# Imputing Job Hunter Interest with Clustering, Logistic Regression and Ensemble Decision Trees

**Elliott Chen, COMP-4449**

## Business Needs

- We provide a two-sided marketplace to job hunters and those who are looking to hire
- Our mobile application depends on an algorithm that decides on how to feed content to our users
- Older iterations of the algorithm have tended to rely on inelegant brute force methods that are time-expensive, creating unwelcome long wait times for end users

## Business Needs, cont.

- We hope to leverage a neural net model that can make predictions on appropriate content to show our users more quickly and with better accuracy, but much of our potential training data does not yet have classifications
- The goal of this project is to infer those classifications so that training the new machine learning model can proceed

## Project Plan

- Since we are inferring many classifications compared to the complete training data that we do have, establishing a benchmark for success will be complicated
- Also since we are inferring many classifications and with relatively unstructured data, we will approach the problem from multiple angles
- We will attempt to use clustering, logistic regression, and random forests to predict whether a job hunter (applicant) would be interested in applying for a given job

## Project Plan, cont.

- Performing logistic regression or random forest classification once the data is wrangled will be relatively straightforward—our label will be a binary variable
- For clustering, we hope that applicant/job pairings will naturally form groups that are correlated with positive or negative interest in an opening; we will then be able to associate certain clusters with positive/negative interest creating a classification

## Project Plan, cont.

- Our main metrics considered will be precision and recall
- The cost of false positives is feeding our end users content that is inappropriate for them; although we surmise that job hunters have a certain tolerance for irrelevant jobs and vice versa, seeing too many will sour user experience and depress growth
- The cost of false negatives is junking content that our users would like to see; although they don’t know what they missed, we do not yet have so many profiles that we want to “waste” them in such a manner if we can help it

## Data

- We have ~1,400 rows of training data stored in a PostgreSQL database; each row represents an applicant expressing positive/negative interest in a job, or a job expressing positive/negative interest in an applicant, as well as the applicants' and jobs' profiles
- In addition, we have roughly ten times that number of pairs of resumes and job posts that we hope to create binary classifications for

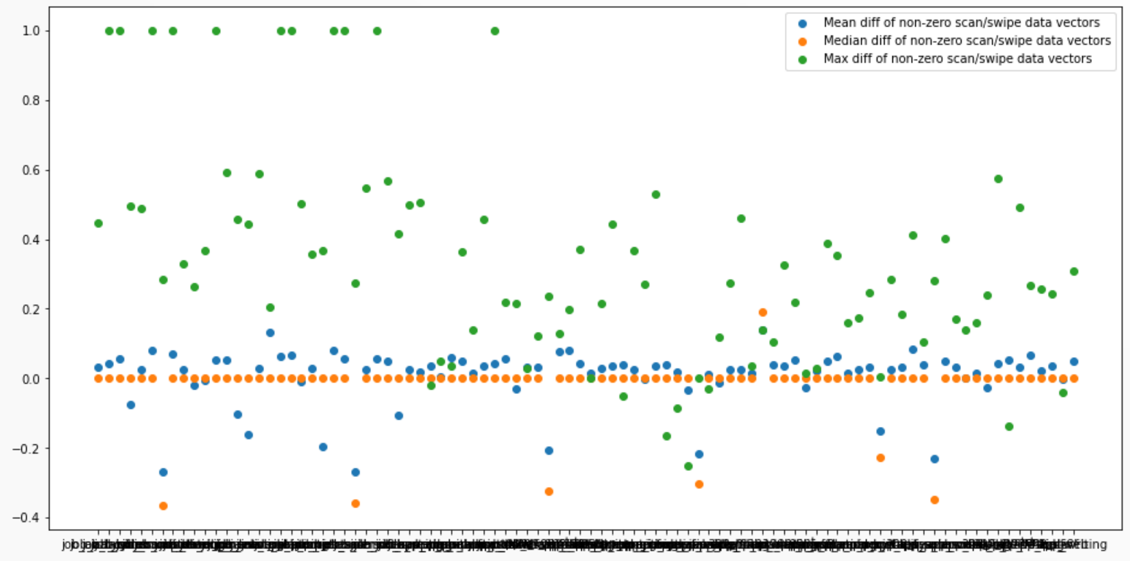
## Data Wrangling

- Perforce we have to train a model on the smaller dataset; but since we hope to extend it on the larger one, we will need to marry the two datasets somehow
- We focus on extracting strings representing an applicant’s available and a job’s required technical skills and qualifications
- Results are fed into sklearn’s TfidfVectorizer class to create features that track the frequency of skills mentioned in an applicant or job profile

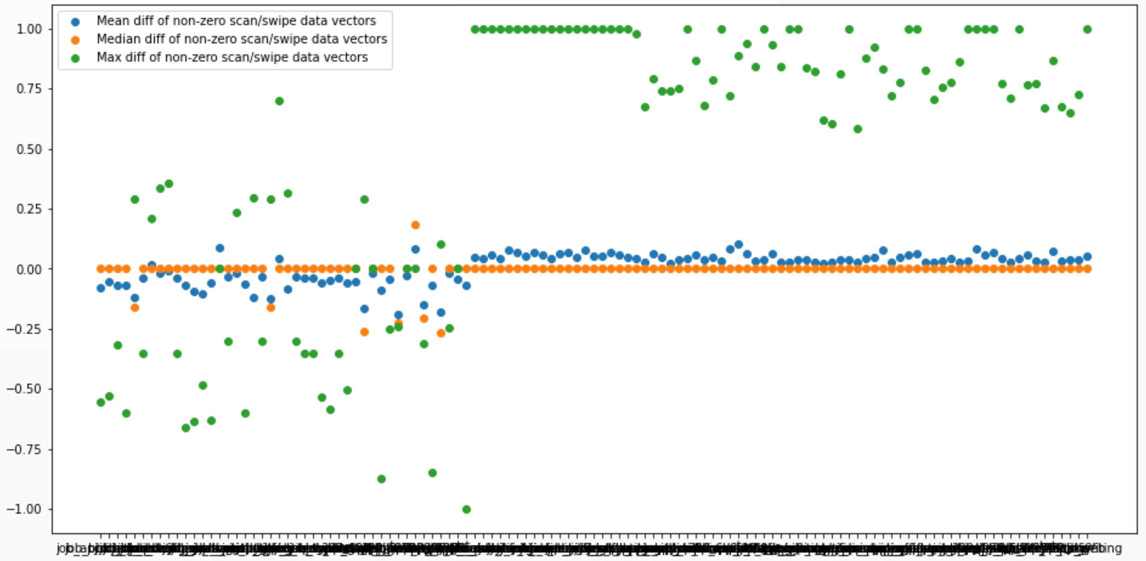
# Data Wrangling, cont.

- Two further issues need to be addressed: first, whether the features' observations differ noticeably when the training and testing data are vectorized together

Difference between training/test data features vectorized together



Difference between training/test data features vectorized independently

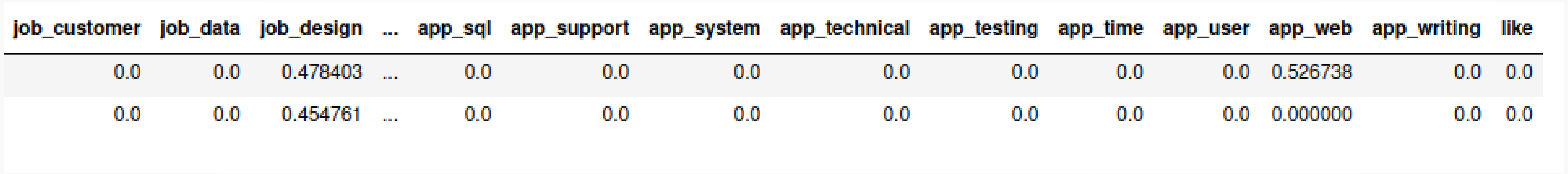


## Data Wrangling, cont.

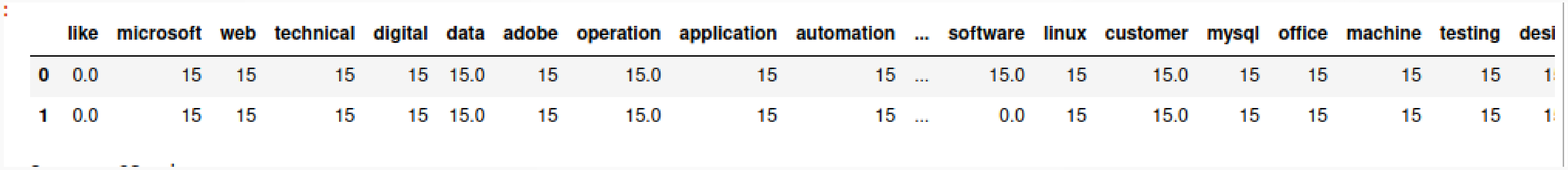
- Secondly, whether it makes sense to combine two features measuring a job’s emphasis on a skill versus an applicant’s emphasis on a skill
- For example: given a row that contains the column data job_Python=0.25 and applicant_Python=0.20, our models might not be able to intuit the connection between these two features
- Might it make more sense to replace the two columns with a single column tracking the relationship between the two, most likely as a percentage?
- In the example given, a single column observation Python=1.25 would replace the job and applicant columns

# Data Wrangling, cont.

- Given these two questions, four dataframes were generated to see which would yield the best results
- Sample dataframe rows, one with separate job/applicant skill features and one with combined features:



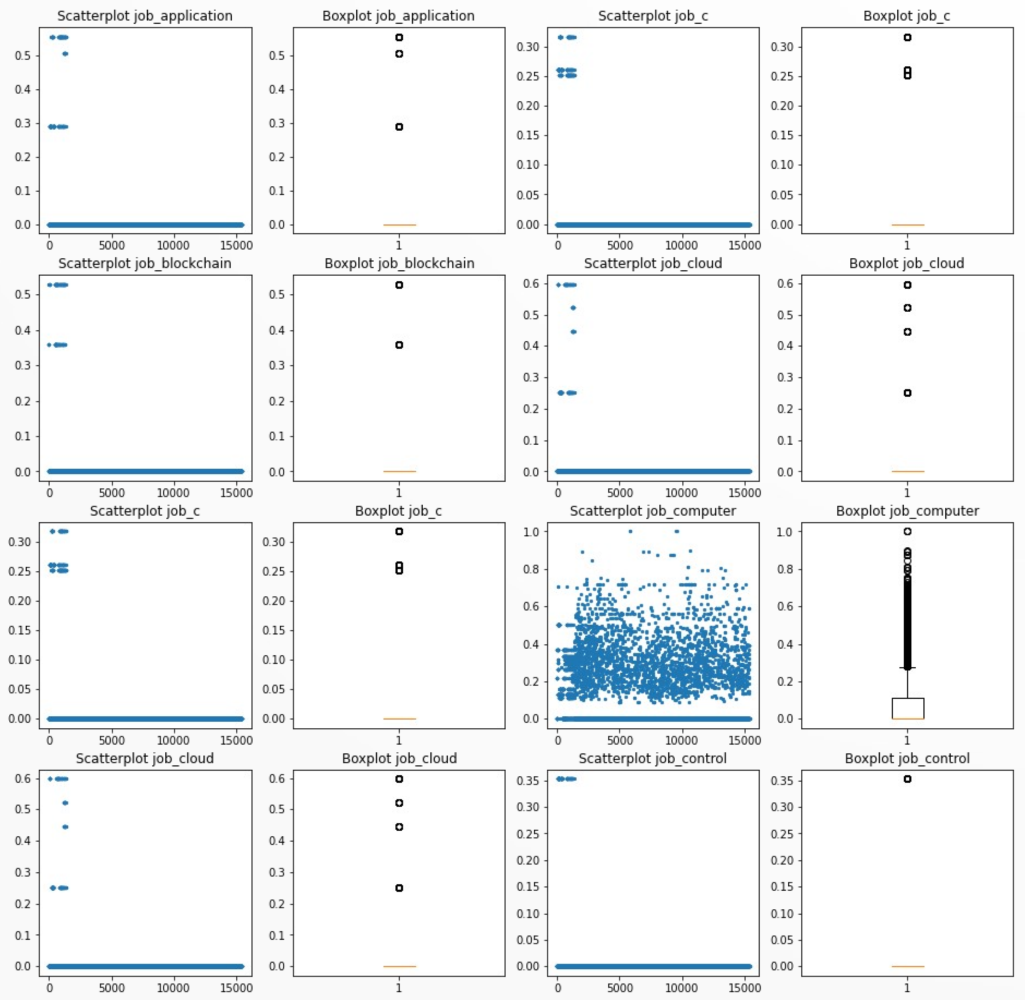
| job_customer | job_data | job_design | ... | app_sql | app_support | app_system | app_technical | app_testing | app_time | app_user | app_web | app_writing | like |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 0.0 | 0.478403 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.526738 | 0.0 | 0.0 |
| 0.0 | 0.0 | 0.454761 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 |



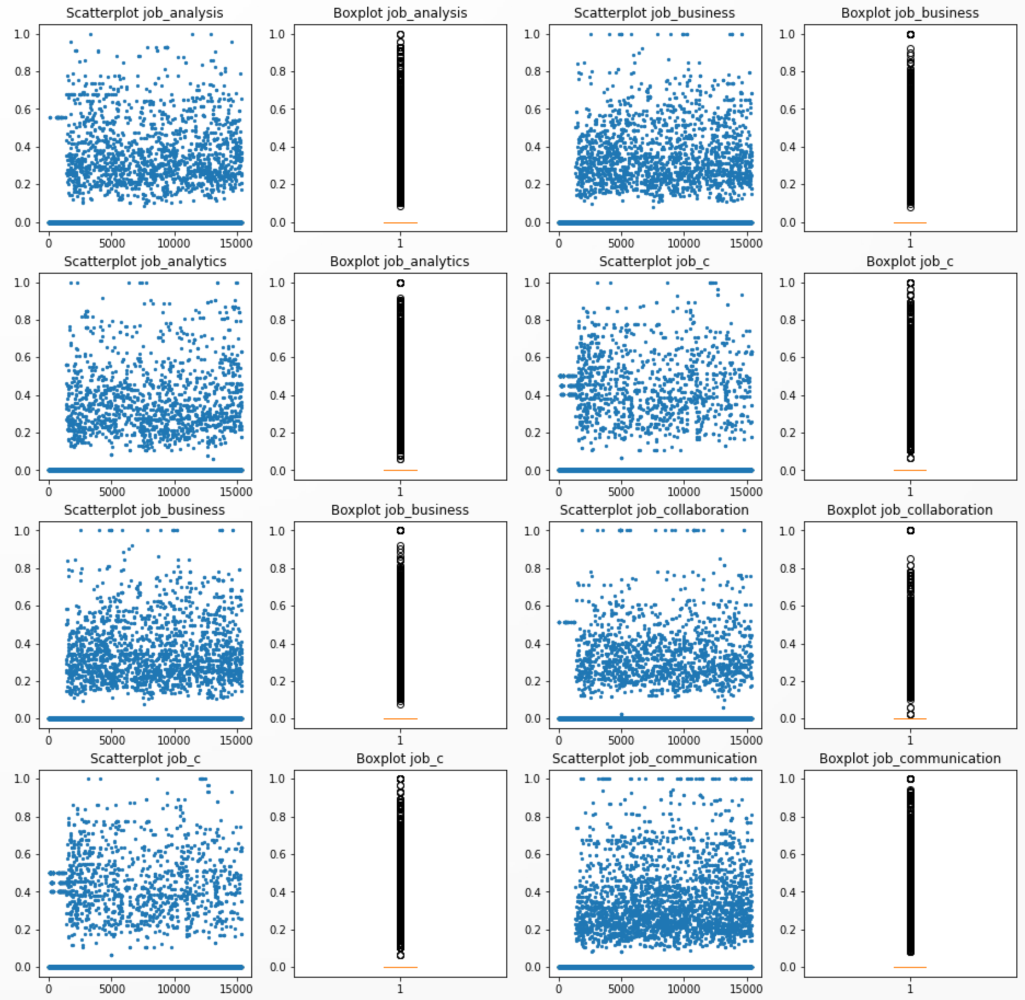
| | like | microsoft | web | technical | digital | data | adobe | operation | application | automation | ... | software | linux | customer | mysql | office | machine | testing | desi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 15 | 15 | 15 | 15 | 15.0 | 15 | 15.0 | 15 | 15 | ... | 15.0 | 15 | 15.0 | 15 | 15 | 15 | 15 | 1 |
| 1 | 0.0 | 15 | 15 | 15 | 15 | 15.0 | 15 | 15.0 | 15 | 15 | ... | 0.0 | 15 | 15.0 | 15 | 15 | 15 | 15 | 1 |

# Data Visualizations

## Independently Vectorized, Independent Job/Applicant Features

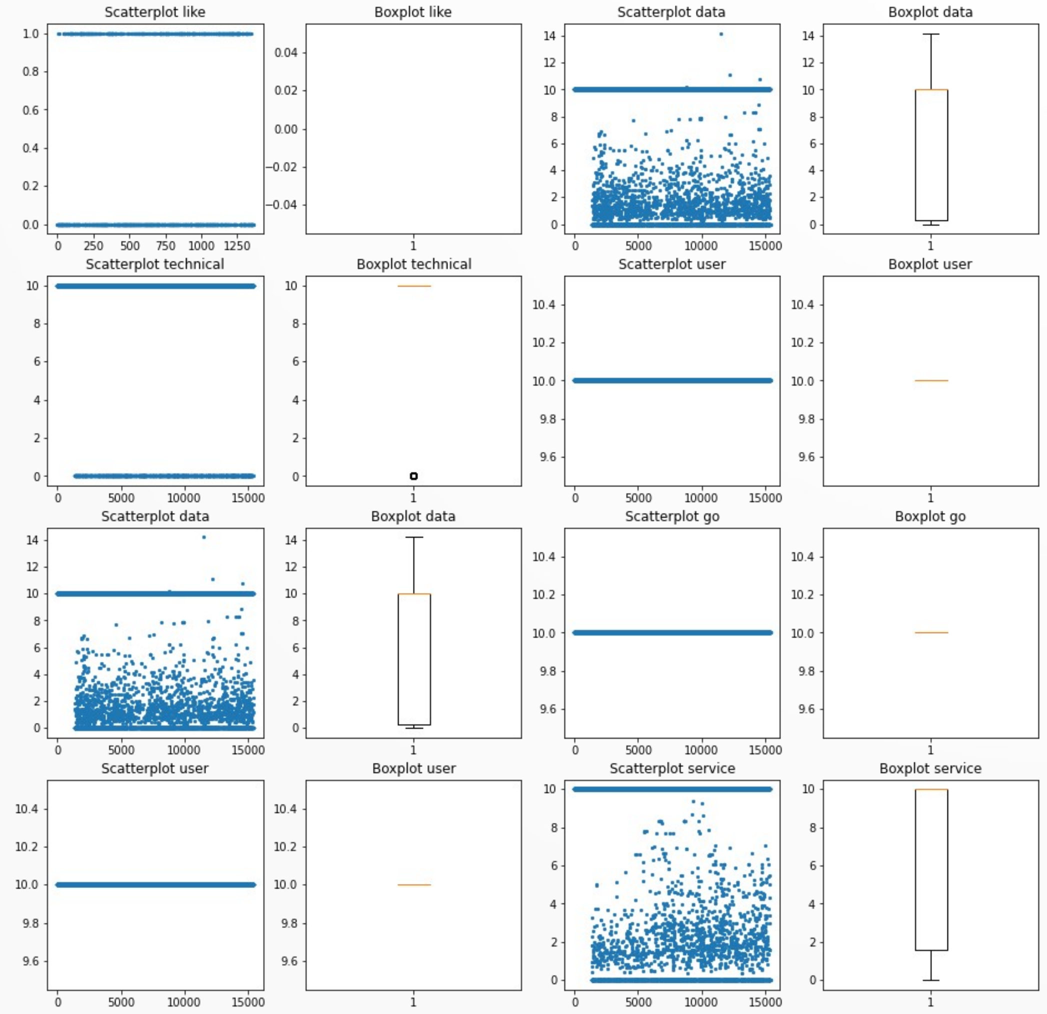


## Simultaneously Vectorized, Independent Job/Applicant Features

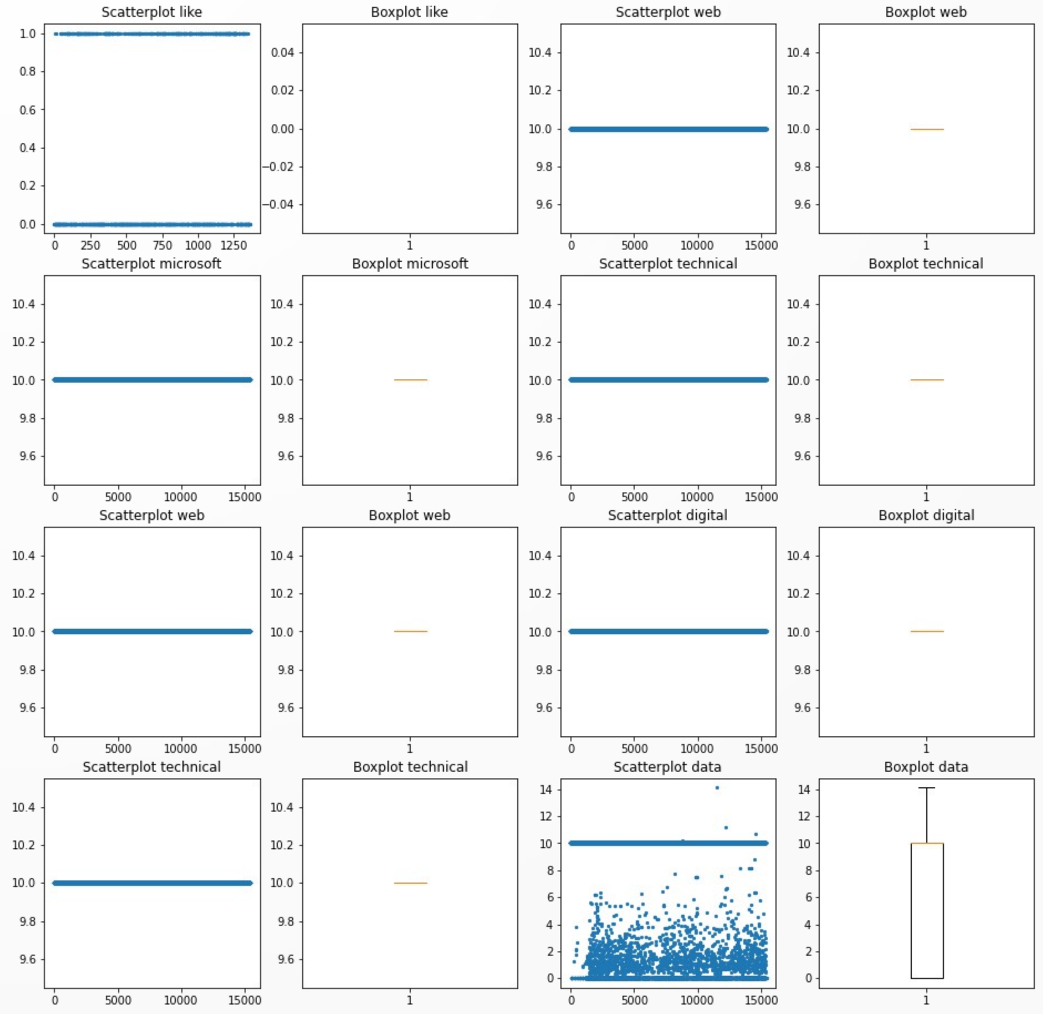


# Data Visualizations, cont.

Independently Vectorized, Merged Job/Applicant Features



Simultaneously Vectorized, Merged Job/Applicant Features

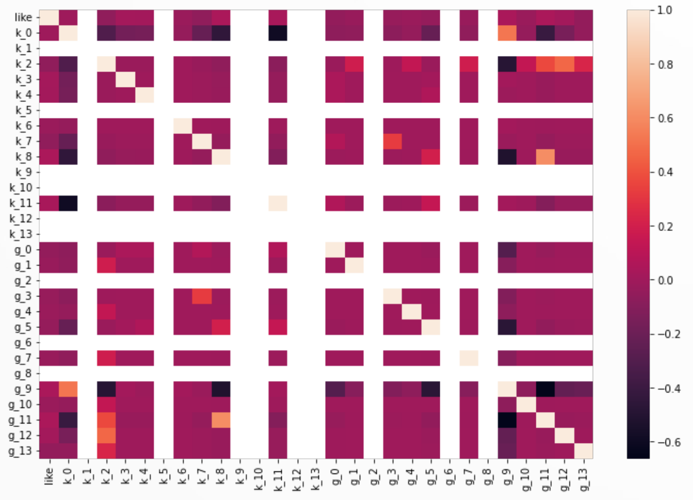
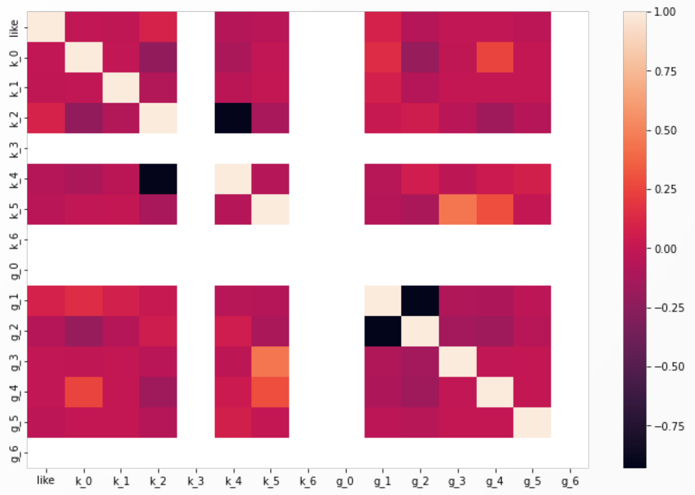
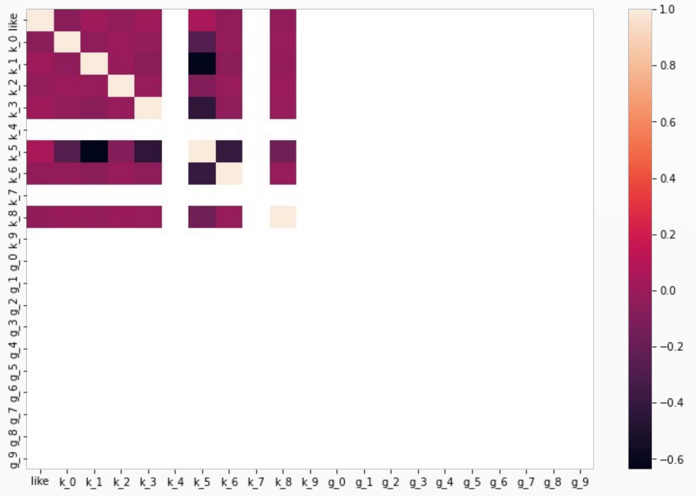
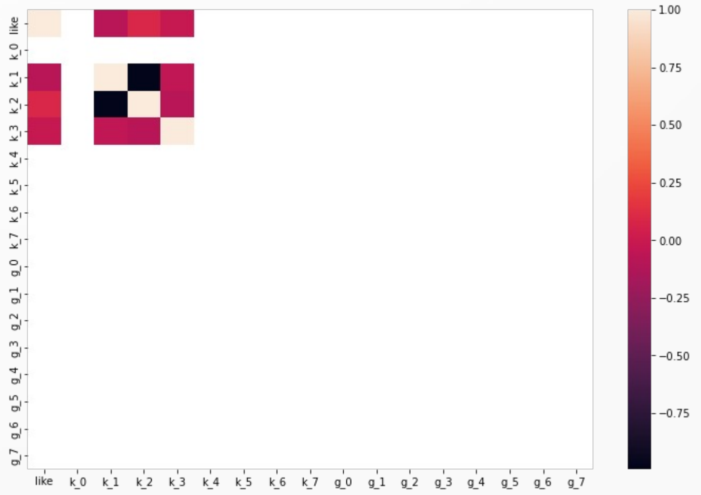


## Clustering

- Using the elbow method to determine an appropriate number of clusters for each dataframe after performing some dimensionality reduction through PCA, we arrive at a range from 7 to 14 clusters
- We will attempt KNN and Gaussian distribution clustering, generating labels; we will then see if there is any correlation between our labels and the known outcome variables to see if clustering is a viable path forward

# Clustering, cont.

- In short: there were not enough notable correlations, and we will have to move on to logistic regression

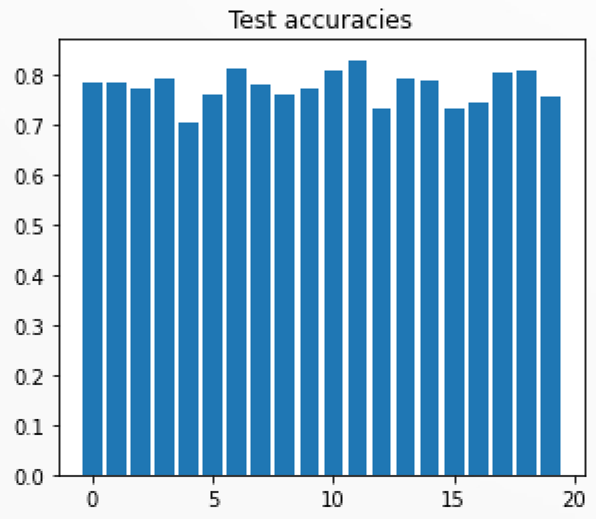





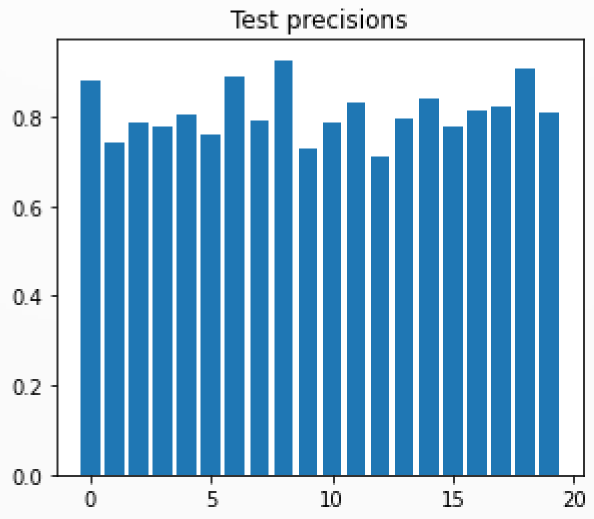
## Logistic Regression

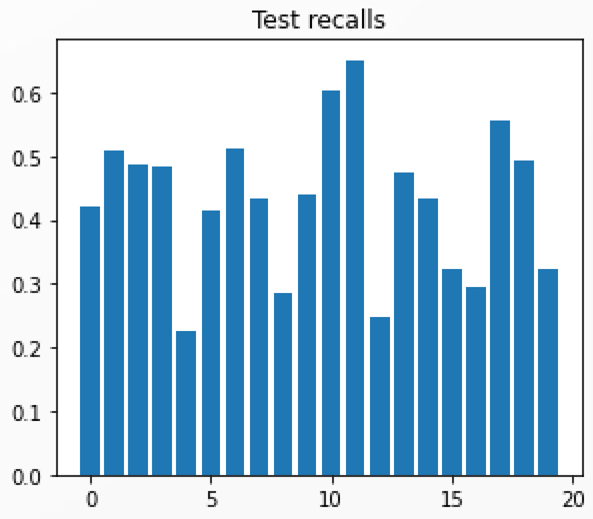
- After performing PCA, logistic regression models failed to deliver good results for any of our four dataframes; in some cases, they failed to converge altogether
- Highest testing accuracy, precision and recalls achieved were 69%, 24%, and 65%
- The recall is sufficient and overall accuracy is not atrocious, but this precision is unacceptably low

# Ensemble Decision Trees

- Our random forests gave good results; out of twenty models trained, five for each dataframe, the one with best accuracy (83%) and recall (65%) still delivered a very acceptable precision (83%)
- This model was trained on the dataframe where the expressions of interest and resume/job pairings were vectorized simultaneously, and where the applicant/job skill features were merged together







## Conclusion and Next Steps

- Our selected model is a random forest classifier with a max depth of 12, max features of 0.25, and 75 estimators
- When creating classifications for the resume/job data, this model predicted that roughly one in ten people who compared their resume to a job would express positive interest in that job
- Time to start tinkering with a neural net!