JOB TITLE RECOMMENDATION SYSTEM

Adithi Mysore Mohan, Manaswini Chilla, Nidadavolu Sri Satya Sai Pavan, Dinuka De Silva

CSCI-P556 Applied Machine Learning

ABSTRACT

With the availability of abundant data on profiles of various people online, we focused our research on profiles specific to the IT industry. The idea here is to collect data of people with diverse IT roles and their corresponding skill set, experience and company, and improve the quantity and quality of this data by data augmentation methods. As observed with various existing research on this topic, collaborative filtering algorithms are popular with this kind of recommendation system. This project, after preprocessing the data, getting the desired structure of the dataset, focuses on training the same with multiple machine learning algorithms. A comparison has been made among the performance mainly accuracy of these trained and tested models, and two models that have fairly performed well, have been chosen best suited with the dataset in hand. The various algorithms trained include Support Vector Machines, Random Forest classifiers, K Nearest Neighbors, Decision trees, Gaussian Naïve Bayes, Multinomial Naïve Bayes, and Multi-layer perceptron. Out of these, we found Support Vector Machines, and Random Forest Classifier to be the most appropriate classifiers for the dataset. As an additional task, we scanned through a couple resumes, collected the skill set, ran through both of these classifiers, and predicted the roles for each of these resumes.

Index Terms— Machine Learning, Data, Jab Recommendation, Classification

1. INTRODUCTION

"There is an abundance of data available in the digital world and if it is harnessed effectively and correctly it can provide terrific insights" -Gian Filgoni.

Starting our though process at this step, and given our common exposure to IT world, we agreed on working on a job recommendation system. As a part of this project we have attempted to study a data set comprising of IT data that includes an individual's role, such as web developer, data scientist, skills, which include various programming language, experience and company, applied various machine learning algorithms, to compare accuracy and other performance metrics, and choose an appropriate model that predicts a suitable role, given current set of qualifications.

Additionally, we have also made an attempt to write a script that reads through resumes, collects the skills set, which is fed to our trained model and it predicts an appropriate role.

2. RELATED WORK

There exists multiple research papers and projects online but we were motivated by the paper "Implementation of an Automated Job Recommendation System Based on Candidate Profiles", in this paper they aim to implement a recommendation system for job hunting for which, given data set, they compared two collaborative filtering algorithms. We took up this problem statement, and decided to train and evaluate few other machine learning algorithms to see if they perform better. Data being a vital part of machine learning, we wanted to explore whether we could collect a huge volume of real time data, using web crawling websites like LinkedIn, Indeed etc. Also, based on finite attributes, whether we can successfully predict job role/ future company. We have limited this study to IT industry related data, we did face certain challenges with dataset, which is different from other implemented projects online, and attempted some classification algorithms which are not so explored with recommendation system. On a majority basis, collaborative filtering algorithms are more popular with these kind of problem statement.

3. DATA COLLECTION

Our initial approach to collect data was through web crawling using python's beautiful soap and lxml modules. We tried this on job search engines like LinkedIn and Indeed. But this initial approach did not yield us a lot of data since there were added security features for these public sites. So, we had to bypass the CAPTCHA to these sites which proved to be difficult. Also, the session used to get time out if the number of pull requests were more than a certain number in a fixed amount of time.

We wanted to find a dataset with features like Skills, JobTitle, Experience and CompanyName. The data sets available online did not have the relevant features that we required. Hence, we performed data augmentation based on the limited data we had obtained via web crawling(skills and company names) and generated more data.

In addition to this, there's a function from Scikit learn library, known as 'makeRegression'. Given a number of in-

formative features, it will generate data which has correlation with the features. Instead of using this inbuilt function, we wanted to test out data augmentation and see what type of results it gave. We used this augmented data for training and testing our models.

4. DATA PRE-PROCESSING

With this data set, we have cleaned it, from a raw format, we have converted the data as a role with corresponding list of skills, experience and location. We collected a unique set of these skills and added these as columns, which will have a value of one if, for that record, that skill is present and 0 otherwise. We then studied it thoroughly, we have visualized this data, as proportion of each of the roles in the data set, top 10 skills for each of these roles, the proportion of a skill in each role and so on. At this stage, the data set has around 22,000 records with 47 columns, with 44 skills and 7 profiles.

With this data set, we have cleaned it, from a raw format, we have converted the data as a role with corresponding list of skills, experience and location. We collected a unique set of these skills and added these as columns, which will have a value of one if, for that record, that skill is present and 0 otherwise. We then studied it thoroughly, we have visualized this data, as proportion of each of the roles in the data set, top 10 skills for each of these roles, the proportion of a skill in each role and so on. At this stage, the data set has around 22,000 records with 47 columns, with 44 skills and 7 profiles.

5. APPROACHES

We identified a set of supervised classification algorithms to train and analyze the accuracy and evaluation matrices.

5.1. Naive Bayes

From the pre-processed data, a Gaussian Naive Bayes classifier was trained assuming skills are independent from each other, given the job title. The results may not be prominent since there's many missing prior probabilities within the distribution. Also, our data set may not be strictly adhering to the Gaussian distribution, owing to a lower accuracy(36%).

5.1.1. Naive Bayes with Laplace Smoothing

To overcome classification error caused by the missing prior probabilities, Laplace smoothing was incorporated and and the overall accuracy increased to 50%. Multinomial Naïve Bayes considers a feature vector where a given term represents the frequency.A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. Here the pseudo count is 0.

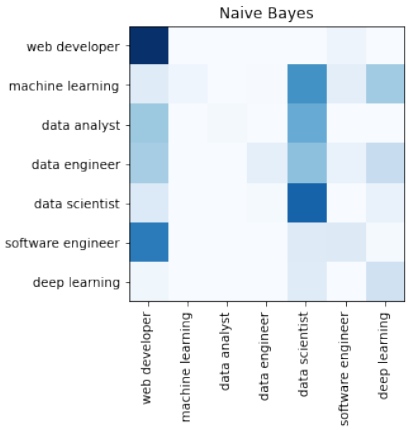

Fig. 1. Naive Bayes Classifier - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.71	0.02	0.03	662
machine learning	0.81	0.08	0.15	713
data analyst	0.31	0.77	0.44	614
data engineer	0.22	0.19	0.21	137
data scientist	0.83	0.04	0.08	629
software engineer	0.37	0.12	0.18	832
deep learning	0.37	0.96	0.53	967
accuracy			0.36	4554
macro avg	0.52	0.31	0.23	4554
weighted avg	0.54	0.36	0.25	4554

Table 1. Naive Bayes Classifier - Evaluation Matrices

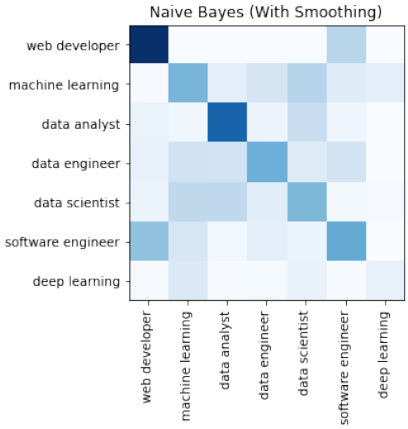

Fig. 2. Naive Bayes Classifier with Laplace Smoothing - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.58	0.64	0.61	662
machine learning	0.52	0.39	0.45	713
data analyst	0.37	0.36	0.37	614
data engineer	0.40	0.06	0.10	137
data scientist	0.37	0.38	0.37	629
software engineer	0.44	0.41	0.43	832
deep learning	0.62	0.80	0.70	967
accuracy			0.50	4554
macro avg	0.47	0.43	0.43	4554
weighted avg	0.49	0.50	0.49	4554

Table 2. Naive Bayes Classifier with Laplace Smoothing - Evaluation Matrices

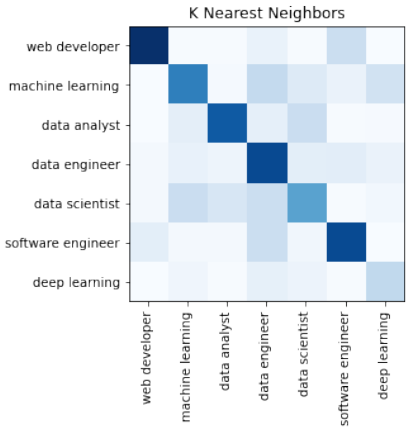

Fig. 3. K Nearest Neighbors Classifier - Confusion Matrix

5.2. K Nearest Neighbors

The K Nearest Neighbors Classifier was able to achieve an overall accuracy of 67% which is better than Naive Bayes even though the calculation complexity is involved in the prediction that it has to traverse through the entire training set to find the k nearest neighbors by calculating the distance against each training data point.We're using Minkowski as the distance metric and n(number of neighbors being 5. To avoid misclassification, we set the value of K to an odd number

5.3. Decision Trees

The decision tree classifier was able to reach an overall accuracy of 66%. But, decision tree may be not the best model for this problem since small changes in the training data could lead to a major difference in the tree and the classification results.

	precision	recall	f1-score	support
web developer	0.77	0.67	0.72	662
machine learning	0.49	0.72	0.58	713
data analyst	0.50	0.44	0.47	614
data engineer	0.48	0.21	0.29	137
data scientist	0.60	0.55	0.58	629
software engineer	0.69	0.72	0.70	832
deep learning	0.88	0.80	0.84	967
accuracy			0.65	4554
macro avg	0.63	0.59	0.60	4554
weighted avg	0.67	0.65	0.65	4554

Table 3. K Nearest Neighbors Classifier - Evaluation Matrices

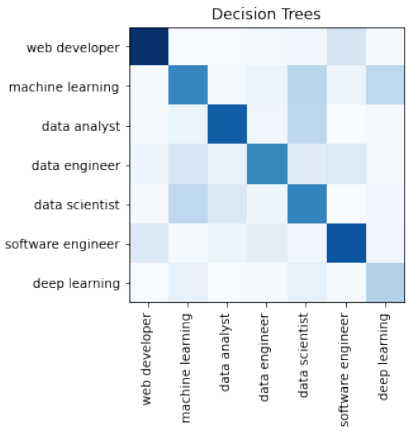

Fig. 4. Decision Trees - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.73	0.70	0.71	662
machine learning	0.72	0.56	0.63	713
data analyst	0.45	0.57	0.51	614
data engineer	0.46	0.26	0.33	137
data scientist	0.54	0.57	0.55	629
software engineer	0.70	0.73	0.71	832
deep learning	0.82	0.85	0.84	967
accuracy			0.66	4554
macro avg	0.63	0.60	0.61	4554
weighted avg	0.67	0.66	0.66	4554

Table 4. Decision Trees - Evaluation Matrices

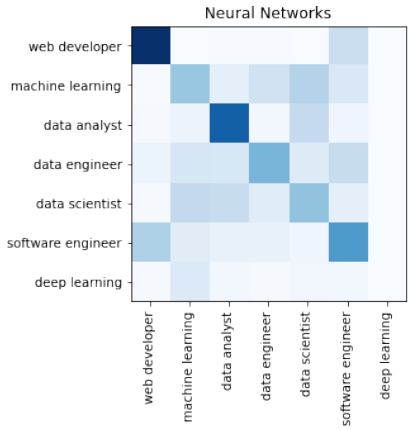

Fig. 5. Neural Networks - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.58	0.68	0.62	662
machine learning	0.53	0.39	0.45	713
data analyst	0.35	0.34	0.34	614
data engineer	0.00	0.00	0.00	137
data scientist	0.35	0.32	0.33	629
software engineer	0.43	0.49	0.46	832
deep learning	0.71	0.83	0.77	967
accuracy			0.52	4554
macro avg	0.42	0.44	0.42	4554
weighted avg	0.49	0.52	0.50	4554

Table 5. Neural Networks - Evaluation Matrices

5.4. Neural Networks

A multi-layer perceptron with two hidden layers of size 100 and 200 was trained over 1000 iterations. The model parameters were optimized through stochastic gradient descent from a batch size of 100 and the learning rate of 0.01. We're using ReLU activation function because it does not saturate for the positive value of the weighted sum of inputs. This gave an accuracy of about 53%.

5.5. Support Vector Machines

Given the nature of feature space which is a set of skills, Support Vector Machines can be considered as a suitable algorithm in terms of accuracy. In other words, support vector machines breaks down the multi class classification problem in to a set of binary classification problems where each class is assigned a support vector that is linearly separated from every other class through the higher dimensional feature space. Support vector machines were able to achieve an accuracy of 67% overall. On the other hand it took more training time considerably since the training time is proportion to the number of classes.

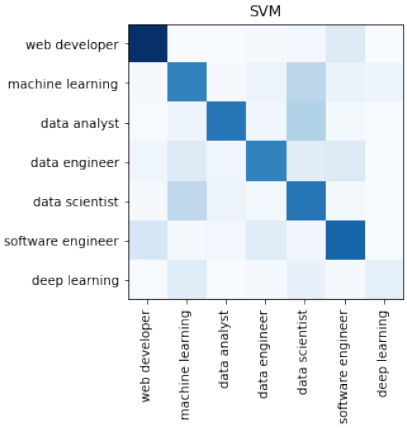

Fig. 6. Support Vector Machines - Confusion Matrix

	precision	recall	f1-score	support
web developer	0.84	0.64	0.73	662
machine learning	0.74	0.60	0.66	713
data analyst	0.47	0.64	0.54	614
data engineer	0.65	0.08	0.14	137
data scientist	0.54	0.60	0.57	629
software engineer	0.68	0.70	0.69	832
deep learning	0.81	0.88	0.84	967
accuracy			0.67	4554
macro avg	0.67	0.59	0.60	4554
weighted avg	0.69	0.67	0.67	4554

Table 6. Support Vector Machines - Evaluation Matrices

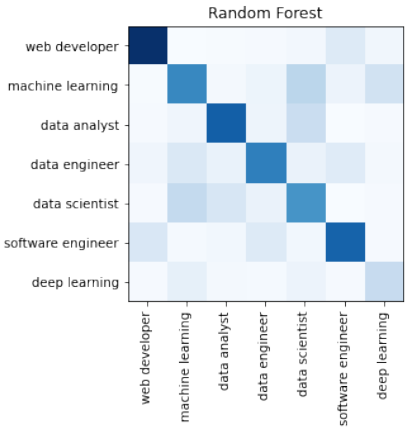

Fig. 7. Random Forest Classifier - Confusion Matrix

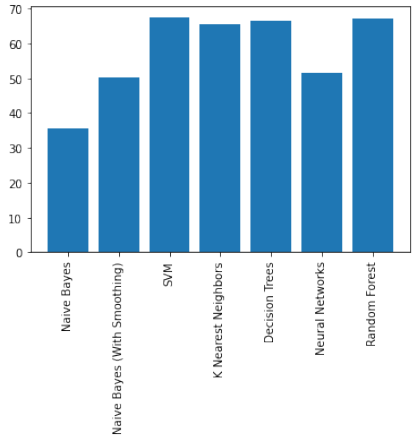

Fig. 8. Summary of the Accuracy

	precision	recall	f1-score	support
web developer	0.73	0.72	0.72	662
machine learning	0.68	0.61	0.64	713
data analyst	0.47	0.53	0.50	614
data engineer	0.48	0.21	0.29	137
data scientist	0.55	0.58	0.56	629
software engineer	0.72	0.70	0.71	832
deep learning	0.82	0.87	0.84	967
accuracy			0.67	4554
macro avg	0.63	0.60	0.61	4554
weighted avg	0.67	0.67	0.67	4554

Table 7. Random Forest Classifier - Evaluation Matrices

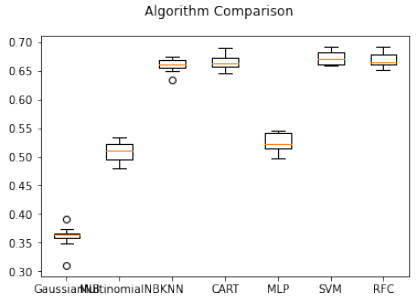

Fig. 9. Comparison of Algorithms

5.6. Random Forest Classifier

This can be considered as a good candidate for the problem given the multi class classification capabilities of random forest classifier over support vector machine where the multi class classification is broken down in to multiple binary classification problems. As also expected, the classifier was able to achieve an overall accuracy of 67%.

6. SUMMARY OF RESULTS

From our results, we can see that SVM, RFC and K Nearest Neighbors perform better, followed by, Decision trees. Naïve Bayes gives the lowest accuracy, which was improved a bit by performing Laplace Smoothing. Neural networks, multilayer perceptron gives a slightly better accuracy than Naïve Bayes with smoothing.

Random forest classification is better suited for this problem as it is intrinsically suited for multi-class classification problem. While SVM is giving out a fairly decent accuracy, it is more appropriate for binary classification. As this is a multi-class classification, we're essentially breaking down this multi-class problem into multiple binary classification problem.

Although the overall accuracy is around 66 to 67%, the accuracy for each role classified is greater than or equal to 88%. This maybe because many roles have a set of common skills which may cause the overall accuracy to drop.

This image shows the algorithm comparison in terms of accuracy.

From our results, we were able to analyze how prevalent certain skills are for a particular job position.

For instance, for the role of a data scientist, skills like python, r, SQL and Hadoop are more important than the other skills.

In addition to this, we took a bunch of resumes, treated it as bag of words and trained SVM and RFC on it. This gave out a job position as an output for each resume.

7. CONCLUSIONS

We were curious how job search engines work. We explored a few sites like LinkedIn, Indeed and Glassdoor and we wanted to see how exactly it works. While studying this problem statement, for now, just concentrating on the skill set, we were relatively able to extract meaningful information.

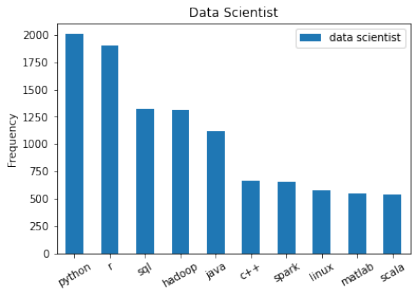

Fig. 10. Distribution of Skills

Model	Accuracy
Naive Bayes	36
Naive Bayes with Laplace Smoothing	50
K Nearest Neighbors	67
Decision Trees	66
MLP	53
SVM	67
RFC	67

Table 8. Classification Accuracy of Algorithms

What we liked about the project is that, we may extract more meaningful information, if we incorporate location, or study the keywords in the objective of resume, experience, previous companies he has worked in, his educational qualifications etc.

We liked exploring new data analysis and visualization concepts. Although web crawling was a huge challenge to us, we liked working on it.

We learnt how to implement various classification techniques and compare them, along with a few data analysis and visualization techniques. We also learnt how to perform data crawling to an extent.

Initially we thought of predicting the company names but that proved hard because of the limitations on web crawling. So, we decided to keep the job position as the target label.

We still have doubts on the scale to which web crawling can be performed.

One major challenge was the data. Given a good dataset, based on real time profiles, we could have achieved better performance, and given more time, we could have implemented more attributes while training.

8. FUTURE WORK

We wish to incorporate location, or study the keywords in the objective of resume, experience, previous companies he has worked in, his educational qualifications etc. This might be a good research topic to see how we can study all attributes at the same time, or study them individually with the target label. Within the deadline of this project we wish to incorporate more roles and skills, and see how our model behaves. For further scope of this project, we are looking forward to incorporating aforementioned attributes, and exploring any existing/hybrid models which may be capable of studying a full resume.

Acknowledgements We would like to thank Professor Donald Williamson and Junyi Fan for all their guidance throughout this project and the course. We would like to thank Luddy School of Informatics for all the technical assistance provided. We would like to thank our peers for providing a competitive platform and giving us enough motivation to perform better.

9. REFERENCES

[1] Implementation of an Automated Job Recommendation System Based on Candidate Profiles,

Vinay Desai, Dheeraj Bahl, Shreekumar Vibhandik,Isra Fatma.
https://www.irjet.net/archives/V4/i5/IRJET-V4I5

[2] Applying Classifications Techniques in Job Recommendation System for Matching of Candidates and Advertisements,

Gözde Özcan, Şule Gündüz Öğüdücü.
https://infonomics-society.org/wp-content/uploa

[3] Matching People and Jobs: A Bilateral Recommendation Approach,

Jochen Malinowski, Tobias Keim.
https://citeseerx.ist.psu.edu/viewdoc/download?

[4] Web crawler,
https://en.wikipedia.org/wiki/Web$_c rawler$

[5] Summary of web crawler technology research, Linxuan Yu, Yeli Li, Qingtao Zeng, Yanxiong Sun, Yuning Bian and Wei He.
https://iopscience.iop.org/article/10.1088/1742

[6] Understanding Data Augmentation — What is Data Augmentation how it works?, Great Learning team
https://www.mygreatlearning.com/blog/understand

[7] Multi-class classification,

https://en.wikipedia.org/wiki/Multiclass$_c lassifica$

[8] Performance Comparison of Multi-Class Classification Algorithms, Gursev Pirge.
https://gursev-pirge.medium.com/performance-com

[9] A Simple Explanation of the Bag-of-Words Model, Victor Zhou.
https://towardsdatascience.com/a-simple-explana

[10] Introduction to Data Visualization in Python, Gilbert Tanner.
https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed