Stackoverflow Professional Developers Studies

Hetu Feng, Xiao Yang

PROBLEMS AND BACKGROUND

Job Seeker Classifier:

There are multiple reasons that lead a person to decide to look for new opportunities and find a new job, for example, salaries, family, career choice and etc. Therefore, in this project, we want to find out what type of people with certain attributes are looking for a new job, and based on that, we are creating a classification model that with certain information, can predict if a person is going to do job seeking.

Stackoverflow has a Job section that works like LinkedIn's job section, but not many people have heard of or used it. One of the reasons could be that the marketing wasn't done so well. Therefore, with the fact that Stackoverflow has a good number of professional developers user base, and our model is built upon these users, analysis and classification model created in this project could help Stackoverflow identify target customers of their Job section, so that they can conduct a more effective marketing strategies..

Salary Estimator:

Another topic that is usually closely related to jobs is salary, and thus on top of the job seeking classifier, we want to create a regression model that can estimate a user's salary based on certain factors such as education level, development skills and professional coding experience.

Popular job posting websites including Glassdoor, Indeed and LinkedIn all have a salary estimation services that can give job seekers a general idea of salaries they can earn. Therefore, our salary estimator could become a new feature that Stackoverflow can roll out to improve the service. This salary estimator will specifically for the United States, because most of the respondents in the dataset are from US.

DATASET

This dataset is Stackoverflow 2019 Developer Survey which can be downloaded from Stackoverflow's website. This survey has 88883 instances with 85 columns. Out of 85 columns, many of them are not related to the topic of the project, so we will drop them before the analysis. Because this dataset is in survey form, it has many missing values and most of the variables are categorical with only 6 columns contain numerical variables. This requires us to perform extensive data cleaning to make sure the variables are usable. Columns like '*Past Development Experience*' allow multiple different answers in one cell, so we must also devise a weighting method to transform these data.

METHODS

Data Preparation:

As mentioned in the last section, the dataset is not clean enough for data modeling because many columns are not relevant to this analysis, a lot of missing values exist and most of the values are categorical. To make the data ready for machine learning, we firstly investigate each column and drop irrelevant ones, such as 'code review hrs', 'SO visit frequency', 'Age 1st to code', 'Web Frame desire next year' and etc. 70% of the 85 columns are dropped and the remaining data columns are as below.

```
Index(['MainBranch', 'OpenSource', 'Employment', 'Country', 'Student',
       'EdLevel', 'UndergradMajor', 'OrgSize', 'DevType', 'YearsCode',
       'YearsCodePro', 'CareerSat', 'JobSat', 'MgrIdiot', 'JobSeek',
       'ConvertedComp', 'Age', 'Gender', 'Employment_dummy', 'JobTypeScore',
       'OrgAvgSalary', 'AgeRange'],
```

These data need to be cleaned and transformed to numbers for scikit-learn modules, and we will illustrate some representative cleaning we performed here in the report. For columns like EdLevel (Education Level), the values are complicated and long, looking like '*Bachelor's degree (BA, BS, B.Eng., etc.)*', '*Master's degree (MA, MS, M.Eng., MBA, etc.)'*. To make it ready for modeling, we transform them to '*BS', 'MS', 'PhD'* and *'Others'*, and assign numerical values to each one, based on the level of education from low to high, as the following code shows.

```
stackoverflow['EdLevel'].replace({'Others':0, 'BS':1, 'MS':2, 'PhD':3}, inplace = True)
```

'UndergradMajor' has more than 20 different data values and we can't assign a numerical value to each one because using OneHotEncoder will generate too many columns, and we can't simply assign each major a number as the majors are not orderay. The solution is that we change the value of each major to either '*CS'* or *'Non_CS'* since for developers' community, the difference makes when one's undergrad major is CS related or not. To fill the missing values for 'UndergradMajor', we assign '*No Major'* to instances whose 'EdLevel' equals 0.

'Gender' column has several different values, so we transform them into 3 categories as '*Male', 'Female'* and *'LGBTQ'* and create 3 new columns which are 'Male', 'Female' and 'LGBTQ'. They contain only 1 and 0 for each instance, so for instance for a female, it will have 1 as the value at the 'Female' column and 0 at 'Male' and 'LGBTQ' columns. This will avoid the situation when the model generates the result as 'Gender < 0.5', which is uninterpretable.

The 'DevType' column allows users to put all past experience, so each cell contains a long string with multiple values, for example one cell could look like *'Developer, full-stack, Data or Business Analyst, Data scientist or machine learning specialist, Database administrator'*. To normalize the data, we first assign a 'score' for every job mentioned in this column based on its average salary in the United State. We consider the highest average salary as 100 and scale down other salaries based on this ratio. For each user, we select the DevType with the highest salary as the base score, multiply 0.01 to each remaining Devtypes and add them to the base score to have the final score. We then create a new column called 'JobTypeScore' to store all the normalized numerical value for 'DevType'.

Columns like 'JobSat', 'CareerSat', 'MgrIdiot' take users' satisfaction level for their current job, career and manager, so we transform the categorical variables to numerical according to level of satisfaction, like the following code shows.

```
stackoverflow['JobSat'].replace({'Very satisfied':4,
 'Slightly dissatisfied':1,
 'Slightly satisfied':3,
 'Very dissatisfied':0,
 'Neither satisfied nor dissatisfied':2},inplace = True)
```

Treating missing values is an important step in this analysis and we applied different methods depending on the situation. For columns that have less than 1500 N/As, we simply drop the data instance, but for columns containing more than 1500 N/As, we fill them by either mean,or mode, or random filling or inference from other columns. For 'YearsCodePro', we use value in

the 'YearsCode' minus 4 to fill N/As at first, because people usually take 4 years to study coding and go pro after that. The rest of the missing values are filled by mean. Values in the 'OrgSize' and 'Employment' are hard to infer, so we just randomly fill them by forwardfill and backfill. For categorical variables like 'DevType', we use mode to fill the missing values. After the data preparation stage, we have 61172 data instances ready for modeling.

Data Modeling

In this project, we consider three classification models. They are Decision Tree Induction, Random Forest, and Support Vector Machines.

Decision Tree Induction is simple to understand and can be clearly visualized in Python. It also handles categorical and discrete features best. The classification results are intuitive and help answer our questions on which factors make people want to seek new jobs.

Random Forest is an ensemble learning method for classification by constructing a multitude of decision trees at training time and output the class that is the mode of the classes. It is good for this analysis because it works like decision tree but could potentially achieve better accuracy than the decision tree without creating a highly overfitting model.

Support Vector Machine uses one-versus-all classification to map an instance to one category or the other, so that the separated categories are divided by a clear gap that is as wide as possible. It also handles multiclass classification well because it reduces the problem into multiple binary classification problems.

Because we have a lot of categorical variables from the dataset, the correlation heatmap doesn't generate any insights about strong correlation between two variables, so we jump right to the modeling step.

The job seeking classification model uses Decision Tree Induction, Random Forest and Support Vector Machines algorithms. The independent variables are 'Age', 'JobSat', 'CareerSat', 'MgrIdiot', 'EdLevel', 'Employment' and gender dummy variables. The label is 'JobSeek' which has value 1 for seeking new jobs and 0 for not seeking new jobs. To avoid overfitting, we use 70% of the dataset for training and 30% for testing to perform cross-validation. The data looks like the table below.

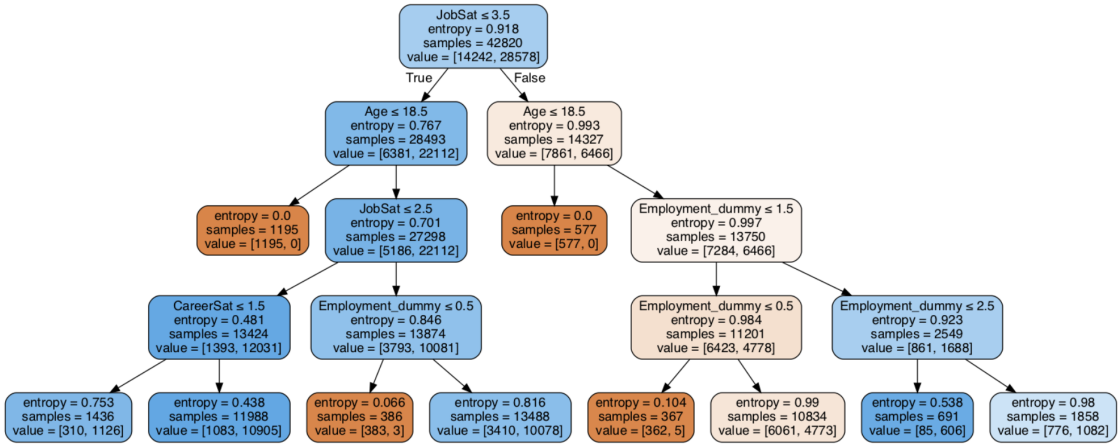
	Age	JobSat	MgrIdiot	EdLevel	Employment_dummy	CareerSat	LGBTQ	Man	Woman
0	14.0	3	3	0	0	5	0	1	0
3	22.0	3	3	1	1	5	0	1	0
5	28.0	3	3	1	1	5	0	1	0
7	24.0	3	3	1	2	5	0	1	0
9	24.0	1	2	2	1	2	0	1	0

The salary prediction model for US uses multiple linear regression algorithm. The independent variables X are 'JobTypeScore', 'YearsCodePro', 'Age' and 'EdLevel', and the dependent variable Y is 'ConvertedComp' which is a respondent's annual income. However, there are no apparent correlations between each X and Y form the scatterplots, and the regression model returns a r-square value as 0.0057. We think this happens for 2 reasons. Firstly, the independent variables we choose heuristically don't cause change of salary, but these are the only variables we can work with from the dataset. Secondly, there could be a lot of errors in the data that we are unable to correct as this is an unofficial survey and people could put fake or wrong salary information. To pivot from this, we decide to do an income level classifier by dividing the 'ConvertedComp' into 3 brackets. Income below $70,000 is 'low', being marked as 0, income above $70,000 but lower than $200,000 is 'Medium', being marked as 1, and income above 200,000 is 'high', being marked as 2. The new variable containing the 3 income levels will be 'CompLevel', and it will be used as the label. We will use Decision Tree and SVM algorithms with the same setup as the job seeking classification model to see if we are able to find any patterns and results.

EXPERIMENTS

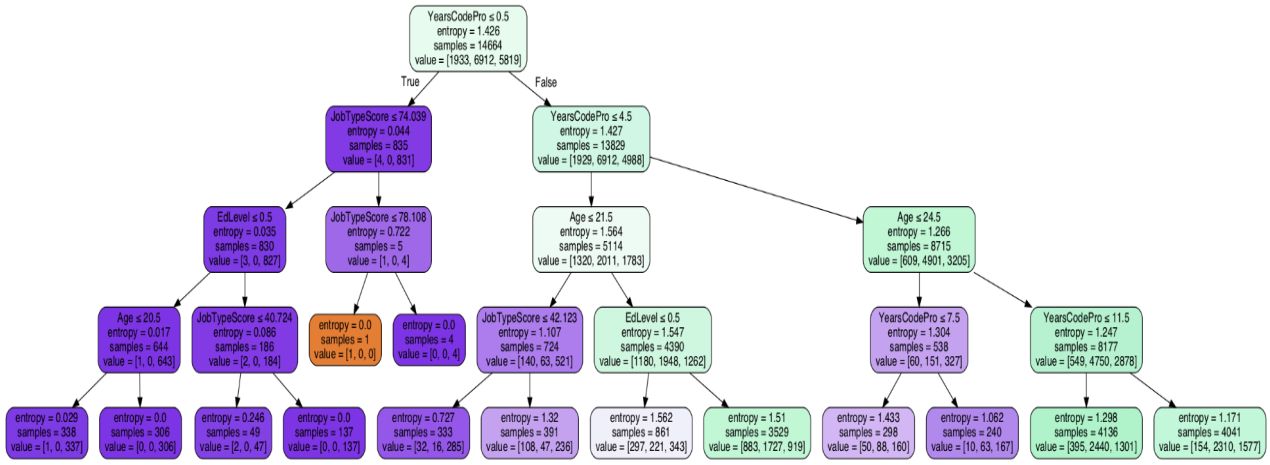
Job Seeker Classifier:

We firstly let the tree grow to the fullest and find out the accuracy is 71.4%. After pre-pruning the tree by switching the maximum depth of the tree between 4 and 5, we decided to use 4 as it produces 75.2% accuracy which is only 0.2% lower than 5 but generates a tree that is cleaner and easier to interpret. The result from Random Forest and SVM are close to the Decision Tree, with accuracy as 73.2% and 75.4% respectively. The final tree graph is as below.



Income Level Classifier:

Pivoted from the salary predictor, the income level classifier uses the same method as the Job Seeker classifier. For the Decision Tree, we set the maximum depth to 4 so that it is easier to see the top attributes the model takes to split. The final accuracy of the tree is 57.9% and that of SVM is 58.9%. The tree graph is as shown below.



Other Analysis Findings:

Statistical analysis indicates that a person with a higher degree usually has a higher salary. However, an interesting find is that Master's have a lower salary than Bachelors. Besides, our analysis suggests that people at any age tend to find a new job.

CONCLUSION

Job Seeker Classifier:

Because the Decision Tree model splits at JobSat first and then age, Job Satisfaction and Age should be the most important attributes to find out if a user is looking for a job or not. The accuracy of the 3 classification models are very close, but since the SVM model produces best accuracy score, we suggest using the SVM model to perform job seeker classification.

Income Level Classifier:

Firstly, the salary predictor model we originally wanted to do doesn't fit this dataset, as continuous numerical values are the best for a regression model. The accuracies from both Decision Tree Model and SVM are pretty low, so this classifier would not work either. Independent variables that are transformed from the categorical variables could distort the model and that might be the reason to cause this model to fail.

REFERENCES

[1] Scikit learn
https://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[2] Pandas
https://pandas.pydata.org/pandas-docs/stable/

[3] Passing Categorical Data to Sklearn Decision Tree
https://stackoverflow.com/questions/38108832/passing-categorical-data-to-sklearn-decision-tree

[4] Decision Trees and Random Forests
https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991