Job Salary Prediction

A theoretical study to predict job salaries based on job descriptions

Sadman Shawmik, Lamia Tasnim, Shiny Raisa
Department of Computer Science and Engineering
BRAC University
Dhaka
sadmanh23@gmail.com, lamanju2011@gmail.com, shinyraisa@gmail.com

***Abstract*—In this paper, we describe our model of a prediction engine that predicts a normalized salary for jobs based on different criteria of the job description. This paper is divided into parts describing our dataset, equipment, algorithm, and prediction model. We also compared results between variations of our algorithm parameters, as well as with similar projects that incorporate other algorithms.**

Keywords—prediction; salary; machine learning; regression; random forest

I. Introduction

Every job seeker longs to secure the best job with the best salary in their own location and fields. However, considering the excessive competition, the inconsistent job circulars and the lack of transparency in the recruitment procedures, this goal is not always satisfied. As such, in these circumstances, an efficient and smart salary prediction machine has become indispensable.

Considering the competitiveness of the job market, we intended to build a prediction engine that can predict salaries based on different criteria like company, location, position, etc. Explicitly speaking, the predicting model is assumed to be able to greatly improve job seekers' search experience, assist employers and job seekers in estimating the optimal market worth of different positions, and make the job search more effective & transparent.

Although a few similar projects have been developed prior to ours, seldom they have reached to efficient and effective conclusions. Moreover, what sets our project apart is that we have tried to use a minimalistic technique and have attempted to create a complex but fascinating outlook with the simplest and most basic methods taught in our class.

II. Methodology

A. Data Collection

The dataset for this project was acquired from Kaggle [1], who, in collaboration with Adzuna, launched a competition under the Kaggle Startup Program. The data used in this project was scraped from different job boards based in the UK by Adzuna, a search engine for job advertisements. The dataset Adzuna provided has thousands of entries, which is mostly unstructured text with a few structured entries, which can be in a hundred of different formats since the data was obtained from over a hundred of different sources of records. Our evaluation dataset is simply a random subset of ads for which we know the salary, or for which the salary column is already filled. However, in the training dataset, the salary column is kept empty. The fields in the dataset are *Id, Title, FullDescription, LocationRaw, LocationNormalized, ContractType, ContractTime, Company, Category, SalaryRaw, SalaryNormalized, and SourceName*.

In the dataset, we use 80% of the dataset for training and 20% for testing purpose. It is quite rare to use 50/50, 80/20 is quite a commonly occurring ratio. A better practice is to use: 60% for training, 20% for cross validation, 20% for testing.

B. Equipments

In this project, we used Python 3.6.3 as our development language, and Anaconda Navigator as our Integrated Development Environment. The built-in Python libraries and packages that we used are pickle—a module for serializing and deserializing Python object structure by converting the object into a byte stream or converting a byte stream back into the object structure; and features—a package for creating feature sets from the dataset. Other built-in libraries used in the project include csv, json, html.parser, os that have been used for different operations and functions.

For the actual data analysis, we used some popular libraries that are widely used for a wide array of Machine Learning projects. These libraries include numpy, pandas, and sklearn.

NumPy is a fundamental package for scientific computing using Python. It contains powerful features like N-dimensional array object, broadcasting functions, linear algebra, Fourier transform, random number, etc. [2].

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with

"relational" or "labeled" data both easy and intuitive. It can be used for tabular data, ordered and unordered time-series data, arbitrary matrix data, and any other forms of observational and statistical datasets.

Scikit-learn is a Machine Learning library built on top of SciPy and contains powerful tools for machine learning and statistical modeling such as classification, regression, and clustering algorithms including Support Vector Machines, Gradient Boosting, K-means, and of course, Random Forests—which we will be using in this project.

C. *Algorithm*

The base algorithm used in the Job Salary Prediction model is Random Forest, which is a supervised algorithm. A supervised algorithm is where both the inputs and outputs are known. In this algorithm, the processes of finding the root node and splitting the feature nodes run randomly. If there are enough trees in the forest, the Random Forest algorithm does not overfit the model, which is one of the reasons why we chose this algorithm over others. This algorithm can also handle missing values, and the classifier can also be modeled for categorical values [3].

The random forest algorithm works by first selecting K features from a total of m features where K < m. Among the k features, the node d is calculated using the best split points [4]. The nodes are then split into daughter nodes using the best split. These steps are repeated until X number of nodes are reached, and lastly a forest is built by repeating all of the above steps for N number of times to create N number of trees.

In this project, we have generated 50 and 100 trees in two iterations, and compared our benchmark data using a Python script that plots the data from the CSV files in a line graph.

D. *Data Manipulation*

In order to train the prediction model, the dataset is first read in using *read_csv*, a function in the pandas library, which is a prerequisite library for the prediction model. Once the data is fed in, the transformed data is fit using the final estimator. The training target here is the *SalaryNormalized* column in the training data. The classifier is then saved to a file called *random_forest_rev1.pickle*.

E. *Prediction*

To predict the salaries from the test dataset, the predict.py file is run, which loads the classifier, makes the predictions for each job ID in the test dataset, and writes the *SalaryNormalized* output to the file.

F. *Limitations and Range of Validity*

In the dataset, the normalized location from within our own location tree, interpreted by us based on the raw location, is not perfect. The category column is inferred in a very messy way based on the source the ad came from, so there is a lot of noise and error in this field. Moreover, since these ads are collected from job boards based in the UK, only people living there and looking for jobs online can benefit from our model.

III. Results

A. *Prediction Results*

Running the predict.py file generates a CSV (Comma Separated Values) file that contains two columns—Id and *SalaryNormalized*.

B. *Result Comparison*

For generating the prediction data, we used two iterations of 50 and 100 number of trees in each. As a matter of fact, the higher the number of trees, the more accurate the result is. Also, it should be noted that it takes roughly double the time to generate 100 trees than it does to generate 50, so to optimize CPU and memory usage, we decided to run only two iterations. However, for more accurate results, a higher number of trees can be generated, but the difference will still be very negligible.

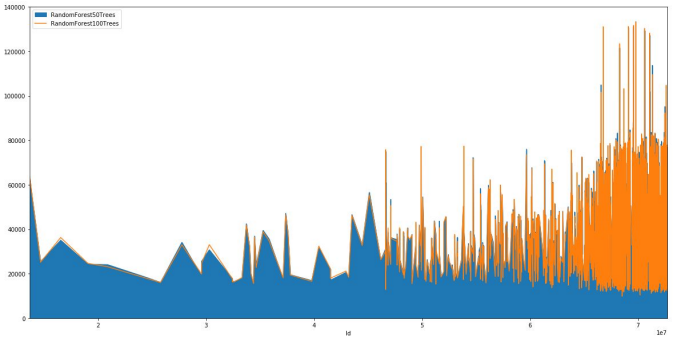

Fig. 1. Result comparison between 50 trees and 100 trees

The graph above shows the result comparison between the prediction files output for 50 and 100 trees. The blue line shows the result for 50 trees and the orange shows the result for 100 trees. It can be concluded that the difference in results for using 50 and 100 numbers of trees is extremely minimal, so using 50 trees is optimal, considering CPU and memory resource usage.

IV. Discussions

To test the validity of our results, we compared the generated predictions with another set of predicted data that we obtained from Github [7]. The model we compared our data against uses a different algorithm for generating their prediction model, which is Linear Regression. We chose this model as a metric for comparison because this was placed among the top 10 in the competition that Kaggle and Adzuna launched.

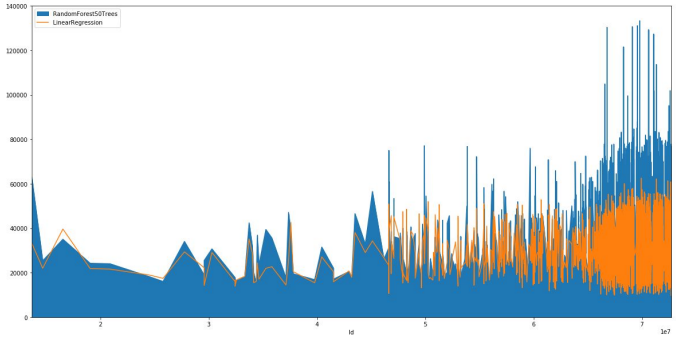

Fig. 2. Result comparison between Random Forest and Linear Regression

V. Conclusions

We have described our iteration of Job Salary Prediction, using the dataset from a Kaggle-hosted competition.

This prediction engine, based on the Random Forest algorithm, can successfully generate a normalized predicted salary based on different criteria from job advertisements.

We also demonstrated that using twice the number of trees to generate the forest has negligible effect over the differences in the prediction results, so 50 trees are recommended for optimal resource usage.

Further plans regarding this model includes integrating it with Django, a Python-based web framework, and building a web front-end to convert this model to a robust web app that can be used to search for salaries based on certain criteria.

Acknowledgment

We would like to thank Dr. Iftekharul Mobin, Assistant Professor and Dr. Mohammad Zavid Parvez, Assistant Professor at the Department of Computer Science and Engineering, BRAC University, for their continuous support, guidance, and thoughtful feedback throughout the development lifecycle of this project.

References

[1] Kaggle and Adzuna. Job Salary Prediction Competition, 2013. Dataset available on kaggle.com/c/job-salary-prediction/data.

[2] W. McKinney, “Python for data analysis: Data wrangling with Pandas, NumPy, and IPython." O'Reilly Media, Inc, 2012.

[3] T. G. Dietterich, "Ensemble Methods in Machine Learning" Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science, vol 1857.

[4] A. Liaw and M. Wiener, “Classification and Regression by randomForest” R news, 2(3), pp.18-22, December 2002.

[5] F. Pedregosa, "Scikit-learn: Machine learning in Python." Journal of machine learning research pp. 2825-2830, October 2011.

[6] Ryszard S Michalski, Jaime G Carbonell and Tom M Mitchell, "Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. Machine learning: An artificial intelligence approach." Springer Science & Business Media, 2013.

[7] Alec Radford. Job Salary, 2013. Code available from github.com/Newmu/Salary-Prediction.