Machine Learning Engineer Nanodegree
Capstone Report

Isaac Robson
February 13, 2018

Capstone Report

I. Definition

Project Overview

This project attempts to cluster geo codes in France (similar to zip codes in the US) via economic/demographic data regarding business activity in each geo code. The three datasets are taken from the *Institut national de la statistique et des études économiques* (INSEE) website, and we refer to them as '**firm**', '**pop**', and '**tiers**'. Each dataset has ~36,000 entries in a number of columns, but we'll only be using 10 features from '**firm**', 8 from '**pop**', and one from '**tiers**'. From the '**firm**' data, we use the number of businesses in each size category (e.g. how many firms have between 1-5 employees? How many firms have between 100-199 employees?) as a proxy for economic and population size in each geo code - this allows us to cluster the geo codes into different size categories based on the relative frequency of businesses of each size they have. We then use the '**pop**' data, whose columns contain comparable information on population size, (e.g. how many people are between 0-14? How many people are 90+?) to create a 'learnability' metric that helps with cluster validation. Finally, from '**tiers**', we have a 4-tier urban/rural classification from INSEE for each geo code in France that serves as a benchmark and motivation for this new metric.

This project originates from questions about the validity of clustering, both from an engineering and demographic perspective. How can we evaluate human-created clusters, such as the urban/rural tiering provided by INSEE? Do traditional metrics like silhouette score have any interpretability for these clusterings? And if so, how does it compare to algorithm-created clusters, such as the assignments from k-means or Gaussian mixture models? And how can we use traditional and novel metrics to 'score' human-created metrics for effectiveness or usefulness? Throughout the project, we also briefly address several other issues such as sparsity, data visualization, and preprocessing for discrete data types in the social sciences/humanities.

Our project consists of three sections in the Jupyter Notebook. Part 1 entails the actual clustering, while Part 2 consists of defining and utilizing 'learnability' and the scoring of our clusters against our more traditional metrics. The third and final part is a Conclusion section with a more substantial summary of the project's accomplishment. We will refer to Part 1 and Part 2 throughout this document, although neither part is standalone in our project.

Problem Statement

City officials and urban planners frequently attempt to plan the landscape and policy of a region in conjunction with business interests in the area. For instance, officials submitting bids to Amazon's famous HQ2 would have to consider the effects of Amazon's presence on the city's revenue forecasts, infrastructure planning, school system, and many more dynamical challenges that are frequently subject to feedback loops. While solving these challenges is beyond the scope of this project, we do explore the interface between business and government in a very limited sense. Our particular emphasis is the validity of INSEE's four-tier urban-rural classification, which we believe may not best partition the business and demographic meanings of urban and rural.

From an engineering perspective, we are primarily concerned with evaluation of clusterings, e.g. how can we determine if unsupervised clusters are valid? By examining the predictability of our clusterings with a disjoint (but correlated) dataset, we can train a supervised learning algorithm (i.e. logistic regression) to predict these clusters and validate our unsupervised results. This takes the quasi-supervised approach of Tibshirani and Walther (2005) a step further, but not beyond comparison. Doing this ties heavily to using unsupervised learning to label training data, but our primary concern here is to examine the efficacy of our clusters. Finally, we can explore the potency of data preprocessing for sparse data in the digital humanities. By simple A/B testing at each stage, we can look at success and failure in transformations and model selection techniques when dealing with sparsity and skew in data.

Metrics

We use 'learnability,' our supervised learning accuracy metric, to select a handful of clustering candidates for further advancement. This is the same as the typical 'score()' function of many sklearn algorithms, albeit on data we have used an algorithm to cluster instead of having some formal ground truth. More specifically, the 'learnability' should be the prediction accuracy of sklearn.linear_model.LogisticRegression trained on the data from '**pop**', which may or may not be transformed.

$$learnability = \frac{\#\ of\ correctly\ predicted\ clusters}{total\ \#\ of\ predicted\ points}$$

The idea for this metric is that silhouette score may not best capture the human-logic design aspects of the INSEE clusters, but a simple supervised prediction score would be much more understandable and appropriate to compare a human-created clustering to an algorithm-created clustering. Ultimately, we'd like to be able to compare the handful of clustering candidates selected from our 'learnability' metric and compare them to our INSEE benchmark with traditional metrics like sklearn.metrics.silhouette_score or Tibshirani's and Walther's 'prediction strength.'

Prediction strength is a metric for unsupervised learning, although we do create a comparable score for our benchmark. Below we define it in Figure 1 (as defined in the Jupyter Notebook).

For our pre-labelled INSEE benchmark, we create a prediction strength-themed score that instead of using KMeans uses a logistic regression on a 50/50 train-test-split to compare the predictions learned from the training data with those learned from the testing data. As a general rule for prediction strength, scores approaching one represent a 'valid' clustering, so having a score for INSEE's is somewhat redundant since we can assume INSEE created the classification system with some logic. Having prediction strength gives us another test to see if our algorithm-created clusters are sensical that's slightly easier to interpret to silhouette score.

Step 1: Split the data into 50/50 train-test-split
Step 2: Find 'n' centroids for the training data with the selected algorithm
Step 3: Cluster the testing data with 'n' centers via the selected algorithm to find 'true labels'
Step 4: Assign the testing data to clusters from the training centroids to find 'predictions'
Step 5: Define 'prediction strength' as the accuracy for the cluster with the lowest accuracy, e.g.

$$\text{'prediction strength'} = \min_{1 \le k \le n} \frac{|C_{k,k}|}{|\cup_{i=1}^{n} C_{k,i}|}$$

Where $|C_{k,k}|$ is the number of elements in test cluster k predicted to be in cluster k
And $|\cup_{i=1}^{n} C_{k,i}|$ is the number of elements in test cluster k assigned to any cluster (= total number of elements in cluster k)

Figure 1, Prediction Strength

II. Analysis

Data Exploration

We have three distinct datasets in this project, two of which have samples listed below and in the Jupyter Notebook. We see that both contain some degree of sparsity,

Geo Code	Unknown Size	Size 1-5	Size 6-9	Size 10-19	Size 20-49	Size 50-99	Size 100-199	Size 200-499	Size 500+
01001	22	1	2	0	0	0	0	0	0
01002	9	1	0	0	0	0	0	0	0
01004	577	272	63	46	24	9	3	2	0
01005	73	20	3	1	2	0	0	0	0
01007	87	20	10	5	2	0	0	0	0

(30051, 9)

Figure 2, '**firm**', which after removing nominative columns contains discrete counts of the number of businesses of each size in a geo code.

Note the many sparse entries. even after adjusting for entirely empty rows as visible in the following figures. The third dataset is sliced to serve as our benchmark, and is trivial discretized categories from 1-4, represented as 0-3 in our data, and can be seen at the end of the Notebook in a parallel coordinates plot with counts for each category listed as well.

	Code géographique	Région	Département	Libellé géographique	Population en 2014 (princ)	Pop 0-14 ans en 2014 (princ)	Pop 15-29 ans en 2014 (princ)	Pop 30-44 ans en 2014 (princ)	Pop 45-59 ans en 2014 (princ)	Pop 60-74 ans en 2014 (princ)	...	Pop 25-54 ans Autres en 2014 (compl)	Pop 55 ans ou plus en 2014 (compl)	Pop 55 ans ou plus Agriculteurs exploitants en 2014 (compl)	Pop 55 ans ou plus Artisans, Comm., Chefs entr. en 2014 (compl)	C
1	01001	84	01	L'Abergement-Clémenciat	767	161	102	132	189	125	...	0	240	10	0	
2	01002	84	01	L'Abergement-de-Varey	239	54.0947	27.5391	68.8477	36.3909	33.4403	...	0	78.6831	0	0	
3	01004	84	01	Ambérieu-en-Bugey	14022	2778.16	2958.44	2641.52	2603.3	1852.89	...	579.458	3869.46	0	74.2814	9
4	01005	84	01	Ambérieux-en-Dombes	1627	335.65	251.235	322.586	375.848	232.141	...	30.1482	447.199	0	5.02471	2
5	01006	84	01	Ambléon	109	11.8909	15.8545	14.8636	28.7364	26.7545	...	0	29.7273	0	0	4

5 rows × 108 columns

Figure 3, ‘**pop**’, complete with the same nominative columns and a number of extra unutilized features for each geo code. We extract the number of persons of each age category in a geo code, which are seen here in the ‘Pop X-Y ans 2014’ where X and Y specify an age range (with decimals to adjust for estimates).

We adjust for a large amount of skew with power transforms (Box-Cox) and eliminate an outlier in each, all while accounting for sparse (0) entries in transformations. Below we display tallies of the sparsity to examine the challenges in each dataset.
We can see that we reduced a large amount of the sparsity in the first few columns of ‘**firm**’ by eliminating the rows with many zero entries. The left is the number of zeros in each feature before dropping nominal features (and ‘Total Busis,’ which is collinear), while the right is the number of zero entries in each column after eliminating all rows with a very large number of zero entries. This still leaves us with a lot of sparsity for the larger business size columns, but on the whole it’s a smaller and more manageable dataset to work with.

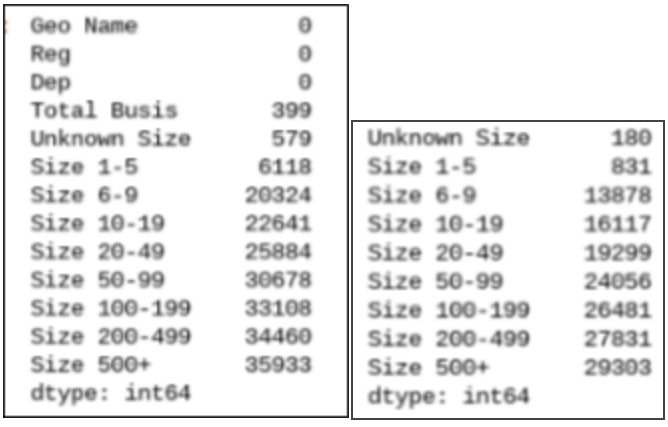

Figure 4, Sparsity in ‘**firm**’

Dropping so many entries (rows) does of course require us to drop entries in our other datasets as well, but we had to use a join operator regardless to ensure we both matched the data we examine only geo codes that exist in both '**firm**' and '**pop**' (as well as the benchmark, '**tiers**', although this had most of the same geo codes from either). Thus, we can safely drop a number of rows without issues of data integrity (we also have ~ 30,000 data points, plenty large to afford multiple subsetting operations).

```
Age 0-14        17
Age 15-29       12
Age 30-44        4
Age 45-59        0
Age 60-74        2
Age 75-89       20
Age 90+       3475
dtype: int64
```

To the left is an equivalent sparsity count for the '**pop**' data, which is much less sparse but still presents challenges with power transforms (which were resolved by adding one to the entries before the power transforms, which preserves sparsity).

The '**tiers**' data did not contain any sparsity or skew, as it was purely categorical labels that served as our benchmark. Below is the count of each tier, which shows the relative distribution of the tiers by INSEE's standards.

Figure 5, Sparsity in '**pop**'

```
Cluster counts for Benchmark:
0: 16542        1: 2282         2: 5194         3: 5043
```

Figure 6, Benchmark Data

Finally, we did end up exploring the variance inflation factors (VIFs) for our data, which revealed a good deal of collinearity (also expressed in our PCA results). This didn't pose any computational or identifiability challenges for our data, but it's good practice to be aware of and, if necessary, use an adjuster such as PCA (which was used here, although primarily for additional benefits such as computation speed and visualization). VIFs above 5 indicate significant collinearity.

```
VIFs for the cleaned, original data:
[  36.48  102.18  108.73   97.26   56.69   24.26   14.39    9.49    4.18]
```

Figure 7, VIFs for '**firm**'

Exploratory Visualization

Many visualizations were used, including boxplots and scattermatrices for the datasets. Below we sample a few of the visualizations used and explain their importance.

Figure 8 is a scattermatrix of the features in '**firm**'. It's clear that there exists a large amount of collinearity and skew, as evidenced by the nearly vertical lines on the

distribution with flat but extended KDE plots for each feature. This can also be observed in the boxplot below in Figure 9, which shows a much more single-dimensional but clearer visualization of the skew in the '**firm**' data.

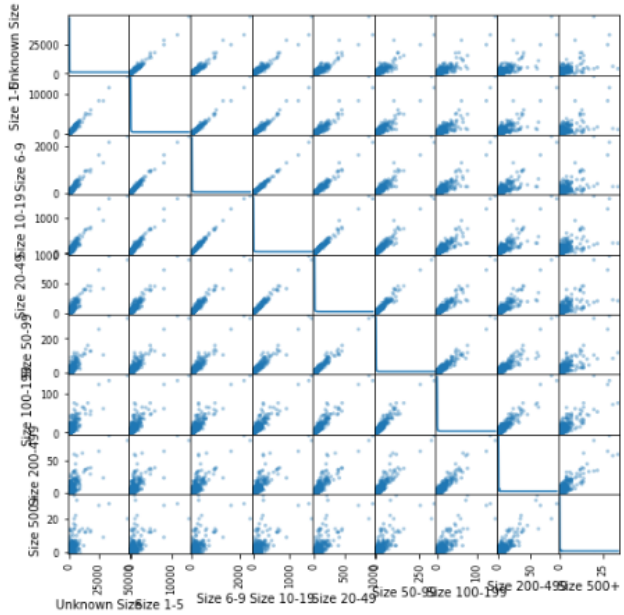

Figure 8, Scattermatrix of 'firm'

Algorithms and Techniques

Throughout the paper, we make use of sklearn.KMeans and sklearn.mixture.GaussianMixture, which are both unsupervised clustering algorithms. We don't tweak many of the parameters, other than specifying the the number of clusters, which we vary highly in order to create a diverse set of clusterings. We selected KMeans and GaussianMixture because they both tend to be robust and fast, along with numerous other reasons we now detail.

GaussianMixture in particular is one of the fastest mixture models, with few difficulties when the sample size is large. Thankfully, we have ~ 30,000 entries, so there is no shortage of data, eliminating one of the few drawbacks of GaussianMixture. It also assumes normally distributed data, but we can normalize our data with something like the Box-Cox transformation or many other scale/power transforms to even further improve the results of this algorithm.

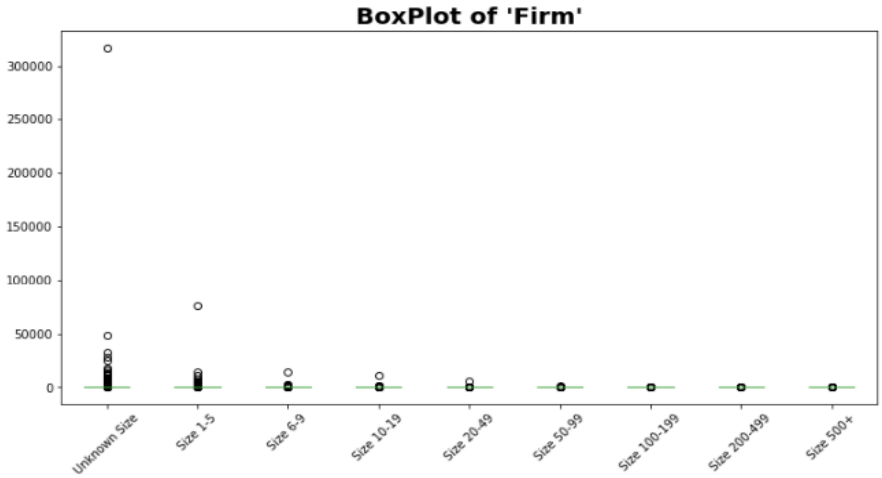

Figure 9, Boxplot of Sparsity-Reduced 'firm'

KMeans is a flexible clustering algorithm that offers quick convergence and multiple initializations in the same run. Like GaussianMixture, this means we can run it numerous times without crashing our computer. KMeans is sensitive to differing scales across dimensions, but using Box-Cox or a scaling preprocessing technique like MaxAbsScaler would likely create rich datasets for KMeans clustering. KMeans typically runs on Euclidean distance, which can be computationally intensive without PCA, but from discussing VIFs above, and for the purposes of visualization, we'd like to use PCA already, meaning KMeans should perform well on our data.

In order to use both KMeans and GaussianMixture, we created a number of dictionaries that could both pass different cluster count parameters and query for different access methods with the getattr() function. This is due to the fact that KMeans inherits from sklearn.cluster while GaussianMixture inherits from sklearn.mixture, and the two have different access methods and parameter names for the otherwise comparable algorithms.

Below, in Figure 10, is a chart of equivalences for the dictionaries utilized by the algorithms as they are stored. In the code, we attempted to store the values into local variables that better enunciate the item of interest. ' - - ' Indicates no local variable was used.

Description	Dictionary	Local Variable	KMeans Equivalent	GMM Equivalent
Algorithm	algos	- -	KMeans	GaussianMixture
Cluster Count	count	cluster_count	n_clusters	n_components
Centroid Locations	centers	cluster_dict	cluster_centers_	means_
Predictions	preds	preds	predict()	predict()

Figure 10, Algorithm Kwarg/Attribute Lookup Chart

Benchmark

The Benchmark can be partly observed in Figure 6. This four-tier classification by INSEE is an interesting benchmark as it interweaves quantitative population count with proximity to large urban centers to create a working definition of urban and rural. The actual scale has multiple subtiers for things such as suburbs of major cities, but ultimately it follows a 'most urban' to 'most rural' classification.

Given either the '**firm**' or '**pop**' data, it would make sense that the urban-rural scale correlates well with higher-density geo codes. Using our 'learnability' metric gives us a sense of approximately how 'learnable' a real urban-rural scale should be, and also serves as an interesting point of analysis within the context of silhouette score - is silhouette score or other unupservised learning metrics viable for human-created classifications (or 'clusters' as we choose to interpret them)?

We should note that in Part 2, the supervised learning section of our project, the benchmark achieves a 'learnability' score of ~ 0.65, something we found surprising as it suggests that INSEE's clustering relies heavily on proximity to large urban centers to create its tiers. Additionally, it is important to mention using silhouette score on the benchmark is dataset-dependent, e.g. a silhouette score varies based on the data being used. Our 'learnability' metric however, is much more indpendent as it uses an extraneous dataset.

III. Methodology

Data Preprocessing

After we had removed some sparsity from our '**firm**' data, we derived a number of interesting variations on our datasets from the original, sparsity-reduced'**firm**' data after eliminating an outlier. Note that the typical definition of outlier would render most, if not all, of our features exhibiting 'Size 500+' quantities greater than zero, so we were very conservative in our outlier elimination.

After the cleaning listed above, we formed both a Box-Cox-transformed dataset and a dataset transformed by sklearn.MaxAbsScaler then variance-stabilized with a single square root transformation. Both of these transformations utilize a power transform, which we ultimately deemed necessary from the large amount of skew and variance present in the original data. These specific techniques are very useful because they can increase the normality of the data, and also assure us that the features are weighted more equally in a clustering. While our Box-Cox'd data doesn't appear to be normalized, which does raise flags about the appropriateness of this transformation, we see that it substantially reduces the skew across each features, justifying the transformation. A similar reduction was visible in the square-rooted, MaxAbsScaled data, and both are visible in Figure 11 below. Scattermatrices of these also exist, which do show a reduction in collinearity, however, for brevity we omit them here.

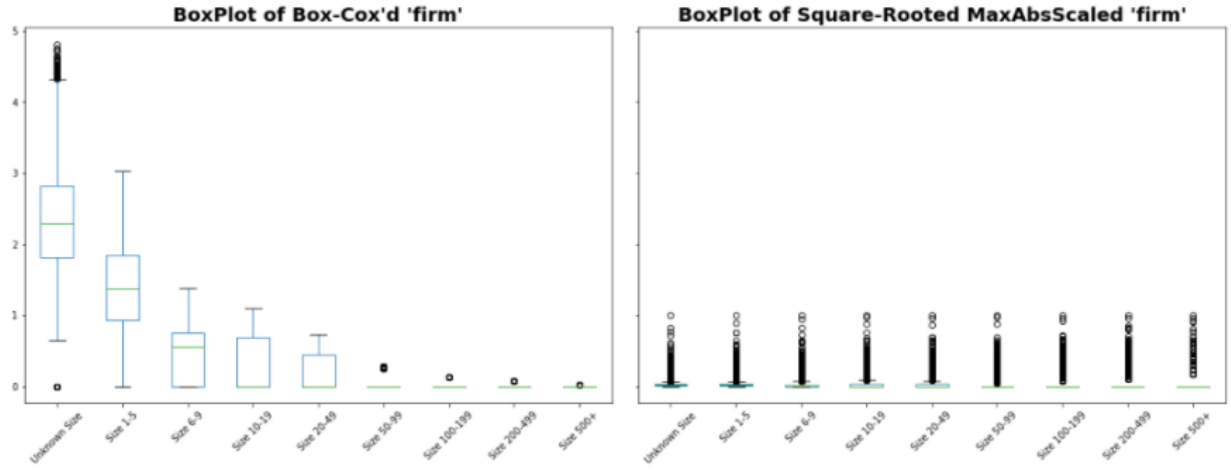

Figure 11, BoxPlots of 'firm'

We then created PCA'd versions of the original, the Box-Cox'd, and the square-rooted, MaxAbsScaled data. Note that each of these transformations was applied to the features individually, which destroyed much of the information present in the covariance matrix of the data. However, the transformations also injected new capabilities such as easier visualizations and new outcomes such less collinear data that ultimately resulted in valid but more oftentimes more interesting clusterings.

Additionally, for the '**pop**' data in Part 2, our supervised section, we followed a much more streamlined transformation of our data, which since it conformed much better to a typical Box-Cox transformation (largely due to substantially less sparsity than '**firm**'), we didn't see much need for additional poking around, although we did apply PCA for increased computability. Figure 12 shows this conformity, where many of the features in '**pop**' appear normally distributed,

Within the code itself, we create side-by-side clusterings across both the pre- and post-PCA datasets. This allows us to see the effects of the transformations and evaluate the strengths, weaknesses, and uniqueness of each (subsequent) transformation.

Implementation

Similar to the content in the 'Algorithms and Techniques' section above, we utilized multiple dictionaries to abstract a layer from our datasets and algorithms. This is a slightly more complicated process, but it allows for much more scalable code without having to override or otherwise modify the underlying classes in Python. One should note that this technique hinges upon pass by reference. For something such as cluster count, using a dict isn't much of a leap, however when dealing with datasets, predictions, and clusterings, this becomes a much more exciting challenge but saves us

hundreds of lines of code and greatly condenses the Jupyter Notebook.

We could have been more selective in which datasets we analyzed by incrementally making analysis decisions, but these decisions frequently pose a ‘free lunch’ dilemma and don’t leave as much wiggle room for model selection post- clustering. Hence our decision to create an ample number of clusterings to choose from. The variety of clusterings created by this approach is displayed below in Figure 13, which displays a number of the post-PCA clusterings (the pre-PCA clusterings have much higher dimensionality and more complex ‘chartability’ than we care to deal with here).

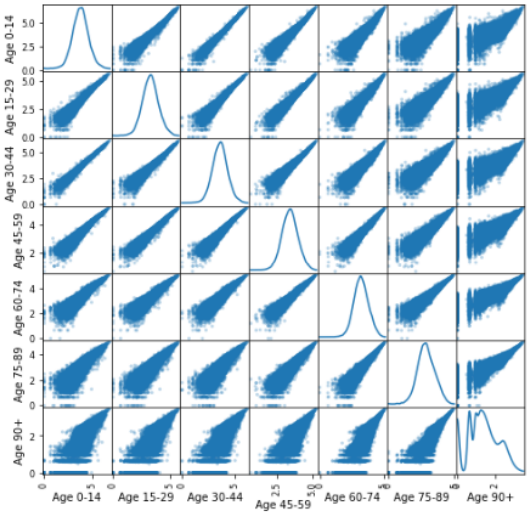

Figure 12, Box-Cox’d ‘pop’

Within Part 2, we follow a similar procedure to Part 1 to process our datasets, albeit in a slightly simpler fashion. We keep an abstracted approach to our algorithms, and this allows us to use common decision structures for our ‘learnability’ metric and inverse functions.

One of the more interesting implementation challenges was the creation of cluster_matcher() as the crux of our design of ‘prediction strength’ from Tibshirani and Walther (2005). We simply enumerate all possible combinations of cluster-to-cluster mappings and calculate the Euclidean distances between each centroid combinations to determine the best mapping that preserves the original intent of prediction strength, namely to determine modulation in assignment of a point to ‘a specific cluster’. Due to our clustering algorithms, these ‘specific clusters’ can only be identified via centroid location or numeric cluster identity randomly assigned each iteration by the clustering algorithm, which necessitates this best mapping determination from cluster_matcher().

In simpler terms, cluster_matcher() converts the test labels in pred_strength(), our prediction strength calculator, to match the corresponding training label, ensuring our prediction strength is calculating the correct accuracies. cluster_matcher() can be seen below in Figure 14.

Finally, we did require a tweak to Tibshirani’s and Walther’s (2005) prediction strength metric as our benchmark, having pre-defined labels, was unsuited to the unsupervised learning intention of prediction strength. This tweak involves substituting a supervised learning algorithm, for which we chose the already mentioned logistic regression, and then predicting the test labels and training labels in a similar fashion. Note that this improvisation does not carry much validity, although it does reveal some about the innate statistical structure of the benchmark. Ultimately, the prediction strength

metric does have some objective validity as scores above 0.9 are typically considered to be valid clusterings, with little need for a ‘benchmark’ save the comparison to other cluster counts if possible (which we omit as we choose to compare clusterings of the same cluster count as our benchmark - four).

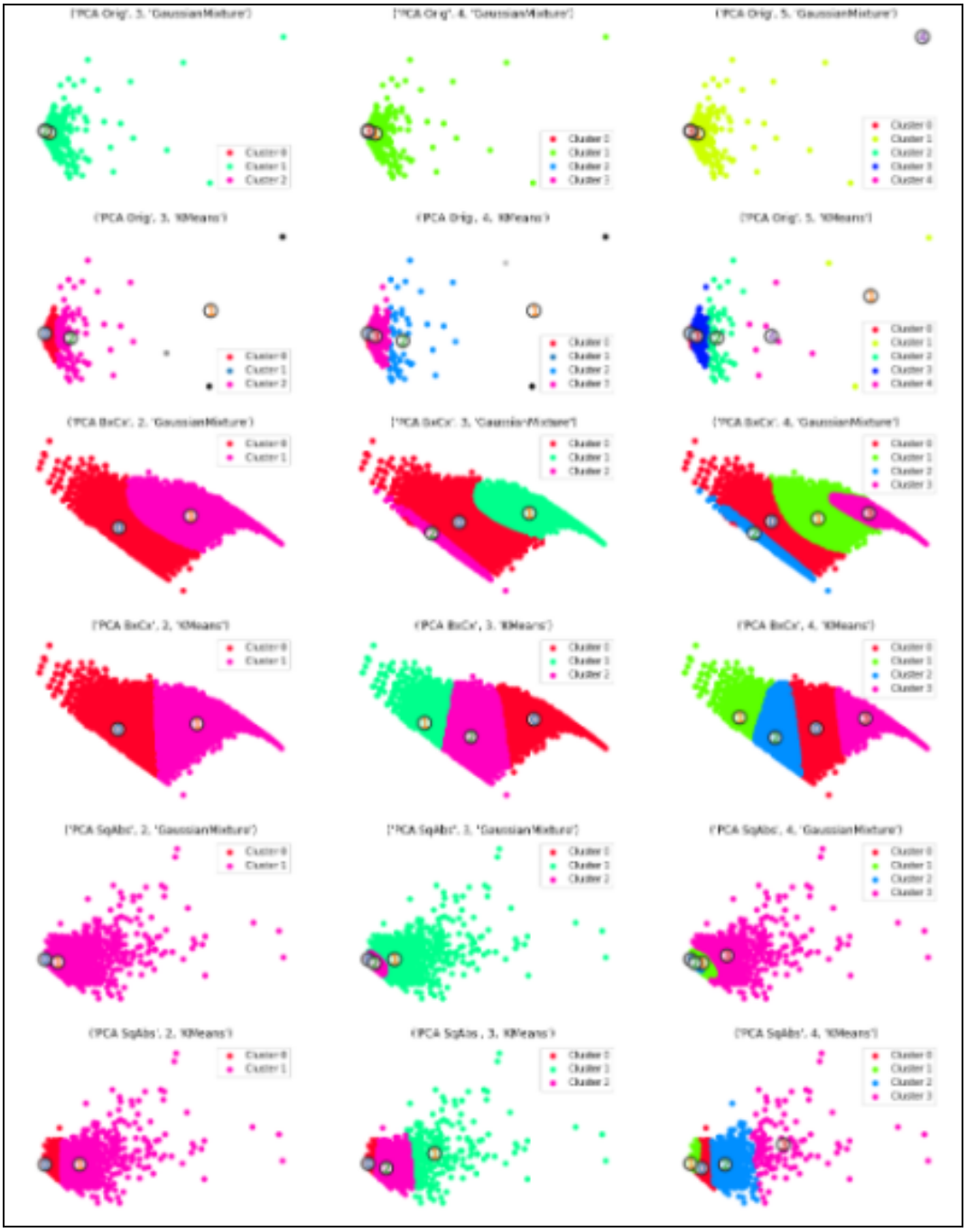

Figure 13, Sample Clusterings Visualized

```
def cluster_matcher(train_centers, test_centers):
    cluster_matching_dict = {}
    cluster_distance_dict = {}

    for train_cluster_num,train_cluster_coords in enumerate(train_centers):
        for test_cluster_num,test_cluster_coords in enumerate(test_centers):
            cluster_distance_dict[(train_cluster_num, test_cluster_num)] = np.linalg.norm(train_cluster_coords - test_cluste

    # this combinatoric solution helps us find the minimum distance
    combo_list = [zip(x, range(len(test_centers))) for x in itertools.permutations(range(len(train_centers)), len(test_cente

    # large distance to be overwritten
    min_dist = 10000000000

    for combo in combo_list:
        total_dist = 0
        for mapping_point in combo:
            total_dist += cluster_distance_dict[mapping_point]
        if total_dist < min_dist:
            min_dist = total_dist
            cluster_matching_dict = dict(combo)

    #display(min_dist)

    return cluster_matching_dict
```

Figure 14, cluster_matcher()

Refinement

As we had created a number of distinct datasets and a large number of clusterings with variable cluster counts, our refinement process largely consisted of using simple rules to eliminate 'poor' clusterings. Had we only one dataset at this point to use, we would consider exploring transformations of our data in order to better cluster.

This process entailed a maximum cluster ratio, which consisted of dividing the size of the largest cluster by the size of the smallest cluster for each clustering. The results of this analysis can be seen in Figure 15. We then proceeded to eliminate all clusterings with a > 50 large-to-small ratio, as these would likely be poor indicators of urban or rural, at least for comparison to INSEE's tiers.

Additionally, as we mentioned briefly in the 'Implementation' section, we chose to focus on models of the same cluster count as our INSEE data - four. This would ensure that our metrics such as 'learnability,' silhouette score, and prediction strength would provide meaningful and interpretable information. To do this, we picked three of the most 'learnable' clusterings with a cluster count of four, namely for Gaussian Mixture: ('PCA Orig', 4), and ('PCA SqAbs', 4), and for KMeans: ('PCA BxCx', 4). These three were selected as they represented the most 'learnable'

```
GaussianMixture

(Orig, 5)          23608.0
(Orig, 6)          23550.0
(Orig, 7)          23546.0
(Orig, 8)          23520.0
(Orig, 9)          23517.0
(PCA Orig, 5)      16013.0
(PCA Orig, 6)      13821.0
(PCA Orig, 7)      13239.0
(PCA Orig, 9)      12983.0
(PCA Orig, 8)      12982.0
(PCA SqAbs, 8)      6561.0
(PCA SqAbs, 9)      6397.0
(Orig, 3)           4228.0
(Orig, 4)           3933.0
(PCA SqAbs, 7)       820.0
(PCA SqAbs, 6)       280.0
(PCA SqAbs, 5)        72.0
dtype: float64

KMeans

(Orig, 6)          29454.0
(PCA Orig, 6)      29454.0
(PCA Orig, 7)      28929.0
(Orig, 7)          28900.0
(Orig, 8)          27671.0
(PCA Orig, 8)      27662.0
(PCA Orig, 9)      27566.0
(Orig, 9)          27564.0
(PCA Orig, 5)       9820.0
(Orig, 5)           9820.0
(PCA Orig, 3)       7484.0
(Orig, 3)           7484.0
(PCA Orig, 4)       7387.0
(Orig, 4)           7387.0
(Orig, 2)           3004.0
(PCA SqAbs, 9)      2205.0
(PCA SqAbs, 7)      1073.0
(PCA SqAbs, 8)      1059.0
(PCA SqAbs, 6)       539.0
(PCA SqAbs, 5)       424.0
(PCA SqAbs, 4)       200.0
(PCA SqAbs, 3)        86.0
dtype: float64
```

Figure 15, Cluster Ratios

clusterings for each algorithm. The learnability scores for the post-large-to-small ratio list can be seen below in Figure 16.

Finally, we should mention that we did have some preference for the PCA'd data as it captured most of the information present for all of our dataset variations and would be much more computationally-friendly for subsequent detailed analysis.

GaussianMixture		KMeans	
('Orig', 2)	0.941	('PCA SqAbs', 2)	0.973
('PCA Orig', 3)	0.881	('PCA BxCx', 2)	0.914
('PCA SqAbs', 2)	0.865	('Box-Cox', 2)	0.913
('Box-Cox', 2)	0.862	('PCA BxCx', 3)	0.819
('PCA BxCx', 2)	0.852	('Box-Cox', 3)	0.817
('PCA SqAbs', 3)	0.84	('PCA BxCx', 4)	0.697
('Box-Cox', 3)	0.821	('Box-Cox', 4)	0.696
('PCA Orig', 4)	0.788	('PCA BxCx', 5)	0.638
('PCA SqAbs', 4)	0.78	('Box-Cox', 5)	0.633
('Box-Cox', 4)	0.692	('PCA BxCx', 6)	0.547
('PCA BxCx', 3)	0.666	('Box-Cox', 6)	0.532
('Box-Cox', 5)	0.607	('Box-Cox', 7)	0.503
('PCA BxCx', 4)	0.59	('PCA BxCx', 7)	0.491

Figure 16, 'Learnability' Scores

IV. Results

Model Evaluation and Validation

We should note that our three final candidate models, enumerated in the 'Refinement' section above, all scored higher 'learnability' scores than our benchmark. Granted, one question this project is attempting to answer is whether or not this 'learnability' metric is useful or effective. Hence the subsequent analysis with silhouette score and prediction strength.

After we compared each of our candidate models to the benchmark (discussed below), we did examine the vulnerability of our 'learnability' with the Spearman correlation coefficient. The Spearman coefficient, or Spearman r, is the r^2 coefficient for the rankings of two lists. We use it here to compare the rankings produced by our 'learnability' metric with the rankings produced by both silhouette score and prediction strength for the post-large-to-small ratio clusterings. We see that between both the GaussianMixture and KMeans sets of clusterings, the Spearman coefficient was relatively high - between 0.57 and 0.80 - for 'learnability.' This is a much better range than the 0.39 to 0.68 range between silhouette score and prediction strength, which suggests that our learnability metric, while similar to silhouette score and prediction strength, is to some degree an independent but valid metric. The Spearman coefficients can be seen below in Figure 17.

Thus we can conclude that our ‘learnability’ metric indicates trustworthy models, and since prediction strength indicates a randomized 50/50 split of our data, we can also have some assurance of the replicability and robustness of models with a high (> 0.9) prediction strength score.

```
Silhouette and Learnability Spearman:
GMM: 0.8069     KMeans: 0.6127

Silhouette and Prediction Strength Spearman:
GMM: 0.6836     KMeans: 0.3922

Prediction Strength and Learnability Spearman:
GMM: 0.5768     KMeans: 0.7181
```

Figure 17, Spearman Correlation Coefficients

Justification

Each of our candidates substantially outperformed the benchmark on ‘learnability,’ as did several of the other clusterings of size 4. For comparison purposes, one can examine the benchmark ‘learnability’ in Figure 19 to the ‘learnability’ of our candidates listed above in Figure 16. As an added layer of justification for our benchmark’s ‘learnability’ score, we computed it after PCA and with a decision tree classifier, both of which achieved slightly less prediction accuracies.

However, the question remained as to whether they would outperform the silhouette score of the benchmark, and whether our predictions strengths indicated valid clusterings. Note that while our proxy ‘prediction strength’ for the benchmark is not actually prediction strength, we can infer to a limited extent the structure of our datasets from the benchmark ‘prediction strength.’

```
(('PCA Orig', 4), 'GaussianMixture')
Benchmark Silhouette: -0.44129
Benchmark Pred Strength: 1.0
Actual Silhouette: 0.4652
Actual Pred Strength: 0.96861
-------
(('PCA SqAbs', 4), 'GaussianMixture')
Benchmark Silhouette: -0.32061
Benchmark Pred Strength: 1.0
Actual Silhouette: 0.55979
Actual Pred Strength: 0.96972
-------
(('PCA BxCx', 4), 'KMeans')
Benchmark Silhouette: -0.10936
Benchmark Pred Strength: 0.8676
Actual Silhouette: 0.43373
Actual Pred Strength: 0.98544
```

Figure 18, Candidate Performance

In Figure 18, we see the results of each silhouette score and prediction strength for each of the candidates. All of the candidates demonstrate a superior silhouette score, while the proxy ‘prediction strength’ for the benchmark is near perfect - save for the last clustering, which we will elaborate more on soon. It appears for now that our ‘learnability’ metric did assist us in choosing quality candidate clusterings since all of them had very high (> 0.9) prediction

strength scores that exhibit a decent silhouette score. We should note that while not conclusive, the fact that the benchmark prediction strengths were high indicate that our first two datasets were easier to learn from, while our last dataset was less separable by a logistic regression. Interestingly, our last dataset had the highest prediction strength, which suggests that in this case especially, our clustering dramatically outperforms the benchmark as a 'valid clustering,' at least by algorithmically-defined standards (it seems unlikely INSEE would reclassify the urban-rual scale for all of France from the output of an algorithm - at least for now).

We can rest assured that we found a quality clustering method for the urban-rural classification at this point, since the candidates all demonstrate better 'learnability', superior silhouette scores, and high prediction strengths, three indicators of valid clusterings. We can also conclude that 'learnability' is a promising metric for unsupervised learning, especially given the vastly shorter computation time compared to prediction strength or sampled silhouette score, as seen in Figure 20 (computed on all of the valid clusterings, just like the Spearman coefficients). Finally, there is evidence to suggest that silhouette score is a poor metric for human created 'clusterings.' We have of course, stretched the meaning of clustering, but without getting too philosophical, human classifications are based upon 'features' that we choose to emphasize for a specific 'class' of objects, and if these 'classes' are truly distinct, they should exhibit similar 'features,' and cluster accordingly.

```
Total silhouette time: 42.1714
Total pred strength time: 49.3963
Total learn time (above): 13.9447
```

Figure 20, Comparing Metric Computation Times

V. Conclusion

Free-Form Visualization

As we can see in the parallel coordinates plots in Figure 21 below, the benchmark is not as clean of a clustering across the '**firm**' data compared to our algorithm-created clusterings. We could likely achieve a crisper distinction between tiers with the '**pop**' data, and were we to create a true clustering for a French urban-rural scale, we would also include far more datasets to achieve a much less 'subjective' clustering compared to our (likely biased) '**firm**' data. Granted, the engineering challenge of clustering '**firm**' was far more substantial than that of clustering '**pop**', as can be seen in the structural differences of the two in the numerous figures below.

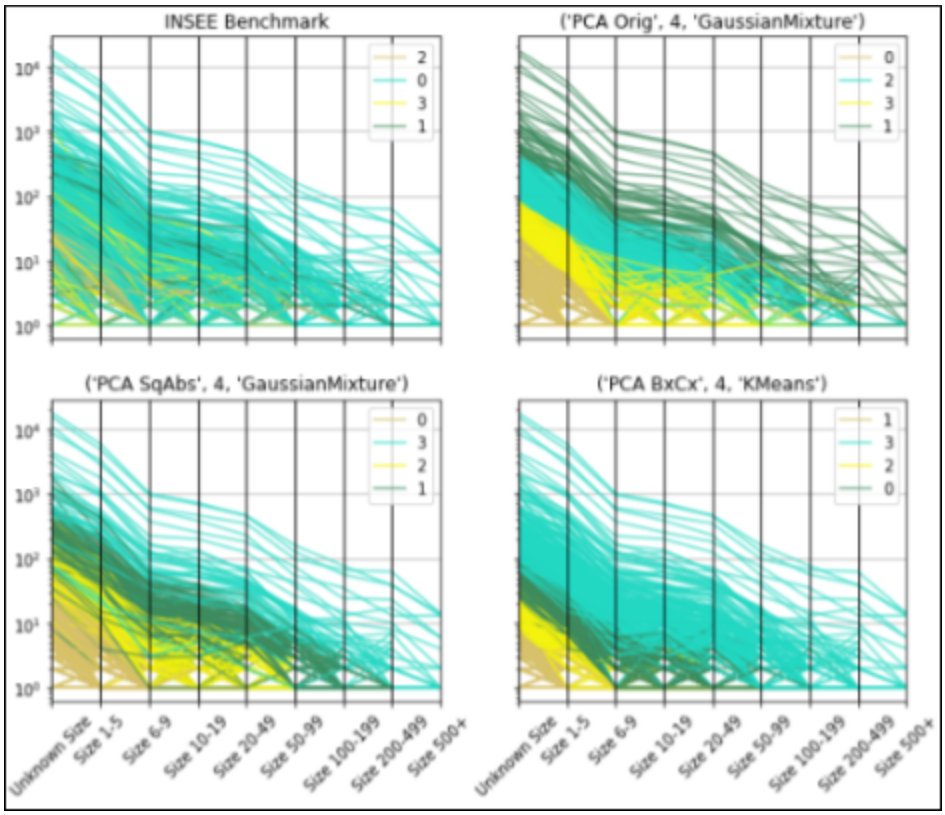

Figure 21, Plots of Candidates vs Benchmark

Parallel coordinates plots work well to visualize the benchmark and our candidates - it displays a sizable sample of each of the clusterings while showing them in like aspect ratio without requiring more than two dimensions. Parallel coordinates plots also show us important characteristics such as the relative size of each cluster and how the clusters compare to one another across features. Here we can see the limitations of our '**firm**' dataset as well, since while it correlates (rather well) with INSEE's four-tier classification, it is undoubtedly not the only factor in INSEE's decision to categorize a geo code as 'more urban' or 'more rural.'

Reflection

In this project, we analyzed '**firm**', preprocessed it into a variety of datasets, created a number of clusterings for each of those datasets, then winnowed our clusterings with common sense practices before using our new 'learnability' metric with an external but related dataset to select a handful of promising candidates. We then compared these candidates to our benchmark with silhouette score and examined the advantages of our new metric over silhouette score while using Tibshirani's and Walther's (2005) prediction strength metric to assure quality clusterings.

One of the interesting aspects of this project was utilizing dictionaries to automate much of the clustering-creation process. This degree of automation makes it easy to create a large number of unique clusterings (without large amounts of code), which appears to be a viable technique for medium-sized datasets when there is sufficient computation power. Otherwise, much stingier, iterative techniques would need to be used, where only a handful of focused models are created.

One challenging aspect of this project was the combination of skew, collinearity, and sparsity present in '**firm**'. Preprocessing this data required large amounts of forethought and ultimately, we can see that the simple 'PCA Orig' dataset was a candidate for the best clusterings. The dictionaries certainly helped with the latter stages of preprocessing, but sparsity detection and removal while utilizing a power transform (accomplished with the two parameter Box-Cox technique) was a bit convoluted, although the square-rooted MaxAbsScaled transformation was much easier. Finding better preprocessing techniques for sparse, collinear, and skewn data would certainly help.

The final candidates do demonstrate promise, and by using automated clustering-creation techniques, this pipeline is promising for future cluster creations. Abstracted and expanded versions of this pipeline would likely perform well, even in production-grade settings. Makes you wonder if Google's AutoML has a similar setup… Finally, 'learnability' appears to be a viable metric when a second dataset is present to ensure we don't overfit the noise in a single dataset. Of course, it would likely make sense to combine '**firm**' and '**pop**' in order to actually cluster the data, so perhaps the only means of accomplishing this 'learnability' in practice would be some variant of cross-validation. The computation time advantage does make it look attractive.

Improvement

For automating clustering-creation, we could likely use wrappers to speed up the process and clean up the code. This would require a bit more advanced software engineering techniques, but the advantages would include increased interpretability of code and more sustainable automation in future cases.

For the preprocessing comments in the 'Reflection' section above, we could likely create an automated mix & match pipeline for subsets of the data and select an optimal combination with some variant of cross-validation. This would entail a large amount of upfront work in creating the combination-creation machine, but it would probably simplify the process for not-so-pretty data, as it's not always clear what kind of transformations work best.

Finally, we know that '**firm**' is not the ultimate arbiter of INSEE's classification standards, so combining '**firm**' with '**pop**' as well as proximity, geographic, and

workforce data would certainly yield more meaningful clusters. Using ‘learnability’ or some other metric developed for INSEE’s four-tier (or even the entire classification system with subtiers), we may be able to show with a high degree of confidence that INSEE’s system isn’t an optimal classification. Ultimately, this urban-rural spectrum is a very abstract measure, unlikely to directly impact many things either, so it would also be good to incorporate more meaningful economic, employment, or educational information into this system or analyze the joint distribution of such variables as well. With further analysis, we may even be able to quantify the distinction between urban and rural, highlight the diversity in modern society, and better understand the different ways we view the world based on how and where we live.