USING DATA TO FIND THE OPTIMAL MIX OF RETAIL LOCATIONS AND RESOURCES

northhighland®
WORLDWIDE CONSULTING

Data and Analytics

INTRODUCTION

Education

- BS CS, Georgia Tech 2009 – Theory and Machine Learning
- MS CS, Georgia Tech 2011 – Heavy Tail Network Analysis

Work

- Institute of Nuclear Power Operations (2010-16)
  - Build, deploy, maintain a model that predicted nuclear power station performance along 13 key functional areas
- North Highland (2016-)
  - ETL, BI, Advanced Analytics for Fortune 100 retailer

northhighland. WORLDWIDE CONSULTING | Data and Analytics

1. Data & Analytics outside academia
2. Case Study: Reassigning territories for district managers
3. Q&A

WORKING WITH CLIENTS

- Problems are never stated formally
- “Interesting” problems can be few and far between
- But they can build your personal brand

REMAPPING TERRITORIES – PROBLEM DESCRIPTION

- Minimizing travel time for regional managers can reduce incurred travel costs and boost morale
- Aligning districts to strategic goals can help ensure a variety of goals:
  - A level playing field where top talent can be evaluated evenly
  - Specialized focus for individual district owners
  - No one regional leader becomes overburdened compared to the others

AVAILABLE DATA

- Store Metadata – geocoding, age, size, store annual sales category, etc.
- Sales Data – department, class, subclass, SKU grain data anywhere from monthly roll-ups to individual transactions
- Inventory Data
- Online Transactions
- Current Territory

(ABBREVIATED) TOOLBOX OF TECHNIQUES

Technique 1: k-means

- Unsupervised Learning
- Identifies a number of means around the map and builds clusters with equal variance inside them
- Very much a black box-hard to specify, and requires a lot of tuning
- **Use if:** You want to explore your data, equal size isn't as important

Technique 2: Integer programming

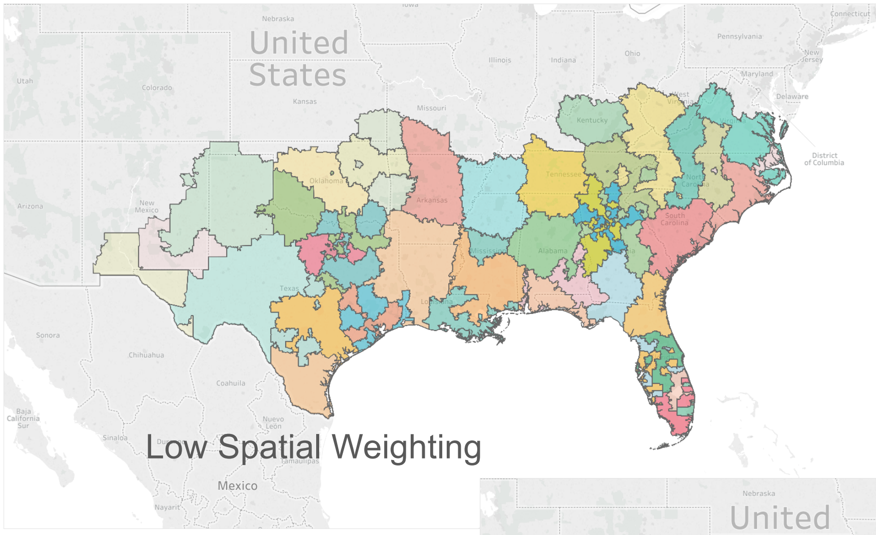
- Can specify exactly what you want, but rules are rigid
- Computationally impossible for large datasets—constraints have to be relaxed
- **Use if:** You have little data

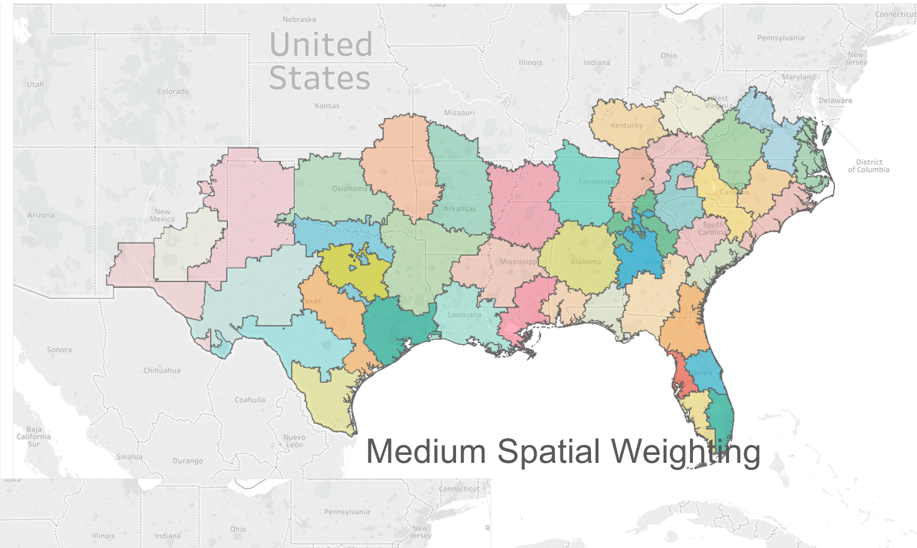
Technique 3: Network construction

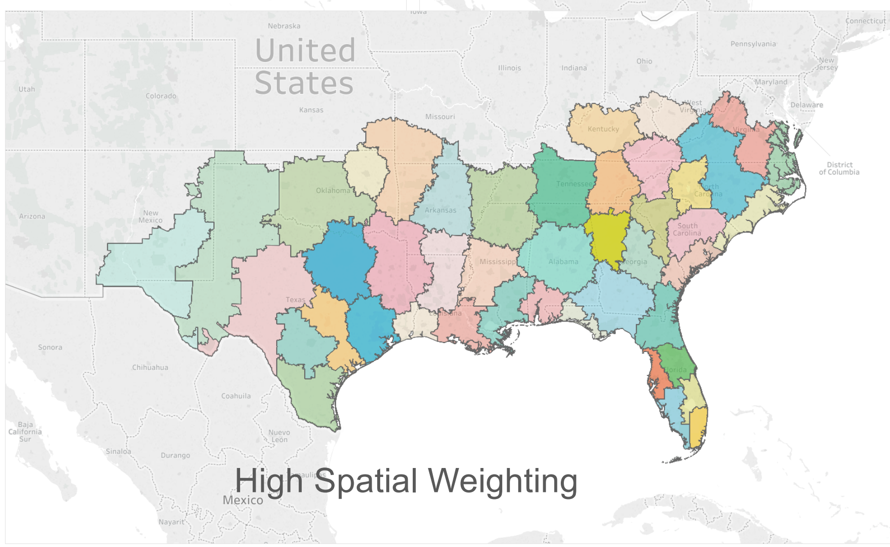
- Randomized (or can be non-random) algorithm to build out a network 'greedily'
- Easy to specify and tune parameters as you go
- **Use if:** Iteration is OK, exact solutions aren't required

PULLING IT TOGETHER

- SQL
- Python
  - Pandas
    - High performance data management/manipulation, SQL-like interface
  - Numpy
    - N-dimensional arrays, math libraries
  - Scikit-learn
    - Huge number of supervised and unsupervised ML algorithms prewritten
  - Networkx
    - Network/Graph analysis library
- Brute force


United States
Low Spatial Weighting
Mexico


United States
Medium Spatial Weighting


United States
High Spatial Weighting
Mexico

NETWORKX

https://networkx.github.io/

- Graph data structure with huge library of built-ins
  - Graph Operations
    - Edge/Node maintenance, weighting, node attributes, etc.
  - Graph Algorithms
    - Connectivity, Neighborhoods, k-core, max-flow, matching, bipartite, approximation algorithms, and on and on…
  - Linear algebra library that takes graph objects
    - Eigenvalue spectrums, laplacians, PageRank
  - Generators
    - Random graph generators (e.g. random normal, Erdős–Rényi, power law)
    - Canonical graphs (Karate club, Florentine families graph)
  - Visualization Tools

GREEDY ALGORITHM OVERVIEW

- Load data
- Using networkx, build an approximately-planar graph based on district mean locations
  - Find the norm of the district centers, pick n-closest
- Set parameters for “optimizer”
- Loop:
  - Pick manager with lowest score, assign them a random district that’s a neighbor as long as constraints are met
    - If that manager has no districts, pick a random district to add.
  - Simulated annealing—jostle where districts are in an attempt to avoid local minima, cooling over time
- Once all districts are assigned, score districts and reshuffle them to minimize variance

LOAD DATA

```python
#Import our data into dataframes
storeList = pd.read_csv(appDir + 'RMM Territories/rmm_map.txt', sep='|', na_values='?')


storeClass = pd.read_csv(appDir + 'RMM Territories/Str_Class_Sales.txt', sep='\t', na_values='?')
#storeList = pd.concat([storeList, storeClass['District Nbr']], join='inner', axis=0)
storeList = storeList.set_index(['STR_NBR'])
storeClass = storeClass.set_index(['Str Nbr'])
allStore = pd.concat([storeList, storeClass], join='inner', axis=1)
```

BUILD GRAPH

```
distrLocCenters = pd.concat([storeIndex['District'], distrList], join='inner', axis=1).groupby(['District']).mean()

distr_distances = np.zeros((distrLocCenters.shape[0],distrLocCenters.shape[0]))
for i in range(distrLocCenters.shape[0]):
        for j in range(distrLocCenters.shape[0]):
                if i!=j:
                    distr_distances[i,j] = np.linalg.norm(distrLocCenters.values[i,:]-distrLocCenters.values[j,:])
```

```
rmmAddr = pd.read_csv(appDir + 'RMM Territories/RMM_Cities_Coord_V2.csv')

rmmAddr = rmmAddr.loc[rmmAddr['DIV_NBR'] == divAssign]
n_Distr = rmmAddr.shape[0]

m_distances = np.zeros((distrLocCenters.shape[0], rmmAddr.shape[0]))

for i in range(rmmAddr.shape[0]):
    m_distances[:,i] += (np.linalg.norm(distrLocCenters[['LAT_NBR', 'LNG_NBR']].values - rmmAddr[['LAT', 'LNG']].values[i], axis=-1))



#Keep only n best neighbors
closestN = 4

for i in range(distrLocCenters.shape[0]):
    a = distr_distances[i].argsort()[closestN+1:]
    distr_distances[i, a] = 0

G2 = nx.from_numpy_matrix(distr_distances)
```

PARAMETERS AND CONTROLS

```
nx.set_node_attributes(G2, 'RMM', -1)
storeDollar = pd.read_csv(appDir + 'MerchFinance/RegionalRealignment/strDlr.csv', sep=',', na_values='?')
storeDollar = storeDollar.set_index(['STR_NBR'])
storeIndex = pd.concat([storeIndex, storeDollar], join='inner', axis=1)
#Get District Dollar Values
distrDollar = storeIndex[['SLS_DLR','District']].groupby(['District']).mean()
rmmStoreCount = storeIndex[['SLS_DLR','District']].groupby(['District']).count()



#initialize the variables.
nx.set_node_attributes(G2, 'sls', None)
nx.set_node_attributes(G2, 'distNbr', 0)
nx.set_node_attributes(G2, 'move', 0)
nx.set_node_attributes(G2, 'strcnt', 0)

for i in range(len(G2.nodes())):
    G2.node[i]['sls'] = np.round(distrDollar.values[i][0])
    G2.node[i]['strcnt'] = rmmStoreCount.values[i][0]
    G2.node[i]['distNbr'] = rmmStoreCount.index.values[i]
```

ITERATE AND BE GREEDY

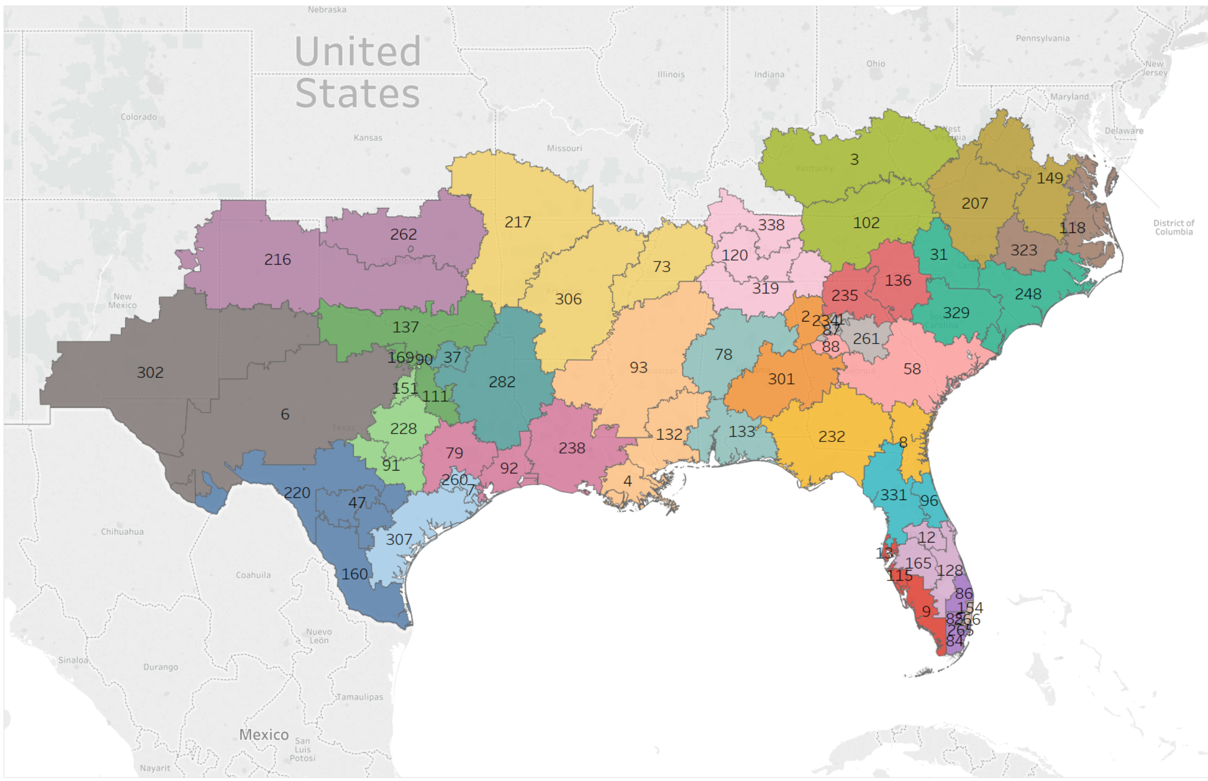
```
in_itns = 0
while ((rmmDistrInfo[rmm, 0] >= maxDistr or rmmDistrInfo[rmm,1]>maxDlr
        or rmmDistrInfo[rmm,2]>maxStr)
        and in_itns < 100):
    in_itns += 1
    possible = np.where(rmmDistrInfo[:, 2] <= rmmDistrInfo[:, 2].min()+5)[0]
    rmm = random.choice(possible)
```

```
else:
    di = distrs[np.random.randint(len(distrs))]
    #choose a random neighbor of that rmm
    ngh = list(G2[di])

    minMove = 999
    for i in ngh:
        if G2.node[i]['move'] < minMove:
            minMove = G2.node[i]['move']

    minDist = 999
    for i in ngh:
        if G2.node[i]['move'] == minMove:
            if distrAssignments[di] != distrAssignments[i]:
                if G2[di][i]['weight'] < minDist:
                    minDist = G2[di][i]['weight']
    mv = -1
    for i in ngh:
        if G2[di][i]['weight'] == minDist:
            mv = i

    dold = distrAssignments[mv]
```

- Pick a random manager from the ones that have approximately the lowest score
- Get a list of possible districts they could have, and randomly pick one of those
- Verify all the constraints (lots of IFs) are met
- Perform some simulated annealing along the way—some random chance to jostle districts from one manager to another adjacent manager occasionally to avoid local minima
- If all districts are assigned, still grab a local district if it improves your score more than it decreases your neighbor's score

RESULTS

Division 1 Map with "Optimal" RMM Hometown

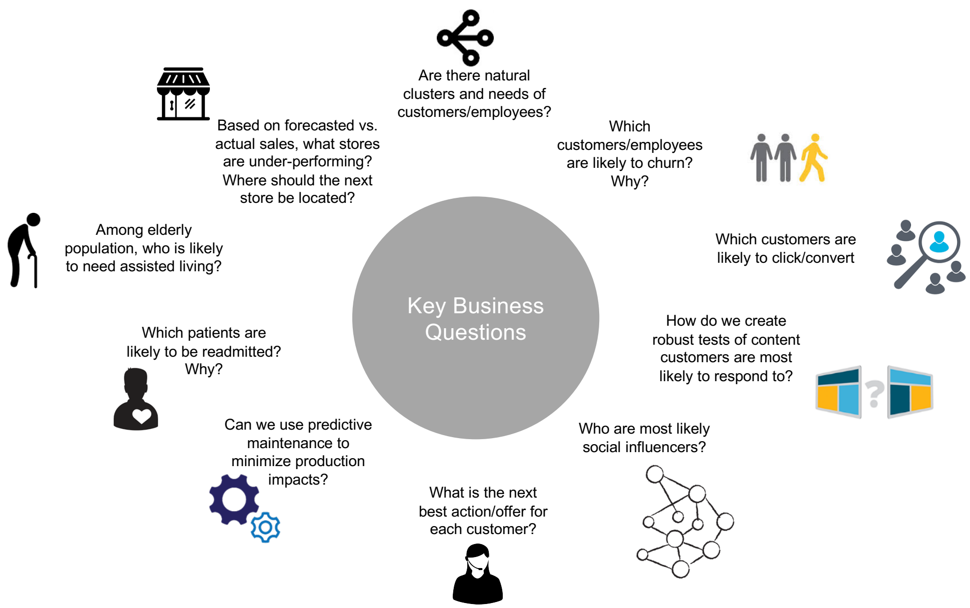


RMM	Distrs	Strs	RMM Hometown
0	3	21	0 - San Antonio, TX
1	3	20	1 - Houston, TX
2	3	26	2 - Atlanta, GA
3	3	25	3 - New Orleans, LA
4	3	26	4 - Dallas, TX
5	3	23	5 - Dallas, TX
6	2	23	6 - Roanoke, VA
7	3	27	7 - Memphis, TN
8	3	20	8 - Dallas, TX
9	2	20	9 - Birmingham, AL
10	2	25	10 - Greenville, SC
11	2	23	11 - Atlanta, GA
12	2	19	12 - Midland, TX
13	3	28	13 - Atlanta, GA
14	3	27	14 - Houston, TX
15	3	28	15 - Nashville, TN
16	2	16	16 - Oklahoma City..
17	3	27	17 - Orlando, FL
18	2	23	18 - Raleigh, NC
19	3	20	19 - Miami, FL
20	2	22	20 - Daytona Beac..
21	3	27	21 - Charlotte, NC
22	2	25	22 - Knoxville, TN
23	2	22	23 - Jacksonville, FL
24	3	24	24 - Tampa, FL
25	4	23	25 - Miami, FL

WHY DO IT THIS WAY?

- Explainable
  - Client has minimal experience and trust of advanced analytics, a simple algorithm makes it easier to get buy-in
- Repeatable, with little variation
  - Similar but not identical results allow fine-tuning / re-running to smooth out client concerns
- Very easy to tweak in live sessions
  - Simple code, simple algorithms mean you can modify on-the-fly in response to questions
- In this case, all solutions are approximations
  - There's no right answer

SOME OTHER PROJECTS



Advanced Analytics Toolkit

- Predictive/Explanatory modeling
- Behavioral segmentation
- Survey segmentation and projection
- Forecasting
- Pricing analytics
- Design of Experiments (A/B and MVT)
- Text/VOC analytics
- Social influence propensity

THANK YOU

www.northhighland.com

CHARLIE MORN

Sr. Data Analyst
North Highland
charlie.morn@northhighland.com

QUICK OIL CHANGE CHAIN

PROBLEM

Our client has a large base of customers that are "oil-only" and have never used them for mechanical services (e.g., belts, brakes, hoses)

SOLUTION

Develop a predictive model used to target customers most likely to convert so they can receive a differentiated experience on their next visit.

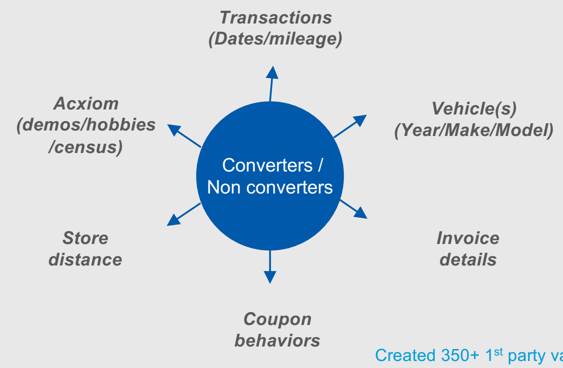
Perform deep data-mining of prevailing customer behaviors to identify ones that tend to lead to conversion and just as important, ones that might turn off customers (e.g., "over-selling")

A sound byte from the modeling process is that air filter replacement recommendations tend to turn customers off and reduce their chance of mechanical conversion by 25%.

RESULTS

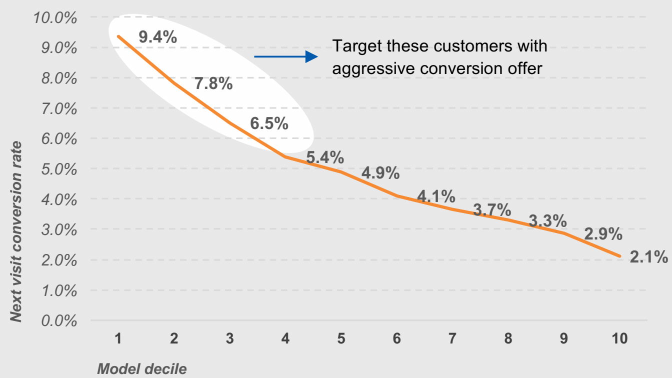
Paid back initial investment at two month mark (based on net EBIT)

At three months (mid-October 2016), converted 1,377 customers for a total of $350k net NEW mechanical revenue.



Created 350+ 1st party variables

PREDICTIVE MODEL PERFORMANCE



Theory matches reality

Decile 1 – Most likely to convert >> highest next visit conversion (9.4%)

Decile 10 – Least likely to convert >> lowest next visit conversion (2.1%)