank Project.



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

Fig 8: POS Tags before excluding stopwords



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

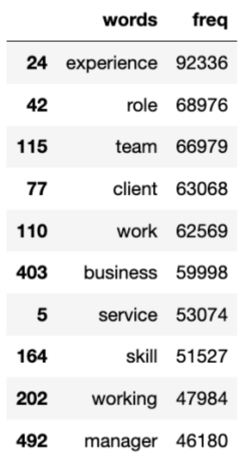
Fig 9: POS Tags after excluding stopwords

You can see in Fig 8 that Noun (NN) , adjective (JJ), preposition (IN), determiner (DT), plural nouns(NNS) are the most common parts of speech from the job descriptions. [6]

From Fig 9, after removing stopwords, there are two important observations in comparison to the previous result

- Prepositions and determiners disappeared from the top 5 set as most of these are present in the stopwords imported from NLTK.
- The counts of nouns and plural nouns have decreased, and the adjectives have increased.
- Verb, gerund or present participle (VBG) and Verb, non-3rd person singular present(VBP) moved to the top 5 list.

I wanted to identify which parts of speech had the highest frequency. For this I performed lemmatization followed by preparing a 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

Fig 10: Frequency Table

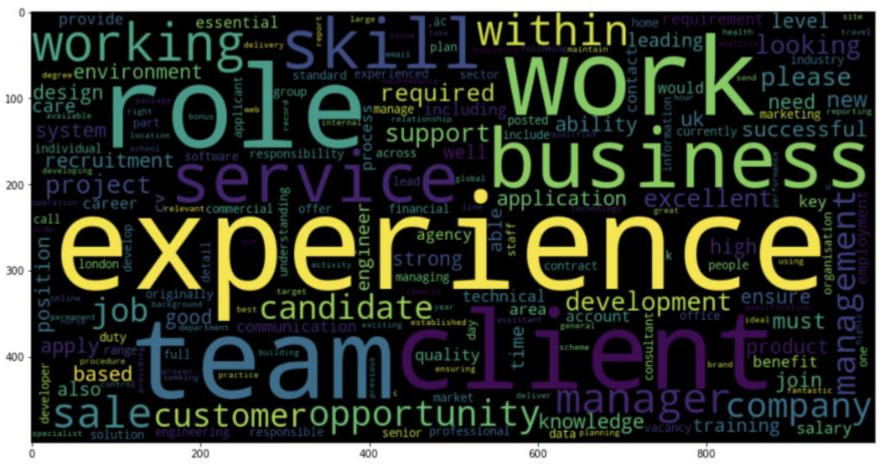


Fig 11: WordCloud

It is clearly apparent from Fig 10 and 11 that experience, role, team, client, work, and business are taking the most frequencies.

Data Preprocessing:
I had to create a target variable. Using the 75th percentile, I put all the salaries in the 'higher range' (1) and the rest as the 'lower range' (0). Later, based on the EDA performed for regression and the link here, I created another proxy column named 'location flag' that returns a Boolean flag if they belong to any of the costly cities list ('London', 'Brighton', 'Edinburgh', 'Bristol', 'Southampton', 'Portsmouth', 'Exeter', 'Cardiff', 'Manchester', 'Birmingham', 'Leeds', 'Aberdeen', 'Glasgow', 'Newcastle', 'Sheffield', 'Liverpool'), if they do, it is 1 else 0. Next, I drop all the exhaustive categorical columns (I cannot perform one hot encoding as the dataset will be too big) and created dummy values for those that had fewer categories.

Machine Learning Analysis:

1. **For Regression**
   I performed Regression using the following four algorithms:
   i. Random Forest Regressor
   ii. SVM Regressor
   iii. Lasso Regularization
   iv. Decision Tree Regressor

   I chose random forest and random forest as it can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed. I considered SVM because it is effective in cases where the number of dimensions is greater than the number of samples.

   I also performed hyperparameter tuning and tree importance but only limited to the RF Regression as the other models took a lot of time to run and I had less time constraint as well. For all the classifiers, I created a training and testing dataset in 80-20%

The performance metrics I've used are as follows:

a) **Mean Squared Error:** In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.
b) **R-Score:** The R-Score is an important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.
c) **Adjusted R-Score**: The R-squared statistic isn't perfect. In fact, it suffers from a major flaw. Its value never decreases no matter the number of variables we add to our regression model. That is, even if we are adding redundant variables to the data, the value of R-squared does not decrease. It either remains the same or increases with the addition of new independent variables. This clearly does not make sense because some of the independent variables might not be useful in determining the target variable. Adjusted R-squared deals with this issue. It considers the number of independent variables used for predicting the target variable. In doing so, we can determine whether adding new variables to the model increases the model fit.

Lastly, I performed Hyperparameter tuning and Feature Importance but only for the RF regressor to check which feature is the best performing one.

2. **For Classification**
I performed Classification using Bernoulli and Multinomial Naïve Bayes Classifier for both of the following scenarios:

i. **Building a model without the job description column:**
After performing data preprocessing for classification, the following are the columns that remain after I drop the values: *ContractType, ContractTime, Category, SourceName, LocationFlag*. I created dummy values for the dataset, and I split my ML classifier into training and testing set of 70-30%. I passed the Bernoulli Naïve Bayes Classifier as it generally produces great accuracies for all NLP oriented problems, including the famous spam and ham email classifier.

ii. **Building a model only with the job description column:**
The analysis is again subdivided into three categories:
a) Without removing stopwords from the data
b) After removing stopwords from the data
c) After lemmatizing the data
After performing each of the steps mentioned above, I performed the Bag of Words (BOW) model using CountVectorizer. After this I performed Multinomial Naïve Bayes Classification. Due to a lot of crashing of my Jupyter Kernel, I could not perform Naïve Bayes classification using the Bernoulli model.

The performance metrics I used here is only **accuracy** due to time constraints.

Results and Discussion:

1. **For Regression**

	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

We can see that RF Regressor and DT Regressors are working well for the following regression problem. We can see that the MSE scores are much lesser as compared to the scores of SVM and Lasso, which means that these regression lines have fitted the lines much better than the latter two. The R-Score of the latter two are in the negative for SVM and Lasso which suggests that the model is not well appropriate.

In the hyperparameter tuning portion, I used RamdomisedSearchCV using 5 fold cross validation. However, my R-Score further reduced to 0.65 which led me to scrap this portion. Finally I used the validation dataset, performed the same preprocessing operations and finally used the same model to build predictions and created a feature importance tree as shown below:

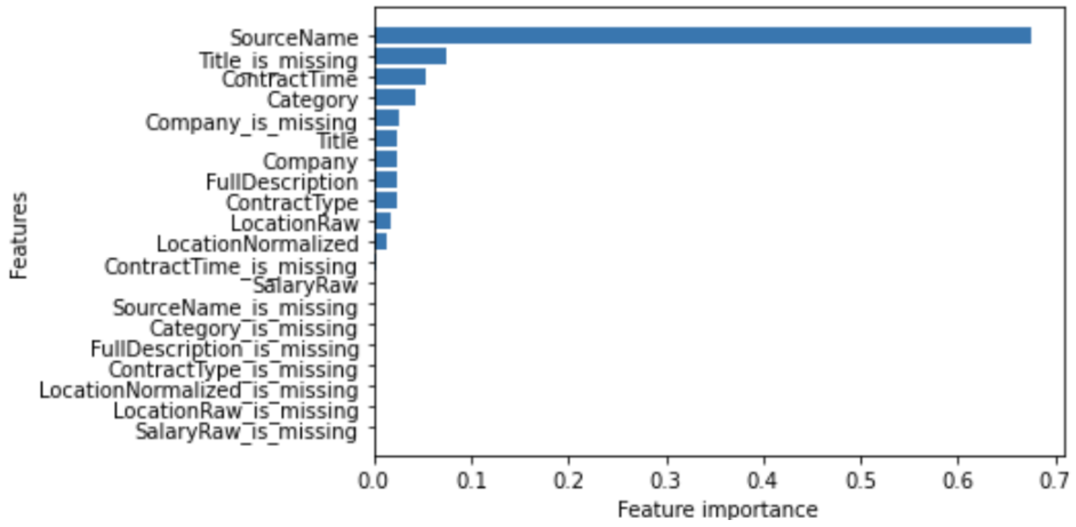


Fig 12: Feature Importance using the RF regressor Model

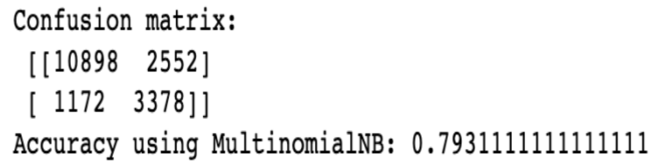
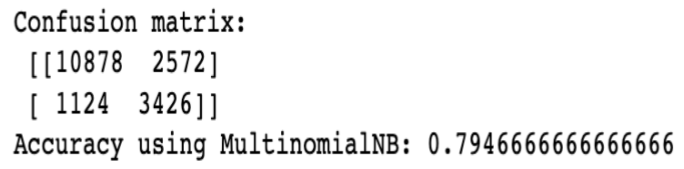
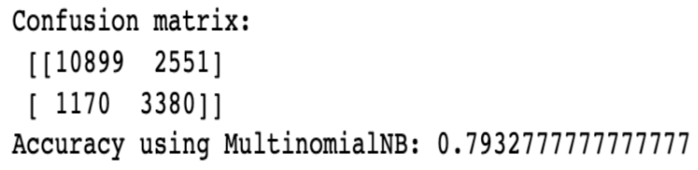
We can observe that the source name decides the predictive power of the model, followed by whether the title even existed or not. Rest of the other columns lie within the same range of the dataset.

2. **For Classification:**

   i. **Without the job description column:**
   The ***accuracy*** of the algorithm was 75.3%, however, with certain implicit errors within the algorithm, we could also calculate the ***baseline accuracy*** which procured 74.93%. Therefore, from this analysis we can observe that the baseline accuracy

does not have a stark contrast with the actual accuracy. Which is when we only performed machine learning analysis with only the job description.

ii. **With the job description column:**

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

We can see that text analysis of the job description alone provided a much higher accuracy than the machine learning prediction without the job description.

Conclusion:
I created a job prediction system to determine what can potentially be the salary of the job based on the variables with and without the job description. I performed regression analysis without the job description and quantified all the categorical values to provide a R-Score around 0.85. Besides, I also performed classification with and without the text variable. For text analysis, I performed the preprocessing operations such as tokenization, stemming, lemmatization, POS Tagging and using Bag of Words Model to then perform ML Classification using Naïve Bayes. While there is still a lot of scope in the classification, the following is what I could potentially do:

- Perform other classification accuracies for the model without job description and also perform comparative ML Analysis using other performance measurement parameters such as precision, recall and F1 Score.
- **Vectorization:** Count Vectorizer (Frequency Counting), Tfidf Vectorizer (Frequency Counting penalizing common words) can be used as an alternative to our simple example which simply indicates whether or not a word is present in the document or not.
- **N-gram:** Bigrams are tokens of words based on two-words per token, trigrams based on three words. These can be used instead of our unigram approach to see if it possibly increases accuracy.
- **Log-probabilities:** Naive (independence/naive assumption) involves multiplication of probabilities (leading to smaller numbers). So in our case floating point precision of computers can introduce a lower cap. In order to circumvent this, log probabilities can be used.

- **Additional Features:** Features like Job Title, Location, Contract Type, Category can be predictors for each job posting in addition to our previous approach. Since, our predictions of salaries are based on text data only, adding information such as Seniority, Department, Location, Full-time/Part-time information can significantly improve our model's performance.
- **Sentiment Analysis Results** can also be incorporated as a feature.
- **Tensorflow Implementation using Keras**

References:

[1] https://www.kaggle.com/c/job-salary-prediction-lkm
[2] https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15
[3] https://www.heavy.ai/technical-glossary/feature-engineering
[4] https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide
[5] https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285
[6] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html