Resume Qualities that Receive a Callback

Anna Donnell, Ashlyn Goila, and Kali Javetski

I. Introduction

Waiting for a job callback can be incredibly stressful due to the uncertainty it brings. Each passing moment feels like an eternity as you anxiously anticipate whether your efforts have paid off. The waiting period often leads to overthinking and second-guessing, as you wonder if you made the right impression or have the right qualities listed on your resume. The stakes feel high, especially if you're eager for a career change or in need of employment, adding to the pressure and anxiety of the situation. As a result, we wanted to discover the most valued qualities of an applicant that determine whether or not they receive a call back.

The initial goal of our model was to use classification learning to analyze which instances get call backs and their key attributes. Since our dataset involves jobs across only two cities, Chicago and Boston, we wondered if we could use one of the city's data to build a good model for the other city. By using one city's data for training and the other for testing, we can learn if our model is able to generalize across major cities or if each city has specific trends in resume attributes. In addition, our biggest concern or challenge with using this dataset was with how we were going to handle the unbalanced data.

II. Data Description

Our data set is from Kaggle, and it is called "Key Resume Attributes Impacting Job Callbacks." The set is made up of 4870 instances and 30 attributes. The attributes provide information on specific jobs, on people's resumes, and on people's demographics. It is important to note that some of the demographic attributes are influenced by the person's name, so it is not necessarily an absolute fact. We will keep this in mind and discuss the raised ethical concerns later. Most attributes are nominal including those that have "1" and "0" as its values because

these correlate with a yes or no question. Lastly, only 2 of the 30 attributes contain missing values which are job federal contractor and minimum experience required, and we kept this in mind as we went into preprocessing.

Attribute	Description	Values
1. job_ad_id	Unique ID associated with the advertisement	numeric
2. job_city	City where the job was located	Boston Chicago
3. job_industry	Industry of the job	other_service business_and_personal_service wholesale_and_retail_trade finance_insurance_real_estate manufacturing
4. job_type	Type of role	secretary retail_sales manager sales_rep clerical
5. job_fed_contractor	If the employer is a federal contractor	0 = No 1 = Yes, NA
6. job_equal_opp_employer	If the employer is an Equal Opportunity Employer	0 = No 1 = Yes
7. job_ownership	Type of company	private unknown nonprofit public
8. job_req_any	If any job requirements are listed (If there are, the other job_req_* fields give more detail)	0 = No 1 = Yes
9. job_req_communication	If communication skills are required	0 = No 1 = Yes
10. job_req_education	If some level of education is required	0 = No 1 = Yes

11. job_req_min_experience	Amount of experience required	‘Some’ mixed in with numeric attributes
12. job_req_computer	If computer skills are required	0 = No 1 = Yes
13. job_req_organization	If organization skills are required	0 = No 1 = Yes
14. job_req_school	Level of education required	none_listed high_school_grad some_college college
15. recieved_callback	If there was a callback from the job posting for the person listed on this resume	0 = No 1 = Yes
16. firstname	The first name that was used on the resume	strings
17. race	Inferred race associated with the first name on the resume	white black
18. gender	Inferred gender associated with the first name on the resume	f m
19. years_college	Years of college education listed on the resume	0 1 2 3 4
20. college_degree	If the resume listed a college degree	0 = No 1 = Yes
21. honors	If the resume listed that the candidate has been awarded some honors	0 = No 1 = Yes
22. worked_during_school	If the resume listed working while in school	0 = No 1 = Yes
23. years_experience	Years of experience listed on the resume	Numeric (1-44)
24. computer_skills	If computer skills were listed on the resume.These skills were adapted	0 = No 1 = Yes

	for listings, though the skills were assigned independently of other details on the resume.	
25. special_skills	If any special skills were listed on the resume	0 = No 1 = Yes
26. volunteer	If volunteering was listed on the resume	0 = No 1 = Yes
27. military	If military experience was listed on the resume	0 = No 1 = Yes
28. employment_holes	If there were holes in the person's employment history	0 = No 1 = Yes
29. has_email_address	If the resume lists an email address	0 = No 1 = Yes
30. resume_quality	Each resume was generally classified as either higher or lower quality	high low

III. Preprocessing

After choosing our dataset and looking at it closer, we realized we chose to work with a heavily numeric dataset that also had various missing values. We decided to start with the spreadsheet version of the data to deal with some discretizing and normalizing the attributes. We also needed to set the class attribute inside the Excel sheet itself as originally it was detecting resume quality as the class attribute but our main focus is if they got a callback from their resume. We moved the callback attribute to the end of the Excel file to declare that as the class attribute. We also decided to delete the job_id attribute. This was specific to what ad the job was found in and had no predictiveness. We assume that there is not an effect on the callback based on which ad the job had been found in.

From there, we had to deal with our most confusing attribute which was job_req_min_experience as this attribute not only had missing values, but had numeric values with nominal values. We made this attribute nominal by discretizing the numeric values around

some information we were already given. We decided to change these manually. All the instances with “some” we kept as “some”. If the value was less than or equal to 2, we assigned to “some” as well. Anything between 2 and 5 was assigned to “moderate” and anything greater than 5 was “alot”. Then anything that was 0 or blank we defaulted to “None”.

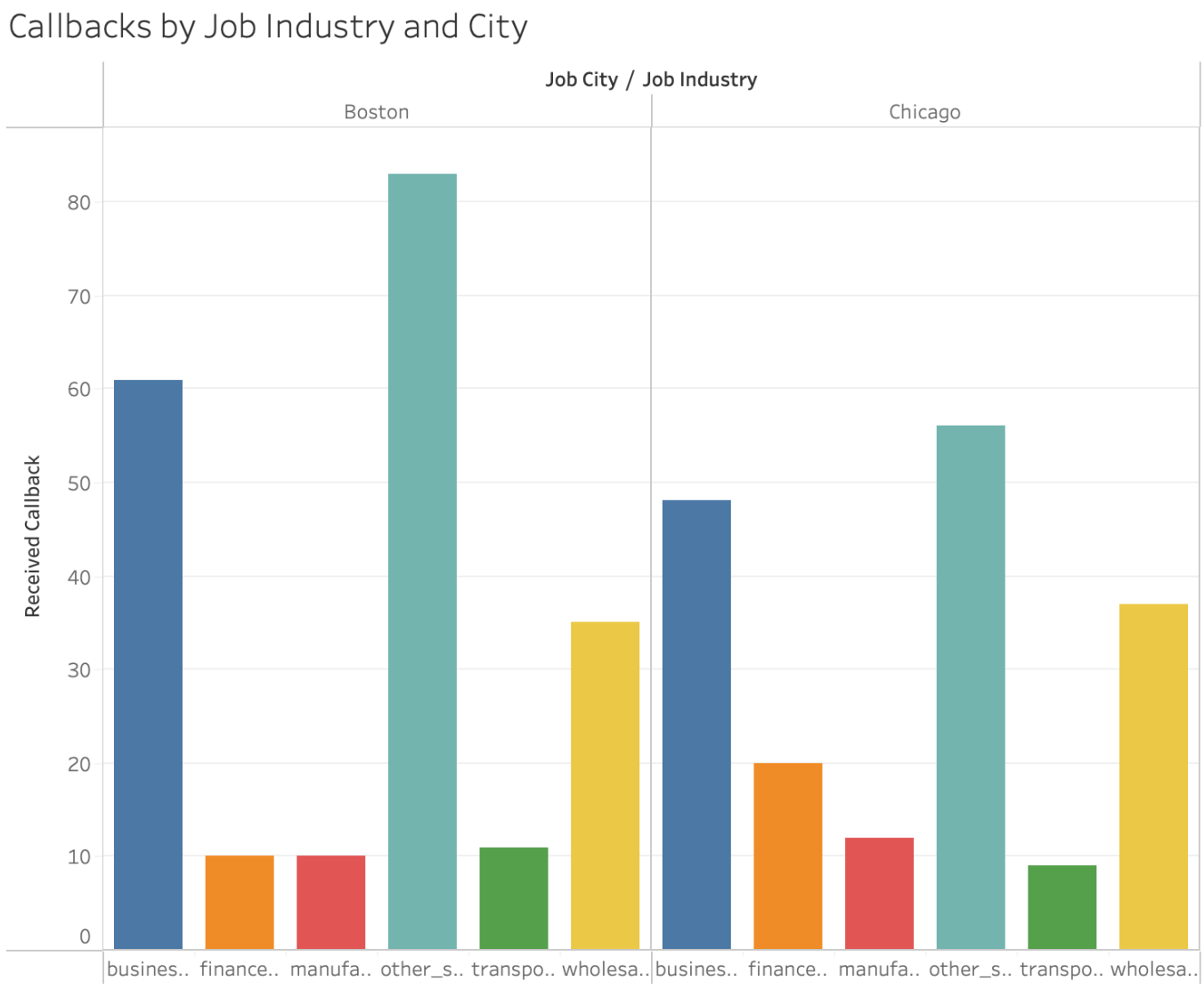
After this, we uploaded the dataset into weka. From there we used a Numeric to Nominal filter to make all the attributes easier to work with in terms of association learning. We did not need to do this for classification but we went ahead and did all attributes so we had it prepped and ready for association learning. The biggest reason for this with classification would have been for the 0 and 1 heavy attributes. There were various attributes with 0 and 1, which we read as yes or no in terms of the attributes they were associated with. Because of this, we knew we wanted to make it nominal where 0 and 1 were their own values that were understood as a “yes” or a “no” rather than number. After discretizing these, we were left with almost all nominal dataset except for one numeric attribute which was years_experience. We then used weka to make this nominal as well. It discretized the years_experience into 3 buckets of (-infinity,9), (9,18), and (18, infinity).

To finalize the dataset, we had to split it up into training and testing data. For this, we decided our easiest split was going to be based on the city the job was in. We split the data so each dataset only consisted of one city which would be either Chicago or Boston. Chicago would be used as our training data to make and build the models while Boston is going to be our testing data that we will use on the models created from the Chicago dataset. An important reason for doing it this way as one goal of this project is to predict if there is a consistent trend of what good resumes are or if it is dependent on the city. By splitting the data like this, as we test Boston on Chicago we can see if Boston can be predicted and if not we know there is some attribute that affects job callbacks and resumes in different locations.

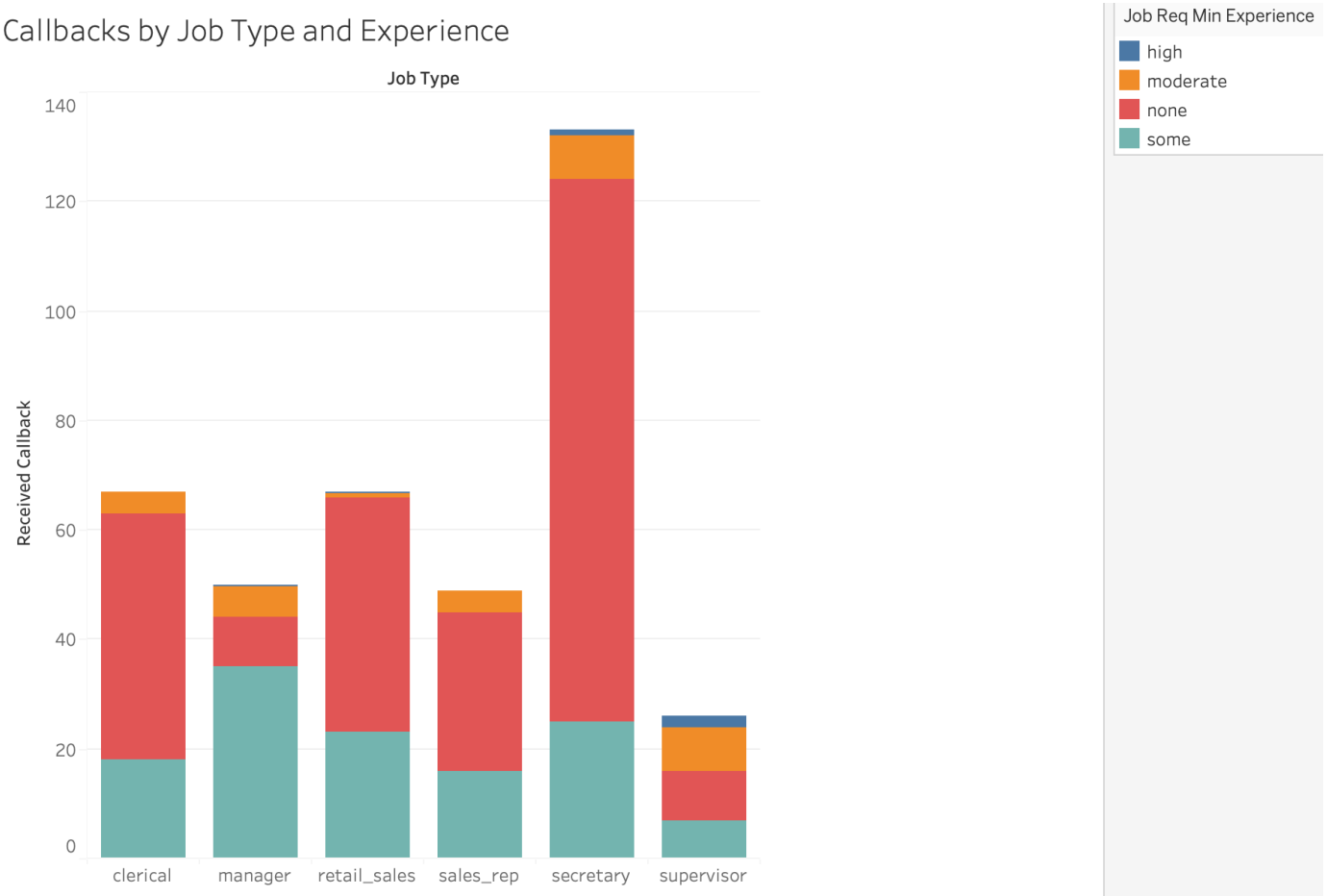
IV. Visualization

To make our data a little bit easier to see, we uploaded our cleaned and discretized data set into Tableau to create visualizations. After creating several graphs and charts with different combinations of attributes, the most informative visualizations appear below. The graph below called “Callbacks by Job Industry and City” displays the job industries that applicants received

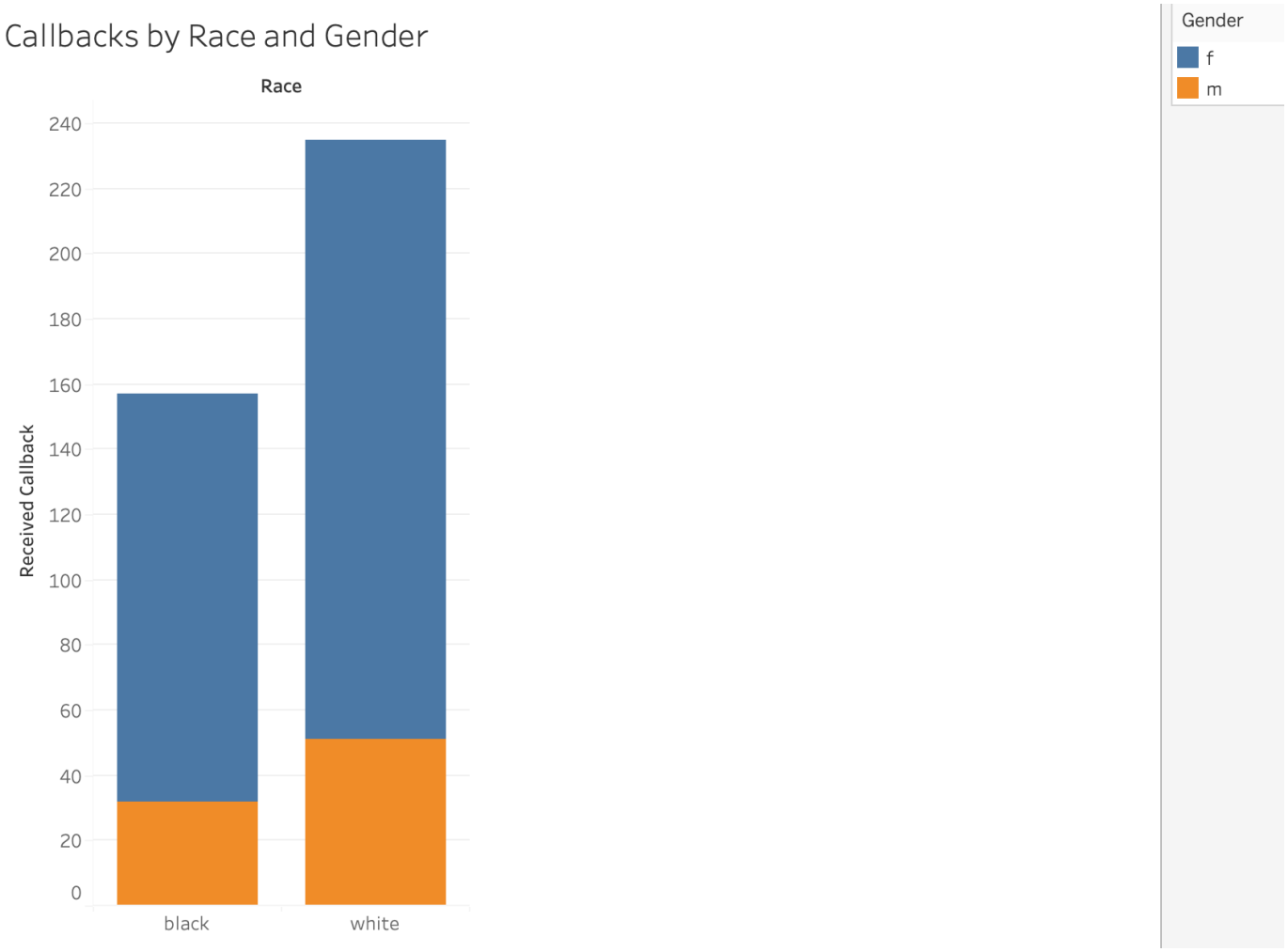
the most callbacks from, split up by city between Boston and Chicago. For both cities, we clearly see that applicants received the most callbacks from the industry labeled “other_service,” followed by “business_and_personal_service” and “wholesale_and_retail_trade.” Within the last three industries, there is more variance between the two cities. Although it is not quite clear what the attribute “other_service” entails, it can reasonably be inferred that service industries, such as restaurants, are lumped into this category. It makes sense that service industries would call back most often because they often require little experience or education and, most of the time, are always looking for help.

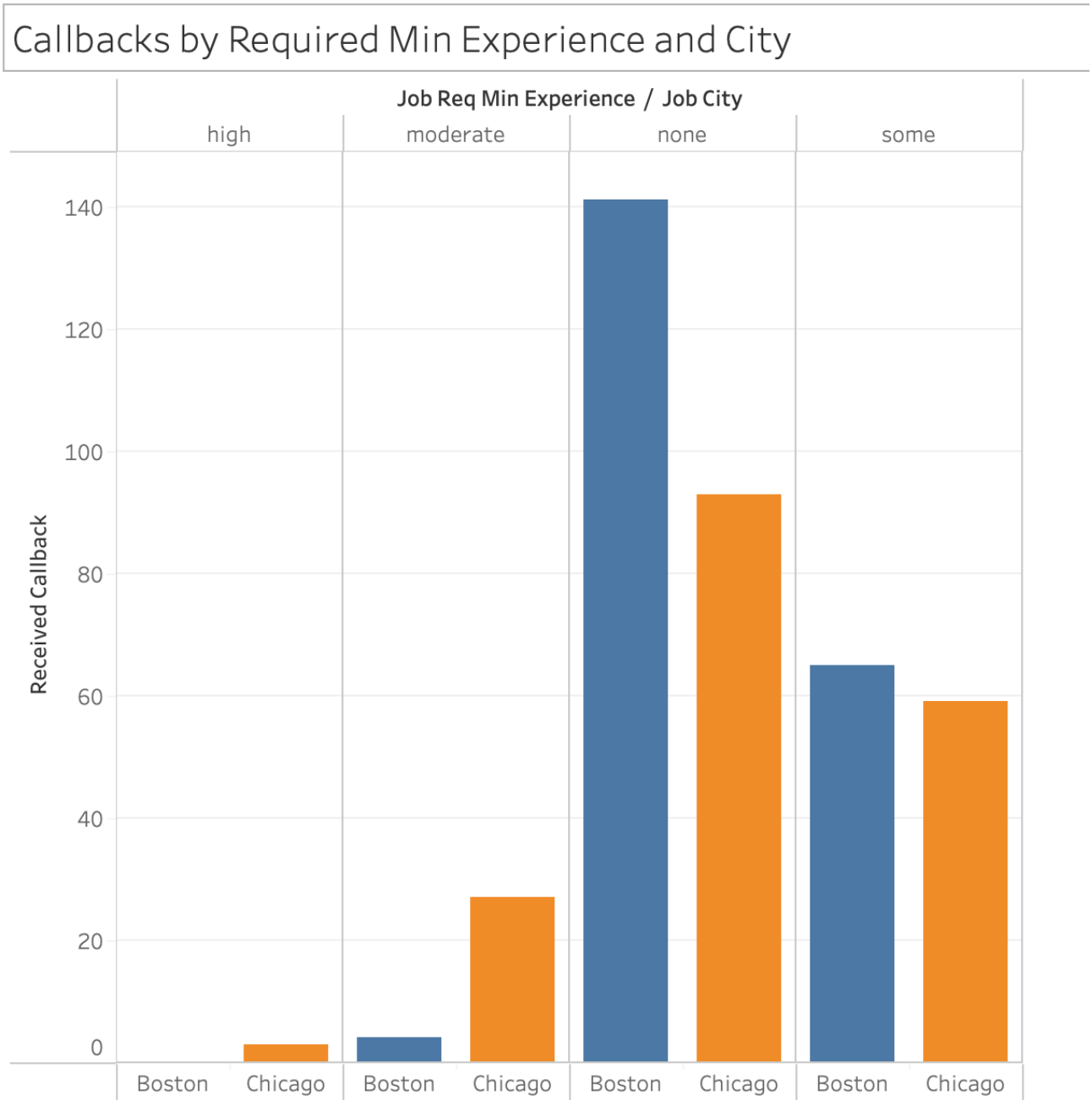


The next graph, “Callbacks by Job Type and Experience,” highlights the amount of callbacks received based on the type of job and filtered by the minimum experience required by that job. This bar chart was not separated by city. The job for which the most callbacks were received was a secretary position. As shown by the color filter, most secretary positions required no experience. This was true across the other job types except for managers and supervisors. Most of those types of positions required at least some experience, with supervisors needing more moderate experience. This may infer that experience is a highly sought after quality on a resume, especially for more complex, higher-paying jobs, and lack of experienced applicants contributes to the lack of callbacks from that job. It is interesting to see that the job with the most callbacks was a secretary. As seen in the following bar chart, there are far more females that received callbacks than males, which, stereotypically, may be correlated to the types of jobs with the most callbacks.



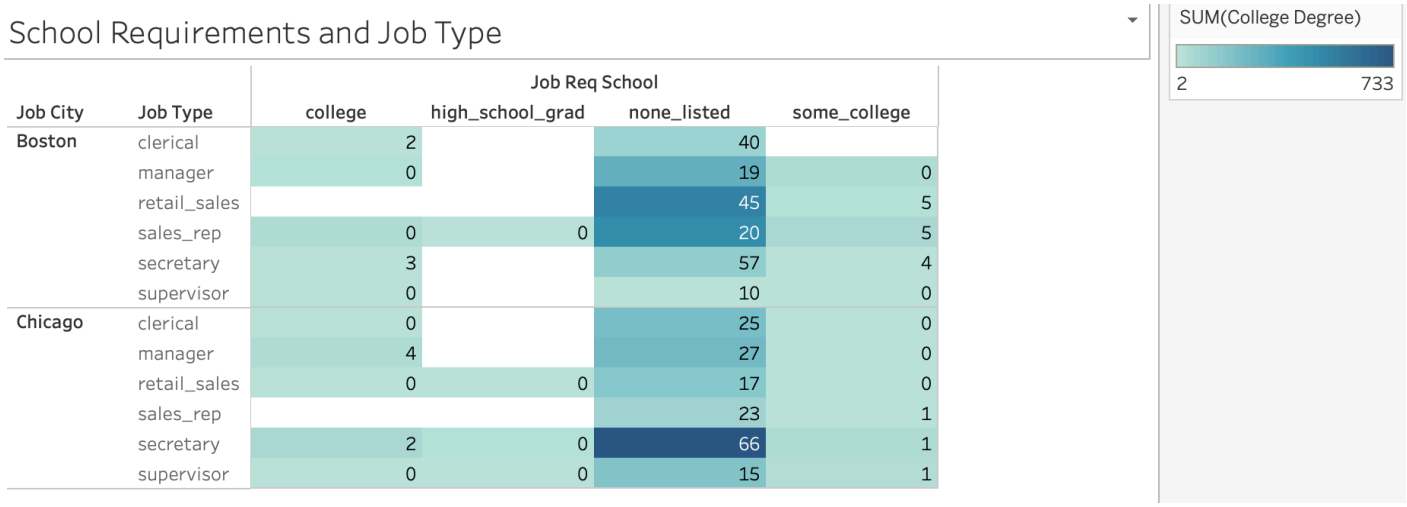
As previously pointed out, females receive significantly more callbacks than males, despite there being more males in the dataset than females. When comparing race, there are more white applicants receiving callbacks than black applicants. Seeing as how there are far less instances in the dataset reported as black than white, however, this is not surprising.





The above chart displays the sum of callbacks received based on minimum experience required and split by city. Most jobs that called the applicant back required no experience, which is not shocking considering that it is easier to fill positions that do not require experience. Boston seems to have significantly more callbacks for jobs with no experience than in Chicago. As the experience requirements increase, however, we see an increase in Chicago's sum of callbacks, which may indicate that different cities have different job needs and, therefore, different required experience levels.

Next, we explored the relationship between education requirements and type of job in relation to the amount of callbacks received by applicants and the sum of college degrees held by applicants. The amount of callbacks is represented by the number inside of each box in the graph, and the amount of college degrees is represented on a color scale from light to dark blue. The graph clearly shows that the jobs with the most callbacks did not have any educational requirements listed and jobs with college requirements had very few callbacks. It is interesting to see how jobs often required either no education or at least some college education, and jobs that required only a high school degree had no callbacks. Even more interesting is how applicants who did receive a college degree still received the most callbacks from jobs where it was not a requirement. It is evident that even with a college degree fulfilling the job's education requirement, it is difficult to receive a callback from a competitive job. This may reasonably indicate a college degree is not one of the most important factors that makes an applicant stand out when trying to secure a job in a tough, competitive climate.



		Job Req School			
Job City	Job Type	college	high_school_grad	none_listed	some_college
Boston	clerical	2		40	
	manager	0		19	0
	retail_sales			45	5
	sales_rep	0	0	20	5
	secretary	3		57	4
	supervisor	0		10	0
Chicago	clerical	0		25	0
	manager	4		27	0
	retail_sales	0	0	17	0
	sales_rep			23	1
	secretary	2	0	66	1
	supervisor	0	0	15	1

V. Analysis and Results

Our class attribute, *received_callback*, for the Chicago training set has 2521 instances for the value 0 (no) and only 182 instances for the value 1 (yes). This means the data is unbalanced because we have an uneven distribution of class attribute values. As a result, we are more likely to predict the majority class, which is that the applicant did not receive a callback. However, predicting the majority outcome rarely says anything interesting about the data. For each classification learning algorithm, we used different filters to tackle this issue and see if we can get a better model. The cost sensitive classifier lets us adjust the cost, or penalty, of making an incorrect prediction in the evaluation phase, and SMOTE extrapolates a "nearest neighbor" instance of the same class value to oversample as many new instances as you request. Again, each model's accuracy tells us how the model performs on the Boston dataset.

a. OneR

The OneR algorithm produces a model based on one input attribute, and so it is considered a baseline strategy. It does not work well if many of the input values have fewer possible values than the output attribute does, but we do not have this issue with our dataset. In addition, the most predictive attribute determined by OneR is not always the one preferred by a decision tree algorithm.

Filter	**Accuracy**	**Total Instances**	**Instances of Unbalanced Value**
None	90.3047%	2703	182
CostSensitiveClassifier (penalty = 15)	43.952%	2703	182
CostSensitiveClassifier (penalty = 10)	58.0794%	2703	182
CostSensitiveClassifier (penalty = 7)	65.097%	2703	182
SMOTE	65.097%	3977	1456
SMOTE	42.0129%	5433	2912

Even though the model with no filter has the highest accuracy by far, it is important to note that it does not correctly classify any instances of the unbalanced value. So the no filter model is not our best model because we are not necessarily interested in predicting the majority class. The cost sensitive classifier with the penalty increased to 7 gives us a decent model with an accuracy of 65.097%. The SMOTE filter that uses 3977 instances interestingly gives us the same accuracy of 65.097%.

b. Naive Bayes Simple

Naive Bayes is different from most algorithms because it does not produce rules. Instead, it can determine the probability of the class given an instance. This algorithm is under the assumption that the evidence or attributes are split into parts that are independent. We wanted to explore this algorithm because it can be more applicable to real world use. If a person gives us their resume information, then we can tell them the odds that they will receive a callback. The applicant might be more interested in this type of information rather than a model giving them a yes or no output.

Filter	Accuracy	Total Instances	Instances of Unbalanced Value
None	76.1773%	2703	182
CostSensitiveClassifier (penalty = 20)	89.4275%	2703	182
CostSensitiveClassifier (penalty = 15)	88.735%	2703	182
CostSensitiveClassifier (penalty = 10)	87.627%	2703	182
CostSensitiveClassifier (penalty = 7)	86.1496%	2703	182

Here, the models' accuracies increase as we increase the penalty for the cost sensitive classifier. The filter with the highest penalty produces the highest accuracy, even over the no filter model. However, we might want to be careful with using a model with a very high penalty because that might skew our results too much. The models with a penalty of 7 and 10 give the accuracies of 86.1496% and 87.627%, which may be good if it can give us an accurate probability of not getting the majority class value.

c. J48

The J48 algorithm creates a decision tree based on rules that allow us to classify different instances. Taking the attribute with the greatest amount of information, this is what the tree uses for the split to get the greatest classification accuracy. Most of the splits for each branch come from the highest normalized information gain that the attribute can give. We wanted to explore this algorithm as we knew it would create a good model for us to see if one city can create a model that would accurately predict the other as well as the attributes that allow for this to happen. This helps us to see patterns in the attributes as well as give us an idea of which attributes provide the most information.

Filter	**Accuracy**	**Total Instances**	**Instances of Unbalanced Value**
None	90.3047%	2703	182
CostSensitiveClassifier (penalty = 15)	67.9594%	2703	182
CostSensitiveClassifier (penalty = 10)	68.2852%	2703	182
CostSensitiveClassifier (penalty = 7)	69.7599%	2703	182
SMOTE	78.7165%	3977	1456

While the model with no filter had a significantly higher accuracy, this was no use to us. Because of the unbalanced data, it appears the one branch of the tree was simply taking the ratio of training to testing instances which gave us such high accuracy that no other branches were produced. The other tests gave us trees that were very detailed allowing it to be too small to see which attributes were used where. However, this test does allow us to consider how well the predictions carry from city to city. As the accuracies of the models are fairly low for what we would like to see, it gives us the impression that the predictive capabilities of a tree for Chicago may not work as well on Boston. This brings into question if there is variability in attributes from city to city on what can predict that callback.

d. Apriori

The apriori algorithm is used for association learning. This is an algorithm that takes item sets from datasets and makes association rules that meet a minimum support and confidence threshold. We decided to explore with Apriori as we wanted to look into if any attributes have similar predictive qualities and produce similar outputs. Starting with a minimum confidence of 90%, taking the top 10 rules, there was nothing we were looking for. As we continued testing, we found most associations came from attributes within the table but none that led to what we were looking for. After going through similar tests, changing the confidence, looking at top 50, top 100 rules, we still were not finding anything. Most of the associations we found came from rules made of smaller itemsets and were reversed of each other. For example, job_req_school =None ⇒ job_req_education = 0. This rule was top for every test and the reverse was the second rule for every test as well. These rules both are rules where the attributes tell us the same thing as one is saying no school is required and the other is saying there is no lowest level of school required. Many other rules like this were the only ones produced, therefore we decided the apriori algorithm was not going to be as useful as we thought.

e. RandomForest

RandomForest is a type of ensemble machine learning that creates a massive web of trees and combines the results of these trees to provide the best possible answer to a question. We decided to employ RandomForest because, as a type of ensemble learning, it combines multiple models to create the best predictions. When initially running the unbalanced data through, we get an accuracy of 88.458%, which is a decent accuracy percentage but still has unbalanced data, so we applied some filters. We then used the cost sensitive classifier to increase the penalty for not correctly classifying “b.” We increased the penalty by five every time we ran the filter. Doing this increased the accuracy to 90.3047%, but even after changing the penalty to 10 and 15, the accuracy remained the same. We then applied the SMOTE filter to increase the number of “b” instances, and by doing this we decreased the overall accuracy percentage to 86.4728%, but even though the overall accuracy decreased the data set was more balanced.

Filter	Accuracy	Total Instances	Instances of Unbalanced Value
None	88.458%	2703	182
CostSensitiveClassifier (penalty = 5)	90.3047%	2703	182
CostSensitiveClassifier (penalty = 10)	90.3047%	2703	182
CostSensitiveClassifier (penalty = 15)	90.3047%	2703	182
SMOTE	86.4728%	3977	1456

f. Bagging

Bagging is a different type of ensemble machine learning. The way it works is that it uses several training sets and builds the same model for each one. Then, it votes for the nominal values or takes the average of the numeric values. This type of algorithm is more complex than OneR and J48 because it is considered a black box algorithm. In other words, we cannot see the inner workings of the algorithm. This meta learner is more likely to produce a more accurate model, but it also forces us to only focus on the input and output which is not necessarily bad if an applicant gives us their resume and is only looking for what kind of outcome they are likely to get. The bagging algorithm was first run with no filter, and we used OneR as a baseline. Then we applied the SMOTE filter and tried different types of trees.

Filter	Accuracy	Total Instances	Instances of Unbalanced Value
None & OneR	90.3047%	2703	182
SMOTE & OneR	65.097%	3977	1456
SMOTE & J48	83.4257%	3977	1456

SMOTE & Random Forest	86.3804%	3977	1456
SMOTE & REPTree	81.2096%	3977	1456

Like we saw in earlier models, the no filter produced a very high accuracy, but at the same time, it did not correctly classify any of the instances of the unbalanced value. Thus, the first model is actually bad. When we used SMOTE with OneR, our accuracy decreased to 65.097%, which is a huge jump. This model is also not ideal because it only picks one predictive attribute, so it makes sense that the accuracy went down. However, using SMOTE with different types of tree algorithms produced much better results. As a result, mixing complex algorithms together can produce more accurate models that are capable of predicting both classes. Our only sacrifice is that we cannot understand or interpret any rules or how the algorithm gets a certain output.

VI. Conclusion

Based on the trends of accuracy resulting from each of the algorithms, we have reason to believe that the qualities that determine whether an applicant gets a callback is shared between both Chicago and Boston. Even though the algorithms that were run with no filter to address the unbalanced data consistently produced the highest accuracies, they did not correctly classify any instances of the unbalanced class. A model that cannot classify all possible cases is considered a bad model, so the fact that these models gave accuracies above 90%, this percentage is misleading. Because the Chicago dataset had so many more instances than the Boston dataset, we did not even attempt to use Boston data as training data because it was so heavily unbalanced. To correct this unbalanced data, we employed the cost sensitive classifier and SMOTE to create a more reliable and accurate model. After applying these filters, the accuracy significantly decreased on certain algorithms like J48 and OneR. However, with some manipulation of the filters, we were ultimately able to raise those accuracies for, making our models more reliable and informative.

When looking at attributes that are most predictive, the level of experience an applicant has makes them more competitive. When looking at education, there were many instances of applicants with college degrees only receiving callbacks from jobs with less overall requirements and skill. Jobs that did require college degrees did not have many callbacks, even to applicants fulfilling this requirement, indicating the competitive nature of those jobs and how holding a

college degree is not necessarily the golden ticket to landing a callback. There were also many of the same trends observed across both cities, with peaks in the data often correlating between the two cities. All in all, we accomplished our goals of balancing the data to build more reliable and accurate models and finding resume qualities across cities that make an applicant stand out.

Data Source:

- https://www.kaggle.com/datasets/utkarshx27/which-resume-attributes-drive-job-callbacks?select=resume.csv