Classifying Data Science Jobs by Skill Requirements

Griffin Brookshire
glbrook2@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Jake Nowokunski
jbnowoku@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Tyrone Wu
tkwu@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Laxmi Aishwarya Thela
lthela@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

1 INTRODUCTION & BACKGROUND

1.1 Problem Statement

When searching for a job in the data science field, it is difficult to determine which skills are most sought after for a given job title. In this project, we will examine data that has been scraped from the job site Indeed and use it to identify distinct data science jobs and the skills that correspond to those jobs. First, we will formulate a way to preprocess the dataset to identify the skills that are requested in each job description. This will require using NLP to search for the list of necessary skills within the job descriptions. The dataset has a wide variety of job titles, and titles may vary for similar positions across different companies. Thus, our next step will be to cluster the jobs by their requirements to determine distinct job classes within the dataset. Last, we will identify the skills that are most important to each job cluster.

1.2 Related Work

1.2.1 Related Work 1. The research paper by Ibrahim et al. [1] compares many approaches to classifying job postings as either real or fake based on their description. The purpose of the research was to identify which classification method was the best in dealing with the given text classification problem. The authors concluded that Random Forest was most suitable, which will be very useful for us to know when creating our own models.

1.2.2 Related Work 2. The paper by Lakshmi et al. [2] performs a comparative study of modified versions of the k-Modes clustering algorithm. The k-Modes clustering algorithm is based on the k-Means algorithm, except it is much better suited for categorical variables. This makes k-Modes a good option for clustering based on the categorical attributes we extract. One of the drawbacks of k-Modes that is discussed in this paper is its tendency to find local optimums instead of global optimums. This is something we must consider when using this algorithm.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.2.3 Related Work 3. The idea presented in the research paper by Sharma [3] is to use a LSTM deep learning model along with word embeddings to extract relevant skills from job postings. The model can also identify emerging skills in the associated field. Part of the process used in this paper involves using NLP to identify certain parts of speech within a text corpus. We can use this approach to analyze job titles within a cluster using noun chunks.

2 PROPOSED METHOD

In this section, we describe our experimental approach as well as the rationale behind it.

2.1 Approach

2.1.1 Data Preprocessing. Our dataset contains raw data about several job postings, including job titles, companies, descriptions, reviews, and locations. For our experiments, we focus primarily on the job titles and descriptions, so the remaining features are discarded. The job titles and descriptions already have the HTML tags removed, but require further cleaning to remove unnecessary symbols, punctuation, and control characters. These are removed using the "re" package, which provides access to the use of regular expressions for removing unwanted characters. The job descriptions are also transformed to have all characters be in lowercase.

After the job titles and descriptions have been cleaned, they are then tokenized to divide them into arrays containing each word in the string. We then remove stopwords from each array, which are common words that do not add significant meaning or context, such as "the", "as", or "of". Tokenization is performed using the "nltk" package, and the list of English stopwords is provided by "nltk" as well.

Next, we extract new features from each of the job descriptions. Each of the new features corresponds to a skill or qualification from the list described in Section 3.3.1. These features are categorical and binary, and describe whether or not the skill is present in the job description. Entries that contain none of the skills in the list are removed from the dataset.

2.1.2 Clustering. Using our new extracted features, we cluster the job postings to identify job postings with similar skill requirements. We use KModes from the "kmodes" package to perform clustering on the data, with the "Huang" method for initialization. We also perform hyperparameter tuning to determine the best number of clusters to create. The goal of our hyperparameter tuning is to determine a number of clusters that classifies the data into distinct

groups. The number of clusters should be as small as possible to avoid overfitting.

Once the data has been clustered, we can process the job titles to find the most frequent words and phrases within each cluster. We use the "spaCy" package to split the titles into chunks, and find the most frequent chunks in each cluster. We can also begin to analyze the importance of each feature within each cluster, as we describe next.

2.1.3 Analysis of Feature Importance. To analyze the importance of each skill or qualification within each cluster, we employ the following methods:

(1) Skill Frequency
We perform a simple frequency count of the number of entries within the cluster that contain each skill, and order the skill frequencies from highest to lowest.
(2) Random Forest Classifier
We use Random Forest Classifiers from the "sklearn" library to determine the relative importance of each skill for classifying entries as a part of each cluster. Rather than using a single Random Forest Classifier, we construct a Random Forest Classifier for each cluster. For each model, we use a one-versus-all approach where the input data is the skill features for each entry and the target values are whether or not each entry is within the cluster. After the models are fit to training data, we use "sklearn" to calculate the permutation importance based on testing data. We use these permutation importance values to determine the most important features for each cluster.
(3) Naive Bayes for Categorical Features
We use Naive Bayes for categorical features from the "sklearn" library to determine the importance of each skill within each cluster Similar to our method for the Random Forest Classifier, we create a Naive Bayes classifier for each cluster, where the input data is the skill features for each entry and the target values are whether or not each entry is within the cluster. After the models are fit with training data, we calculate the permutation importance for each feature using testing data and use these values to determine the most important features for each cluster.

2.2 Rationale

2.2.1 Data Preprocessing. We chose to use a predetermined list of skills to extract our new features. We chose to do this to avoid having to discern what is or is not a skill within a job description. Also, as this is a "data scientist" job dataset, many of the job descriptions will likely contain phrases like "machine learning", "natural language processing", and "data mining", as well as other phrases that are typical of job descriptions like "communication skills" and "bachelor's degree". By using a predetermined list of skills, we are able to better differentiate the jobs by the technical skills and technologies that are used at each job.

2.2.2 Clustering. One of our project's goals is to determine distinct groups of jobs based on skill requirements listed in job descriptions. However, we are not using the job titles as the target attribute for the dataset, and are instead using clustering to determine how to classify each entry. We chose to do this because different companies may use the same names for job titles, even if the requirements for the jobs are completely different. Similarly, different companies may use different titles for positions with the same requirements. Also, as the dataset is a collection of "data scientist" job entries, many of the titles will likely be similar. Thus, job titles are not a useful or accurate target attribute for classifying the data.

We decided to use KModes as our clustering algorithm due to our extracted features being entirely binary categorical variables. The KModes algorithm performs computations using the number of matching dissimilaries, which is simply the number of attributes that are different. This works well for our extracted features because they can only be one of two values: 0 or 1. Other clustering algorithms, such as KMeans, are more appropriate for attributes with continuous values.

Although we discussed how job titles are not a good target attribute in Section 2.2.1, we did still analyze the job titles within each cluster to see if any distinct patterns emerge. We use the "spaCy" package to do this because it can split the job titles into chunks of nouns, and we can use a simple frequency count to find the most common chunks in the titles within each cluster. This gives us an idea of the types of job titles that are within each cluster.

2.2.3 Analysis of Feature Importance. Part of our Problem Statement involves identifying the skills that are most important to each job cluster. We do this in order to determine the skillsets that make each job cluster distinct.

We chose three different methods of analyzing the importance of features: Skill Frequency, Random Forest Classifiers, and Naive Bayes for Categorical Features.

(1) Skill Frequency
Skill frequency within each class is the simplest approach we use, and should set a baseline to compare our other approaches to. Identifying the most frequent skills that appear in each cluster gives us an idea about the skills that define that cluster.
(2) Random Forest and Naive Bayes Classifiers
Since our extracted features are entirely binary and categorical, we cannot use methods like Linear Regression that are meant for continuous attributes. Random Forest Classifiers and Naive Bayes Classifiers are both classifiers that can work well with categorical data. For both classifiers, we use a one-versus-all approach to build models of each classifier for each cluster. This allows us to determine the features that are important within each cluster, rather than the features that are important to distinguishing between the clusters. By fitting each model with training data and computing the permutation importance of each feature using test data, we can determine the features that are most significant for each cluster. Permutation importance measures how shuffling the values of each attribute affects the accuracy of the model. This process works well with categorical data and makes permutation importance a good metric for measuring the importance of each skill within each cluster.

3 PLAN & EXPERIMENT

3.1 Dataset

The US Data Scientist Job Market dataset from Kaggle contains 6,964 instances of data science job postings that were web scraped from the Indeed website. The data includes CSV files of job postings based on city/region; however, for our report, we will be using the dataset that contains all the instances.

As discussed earlier, each instance contains information about the company name, the position name, the location, the job description, and the number of reviews for the job. The attribute that we will be focusing on is the job description, which contains the desired skills and qualifications of a job, as well as irrelevant information, such as a company's mission statement. Upon further inspection of the job descriptions, some instances do not include any description of desired skills/qualifications. Relevant features in the job description must first be extracted, in which we will focus on specific words that make up the skills of a job posting. Extraction of these features are explained later in the report.

3.2 Hypotheses

Our primary hypothesis is determining whether the desired skills in a job description can be used to cluster distinct job classes. Specific job classes, such as Data Engineer, require distinct skills. Therefore, we theorize that these skills, that are reflected in the job descriptions, can be used to cluster job positions/classes.

In addition, we will identify the skills that are most important to each job cluster to get a better understanding of which skills to look out for in the Data Science job market.

3.3 Experimental Design

To extract the relevant features for our model, a series of steps must be performed in order to obtain the desired skills/qualifications from the job description. The process that we followed for this report is included here:

3.3.1 Feature Extraction. As mentioned above, the job description attribute contains a lot of irrelevant information and that must be filtered in order to obtain the desired features. To do this, we will incorporate domain knowledge to pick out certain key words that appear in the job description. This way, we can filter out irrelevant information and gather the desired skills/qualifications from the job description.

- A total of 53 skills have been compiled into the following set:

  {'.net', 'ajax', 'aws', 'azure', 'c', 'caffe', 'cassandra', 'css', 'docker', 'dynamodb', 'elasticsearch', 'excel', 'git', 'hadoop', 'haskell', 'hbase', 'hive', 'html', 'java', 'javascript', 'julia', 'keras', 'matlab', 'mongodb', 'mysql', 'neo4j', 'nosql', 'octave', 'oracle', 'postgis', 'postgresql', 'powershell', 'python', 'pytorch', 'qlikview', 'r', 'rapidminer', 'ruby', 'sas', 'scala', 'spark', 'splunk', 'spss', 'sql', 'ssh', 'stata', 'swift', 'tableau', 'tensorflow', 'tomcat', 'typescript', 'xml', 'xquery'}

The text in the job description attribute and list of skills have been converted to lowercase for easier extraction.

When applying the list of skills to the job description attribute, we generate a vector representation of the skills in the job description by checking if each skill is contained within the job description. Instances that contain none of the skills in the list are removed from the dataset, and we are left with 4,858 remaining instances to work with in our model.

Figure 1: Frequencies of the Upper 27 Skills in the Dataset

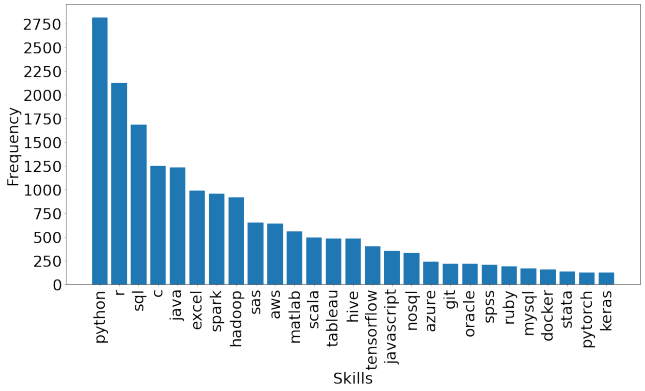
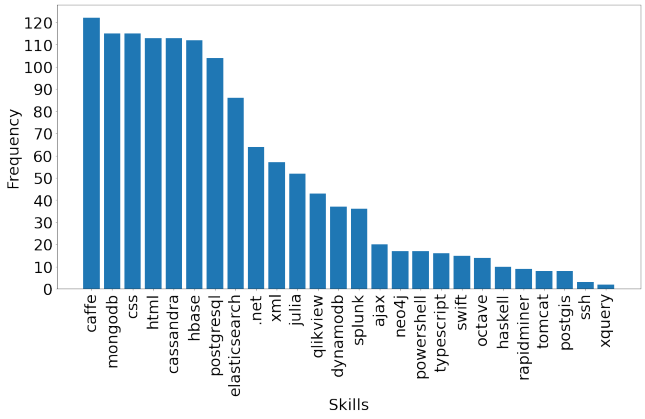


Figure 2: Frequencies of the Lower 26 Skills in the Dataset



The skills that are used to capture the features of the job description have been accumulated across the entire dataset and ordered from greatest to least in Figure 1 and Figure 2. We can see that the frequency of skills decrease in an exponential fashion. While some skills have a lot higher frequencies/occurrences than others in the dataset, all the frequencies are greater than 0. Our next step now is to apply our unsupervised clustering methods to our extracted features.

3.3.2 Clustering. As mentioned earlier, we are limited to vector representations of categorical data, which do not have class labels. This restricts us to unsupervised learning techniques, which were discussed in the proposed methods. KModes, our method of clustering, operates similarly to the kMeans algorithm but with a few minor differences. Instead of using distance to measure how far

apart one instance is from another, it uses *matching dissimilarities* (the total mismatches between attributes of two instances). Additionally, instead of mean, KModes uses *mode* to find and match clusters. One caveat of the KModes algorithm is that it is prone to finding local optima instead of global optima [2]. To combat this, we run the KModes algorithm several times and select the most optimal result. The number of runs is described by the *n_init* parameter below.

For reproducibility, the following fixed parameters shall be used:

- init = "Huang"
- n_init = 10
- random_state = 25

As we are only concerned about identifying and producing meaningful clusters from our data, there will not be any need for splitting the dataset into training and test sets during clustering.

3.3.3 Hyperparameter Tuning. Since we do not have job class labels in our dataset, we must determine the best number of clusters to split the data into. As mentioned in our rationale, choosing a small number of clusters would reduce the possibility of our model from overfitting. Our method for evaluating the number of clusters is through graphing the average dissimilarity against the number of clusters. For every n number of clusters we generate, we compute the average matching dissimilarity between instances in the dataset and the centroid of their cluster. This is repeated every time, from 1 cluster up to 14 clusters, and will total in 14 KModes models examined for tuning. Using the Elbow method, we can find a reasonable cutoff of where the optimal number of clusters appear to work well. The results of our tuning will be later shown in the next section.

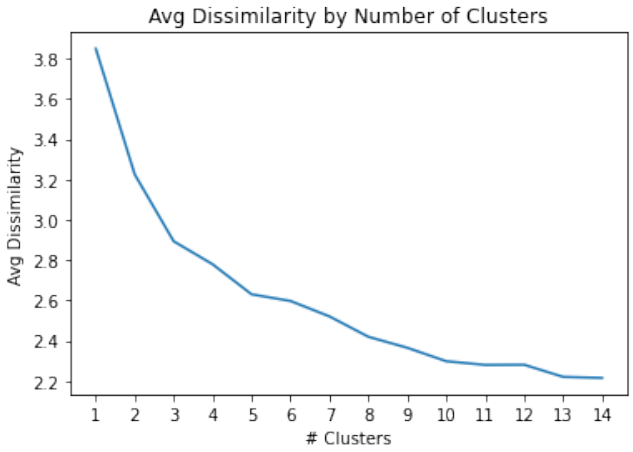
3.3.4 Job Titles for Each Cluster. While job titles in the dataset are not optimal to use as class labels, we can still try to use them to differentiate between the clusters. To do this, we need to determine the appropriate job titles for each specific skill set associated with a cluster. We use the "spaCy" package to separate the job titles into noun chunks, and then identify the words and phrases that most accurately describe each cluster. Using these words/phrases, we can compare clusters and see whether or not these titles are distinct and meaningful for dividing the clusters.

3.3.5 Feature Importance Within Each Cluster. To determine the important skill set within each cluster, we use skill frequencies as well as the permutation importance for Naive Bayes and Random Forest classifiers.

(1) Skill Frequencies
We perform a simple frequency count of skills within each cluster. The skills will be ordered from most to least frequent within each cluster. The results of this process should give us a good baseline to compare the results of the classifiers to.
(2) Random Forest Classifier
Random Forest Classifiers are trained for each cluster, using a "one-versus-many" approach. Each model has a max depth of 10 and a random state of 25 for reproducibility. All other parameters have the default values assigned to them by "sklearn". Each model is trained on training data consisting of 70% of the dataset. The target attribute describes whether or not the job entry is within the given cluster. After each model is fit, we calculate the permutation importance of each feature using the test data (30% of the dataset), a random state of 25 for reproducibility, and 5 repeats. The resulting permutation importance scores are ordered from highest to lowest.
(3) Naive Bayes Classifier
Similar to the Random Forest Classifiers, a model is trained for each cluster. Each model has the default alpha value of 1 and is trained on the 70% of the dataset that has been designated as training data. The target attribute describes whether or not the job entry is within the given cluster. After each model is fit, we calculate the permutation importance of each feature using the remaining data as test data, a random state of 25 for reproducibility, and 5 repeats. The resulting permutation importance scores are ordered from highest to lowest.

In the next section, we will compare the results from each of these methods to determine the most important skills within each cluster.

Figure 3: Average Dissimilarity by Number of Clusters



4 RESULTS

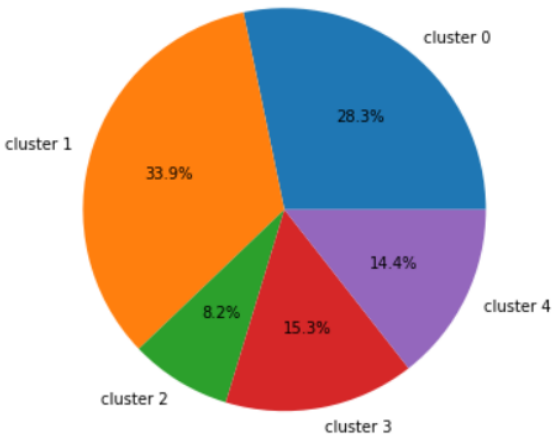
4.1 Cluster Size Selection

The plot in Figure 3 graphs the average dissimilarity against the number of clusters for the 14 models. As the number of clusters increases, the average dissimilarity approaches zero. We use the Elbow method to select an amount of clusters such that increasing the amount of clusters no longer has a meaningful improvement in average dissimilarity. We can see that 5 clusters appears to perform reasonably well without splitting the data too much. After 5 clusters, there does not appear to be meaningful information gain with additional clusters added.

4.2 Clustering Distribution of Dataset

The pie chart in Figure 4 shows the proportion of the data in each cluster. Cluster 1 is the largest with about 1/3 of the data, followed

Figure 4: Distribution of the 5 Clusters



by Clusters 0 with about 1/4 of the data, then followed by Cluster 3 and Cluster 4 with 1/6 of the data each, and finally Cluster 2 as the smallest cluster. Each cluster is large enough to gain insight into the properties of the data within the cluster.

One problem we noticed was that even when given a consistent integer value for the random state, the KModes implementation produces different results when run on different computers or on different dates. This may be a result of the random number generator in the implementation considering additional information beyond the random state seed. This problem may make it difficult to reproduce the clusters when our code is run again in a different environment. However, the distribution and results of the clustering are often similar, although the clusters may be labeled differently (ex. data that would be placed in Cluster 1 in one execution may be labeled as Cluster 0 in another). For consistency, we used the same execution of the clustering to generate all charts and data shown in the remainder of the report.

4.3 Job Titles Within Clusters

We computed the top 5 most frequent noun chunks contained in the titles in each cluster. The top titles are displayed in Figure 5. From this data, we can see that many of the clusters contain the same job title as the most frequent: "Data Scientist." This makes sense for this dataset because this dataset primarily contains information about "Data Scientist" job postings.

While many of the titles in these clusters are similar, there are some noticeable differences. For example, job titles in Cluster 0 are frequently related to Machine Learning, and the most frequent job title in Cluster 3 is "Research Analyst."

These findings confirm our initial assumption that the job titles are poor class labels for the data. This is evidenced by different clusters having data entries with similar job titles despite being clustered into different skill sets. Had we attempted to train models using the job titles as class labels, we likely would have had a low accuracy score because many different sets of skills would be predicted by the model to be "Data Scientist."

Figure 5: Top 5 Job Titles in Each Cluster

```
cluster 0 (1373, 57)
Data Scientist               69
Sr                           57
Machine Learning             20
Machine Learning Engineer    18
Manager                      18
Name: title_join, dtype: int64

cluster 1 (1646, 57)
Data Scientist           298
Senior Data Scientist     79
Sr                        66
Data Engineer             29
Data Science              23
Name: title_join, dtype: int64

cluster 2 (396, 57)
Data Scientist               63
Data Engineer                30
Sr                           24
Senior Data Scientist        21
Machine Learning Engineer    17
Name: title_join, dtype: int64

cluster 3 (742, 57)
Research Analyst       30
Sr                     21
Scientist              16
Associate Scientist    11
Data Scientist         10
Name: title_join, dtype: int64

cluster 4 (701, 57)
Software Engineer                40
Sr                               36
Data Scientist                   27
Software Development Engineer    25
Machine Learning                 21
Name: title_join, dtype: int64
```

As a result of our findings, it is difficult to assign a specific job title in English to each cluster. The clusters are better defined by their most important skills, which we will discuss next.

4.4 Feature Importance Within Each Cluster

4.4.1 Skill Frequency. To analyze the importance of each skill within each cluster, we started by calculating the most frequent skills within each cluster. The results are shown in Figure 6.

There are some noticeable differences in the skills that are important to each cluster. Cluster 3 contains many entries that require "excel", while other clusters do not. Cluster 2 is the only cluster that frequently has data entries containing "hadoop" and "spark" as required skills. "SAS" and "tableau" are frequent skills in Cluster 1. Cluster 4 frequently has entries with "javascript", but "javascript" is not a frequent skill in any of the other clusters. Cluster 0 is the only cluster that has "tensorflow" as a frequent skill.

These results give us a decent understanding of which skills are important to each cluster, and can serve as a baseline to compare the results of our permutation importance experiments to.

4.4.2 Random Forest Classifier. By following the process outlined in Section 3.3.5, we created and trained five Random Forest Classifier models using 70% of the data as training data. We then computed the

Figure 6: Top 10 Skills in Each Cluster - Frequency

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
0	r	python	spark	excel	python
1	python	r	python	sql	c
2	sql	sql	hadoop	r	java
3	c	sas	java	sas	javascript
4	spark	spark	scala	c	aws
5	aws	tableau	r	tableau	r
6	sas	hadoop	sql	spss	matlab
7	hadoop	matlab	hive	python	hadoop
8	matlab	java	c	matlab	scala
9	tensorflow	c	aws	oracle	spark

Figure 7: Top 10 Skills in Each Cluster - Random Forest

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
0	excel	python	scala	excel	c
1	python	r	spark	python	java
2	r	sql	hadoop	r	r
3	sql	java	java	sql	python
4	java	c	python	java	sql
5	c	scala	hive	spark	spark
6	scala	hadoop	sql	scala	excel
7	tableau	spark	r	ruby	hadoop
8	hive	spss	matlab	c	scala
9	hadoop	nosql	javascript	sas	javascript

permutation importance of each feature using the remaining 30% of the data as test data. The most important features for each model are shown in Figure 7, ordered from greatest to least importance.

In these results, there are some skills that appear to be important for many of the clusters. "python" has a high importance for every cluster, and "r", "c", and "sql" frequently appear as important skills for many of the clusters. There are also some differences between the clusters, such as "hadoop", "hive", and "spark" being very important for Cluster 2 but not being very important for the other clusters. Also, "excel" is important to Cluster 0 and Cluster 3, but is much less important to the other clusters.

4.4.3 Naive Bayes Classifier. By following the process outlined in Section 3.3.5, we created and trained five Categorical Naive Bayes Classifier models using 70% of the data as training data. We then computed the permutation importance of each feature using the remaining 30% of the data as test data. The most important features for each model are shown in Figure 8, ordered from greatest to least importance.

Skills that appear to be important to many or all of the clusters are "spark," "r," "elasticsearch," and "python." Cluster 2 is the only cluster that places a high importance on "pytorch," and Cluster 4 is the only cluster that has a high importance on "java." Noticeably,

Figure 8: Top 10 Skills in Each Cluster - Naive Bayes

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
0	spark	spark	python	spark	c
1	r	python	r	r	java
2	scala	scala	elasticsearch	python	r
3	hadoop	hadoop	pytorch	scala	python
4	elasticsearch	elasticsearch	matlab	hadoop	sql
5	matlab	matlab	azure	matlab	spark
6	hbase	hbase	hbase	elasticsearch	excel
7	azure	azure	aws	aws	hadoop
8	aws	aws	spss	hbase	scala
9	tableau	spss	caffe	azure	javascript

Cluster 1 is the only cluster that does not have "r" in its top 3 most important skills.

4.4.4 Discussion. When examining the similarities within each cluster for all three methods of feature importance, the clusters appear to somewhat have a consensus for which skill is important in each cluster. To show the similarities among the three methods, the following section will display the intersection between the top skills of Frequency, Random Forest, and Naive Bayes.

- Cluster 0: {'r', 'hadoop'}
- Cluster 1: {'spark', 'hadoop', 'python'}
- Cluster 2: {'r', 'python'}
- Cluster 3: {'r', 'python'}
- Cluster 4: {'hadoop', 'scala', 'spark', 'r', 'c', 'java', 'javascript', 'python'}

The cluster with the highest intra-cluster similarity appears to be Cluster 4, where eight skills are present in all three methods. The methods only agree on two skills for Clusters 0, 2 and 3, while the methods agree on three skills for Cluster 1.

We expected that our feature importance methods would be largely in agreement in regards to which skills are important for each cluster. However, only Cluster 4 appears to display this result, while the other four clusters agree on very few skills. Looking at the inter-class similarity of the intersections, we suspect that the re-occurrence of "r" and "python" among the clusters is due to their high overall frequency in the dataset, as shown in Figure 1.

One reason that the important features determined by the Random Forest and Naive Bayes methods might differ from the Frequency method is the way feature importance is determined with the one-vs-many technique. When using the one-vs-many method, the permutation importance of features is determined by how important each skill is for classifying an entry into *either* class: "in the target cluster" or "out of the target cluster." As a result, some skills may have a high importance score even if that skill is not necessarily important to the target cluster. In other words, skills may have a high importance score using these methods because they are important for determining that entries *are not* in the target cluster, while the Frequency method focuses only on entries that *are* in the cluster.

4.5 Revisiting Hypotheses

Our primary hypothesis for our experiments was that the desired skills listed in a job description could be used to cluster distinct job classes. Our experiments were only partially able to prove this hypothesis. There were some noticeable differences between the job titles within each cluster, but in general the clusters contained many of the same job titles. Also, while some clusters had skills that were uniquely frequent or important to their cluster, many of the clusters had similar skills that were important, such as "python" and "r."

Our other hypothesis was that by clustering jobs by skills, we could identify sets of skills that prospective candidates might want to learn to get a job in the Data Science field. We partially succeeded at this as well. A few skills, such as "python" and "r," were deemed important in almost every cluster. This suggests that potential candidates may want to be well-versed in these skills to improve their marketability. Also, the skills "spark," "hadoop," and "scala" were important together in Cluster 2. This suggests that if a candidate is familiar with one of these skills, it may be beneficial to them to become familiar with the other two. However, most of the other clusters did not have similar unique groupings of skills.

5 CONCLUSIONS

Throughout our project, we learned a lot about clustering, feature extraction, and feature importance analysis. Next, we will discuss what we learned by examining the limitations of our approach and considering alternate approaches could be taken for this problem.

5.0.1 Limitations of Approach. One limit of our approach was the use of a predetermined set of skills to extract our new attributes. We used our own domain knowledge and a quick, manual scan of the job descriptions to determine a list of skills that we should be able to extract from the dataset. However, this list of skills may not be as applicable to other datasets of job descriptions as it was to this dataset. Also, if these experiments were to be run on a similar dataset in the future, the list of skills may not account for new skills and technologies that may emerge in the future job market.

Another limit of our approach was the extraction of only categorical binary attributes. Since the features we extracted are binary and categorical, we are limited in what we can do with our data and which models we can use. For example, Linear Regression and SVMs are ineffective when the data is categorical. This also prevents us from performing common data transformations, such as normalization or PCA, to help reduce noise within the data.

5.0.2 Alternate Approaches. Using a predetermined set of skills limited the number of skills we were able to identify within the job descriptions. In addition, it also limited our ability to apply our experiments to similar datasets in the future. To remedy this, we could have used a combination of LSTM and word embeddings to identify skills within job descriptions [3]. This method is much more flexible and is capable of extracting previously-unseen skills from the job descriptions. This method could also identify non-technical skills or requirements, such as "Master's Degree" or "Civil Engineering."

One approach we could have used to analyze skills within clusters is Association Rule Mining. By computing the frequent itemsets within each cluster, this method can be used to identify skills that are often seen together in a description. Also, we could identify association rules to suggest new skills to learn based on the skills someone already knows, which might satisfy our second hypothesis better than the experiments we performed.

6 ONLINE MEETING SCHEDULE

Meeting Times:

(1) 4/12/2021 - 1:00 pm - 2:00 pm EST - All Attended
(2) 4/19/2021 - 3:30 pm - 4:30 pm EST - All Attended
(3) 4/25/2021 - 1:00 pm - 2:00 pm EST - All Attended
(4) 4/28/2021 - 1:00 pm - 2:30 pm EST - All Attended

7 GITHUB REPOSITORY

GitHub: https://github.com/piepielovers/CSC522-Final-Project-Report
Google Colab (backup): Google Colab File

REFERENCES

[1] Ibrahim M. Nasser and Amjad H. Alzaanin. 2020. Machine Learning and Job Posting Classification: A Comparative Study. *International Journal of Engineering and Information Systems* 4, 9 (Sept. 2020). 6-14. https://philpapers.org/archive/NASMLA-3.pdf.

[2] Lakshmi, K. et al. 2017. CLUSTERING CATEGORICAL DATA USING k-MODES BASED ON CUCKOO SEARCH OPTIMIZATION ALGORITHM. *ICTACT Journal on Soft Computing*. 8, 1 (Oct. 2017), 1561–1566. DOI:https://doi.org/10.21917/ijsc.2017.0218.

[3] Nikita Sharma. 2019. Job Skills extraction with LSTM and Word Embeddings. University of Technology Sydney, Sydney, Australia. https://confusedcoders.com/wp-content/uploads/2019/09/Job-Skills-extraction-with-LSTM-and-Word-Embeddings-Nikita-Sharma.pdf.