**Job Salary Prediction**

**Abstract:**

Kaggle is an online platform for machine learning and analytics competitions where companies post data and individuals post solutions to compete for monetary prizes and to hone their skills [1]. Recently Adzuna, a UK based classifieds company, hosted a Kaggle competition with the goal of improving job salary pre- dictions for job postings in which the salary does not appear. Using various linear models, random forests and extra treeimplementations we were able to achieve an mean error of 6839.547 corresponding to the top 20% of the of the competitors. This paper describes the details of our feature selection and modeling methods.

**Introduction:**

Often when job advertisements are posted online, the employer neglects to mention the salary. To an individual looking for a job, this poses a dilemma; do they risk wasting valuable time investigating a low paying job, or skip the advertisement and risk ignoring a great opportunity.

Adzuna, a UK based classifieds company hosts many job advertisements; approximately half of which do not list a salary. In order to provide better services to its user base, Adzuna wants to give accurate salary estimates for job postings for which the true salary is not provided. To this end, Adzuna hosted a Kaggle competition with the goal of improving job salary predictions [1].

This paper explores the performance of three different regression techniques & four different classification techniques applied to the Adzuna data. Our best models are very competitive placing us in the top 20% of the competition when run on the withheld testing data.

**Data Structure:**

The training data for the competition consists of a .csv file of approximately 300,000 job postings with the information in each job posting arranged in 12 fields. The dataset was mostly unstructured text, with a few structured data fields. It consisted of the following data fields:

1. Id - A unique identifier for each job ad
2. Title- A freetext 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.
3. FullDescription - The full text of the job ad as provided by the job advertiser.
4. LocationRaw – The location as provided by the job advertiser.
5. LocationNormalized - Adzuna's normalised location from within the location tree, interpreted by kaggle based on the raw location.
6. ContractType - full\_time or part\_time, interpreted by Adzuna from description or a specific additional field received from the advertiser.
7. ContractTime - permanent or contract, interpreted by Adzuna from description or a specific additional field received from the advertiser.
8. Company - the name of the employer as supplied by the job advertiser.
9. Category - which of 30 standard job categories this ad fits into
10. SalaryRaw – the salary field received in the job advert from the advertiser.
11. SalaryNormalised - the annualised salary interpreted by Adzuna from the raw salary.  This is always a single value based on the midpoint of any range found in the raw salary.  ***This is the value we are trying to predict.***
12. SourceName - the name of the website or advertiser of the job advert.

The goal of the competition is then to predict the salary given the non-salary fields. The loss function specified by the competition is the mean absolute error on a withheld testing set.

**Data Cleaning:**

All of the data we received from kaggle was real, live data used in job ads and hence, was clearly subject to lots of real world noise, including but not limited to: ads that are not UK based, salaries that are incorrectly stated, fields that are incorrectly normalized and duplicate adverts. In order to clean this data we essentially used four techniques, namely: tokenization, stemming, excluding stop words & filtering junk values.

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens*, perhaps at the same time throwing away certain characters, such as punctuation. Sometimes, some extremely common words, which would appear to be of little value in helping select documents matching a user need, are excluded from the vocabulary entirely. These words are called *stop words*. The general strategy for determining a stop list is to sort the terms by *collection frequency* (the total number of times each term appears in the document collection), and then to take the most frequent terms, often hand-filtered for their semantic content relative to the domain of the documents being indexed, as a *stop list*, the members of which are then discarded during indexing.

A stemmer is a program that reduces a word to its word stem or base form. For instance, in English a stemmer would remove suffix endings such as -ed, -ly, -s, -ing, and others. Hence, ‘walking’, ‘walked’, ‘walks’ would all be reduced to ‘walk’. This was very useful as our analysis depended on word frequency. We implemented a stemmer from nltk.stem and saved a list of the original word and stemmed form to a file. This allowed us to spot stemming issues.

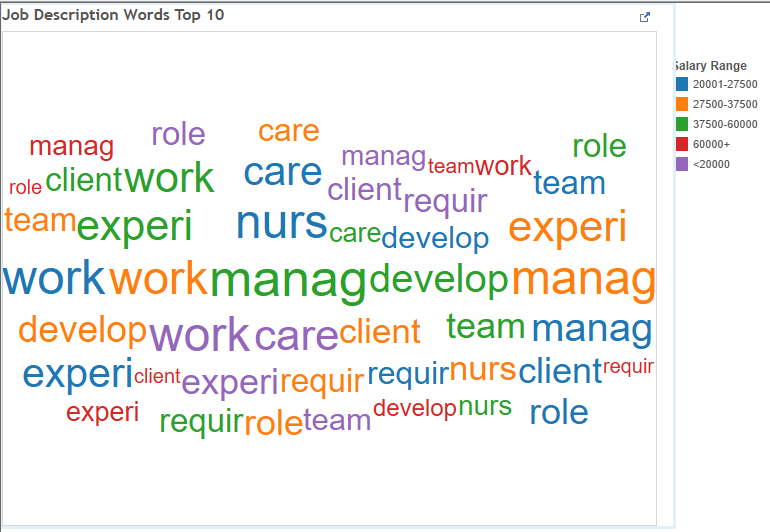


Fig. 1: Word Frequency – Job Description



Fig. 2: Word Frequency – Job Description

**Text Parsing:**

We used Python-NLTK to determine Tf-idf (Term frequency inverse document frequency) for feature analysis of job Title.

In a large text corpus, some words will be very present (e.g. “the”, “a”, “is” in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms.

In order to re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the tf–idf transform.

Tf means term-frequency while tf–idf means term-frequency times inverse document-frequency. This was originally a term weighting scheme developed for information retrieval (as a ranking function for search engines results), that has also found good use in document classification and clustering.

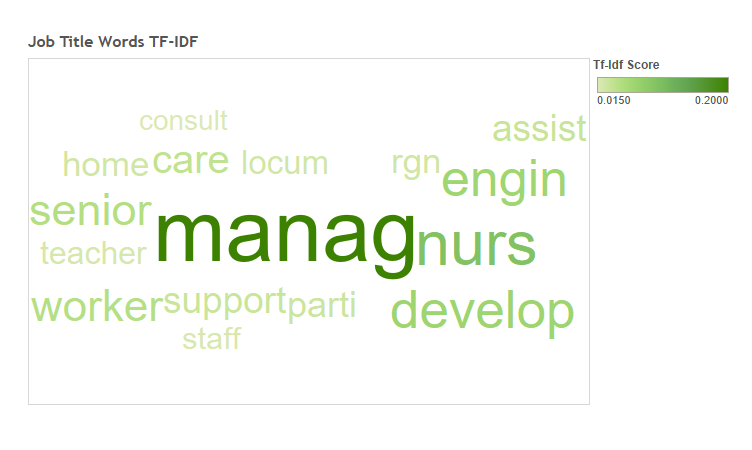


Fig. 3: Text parsing – TF-IDF

**Bag of Words:**

Text Analysis is a major application field for machine learning algorithms. However the raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

In order to address this, scikit-learn provides utilities for the most common ways to extract numerical features from text content, namely:

* tokenizing strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
* counting the occurrences of tokens in each document.
* normalizing and weighting with diminishing importance tokens that occur in the majority of samples / documents.

In this scheme, features and samples are defined as follows:

* each individual token occurrence frequency (normalized or not) is treated as a feature.
* the vector of all the token frequencies for a given document is considered a multivariate sample.

A corpus of documents can thus be represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.

We call vectorization the general process of turning a collection of text documents into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or “Bag of n-grams” representation. Documents are described by word occurrences while completely ignoring the relative position information of the words in the document.

**Exploratory Data Analysis:**

In this section we analyzed different parameters before feature generation. Example: Number of jobs for different categories and their average salaries. We also conducted analysis on job title and description based on seniority. Example: Inter, Senior Analyst, Junior Analyst etc. Also, we checked the distribution over contract parameters.

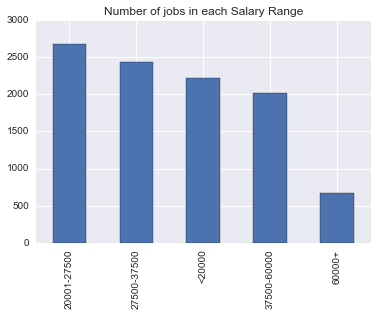


Fig. 4: Number of jobs in each salary range

In the above figure we see that the highest number of jobs fall in the salary range of 20001-27500, and this number steadily decreases towards the salary range of 60000+.

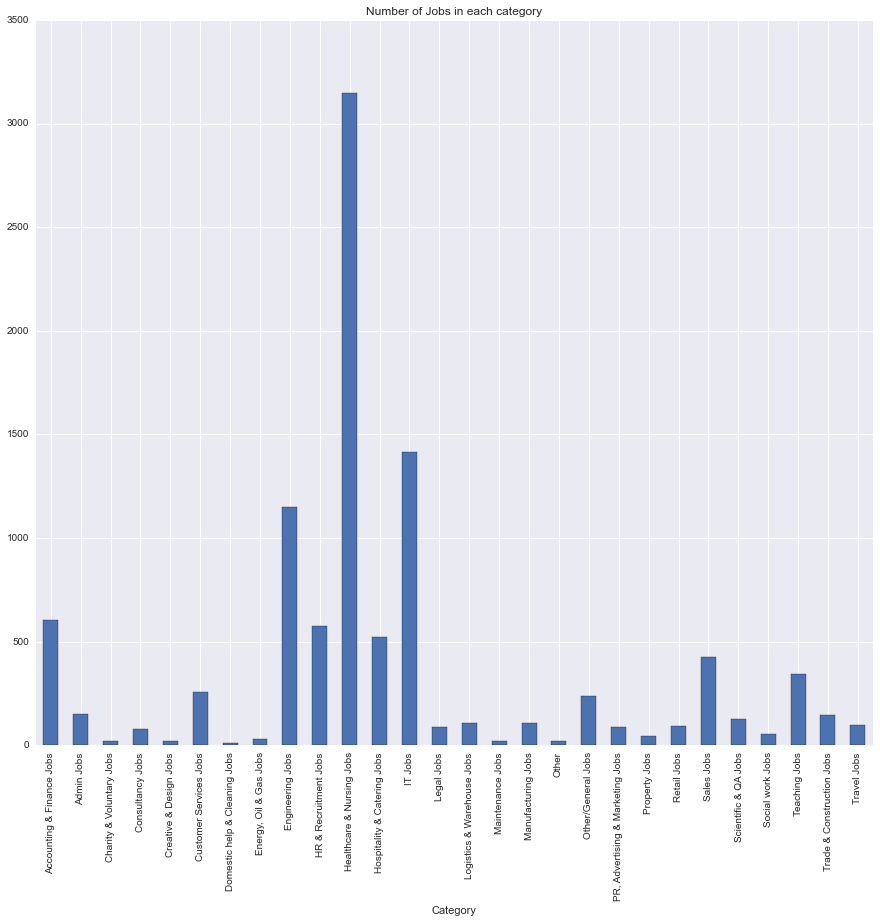


Fig. 5: Number of jobs in each category

In the above figure we have checked the distribution of jobs depending on the functional areas. As the job category or the functional area is a key factor in determining the job salary this bar graph is extremely useful.

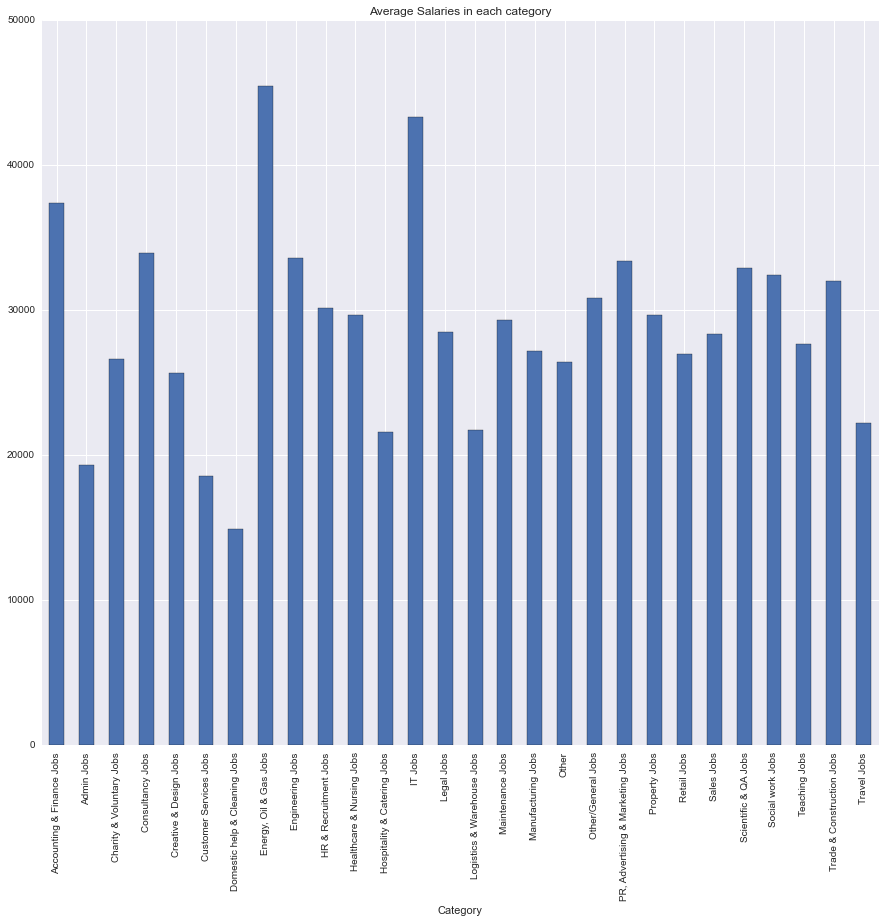


Fig. 6: Average salaries in each job category

Further, we also saw that the salary of a job also depended on the job level that a person was being hired for. Example: the salary of an intern may not be the same as that of a permanent employee at the same company for the same job function. Similarly, a junior level employee would be paid less than a senior level employee.

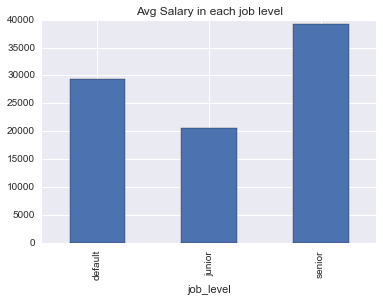
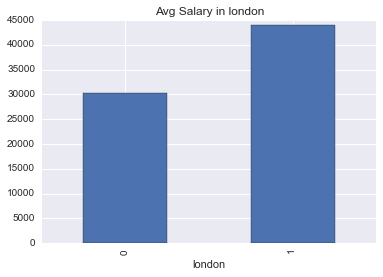
 

Fig. 7: Average Salary in Each Job level Fig. 8: Average Salary in London

Similarly, we also saw that the location of a job had an impact on the Job salary. For example: the salary of employees working in London was higher as compared to employees working outside of London. This may be due to the high living expenses in London as compared to other places. However, we can confirm that location has a role to play in deciding a person’s salary.

**Feature Extraction & Engineering:**

**Feature extraction** is very different from *Feature selection*: the former consists in transforming arbitrary data, such as text or images, into numerical features usable for machine learning. The latter is a machine learning technique applied on these features. We have use bag of words technique for feature extraction in this project.

**Feature engineering** is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. In this project we have used 4 features for analysis, they are as follows:

* Feature -1: Job Title + Description + Location Raw + Location Normalized
* Feature – 2: Feature 1 + Category
* Feature -3: Feature 2+ Contract Type + Contract Time
* Feature – 4: Feature 3+ Job Level+ Location

**Regression or Classification:**

The competition on Kaggle expected MAE as the evaluating criteria; hence it was a regression problem. However, we worked on this problem using both approaches, i.e. regression as well as classification. We divided the normalized salary into bins for classification problem. We used an 80-20(training-validation) split for both of the aforementioned approaches.

**Regression:**

We have used the following regression models for our project:

* Random Forest Regressor
* Decision Tree
* Extra Trees Regressor

To evaluate a regressor model we used MAE(Mean Absolute Error) and MSE(Mean Squared Error) as the evaluating criteria

**Random Forest Regressor:** Random forests is a notion of the general technique of random decision foreststhat are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set.

The algorithm for inducing Breiman's random forest was developed by Leo Breiman and Adele Cutler, and "Random Forests" is their trademark. The method combines Breiman's "bagging" idea and the random selection of features, introduced independently by Ho and Amit and Geman in order to construct a collection of decision trees with controlled variance.

**Decision Tree:** Decision tree builds regression or classification models in the form of a tree structure. It brakes down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree, which corresponds to the best predictor, called root node. Decision trees can handle both categorical and numerical data.

**Extra Tree Regressor:** An extremely randomized tree regressor.

Extra-trees differ from classic decision trees in the way they are built. When looking for the best split to separate the samples of a node into two groups, random splits are drawn for each of the max\_features randomly selected features and the best split among those is chosen. When max\_features is set 1, this amounts to building a totally random decision tree.

**RESULTS:**

|  |  |  |
| --- | --- | --- |
| **MODEL** | **FEATURE** | **MAE** |
| Random Forest Regressor | F-1 | 7676.68 |
| Random Forest Regressor | F-2 | 7217.21 |
| Random Forest Regressor | F-3 | 7012.14 |
| Decision Tree | F-3 | 9258.98 |
| Extra Trees Regressor | F-3 | **6839.547** |

Fig. 9: Results of Regression models

**Classification:**

We used the following models for Classification:

* Random Forest Classifier
* Naïve Bayes
* Support Vector Machine(SVC)
* Extra Trees Classifier

To evaluate a classifier model we used Accuracy as the evaluating criteria and also used Confusion Matrix & Classification Report for further analysis

**Naïve Bayes:** In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate preprocessing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

**Support Vector Machines(SVM):** In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

**RESULTS:**

|  |  |  |
| --- | --- | --- |
| **MODEL** | **FEATURE** | **ACCURACY** |
| Random Forest Classifier | F-1 | .5709 |
| Random Forest Classifier | F-3 | .5820 |
| Random Forest Classifier | F-4 | .5910 |
| Support Vector Machine | F-4 | .5333 |
| Naïve Bayes | F-4 | .4581 |
| Extra Trees Classifier | F-4 / F-3 | **.6126** |

Fig. 10: Results of Classification Models

**Conclusion:**

Looking at the results its is safe for us to conclude that regression models performed much better that the classifiers in this case. Extra tress regressor when run on Feature -3 gave us the lowest MAE(Mean absolute error) and hence the maximum accuracy in prediction. Also, considering the fact that our models ranked among the top 20% of the ones submitted on kaggle(i.e 250+ entries), it is safe to accept our conclusions.

**References:**

1. https://www.kaggle.com/c/job-salary-prediction

2. http://scikit-learn.org/stable/modules/feature\_extraction.html

3. http://www.saedsayad.com/regression.htm

4. <https://en.wikipedia.org/wiki/Support_vector_machine>

5. http://ceur-ws.org/Vol-1202/paper11.pdf