Predicting Salary through Job Description

A Natural Language Processing Project

By: Justin Zhang

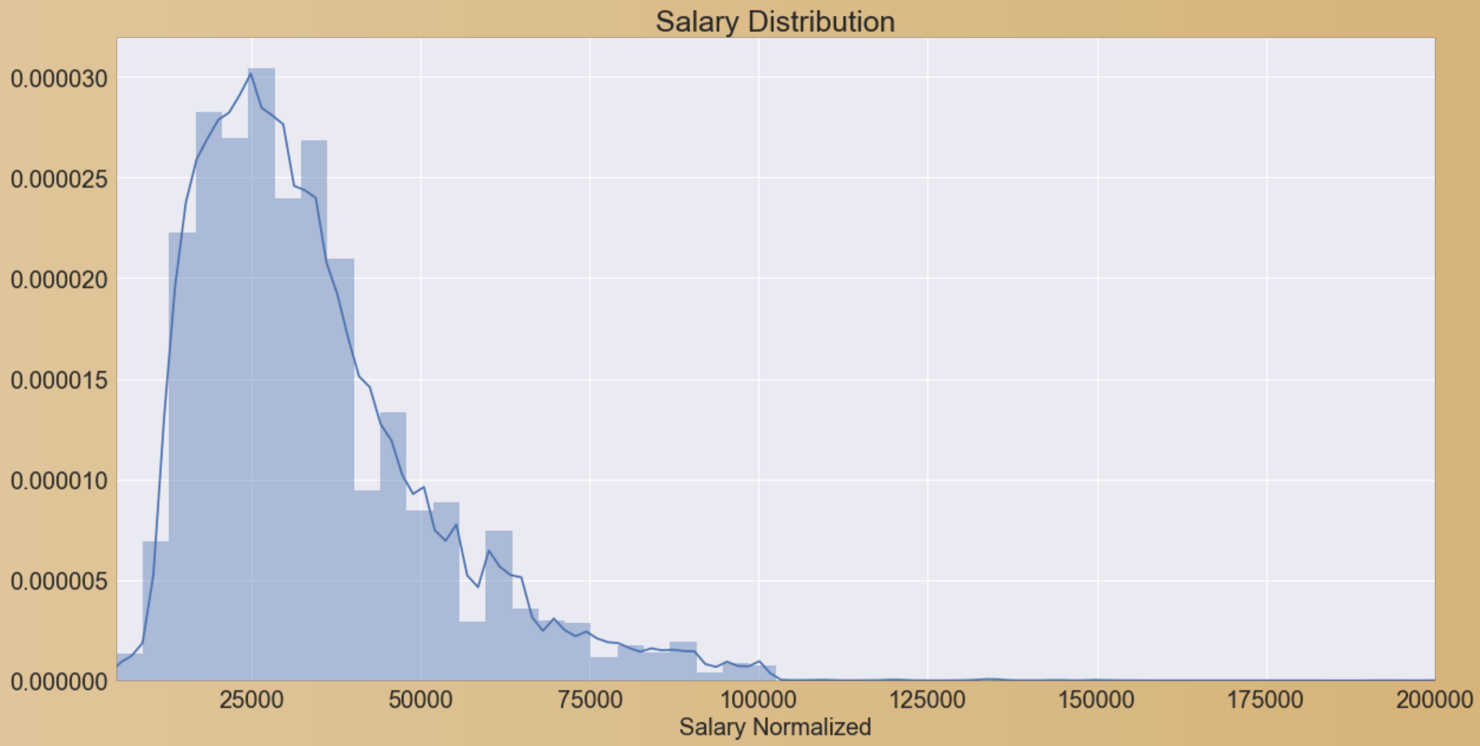
Data Introduction

- Goal: Predict salary using content of job description.

- Data is obtained from kaggle
- https://www.kaggle.com/c/job-salary-prediction/data
- Total sample: 244,768
- Notable columns ‘FullDescription’, ‘Title’, ‘LocationNormalized’, and ‘Category’

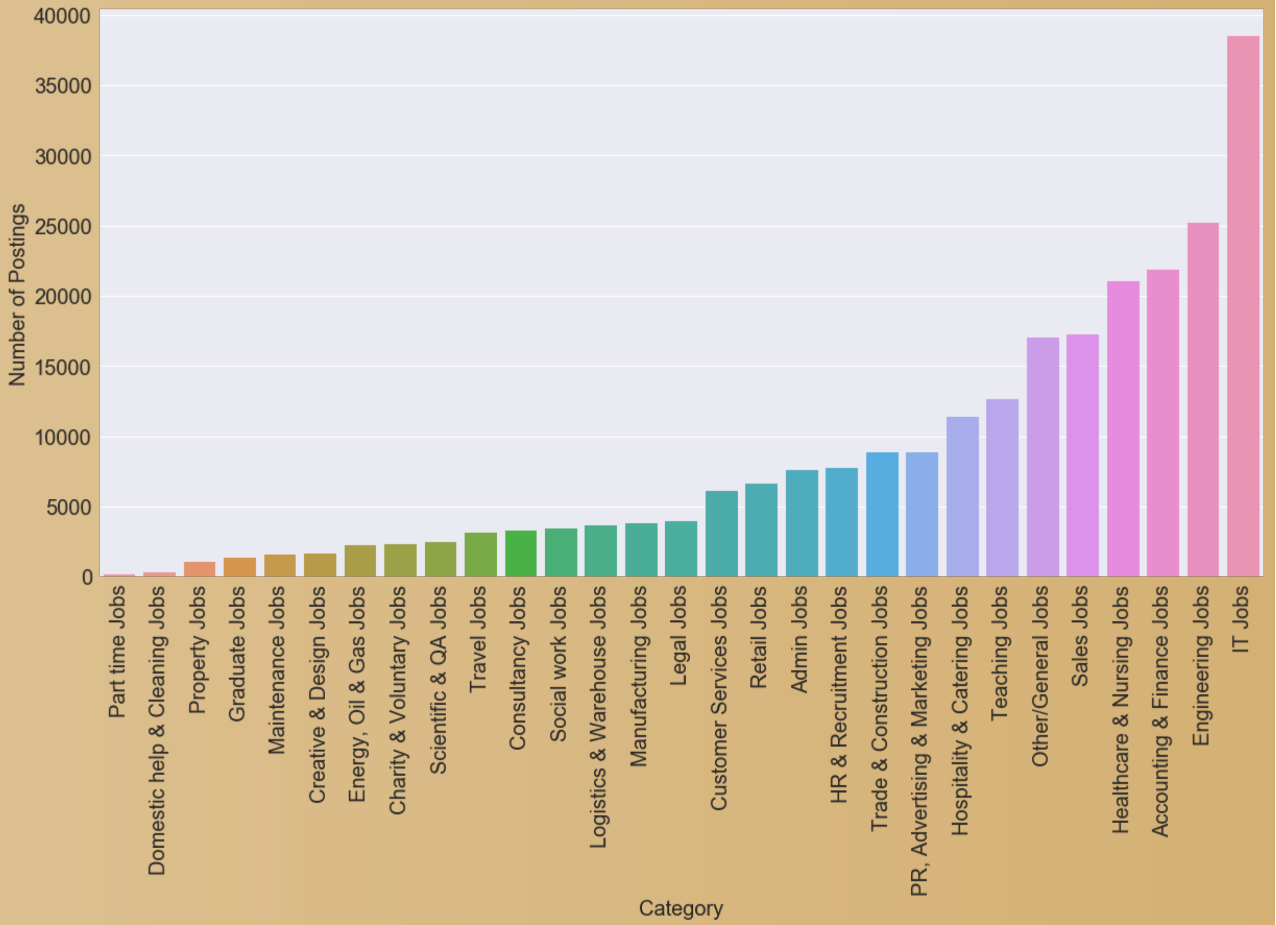
Exploratory Data Analysis

- Data is left skewed.
- Mean of 30,000 and median of 34,122.
- Majority of jobs are paying around 30,000

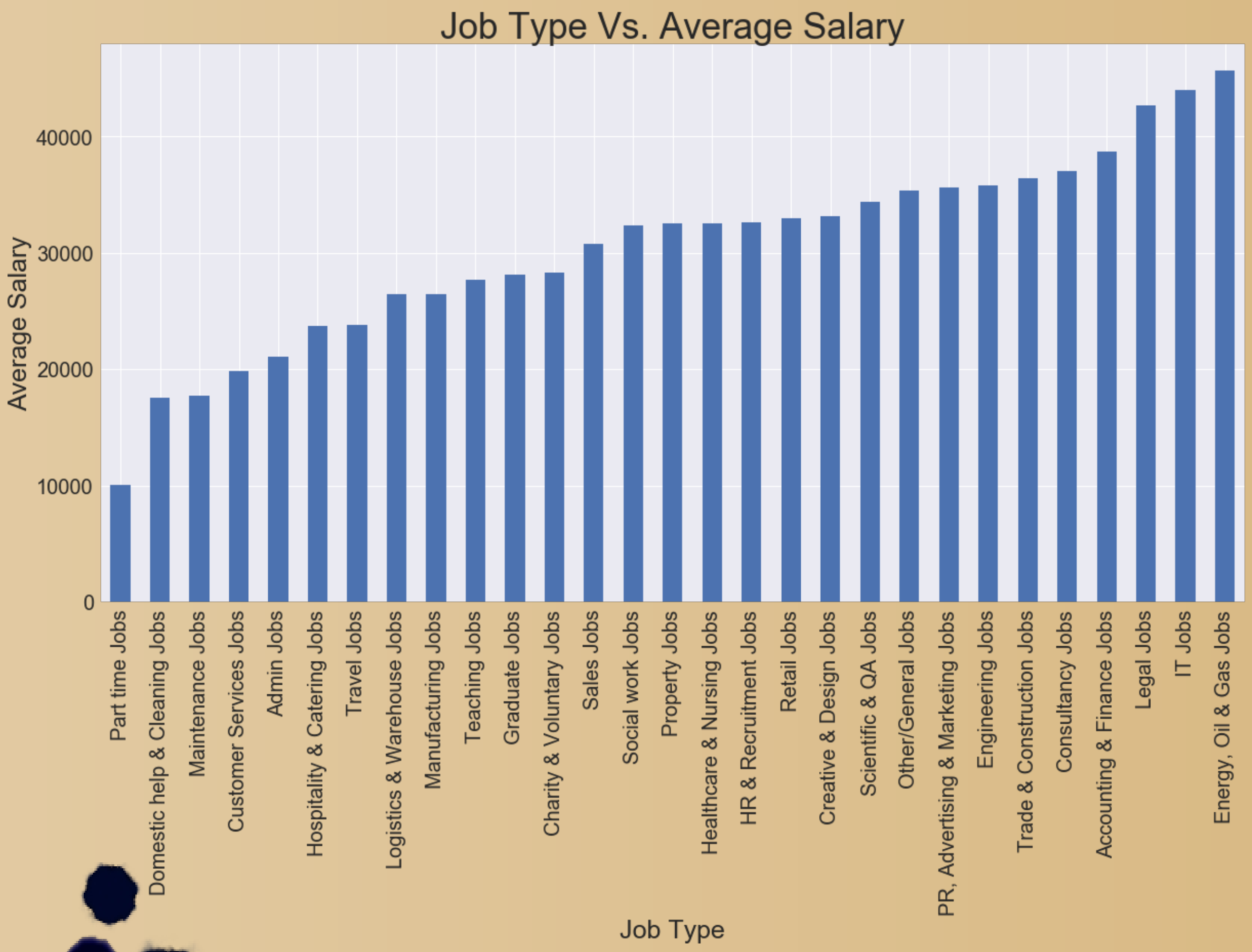


Exploratory Data Analysis

- Aggregation by ‘Category’ column.
- Large amount of IT and Engineering positions.
- Most jobs are full time.
- Does demand determine pay rate?



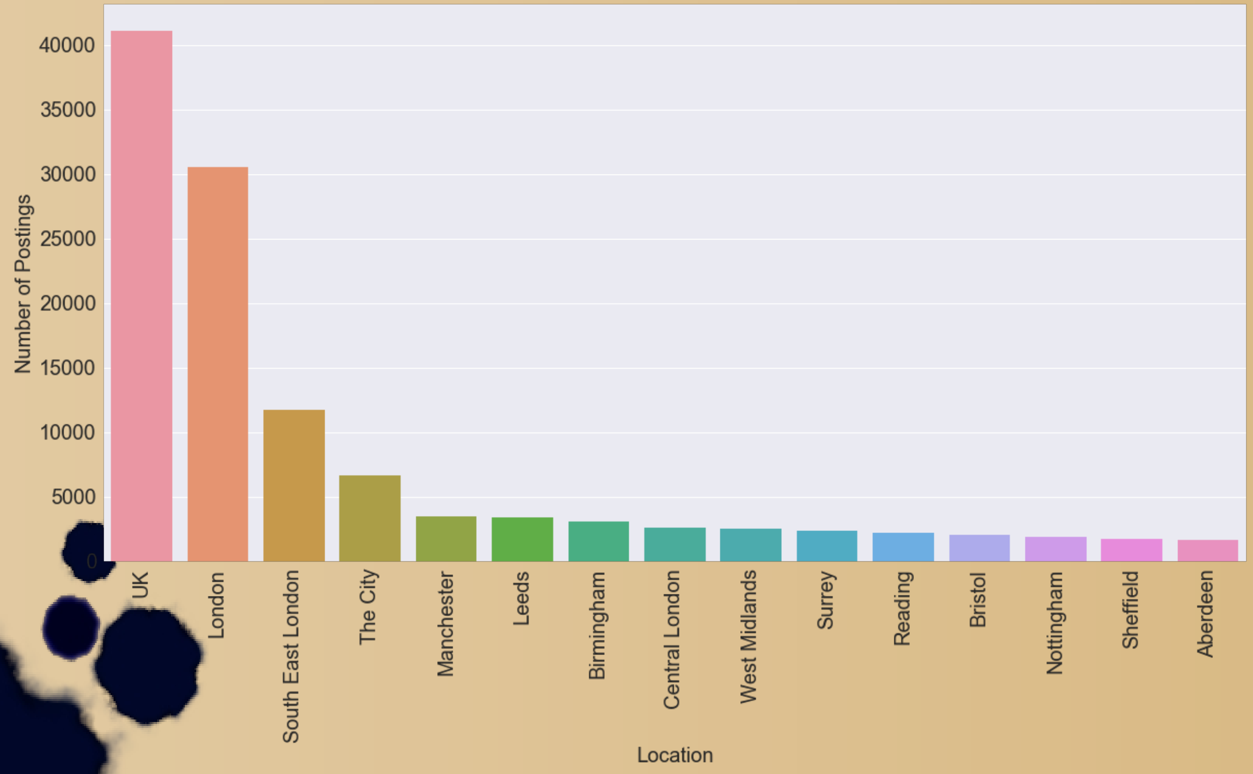
Exploratory Data Analysis

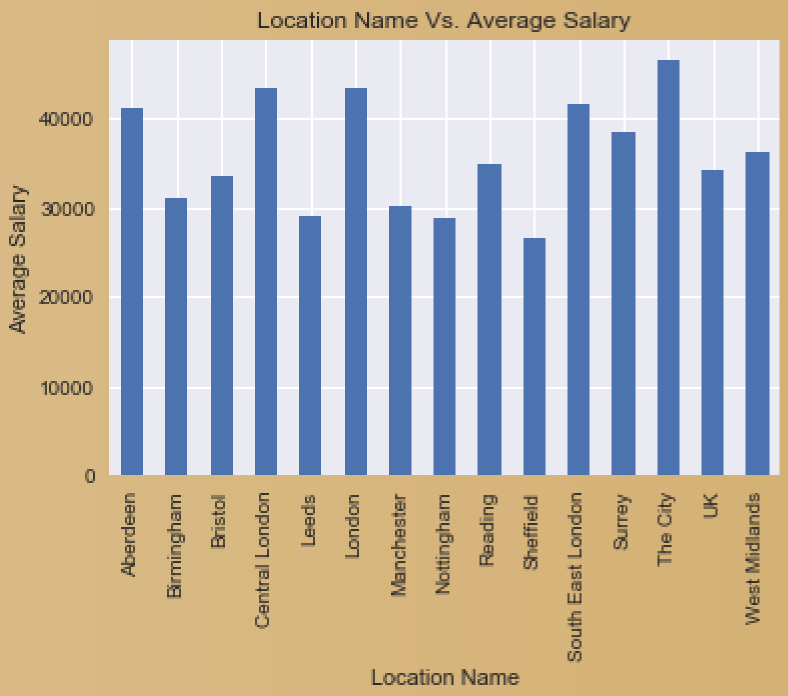


- Part time jobs are low paying, due to possible low skill cap requirement.
- Salary depends on both demand and skill qualification.
- Energy, Oil & Gas is the highest paying due to high education background.
- Part time jobs are not posted as frequently on job posting sites.

Exploratory Data Analysis

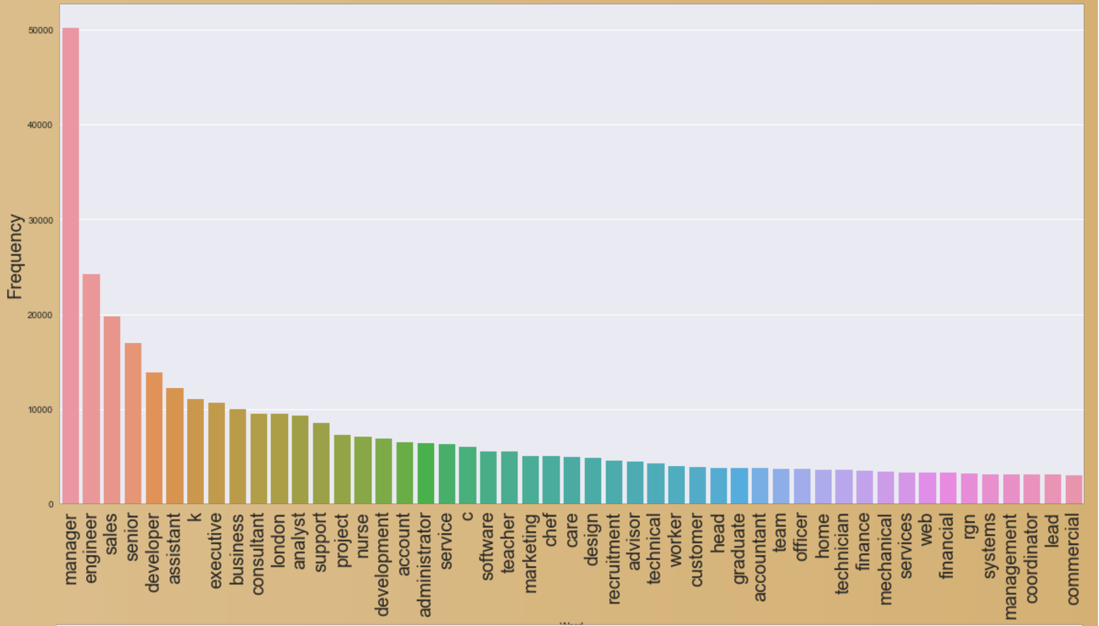
- Top 15 location names are too generalized
- UK/London/South East London are similar. Information is poor quality because data is not standard.
- Average salary fluctuates greatly between the most frequent and least frequent location names.
- Weak relationship. Weak predictive variable.

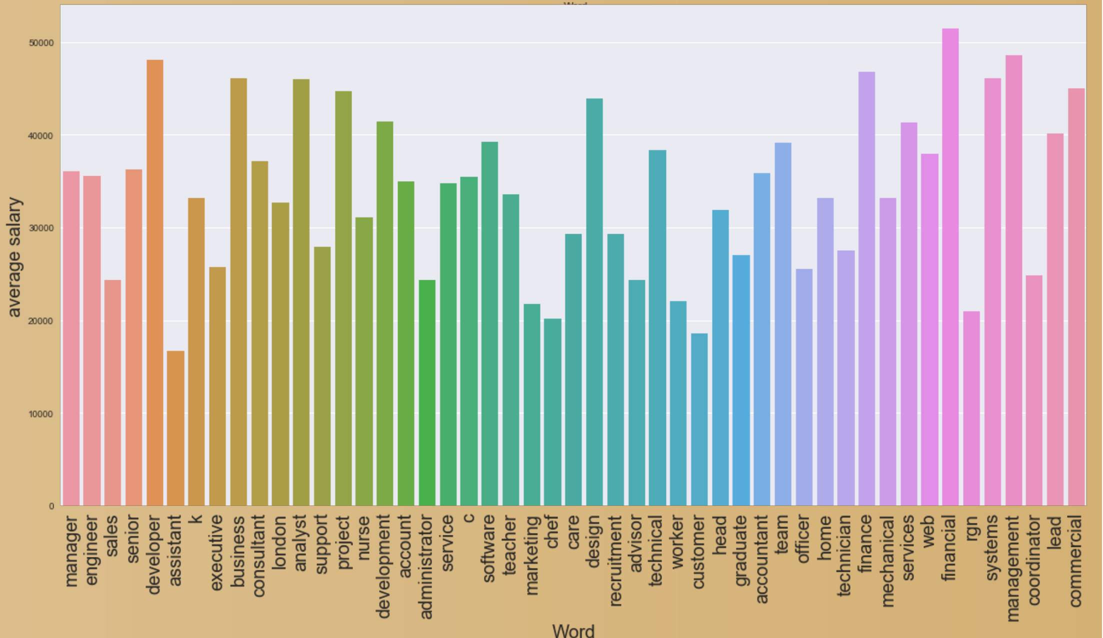




Exploratory Data Analysis

- Top 50 words in ‘Title’. Frequency is the word count.
- Word vs. Average salary. Relationship is not obvious.
- Weak relationship indicates bad predictive power.





Model Building

- Multinomial Naive Bayes
  - Most common for natural language processing
- Random Forest Classifier
  - More complicated, low interpret-ability, high predictive power.
- LinearSVC (Support Vector Machine)
  - High interpret-ability, with high training/prediction speed.

Model Building

- TfidfVectorizer used to construct bag of words.
- Words appearing in many documents are given less weight.
- Corpus is the dictionary made from training set.
- Each row is a document, each word is a token.
- Testing set can be transformed to return a term-document matrix.

Model Building

- Notable parameters for TfidfVectorizer:
  - stop_words
  - max_df
  - ngram_range
- stop_words removes filler words and max_df removes words that appear too frequent out of all the documents.
- stop_words and max_df had minimal effect on the accuracy score.
- ngram_range groups words together by specified number. It construct word phrases, therefore deriving additional meaning from data.
- ngram_range parameter significantly increased accuracy score with range (3,4).
-

Results

	MultinomialNB	Random Forest	LinearSVC
No parameter training	80.42%	99.42%	92.26%
No parameter testing	81.27%	83.55%	87.63%
No stopwords/ max_df=0.7 training	81.42%	99.42%	92.45%
No stopwords/ max_df=0.7 testing	80.55%	84.69%	87.61
With ngram training	92.07%	99.27%	99.56%
With ngram testing	86.42%	84.34%	91.34%
All parameter + GridSearchCV: C=3 training			99.83%
All parameter+ GridSearchCV: C=3 testing			91.45%

Conclusion

The project is turned into a binary classification problem for interpret-ability. Score greater than 50% indicates a working model.

Best model for this project is LinearSVC, it is fast, able to handle large sample of data, and have minimum parameters for tuning.

Resulting model could prove useful for sites such as indeed.com, giving job searchers more power in filtering through millions of applications.