ML Regionalization Model

Chris Whitmore, Shanna Greathouse, Stephanie McNutt

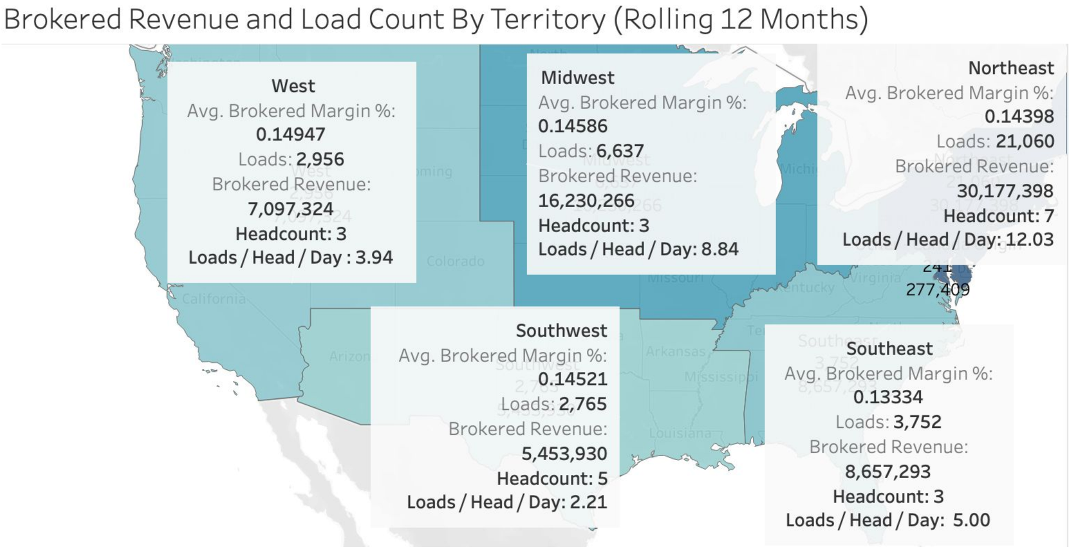
Mission statement: US sales regions will be defined by their market features and not limited by their geography as the workforce embraces the WFH model.

The problem

A client is looking to scale and grow 2x over the next 5 years and has noted satellite offices.

Regions have been defined by geographic proximity for sales coverage, but the lack of development on the west coast and newness of the office is dampens West Coast market development.

Sales associates are often working across regions for enterprise accounts, confusing the customer base and causing poor customer experience.



The solution

ML model to group states based on attributes to build regions, allowing sales to sell to who they know and what they know.

The Process

Gather Data

3 year csv of transaction history
geoJSON of US map

Pandas - Clean Data

Filter to Rolling 12 month from max date in dataset
Modify column types
Rename headers

Set Continuous Variables

Float variables including average margin and total volume grouped by origin state ['NAME']

Set Discrete Variables

Create dummies to change customer and supplier information to numbers for processing

MatPlot - Find the Ideal Region Count

Elbow Method to determine when there are diminishing returns on increasing the region count

SKlearn - Regionalize

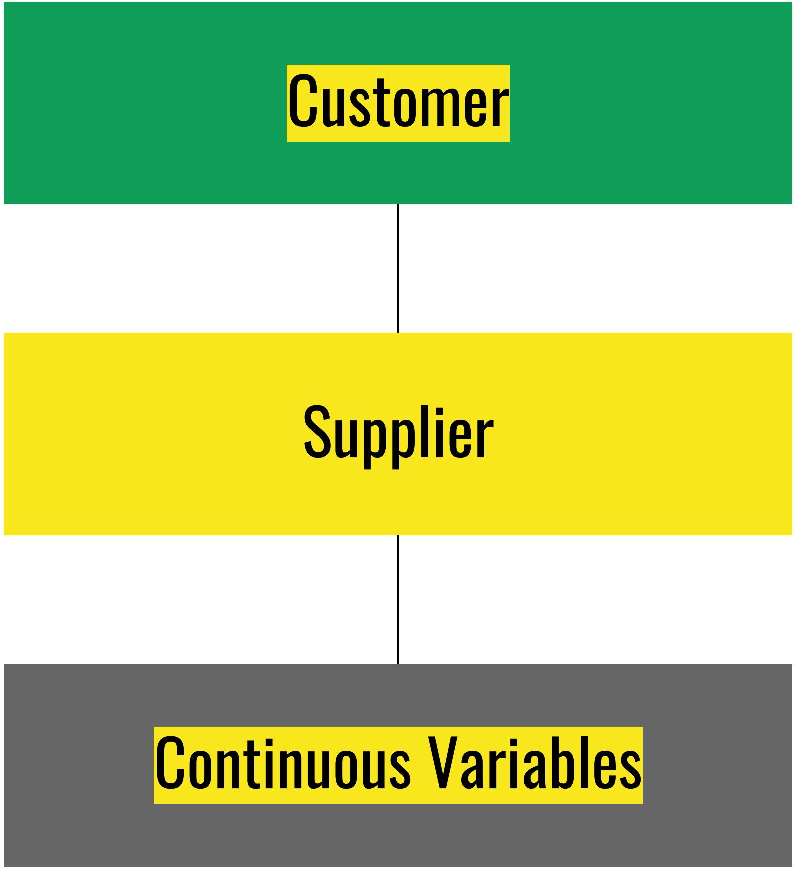
Display the clustered regions based on the defined attributes using the optimal region count

Summarize regional attributes

The technology: ML Geospatial Clustering

Clustered Sales Regions by Attributes

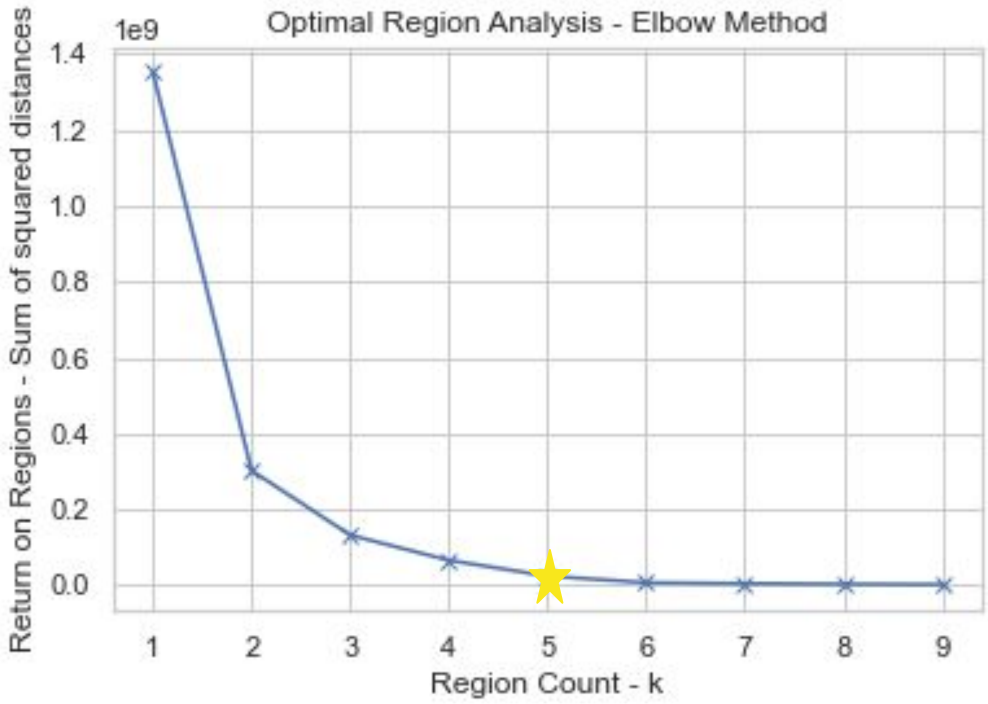
Attributes are defined by average load volume (12 month rolling period), average margin, customer, and supplier. Regions are not defined by geographic proximity to each other.



How many regions?

Using the Elbow Method, we can see at what point regions stop being significantly different. We should have no more regions than are significantly different in order to optimize cost of service.

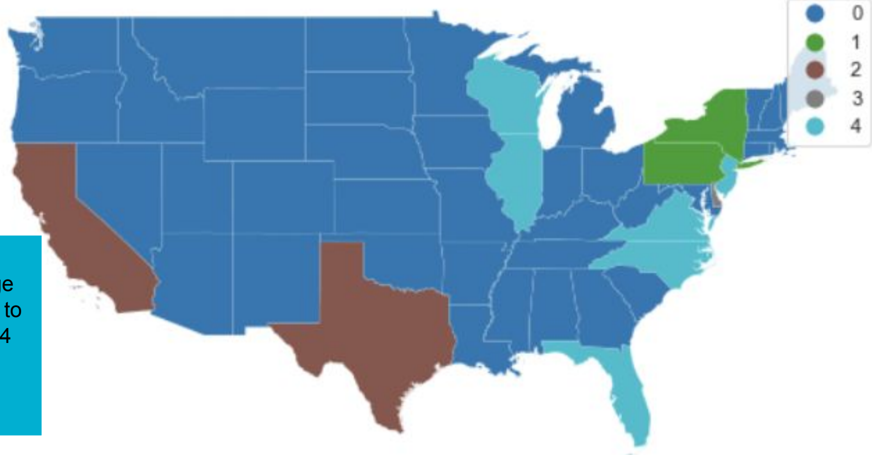
```
In [118]: #Optimal Cluster Calculations
          Sum_of_squared_distances = []
          K = range(1,10)
          for k in K:
              km = KMeans(n_clusters=k)
              #km.fit(X)
              km = km.fit(statesct) #count of Origin state
              Sum_of_squared_distances.append(km.inertia_)

In [119]: Sum_of_squared_distances

Out[119]: [1351964189.8958328,
           302673937.33333325,
           132534766.49211712,
           66259163.158783786,
           24764002.950450454,
           7213420.424242424,
           4991970.007575759,
           3285032.6388888895,
           2257830.138888889]

In [143]: #optimal clusters
          plt.plot(K, Sum_of_squared_distances, 'bx-')
          plt.xlabel('Region Count - k')
          plt.ylabel('Return on Regions - Sum of squared distances')
          plt.title('Optimal Region Analysis - Elbow Method')
          plt.savefig('Elbow Method')
```



The Results

Regions are grouped broadly across time zones based on customers, suppliers, and average margin. By implementing this new regional sales model, sellers can focus on selling even more to the clients they are used to working with, while operations can have more supplier consistency!

Volume by Region range has **leveled** to 8839 - 15874 while prior it was 2,765 - 21,040



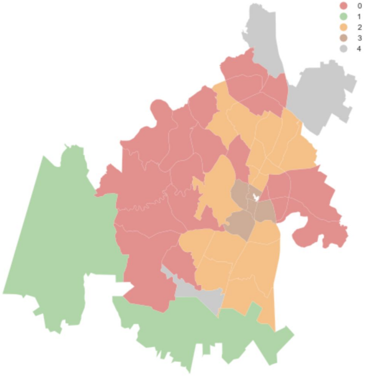
Region	State Count	Mean Margin	Annual Load Volume
0.0	37.0	387.79	14094.0
1.0	2.0	309.792	15874.0
2.0	2.0	394.743	8839.0
3.0	1.0	241.7	12396.0
4.0	6.0	362.823	12404.0

You may have noticed **Delaware** is on its own - *what a strange error!*
Not at all - **Delaware is the client's only market for a unique mode of transportation with therefore unique customers and suppliers!**

```
In [124]: #Cluster in specfic groups
cluster_count = 4
km5 = cluster.KMeans(n_clusters=cluster_count)

In [125]: # Fit to data
km5cls = km5.fit(statedb.drop(['geometry', 'NAME'], axis=1).values)

In [140]: # Map clusters
f, ax = plt.subplots(1, figsize=(9, 9))

statedb.assign(cl=km5cls.labels_)\
   .plot(column='cl', categorical=True, legend=True, \
         linewidth=0.1, edgecolor='white', ax=ax)

ax.set_axis_off()
plt.savefig('Regions')
```

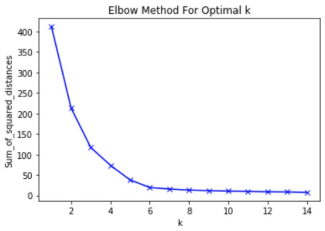
Resources

Our model is based off these methods:



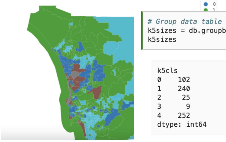
Spatial Clustering - Austin AirBnB
http://darribas.org/gds_scipy16/ipynb_md/07_spatia
l_clustering.html

```
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```



kMeans - Cambridge Blog
https://blog.cambridgespark.com/how-to-determine-
the-optimal-number-of-clusters-for-k-means-clusteri
ng-14f27070048f

```
# Assign labels into a column
db['k5cls'] = k5cls.labels_
# Setup figure and ax
f, ax = plt.subplots(1, figsize=(9, 9))
# Plot unique values choropleth including a legend and with no boundary lines
db.plot(
    column='k5cls', categorical=True, legend=True, linewidth=0, ax=ax
)
# Remove axis
ax.set_axis_off()
# Display the map
plt.show()
```

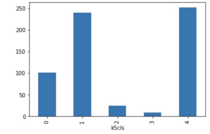


```
# Group data table by cluster label and count observations
k5sizes = db.groupby('k5cls').size()
k5sizes
```

```
k5cls
0    102
1    240
2     25
3      9
4    252
dtype: int64
```

And we can get a visual representation of cardinality as well:

```
_ = k5sizes.plot.bar()
```



Find the attributes - Geographic Data
https://geographicdata.science/book/notebooks/10_
clustering_and_regionalization.html