U.S. Farmers' Markets

General Assembly, 2015
Los Angeles #7
Sharon Wang

Where to Sell?

If you were a local farmer starting your business, how would you know which of the nearby farmers' markets was best to sell your produce?

Getting Started

What I Wanted to Do

Create a Recommendation System to help farmers easily find the optimal location to sell perishables.

What I Really Did

Determine U.S. regions that are most likely to sell specific types of tracked produce at farmers' markets.

Gather Data

Getting the data

- USDA Public Data: National Farmers' Market Directory
- USDA Food Availability per Capita
- US Census: Population Estimates
- US Census: Poverty Estimates

FINDINGS

- National Farmers' Market Directory is voluntarily updated
  - Data as good as the Users inputting data
  - Updates are very frequent
  - USDA website functions like an Excel search
  - Obtained data through 'Export to Excel'

Significant time spent validating completeness and validity of data obtained.

CHALLENGES

- USDA Food Consumption per Capita is a very interesting combination to the information
  - Need more variable to relate datasets in a meaningful way
- US Census population, education and poverty data appeared challenging to merge
  - Already researched topics exist so know this is possible. Example: food

OTHER FUN THINGS...

```
cleanYear
2009    1815
2010     340
2011     754
2012     790
2013    1183
2014    2278
2015    1316
Name: FMID, dtype: int64

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x10a57b2b0>
```

```
region
                1
Midnorth     2351
Midsouth      820
Northeast    1984
Northwest     530
Southeast    1639
Southwest    1046
Territories   105
Name: FMID, dtype: int64

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x10a695710>
```

National Farmers' Market Directory

Data for Analysis:

- Location
  - Address
  - County
  - State
  - GIS
- Forms of Payment
- Seasonal Availability
- Produce
- Prepared Foods
- Beverages
- Other Specialities

Data Cleansing & Analysis

Try #1: Naive Bayes

- Why did I choose to use it?
  - Naive Bayes creates clusters and allows for predictions based on clusters. Given the information provided, this model was reviewed for suitability.
- Successful?
  - Nope.
  - While statistical metrics were obtained from running the models, the metrics were not the results intended. If a users executed this program, the program should response with an answer that resembles a value already existing in the dataset such as "Midnorth" or "Nashville".

Try #2: Logistic Regression

- Why did I choose to use it?
  - Logistic Regression is used to predict the probability of 0 or 1… no or yes… Since the farmers' market dataset consisted mostly of binary values, this model was evaluated for suitability.
- Successful?
  - Not really.
  - This worked best when I started testing probability of Social Media usage based on year. I would have preferred to calculate the probability of existence or not based on special products used. For example, it'd be very interesting to prove that Social Media was not used as much when cash was the only accepted form of payment or Social Media is always present when wine is sold.

Try #3: KNearestClassifier

- Why did I choose to use it?
  - The purpose of this model is intended to provide singular predictions, which seemed likely to achieve the objective in a more straight-forward manner.
- Successful?
  - Yes.
  - This was successful. When predicting the region most likely to sell Wild Harvested produce, the model output "Midnorth." From a business standpoint, the reasonable test checked. In further running this model for the region where the produce would not sell, where it could or could not sell by City, and then by County, all outputs passed the business reasonable test. Due to the success of Model #3, further tested with other produce.

jupyter GA-Project-PredictWildHarvested Last Checkpoint: Last Monday at 6:48 PM (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

In [10]:
```
from sklearn.neighbors import KNeighborsClassifier

print("Shape of Training Sets")
print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print()

neigh = KNeighborsClassifier(n_neighbors=1)
neigh.fit(X, y)
train_score = neigh.score(X_train,y_train)
cv_score = neigh.score(X_test,y_test)

print("Most Likely Region without Wild Harvested Produce: ", neigh.predict([[0]]))
print("Most Likely Region with Wild Harvested Produce: ", neigh.predict([[1]]))
print()
print("Predicted Probabilities: ",neigh.predict_proba([[3000]]))
print()
print("Train Score: ",train_score)
print("CV Score: ",cv_score)
```

```
Shape of Training Sets
X_train:  (6780, 1)
y_train:  (6780,)

Most Likely Region without Wild Harvested Produce:  ['Northeast']
Most Likely Region with Wild Harvested Produce:  ['Midnorth']

Predicted Probabilities:  [[ 0.  1.  0.  0.  0.  0.  0.  0.]]

Train Score:  0.236578171091
CV Score:  0.237028301887
```

```
Shape of Training Sets
X_train:  (6780, 1)
y_train:  (6780,)

Most Likely State without Wild Harvested Produce:  ['Pennsylvania']
Most Likely State with Wild Harvested Produce:  ['Iowa']

Predicted Probabilities:  [[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.
   0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]

Train Score:  0.0346607669617
CV Score:  0.0400943396226
```

```
Shape of Training Sets
X_train:  (6780, 1)
y_train:  (6780,)

Most Likely City without Wild Harvested Produce:  ['Mount Bethe']
Most Likely City with Wild Harvested Produce:  ['Mt. Pleasant']

Predicted Probabilities:  [[ 0.  0.  0. ...,  0.  0.  0.]]

Train Score:  0.000147492625369
CV Score:  0.000589622641509
```

The answers were pretty good. However, it doesn't put the model into the best use.

Region Where Cheese is Mostly Likely Sold

In [9]:
```
from matplotlib.colors import ListedColormap
from sklearn.cross_validation import train_test_split

# Create the train and test sets
X = markets2[['cleanCheese']].values
y = markets2.region.ravel()

# create 80%-20% train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
Shape of Training Sets
X_train:  (6780, 1)
y_train:  (6780,)

Most Likely State without Cheese:  ['Arizona']
Most Likely State with Cheese:  ['Florida']

Predicted Probabilities:  [[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.
   0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
   0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]]

Train Score:  0.0199115044248
CV Score:  0.0165094339623
```

State Where Cheese is Most Likely Sold

In [24]:
```
# Create the train and test sets
X = markets2[['cleanCheese']].values
y = markets2.State.ravel()

# create 80%-20% train-test split
X_train, X_test, y_train, y_test = train_t
```

In [25]:
```
print("Shape of Training Sets")
print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print()

neigh = KNeighborsClassifier(n_neighbors=1)
neigh.fit(X, y)
train_score = neigh.score(X_train,y_train)
cv_score = neigh.score(X_test,y_test)

print("Most Likely State without Cheese: ", neigh.predict([[-1]]))
print("Most Likely State with Cheese: ", neigh.predict([[3000]]))
print()
print("Predicted Probabilities: ",neigh.predict_proba([[3000]]))
print()
print("Train Score: ",train_score)
print("CV Score: ",cv_score)
```

```
Shape of Training Sets
X_train:  (6780, 1)
y_train:  (6780,)

Most Likely Region without Cheese:  ['Southwest']
Most Likely Region with Cheese:  ['Southeast']

Predicted Probabilities:  [[ 0.  0.  0.  0.  0.  0.  1.  0.]]

Train Score:  0.148377581121
CV Score:  0.143867924528
```

nom.

nom.

nom.

Try #4: KMeans

- Why did I choose to use it?
  - The purpose of this model is intended to provide singular prediction based on creation of clusters. Since the latitude and longitude for each farmers' market was provided in an interpreted fashion, the grid-like data can be used to facilitate location predictions.
- Successful?
  - Sort of.
  - This was not successful (yet). Pandas successfully plotted all farmers' markets across the US in a visual shape similar to a map of the US. However, in validating the data values with Google Maps, confirmed the x and y provided are not literal latitude and longitude. Instead, they are pre-interpreted for a coordinate plot. As of the presentation for this project, the model has not successfully predicted the next x and y coordinates. However, I am optimistic this is the right model to complete the prediction.

Not new news.

```
In [198]: markets['new_x'] = markets['x'].replace('NaN',markets['x'].mean())
          markets['new_y'] = markets['y'].replace('NaN',markets['y'].mean())

In [199]: #Attempt scatter plot

          plt.scatter(markets['new_x'],markets['new_y'])

Out[199]: <matplotlib.collections.PathCollection at 0x10df4ff60>
```

jupyter GA-Project-ReviewLocations Last Checkpoint: Last Saturday at 5:34 PM (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

In [206]:
```
import numpy as np
from scipy.cluster.vq import kmeans,vq
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt


mcluster = m.values
kmeans = KMeans()
kmeans.fit(mcluster)

k_range = range(1,10)

k_means_var = [KMeans(n_clusters=k).fit(mcluster) for k in k_range]
centroids = [X.cluster_centers_ for X in k_means_var]
k_euclid = [cdist(mcluster, cent, 'euclidean') for cent in centroids]
dist = [np.min(ke,axis=1) for ke in k_euclid]

avgWithinSS = [sum(d)/X.shape[0] for d in dist]
avgWithinSS
```

Out[206]:
```
[7127.9802982581741,
 2863.0298278306632,
 1466.0377106101482,
 955.04067542482755,
 717.37117423405925,
 585.22097914528104,
 482.11063109676257,
 409.74519349509626,
 363.19769495143214]
```

In [207]:
```
kIdx = 2
# plot elbow curve
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(k_range, avgWithinSS, 'b*-')
ax.plot(k_range[kIdx], avgWithinSS[kIdx], marker='o', markersize=12,
      markeredgewidth=2, markeredgecolor='r', markerfacecolor='None')
plt.grid(True)
plt.xlabel('Number of clusters')
plt.ylabel('Average within-cluster sum of squares')
tt = plt.title('Elbow for K-Means clustering')
```

In [213]:
```
km = KMeans(3, init='k-means++') # initialize
km.fit(mcluster)
c = km.predict(mcluster)

print(c)
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=3, n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
[2 0 0 ..., 2 2 0]
```

… what is this saying?

That Data Product you asked for…

… blog post to come after the algorithms are settled and models operating.

Medium: @SNWang

Next Milestones…

- Finalize the algorithms and replicate for all features.
- Create the ensemble of predictions to allow the system to present the User a business analysis report.
  - Goal of the business analysis report is to facilitate decision making.
  - Assumes web interface for Users.
- Determining monetization of the public service. Donations or service payment required?

"When we try to pick out anything by itself, we find it hitched to everything else in the Universe."

–John Muir, My First Summer in the Sierra