Classifying Architectural Building Typologies Using Unsupervised Learning

MLND Capstone

Jim Peraino November 12th, 2017

I. Definition

Project Overview

In the domains of architecture and urban planning, building typology is an often referenced idea when analyzing neighbordhoods, buildings, and public spaces. Historically, buildings have been classified in two primary ways: by form (or shape), and by use (what happens inside). In the early 19th century, the architect Jean-Nicolas-Louis Durand proposed the notion of architectural typology in his book Precis des Lecons d'Architecture (1). Since then, many efforts have been made to apply machine learning techniques to Urban Design and Architectural analysis, including with software such as GIS (2) and spatial recognition systems for identifying ideal sites for new development projects (3).

As an architect myself, I'm curious about ways that we can quantify qualitative information about buildings, and vice versa. There are relatively few datasets in the field of architecture, and establishing and automating methods for classifying buildings, in particular into qualitative categories, will be essential in this endeavor. Being able to classify a building as a certain typology enables many other kinds of analysis. This information can be used to assess the characteristics of a neighborhood, predict property values, identify areas for new development, or countless other tasks related to the built environment.

Process and code is based on Udacity Machine Learning Nanodegree Project: "Customer Segments"

Problem Statement

Databases of building typologies do not exist for most of the world. To exacerbate this problem, buildings often no longer limit themselves to the typical typologies (schools, homes, hospitals, churches, for instance), and instead, are often mixed-use. However, physical charactistics of buildings can be easily observed or deduced from satelite images (3).

The problem, then, is: How can we automate the process of classifying buildings into their respective typologies by analyzing this physical information?

The problem is quantifiable and measurable because we can logically assess any individual building as to the makeup of its use. It is a problem that is replicable anywhere on earth that there are buildings, since google earth shows satelite images from which we can mine physical data.

Unsupervised learning can be used on this task. As was used in the Customer Segment project, this solution can follow the process of: Data Exploration, Data Preprocessing, Feature Transformation, Clustering, and drawing conlusions(5). For more detail, see the project design section.

New York City's Department of City Planning puts out a dataset called The Primary Land Use Tax Lot Output (PLUTO), which contain information about plots of lands, the characteristics of the buildings on that land, and various administrative districts. This dataset was obtained as a download from Kaggle (4). This dataset contains information for over 65,000 lots.

Since my goal is to use physical characteristics to classify buildings into typological groups, I will test different combinations of Physical Characteristics to see which best enable clustering of the data into building typologies. I will then compare the percentages of commercial/residential/office etc. floor areas in each cluster to better understand and evaluate the clusters.

Metrics

The evaluation metric will be the extent to which each cluster is homogenous in terms of its program (residential, retail, etc). This can be determined by calculating the standard deviation of the primary programmatic element in each cluster. A smaller standard deviation would indicate that the cluster is more homogeneous, and a larger standard deviation would indicate that it is heterogeneous.

II. Analysis

Data Exploration

Dataset Features

While the dataset has roughly 80 variables for each lot, those of particular interest are:

Physical Characteristics (Used to create the model)

- Total Building Floor Area
- Lot Area
- Number of Floors
- Lot Frontage
- Lot Depth
- Building Frontage
- Building Depth
- Floor Area Ration (FAR)

Other Data (Used as the benchmark)

- Commercial, Residential, Office, Retail, Factory, Storage, Garage, and Other Floor Area
- Building Class
- Zoning District

In [1]:
```
### SOURCE: Code is based on Udacity MLND Customer Segments Project
# Import libaries necessary for this project
import numpy as np
import pandas as pd
from IPython.display import display

# Import supplementary visualizations code visuals code visuals.py
import visuals as vs

# Pretty display for notebooks
%matplotlib inline

# Define column sets
col_Import = ["Address", "SchoolDist", "LotArea", "BldgArea", "BldgFront", "BldgDepth", "NumFloor
s", "NumBldgs", "LotFront", "LotDepth", "ComArea", "ResArea", "OfficeArea", "RetailArea", "Garage
Area", "StrgeArea", "FactryArea"]
col_PhysAll = ["LotArea", "BldgArea", "BldgFront", "BldgDepth", "NumFloors", "NumBldgs", "LotFron
t", "LotDepth"]
col_Phys = ["BldgFront", "BldgDepth"]
#col_PhysPlus = ["LotArea", "BldgArea", "NumBldgs", "FAR2", "NumFloors"]
#col_PhysPlus = ["BldgFront", "BldgDepth", "BldgArea", "LotArea"]
col_PhysPlus = ["BldgFront", "BldgDepth", "FlrShape", "FAR2", "NumFloors"]

# Load the PLUTO Buildings Dataset for Brooklyn
try:
    data = pd.read_csv("BK.csv",
                       low_memory=False,
                       usecols= col_Import)


    print "PLUTO Brooklyn Dataset has {} samples with {} features each.".format(*data.shape)

except:
    print "Dataset could not be loaded."
```

```
PLUTO Brooklyn Dataset has 277131 samples with 17 features each.
```

In [2]:
```
# Display a description of the dataset
display(data.describe())
```

	SchoolDist	LotArea	BldgArea	ComArea	ResArea	OfficeArea	RetailArea	Garage
count	276810.000000	2.771310e+05	2.771310e+05	2.771310e+05	2.771310e+05	2.771310e+05	2.771310e+05	2.771310
mean	19.169708	6.028453e+03	5.177061e+03	1.460424e+03	3.637264e+03	2.927963e+02	2.485272e+02	8.346268
std	3.911379	4.300028e+05	5.477153e+04	5.126787e+04	1.870863e+04	7.991499e+03	3.746235e+03	3.386011
min	13.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000
25%	16.000000	1.900000e+03	1.648000e+03	0.000000e+00	1.400000e+03	0.000000e+00	0.000000e+00	0.000000
50%	20.000000	2.134000e+03	2.368000e+03	0.000000e+00	2.052000e+03	0.000000e+00	0.000000e+00	0.000000
75%	21.000000	2.833000e+03	3.312000e+03	0.000000e+00	2.860000e+03	0.000000e+00	0.000000e+00	0.000000
max	32.000000	2.073912e+08	2.400000e+07	2.400000e+07	1.800000e+06	1.239070e+06	1.263000e+06	1.285000

Dataset Statistics As noted in the table above, there is a large range in the physical characteristics in the data. The number of floors in the middle 50% of the data ranges from 2-3 floors, however the maximum number of floors is 119. This makes sense when thinking about a city composed mostly of small buildings but with several taller buildings near its downtown area. This trend persists throughout the building characteristics, with characteristics such as Lot Front and Lot Depth staying relatively consistently between 20 and 100' for more than 50% of the data, but with significant shifts at the upper end (with some lot depths more than a mile long). Tall buildings are not outliers in this case, since they are a distinct typology, they just occur less often, so we will need to be sure to pick samples that account for this.

Abnormalities The dataset includes all lots in Brooklyn, even those without buildings. These will need to be removed as outliers/irrelevant because including them results in several features being listed with values of zero (Number of Floors, Building Front, etc). Notably, large buildings should *not* be removed as outliers since they may correspond to a specific type of building.

Empty lots are removed from the data below.

With over 200,000 samples, the data set is not very facile. After experimentation, I've reduced the data to about 10,000 items by limiting the study to a zone defined by School District 13.

Selecting Samples

Several representative building types are identified to put real buildings to the data.

- Single Family Row House: 48 HICKS STREET
- Church: 113 REMSEN STREET
- Warehouse: 127 CONCORD STREET
- Bar: 708 MYRTLE AVENUE

In [3]:
```
# Identify addresses for samples
addresses = ['127 CONCORD STREET', '48 HICKS STREET', '113 REMSEN STREET', '708 MYRTLE AVENUE']

indices = []

# Get index of each sample
for i in addresses:
    index = data[data.Address == i].index[0]
    indices.append(index)

# Create a DataFrame of the chosen samples
samples = pd.DataFrame(data.loc[indices], columns = data.keys()).reset_index(drop = True)
print "Chosen samples from PLUTO database:"
display(samples)
```

```
Chosen samples from PLUTO database:
```

	SchoolDist	Address	LotArea	BldgArea	ComArea	ResArea	OfficeArea	RetailArea	GarageArea	StrgeArea	Factr
0	13.0	127 CONCORD STREET	6900	7650	7650	0	0	0	0	7650	0
1	13.0	48 HICKS STREET	1819	4682	1100	3582	0	1100	0	0	0
2	13.0	113 REMSEN STREET	2500	5496	5496	0	5496	0	0	0	0
3	13.0	708 MYRTLE AVENUE	4529	4000	4000	0	0	4000	0	0	0

In [4]:
```
# Limit samples to sites in school district 13 with only one building and within certain physical characteristics thresholds
data = data.loc[data['NumBldgs'] == 1]
data = data.loc[data['NumFloors'] > 0]
data = data.loc[data['BldgArea'] > 2500]
data = data.loc[data['BldgArea'] < 10000]
data = data.loc[data['BldgFront'] > 0]
data = data.loc[data['BldgDepth'] > 0]
data = data.loc[data['BldgFront'] < 400]
data = data.loc[data['BldgDepth'] < 400]
data = data.loc[data['SchoolDist'] == 13]

# Display a new description of the dataset
print "The reduced PLUTO Brooklyn Dataset has {} samples with {} features each.".format(*data.shape)
display(data.describe())
```

```
The reduced PLUTO Brooklyn Dataset has 9917 samples with 17 features each.
```

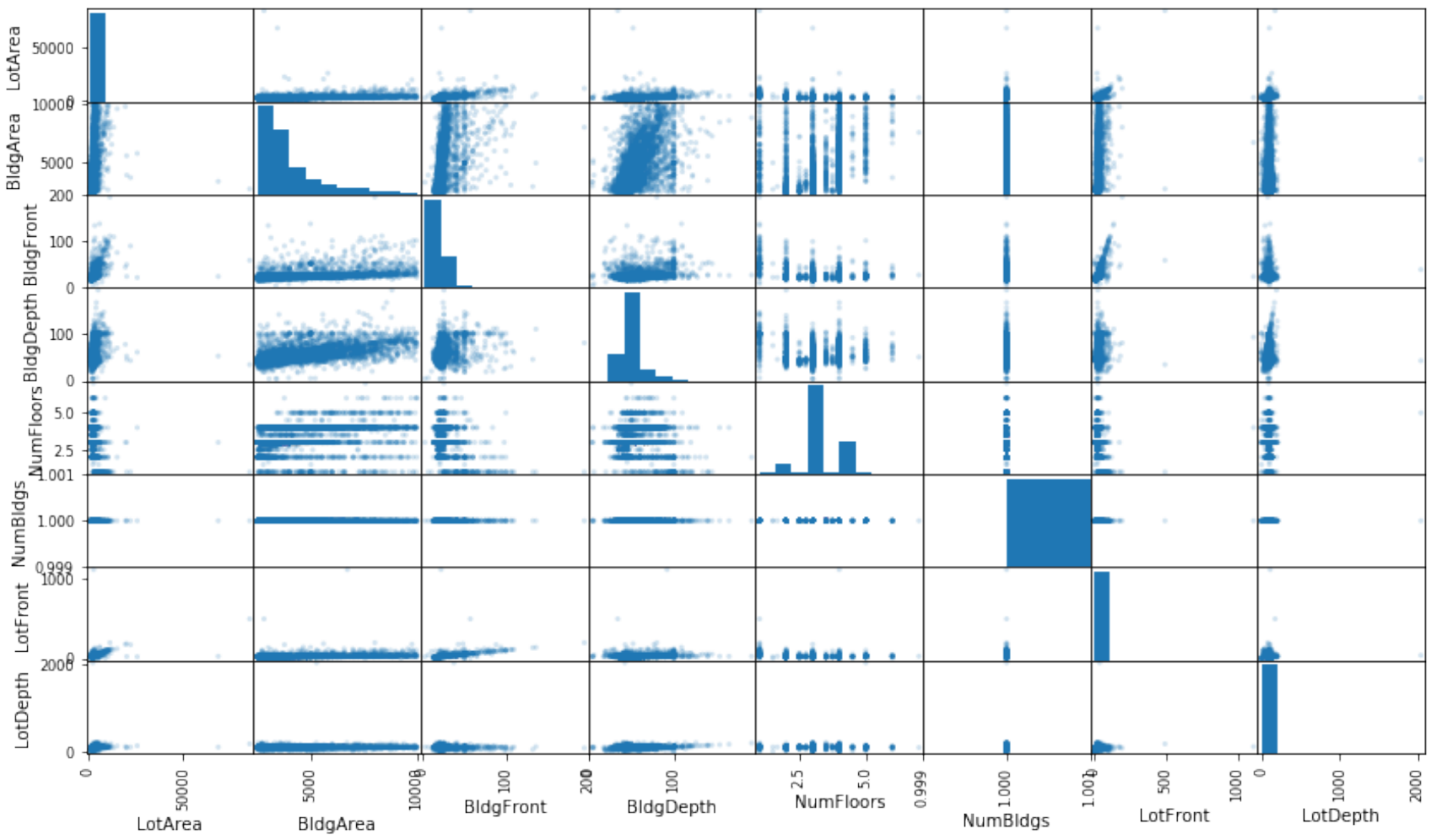
	SchoolDist	LotArea	BldgArea	ComArea	ResArea	OfficeArea	RetailArea	GarageArea	Str
count	9917.0	9917.000000	9917.000000	9917.000000	9917.000000	9917.000000	9917.000000	9917.000000	9917.
mean	13.0	2165.642533	3978.408289	420.441767	3242.642432	43.769588	210.337300	18.076233	26.10
std	0.0	1436.405865	1476.650759	1248.366401	1620.947807	409.194583	706.907882	303.380337	355.7
min	13.0	640.000000	2502.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	13.0	1800.000000	3000.000000	0.000000	2415.000000	0.000000	0.000000	0.000000	0.000
50%	13.0	2000.000000	3516.000000	0.000000	2850.000000	0.000000	0.000000	0.000000	0.000
75%	13.0	2225.000000	4400.000000	0.000000	3652.000000	0.000000	0.000000	0.000000	0.000
max	13.0	85289.000000	9975.000000	9966.000000	9975.000000	8728.000000	9944.000000	9800.000000	9600.

These sample data are similar to the original data set.

Exploratory Visualization

The scatter matrix below allows us to undertand correlations between the seven identified physical features.

In [5]:
```
pd.plotting.scatter_matrix(data[col_PhysAll], alpha = 0.2, figsize = (14,8));
```



From this visualization, it is clear that, as might be expected, many of these variables have a linear correlation. This makes sense, since these features are all related to size. A building that has an extremely large depth will also likely have a large front. A building with more floors will likely have a higher area.

It is clear, then, that some additional preprocessing is necessary to tease out some additional relationships. While size is of course an important factor, it is not the only way that we can predict typology. Ratios of building depth to building front may give a clue as to the shape of the building, for instance, which could be a predictor.

Algorithms and Techniques

There are multiple clustering algorithms to consider, including K-Means clustering and Gaussian Mixture Model (GMM) clustering. While K-Means clustering is relatively easy to imlement and easy to understand, it is a hill-climbing algorithm, so the output may vary depending on the starting point. GMM allows some overlap between clusters, and also allows clusters to take on different shapes. It might be costlier to run, but it is less sensitive to outliers.

I will select GMM because this data does not have clear clusters that would result with a k-means process, so some overlap or sharing will be necessary. We also know that there are outliers in the data, and GMM is less sensitive to those.

I will follow a similar process to the Customer Segments project. In order to run this, first, I will preprocess the data. Then, I will first create clusters using an sklearn GMM algorithm. Then, I will calculate silhouette scores for multiple numbers of clusters to determine the best number.

Benchmark

Existing methods for classifying building typology enter the use of each building one by one, through analysis of the use. In the case of this dataset, this data is included, and will serve as the benchmark model. This data will not be used in constructing the new model, but will instead be used to compare its performance. The benchmark model should have near 100% accuracy, as each item is entered individually. We expect that the unsupervised learning model will have significantly lower accuracy than the benchmark since it will be using only the information that can be gained from physical analysis.

III. Methodology

Data Preprocessing

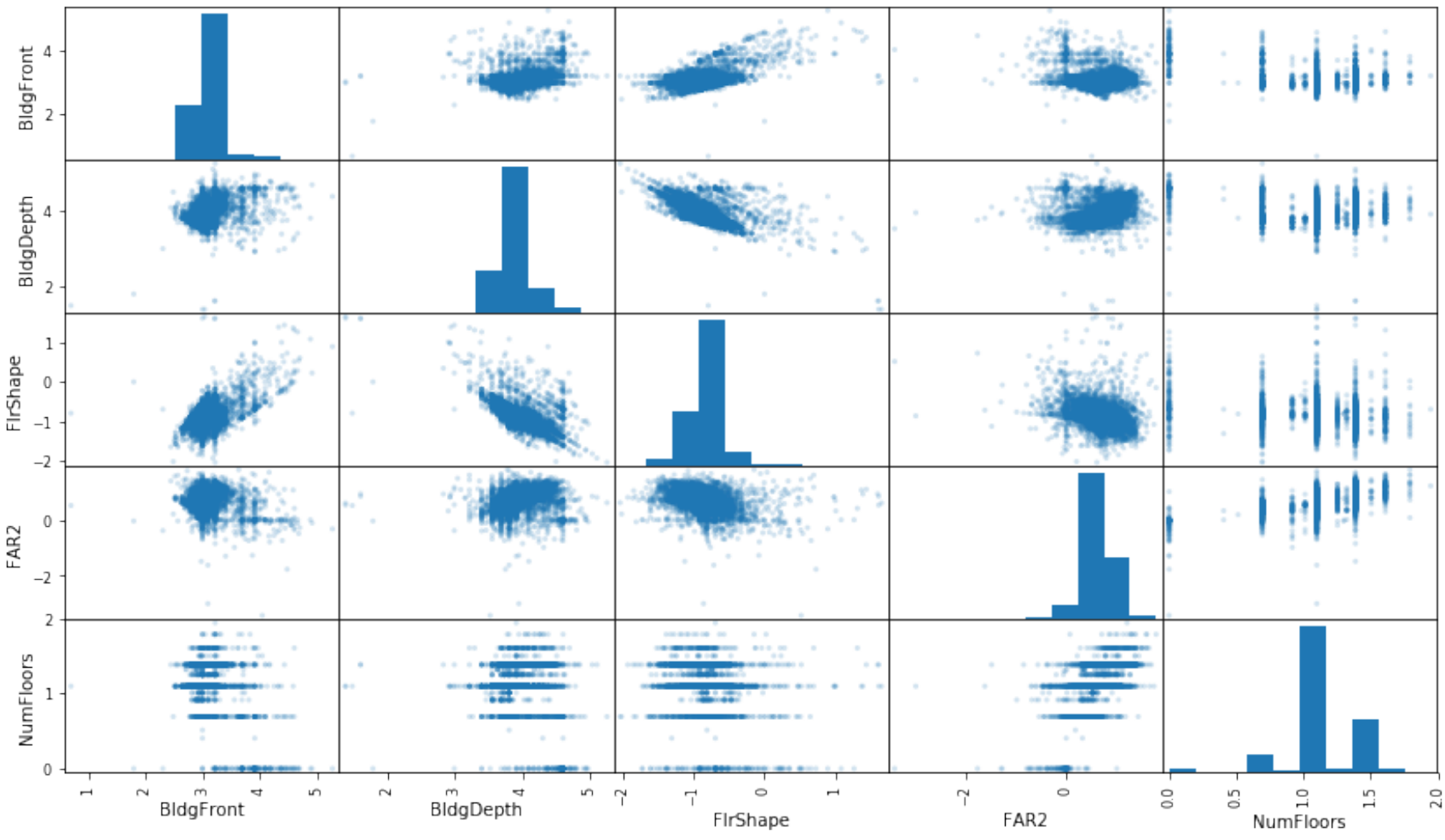
As mentioned in the exploratory visualization section, some additional features need to be created in order to understand the relationship between variables for each building. I will create two new variables:

- **Width to Depth Floor Shape ratio (FlrShape)** = BldgFront / BldgDepth
- **Floor Area Ratio** (FAR) = BldgArea / LotArea

In [6]:
```
# Add new variables to dataset
data['FlrShape'] = data.BldgFront / data.BldgDepth
data['FAR2'] = data.BldgArea / data.LotArea

# Add new variables to samples
samples['FlrShape'] = samples.BldgFront / samples.BldgDepth
samples['FAR2'] = samples.BldgArea / samples.LotArea
```

Then, we can apply non-linear feature scaling to ensure that one feature does not outweight the others.

In [7]:
```
# Scale the reduced data using the natural logarithm
exclude = ['Address']
log_data = np.log(data[col_PhysPlus])

# Scale the sample data using the natural logarithm
log_samples = np.log(samples[col_PhysPlus])

# Produce a scatter matrix for each pair of newly-transformed features
pd.plotting.scatter_matrix(log_data, alpha = 0.2, figsize = (14,8));
```

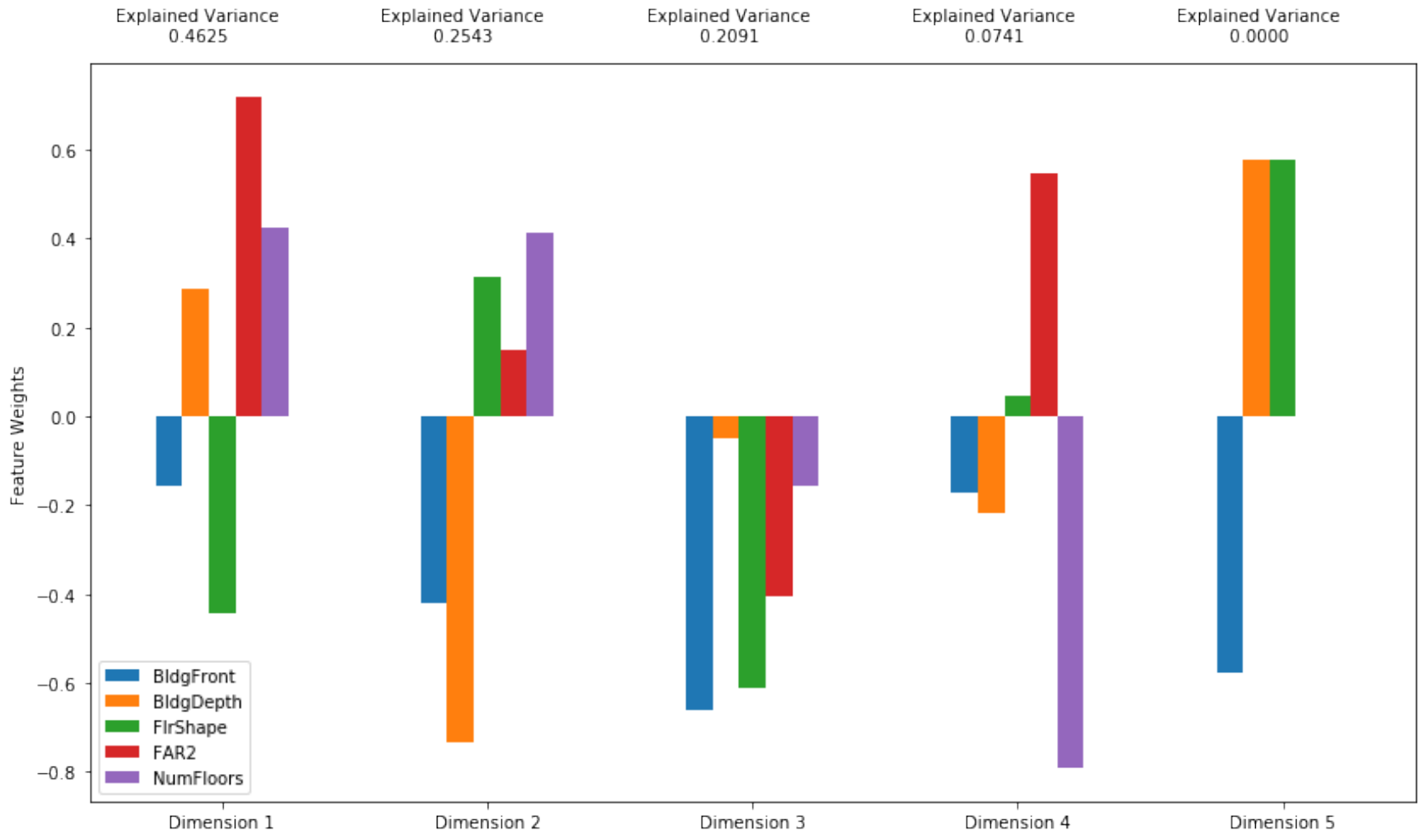


We can see that Floor Shape is correlated with Building Front and that FAR is related most with Building Area and Number of floors, both of which are intuitive. Unlike in the previous scatterplot, there are fewer linear relationships (except in the newly constructed variables), meaning that more variance can be described by BldgFront, BldgDepth, and NumFloors than other variables that are related.

Implementation

Feature Transformation By completing a Principal Component Analysis, we can understand which dimensions best maximize variance.

In [8]:
```
#Apply PCA by fitting the data with the same number of dimensions as features
from sklearn.decomposition import PCA
dataCurated = log_data[col_PhysPlus]

# Eliminate rows with NaN in FAR2 column
good_data = dataCurated[np.isfinite(data['FAR2'])]

pca = PCA()
pca.fit(good_data)

# Transform log_samples using the PCA fit above
pca_samples = pca.transform(log_samples)

# Generate PCA results plot
pca_results = vs.pca_results(good_data, pca)
```



In [9]:
```
#Display sample log-data after having a PCA transformation applied
display(pd.DataFrame(np.round(pca_samples, 4), columns = pca_results.index.values))
```

	Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5
0	-0.3067	-1.4313	-0.2410	-0.3153	0.0
1	0.1713	0.2459	-0.4573	-0.0193	0.0
2	0.4507	-0.3332	-0.0869	-0.2435	0.0
3	-0.8664	-1.3902	0.0631	0.2214	0.0

Dimensionality Reduction

As demonstrated in the graph above, more than 70% of the variance can be explained by two dimensions. Therefore, we can reduce the complexity of the problem by reducing the number of dimensions to two, while still being able to explain the majority of the variance.

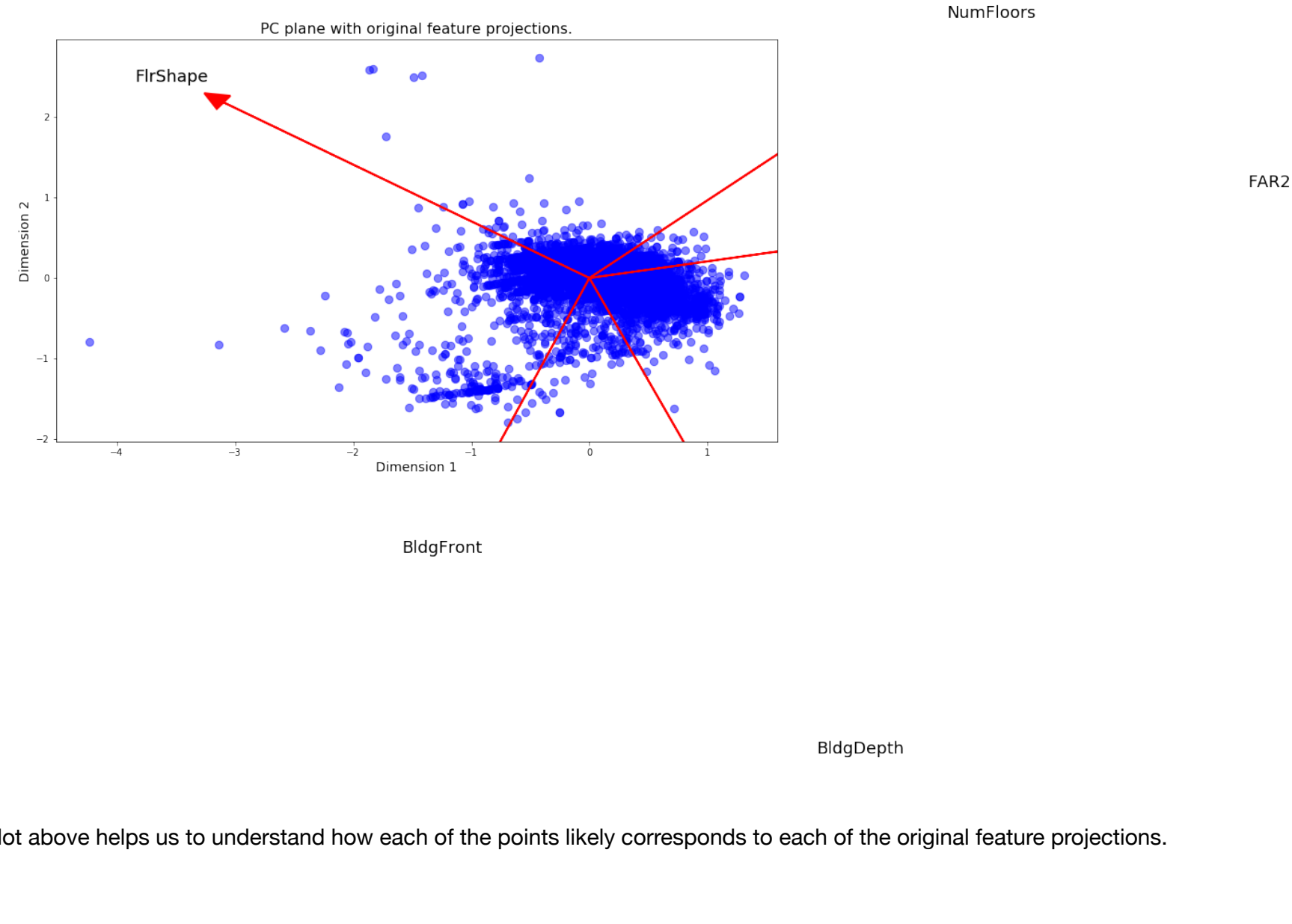
In [10]:
```
pca = PCA(n_components=2)
pca.fit(good_data)

# Transform the good data using the PCA fit above
reduced_data = pca.transform(good_data)

# Transform log samples using the PCA fit above
pca_samples = pca.transform(log_samples)

# Create a dataFrame for the reduced data
reduced_data = pd.DataFrame(reduced_data, columns = ['Dimension 1', 'Dimension 2'])
```

In [11]:
```
# Display sample log-data after applying PCA transformation in two dimensions
display(pd.DataFrame(np.round(pca_samples, 4), columns = ['Dimension 1', 'Dimension 2']))
```

	Dimension 1	Dimension 2
0	-0.3067	-1.4313
1	0.1713	0.2459
2	0.4507	-0.3332
3	-0.8664	-1.3902

In [12]:
```
vs.biplot(good_data, reduced_data, pca)
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x118f1a950>
```



The scatterplot above helps us to understand how each of the points likely corresponds to each of the original feature projections.

Clustering

The next step is to generate the clusters. First, we need to determine the number of clusters. By calculating the mean silhouette coefficient for each number of clusters, we can determine how well it explains the data. Below, I've calculated the silhouette score for each of several amounts of clusters.

In [13]:
```
from sklearn.mixture import GMM

def testNumClusters(numClusters):

    from sklearn.mixture import GMM
    clusterer = GMM(n_components=numClusters, random_state = 35)
    clusterer.fit(reduced_data)

    # TODO: Predict the cluster for each data point
    preds = clusterer.predict(reduced_data)

    # TODO: Find the cluster centers
    centers = clusterer.means_

    # TODO: Predict the cluster for each transformed sample data point
    sample_preds = clusterer.predict(pca_samples)

    # TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
    from sklearn.metrics import silhouette_score
    score = silhouette_score(reduced_data, preds)

    print numClusters,': %.3f' % score
    return;

for i in range (2, 11):
    testNumClusters(i)
```

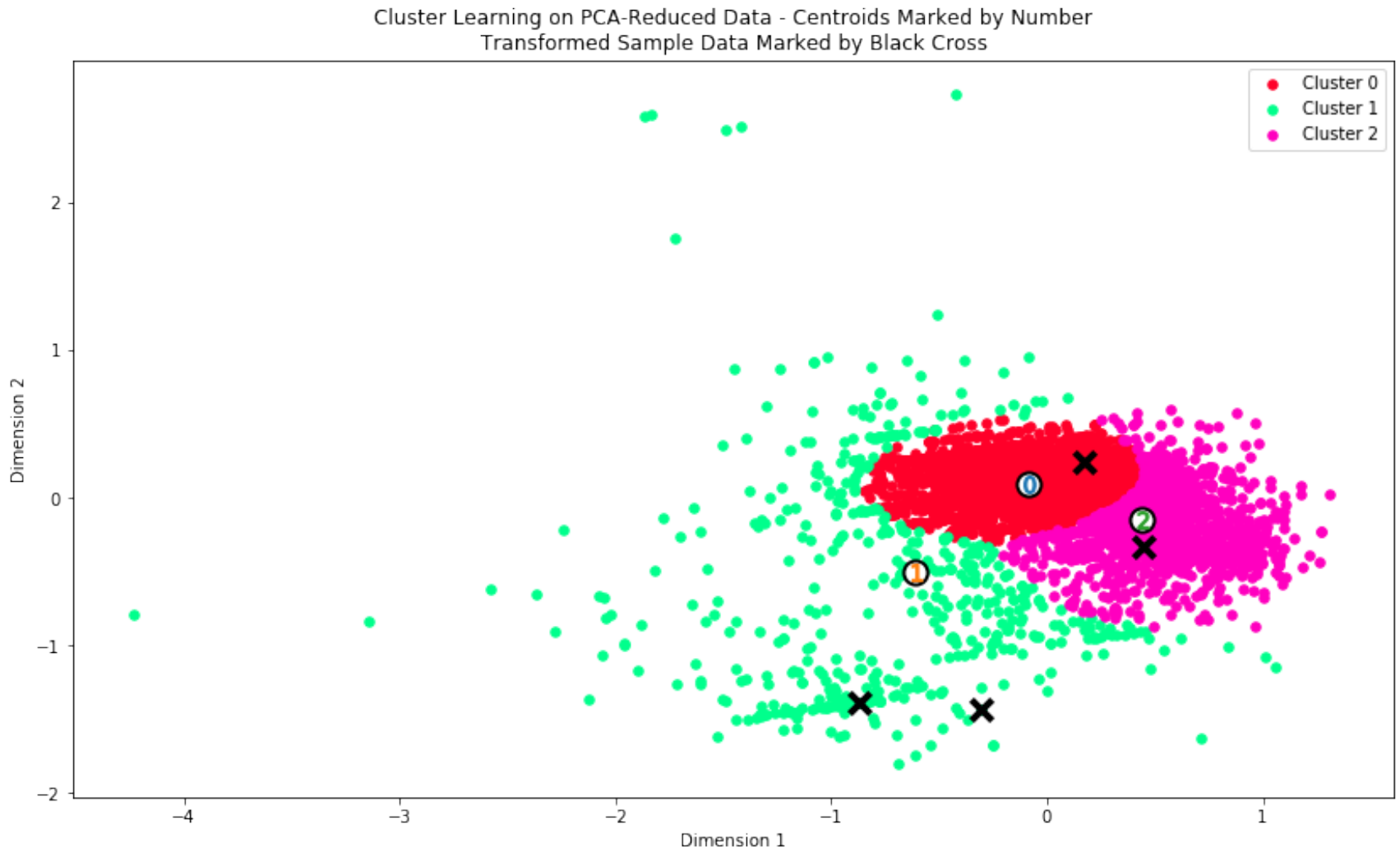
```
2 : 0.497
3 : 0.486
4 : 0.352
5 : 0.308
6 : 0.314
7 : 0.291
8 : 0.245
9 : 0.247
10 : 0.253
```

Although 2 clusters has the best silhouette score, I have identified more than two typologies in the sample data, and want to see how well the model can predict each of 3 different typologies.

In [19]:
```
from sklearn.mixture import GMM
numClusters = 3


clusterer = GMM(n_components=numClusters, random_state = 35)
clusterer.fit(reduced_data)

# TODO: Predict the cluster for each data point
preds = clusterer.predict(reduced_data)

# TODO: Find the cluster centers
centers = clusterer.means_

# TODO: Predict the cluster for each transformed sample data point
sample_preds = clusterer.predict(pca_samples)

# TODO: Calculate the mean silhouette coefficient for the number of clusters chosen
from sklearn.metrics import silhouette_score
score = silhouette_score(reduced_data, preds)

print numClusters,': %.3f' % score
```

```
3 : 0.486
```

In [20]:
```
# Display the results of the clustering from implementation
vs.cluster_results(reduced_data, preds, centers, pca_samples)
```



In [21]:
```
# TODO: Inverse transform the centers
log_centers = pca.inverse_transform(centers)

# TODO: Exponentiate the centers
true_centers = np.exp(log_centers)

# Display the true centers
segments = ['Segment {}'.format(i) for i in range(0,len(centers))]
true_centers = pd.DataFrame(np.round(true_centers), columns = good_data.keys())
true_centers.index = segments
display(true_centers)
```

	BldgFront	BldgDepth	FlrShape	FAR2	NumFloors
Segment 0	21.0	45.0	0.0	2.0	3.0
Segment 1	29.0	59.0	0.0	1.0	2.0
Segment 2	21.0	61.0	0.0	2.0	4.0

The scatterplot and graph above show the clusters and their centers. By inversely transforming the centers to undo the log transform, we can understand what each cluster represents.

Segment 0's center has a relatively small building front and building depth, with 3 floors. This likely corresponds with a single family home.

Segment 1's center has a relatively large building front and building depth, with an FAR of 1. This likely corresponds with a warehouse-like building.

Segment 2's center has a relatively small building front but a longer building depth, and is taller. This likely corresponds with a retail or commercial building.

Refinement

The process indicated above is the result of several rounds of iterative refinement. In particular, several strategies were used for refining the model's performance.

- **Limiting data**: Initially, I had considered the full dataset, which made processing times too large. Then, I tried reducing the number of points through randomly culling indices, but found that that eliminated variation in the data. Then, I tried limiting the data by culling out data that was over or below a certain square footage threshold. This also resutled in limited variation. Finally, I settled on only using data from a certain school district, which reduced the number of samples below 10,000.
- **Adjusting number of Dimensions**: Initially, I noticed that the Principal Component Analysis indicated that additional dimensions could continue to better describe the data, up to 4 dimensions. However, when I adjusted the process to include 3 and 4 dimensions, I found that it did not improve my results. Rather than spliting my sample points into different clusters, these sample points always remained within one or two clusters when the dimensions were increased.
- **Testing Features to Include**: Several Multiple combinations of features were tested, including:
  - ["LotArea", "BldgArea", "NumBldgs", "FAR2", "NumFloors"]
  - ["BldgFront", "BldgDepth", "BldgArea", "LotArea"]
  - ["BldgFront", "BldgDepth", "FlrShape", "FAR2", "NumFloors"]

    Ultimately, this last combination resulted in the highest silhouette scores for 3 or more clusters, and the best accuracy for my sample points.

IV. Results

Model Evaluation and Validation

In [23]:
```
# Display the predictions
for i, pred in enumerate(sample_preds):
    print "Sample point", i, "predicted to be in Cluster", pred
```

```
Sample point 0 predicted to be in Cluster 1
Sample point 1 predicted to be in Cluster 0
Sample point 2 predicted to be in Cluster 2
Sample point 3 predicted to be in Cluster 1
```

The tests above test the sample points to see which clusters they fall within. They fall into the following segments:

Segment 0 likely corresponds with a single family home. Sample point 1, which is a single family home, is located within this cluster, supporting this conclusion.

Segment 1 likely corresponds with a warehouse-like building. Sample points 0 and 3, which are a warehouse and church, respectively, are located within this cluster, supporting this conclusion.

Segment 2 likely corresponds with a retail or commercial building. Sample point 2, which is a retail building, is located within this cluster, supporting this conclusion.

Justification

The final results are not stronger than the benchmark, but this is understandable. I wanted to see how well a system could classify building typologies without using data about what goes on inside of it. The benchmark does use data about what goes on inside of it, and is therefore accurate about its use nearly 100% of the time. My model does successfully classify some types of residences, retail, and office buildings, but does lose some accuracy when buildings have similar proportions in plan--for instance, it is not able to distinguish between a church and a factory, which both may be similar dimensions in plan, and are both technically only one story tall.

V. Conclusion

Free-Form Visualization

(See visualizations above in clustering secion)

Reflection

By using an unsupervised learning process on a subset of data about a city's building stock, we can begin to classify buildings by typology. In particular, I was interested to find certain cases where buildings were classified to have similar forms, but had different uses, such as a church and a factory.

Difficulties included working with the dataset, and carefully considering only certain columns of the data without removing them from the set so that they could be referenced later. Also, finding a good balance of features to include for accuracy took a lot of trial and error.

The final solution does meet my expectations, given that I intentially limited the type of data I could use from the analysis. There are better methods to solve this problem that use more data, however, as I will discuss in the improvement section, there are more ways to improve this method.

Improvement

Further improvements would likely revolve around the type of data that can be used. I think that a better computer vision software could take into account the shape of buildings (square vs. courtyard vs. H, etc) which could then be one-hot encoded as features to better describe the shape, since building lenght and depth are relatively blunt descriptions of buildings. This kind of algorithm was one that I researched but did not know how to implement. I think a better solution, that uses better data through more advanced computer vision could be a better solution for this problem.

References:

(1) https://books.google.com/books?hl=en&lr=&id=wilTAAAAcAAJ&oi=fnd&pg=PA1&dq=precis+durand&ots=hY1zElBwh2&sig=clePUvTcCCBGFLNlBVFp0j4XPJw#v=onepage&q=prec (https://books.google.com/books?hl=en&lr=&id=wilTAAAAcAAJ&oi=fnd&pg=PA1&dq=precis+durand&ots=hY1zElBwh2&sig=clePUvTcCCBGFLNlBVFp0j4XPJw#v=onepage&q=prec)

(2) https://pdfs.semanticscholar.org/24c5/4f200302943cde042aee677956adc0b2498e.pdf (https://pdfs.semanticscholar.org/24c5/4f200302943cde042aee677956adc0b2498e.pdf)

(3) http://certainmeasures.com/spatial_recognition.html (http://certainmeasures.com/spatial_recognition.html)

(4) https://www.kaggle.com/new-york-city/nyc-buildings (https://www.kaggle.com/new-york-city/nyc-buildings)

(5) Udacity MLND Customer Segment Project