Job Hunting Strategy - Career Development Recommendation

Cheng-chun Chao, Sean Chuang, Xinyu Wang, Yijun Liang, Yuanyuan Xiao

Project Overview:

Job hunting can be quite frustrating, and it is not just about an eye-catching resume. You need to know how your knowledge level locates you in the job experience level, what kind of extra skills you need to master, also you would like to know the appropriate salary range for the job you are looking for so you can negotiate with HR for a better offer. In order to help out, our team built a corresponding recommendation engine, where users provide their basic information and we can run our recommendation engine. Our website will return suggested jobs and other related information such as salary, needed skills, and useful courses/books. Our application specifically targets people who lack job searching experience, for example, fresh graduates. Our recommendation engine can help them understand based on their current skills set, what kind of jobs fits users, and what's the reasonable range of salary for them to ask during offer negotiating. Additionally, we can also help HR who would like to recruit talents and see what necessary skills they need to screen for the candidate's resumes.

In order to achieve our goal, we split our project into three phases. First, we need to collect data to build our job recommendation and average salary prediction model. Second, in order to present the necessary skills for suggested jobs, we decided to web scrap such information from online resources. Finally, we need to build a user-friendly application to visualize our results.

We've decided to build our minimum viable product by focusing on recommending jobs within machine learning and data science due to the limited observations from the available data. And hope if time allows, we will be able to add additional functions to our web.

Problems We have Solved and How it Works:

(a) Data cleaning

For the data cleaning part, we aimed to combine data sets acquired from Kaggle and web scraping, and then selected important features for our machine learning model. Unstructured and excessive features were the main problems we focused on. It is because our data was derived from plain text surveys and it includes numerous multiple choice questions and we have around 100 features in total. As a result, we took steps including data aggregation and feature selection to reduce the number of features. We also removed features that contained too much NA values. From the missing data, we imputed them with guess matrices, either the mode or average value. Lastly, we encoded and converted plain-text responses into either numbers or categorical variables for further analysis.

(b) Model Building

Since we identified our case as a classification model, we chose 6 machine learning algorithms to predict job titles and salary range: logistic regression, decision tree, random forest, K-nearest neighbors, support vector machine and perceptron. Based on the comparison of the

test set models under random states, the support vector machine algorithm overall has the highest accuracy at around 54% which is about 20% more accurate than the baseline model. Since the algorithm outputs the class with highest probability, we modified the model to output top 3 classes ranked by prediction probability values which gives users more options while selecting jobs. Since the predicted salary ranges need to be highly correlated with the predicted job titles along with skillset factors, we take the predicted job titles as a parameter of new input along with other user inputs for salary range prediction.

(c) Web UI design

In order to record user input and pass on to the model we trained, we found that the current out-of-the-box web page design tool (e.g. WIX) did not have sufficient flexibility for us to provide a better user experience. Instead, our team utilized Python Flask web framework to design our webpage. There were three milestones we achieved for this part. First, we utilized HTML methods to pass user input to python variables. These variables were then processed as inputs for our model and generated outputs to display in the result page. Secondly, we adopted Joblib Python packages to read the pickle files exported from the trained models in Scikit-Lean. As a result, our website incorporated the models we built, directly making predictions based on users' input information. Last but not least is that we designed how we displayed our information. We adopted web design best practices to allow users to understand what information we need from them and also a concise view of results. Since the website asks users to put their information, we included error handling features making sure users fill in all required fields.

Conclusion:

Finally, we've successfully built our application. As attached picture shown below: if you enter your name, age, year of experience, expected salary, mastered skills, and the skills you want to approve, the information will be passed into backend model, and through the result page, we will let you know what is your experience level, what jobs we will recommend, the corresponding salary, and what are the skills commonly required for those jobs.

Additionally, if you let us know what specific programming language you would like to improve, we will also share some related book information with you.

Limitation and Future Improvement:

One limitation we are having now is due to the specific character of our database, the salary ranges don't differentiate that much for different jobs as most of our observations are people quite experienced, and they tend to have higher salaries than the average. We will take time to see if we will be able to integrate some additional databases to gain a better estimation for users.

All materials were uploaded to out GitHub repository as the link below
https://github.com/joliang17/DataX_Project