FOCUS ARTICLE

Geochemical Facies Analysis Using Unsupervised Machine Learning

Brendon Hall
ENTHOUGHT, HOUSTON, TX

This tutorial will demonstrate step-by-step how to build a practical geoscience machine learning workflow using Python and scikit-learn. The practice of extracting insights from large datasets has certainly caught mainstream attention in recent years. Software packages are emerging with multivariate statistics and machine learning algorithms built in to workflows. As this trend continues, it is important to understand how these algorithms work, know their strengths and limitations, and avoid treating them as a black box.

Python has become a popular programming language for scientific applications as it is straightforward to learn, has a large community of users and contributors that have created many open (and free) scientific libraries. The scikit-learn library provides a state-of-the-art collection of machine learning tools for Python and contains many tools for data mining and analysis including classification, clustering, and regression algorithms.

In this tutorial, we demonstrate how dimensionality reduction and unsupervised machine learning can be used to analyze X-ray fluorescence measurements of cuttings samples. Dimensionality reduction is the process of reducing the number of random variables under consideration by obtaining a smaller set of features that describe most of the variation within the dataset. Clustering is an unsupervised machine learning technique that learns a grouping from the data itself and doesn’t require training data. The groups consist of samples with similar characteristics, which can be considered as distinct geochemical facies. This will be implemented using Python, the scikit-learn library (Pedregosa et al. 2011), and other popular Python data science tools. These tools are open source and available online. Enthought’s Canopy provides an easy to use interface to install these packages (and many more), as well as an integrated analysis and visualization environment. It is available for free at www.enthought.com/product/canopy/.

The first step is importing the tools we need for this tutorial.

```python
import pandas as pd
import numpy as np

# Machine Learning Libraries
from sklearn.preprocessing import scale
from sklearn.decomposition import FactorAnalysis
from sklearn.cluster import KMeans

# Visualization Libraries
import matplotlib.pyplot as plt
import seaborn as sns
```

XRF Cuttings Analysis

X-ray fluorescence (XRF) is becoming a common wellsites cuttings analysis technique (Carr et al, 2014). Portable XRF provides rapid elemental composition measurements of cuttings as they are drilled. These devices measure the fluorescent x-rays emitted by a sample when illuminated by an energetic X-ray source. Elements in the sample will emit X-rays at specific wavelengths, and the spectrum of the emitted x-rays can be measured and used to quantify the amount of the corresponding elements present in the sample. Trends in element concentration can be used to infer sediment depositional environment, sources, and indicate conditions conducive to the preservation of organic material. XRF data can be used for geologic characterization, optimizing well placement, and providing additional guidance for geosteering.

The data for this tutorial consists of XRF measurements of cuttings from the lateral section of an unconventional well. The measurements were made at approximately 10 m intervals. In this case, the raw data consists of 22 measurements for each sample. Each measurement gives the weight percentage of a chemical component. The data is read from a csv file into a dataframe using the pandas library, which provides many convenient data structures and tools for data science.

```python
geochem_df = pd.read_csv('XRF_dataset.csv')
```

A dataframe is a 2-dimensional labeled data structure, where each row represents a different sample, and the columns are measurements on the sample. The contents of the dataframe are shown in Table 1.

<table>
<thead>
<tr>
<th>Well Name</th>
<th>Depth</th>
<th>Quartz</th>
<th>...</th>
<th>SO3</th>
<th>Cl</th>
<th>Zr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Well 1</td>
<td>3173.97</td>
<td>27.56</td>
<td>...</td>
<td>1.20</td>
<td>0.28</td>
</tr>
<tr>
<td>1</td>
<td>Well 1</td>
<td>3183.11</td>
<td>42.92</td>
<td>...</td>
<td>0.81</td>
<td>0.26</td>
</tr>
<tr>
<td>2</td>
<td>Well 1</td>
<td>3192.26</td>
<td>44.55</td>
<td>...</td>
<td>0.76</td>
<td>0.23</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>266</td>
<td>Well 1</td>
<td>6255.50</td>
<td>45.04</td>
<td>...</td>
<td>0.97</td>
<td>0.02</td>
</tr>
<tr>
<td>267</td>
<td>Well 1</td>
<td>6273.78</td>
<td>41.21</td>
<td>...</td>
<td>1.05</td>
<td>0.02</td>
</tr>
<tr>
<td>268</td>
<td>Well 1</td>
<td>6296.64</td>
<td>46.72</td>
<td>...</td>
<td>0.77</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 1. Partial contents of the dataframe after loading the dataset. The entire dataset consists of 269 cuttings samples and 22 measurements of each sample.
In machine learning, the term feature is used to refer to the attributes of the objects being considered that can be used to describe, cluster, and classify them. In this case, the objects being studied are cuttings samples, and the features are the 22 XRF measurements. We can use feature engineering to augment this dataset. This refers to the process of using domain knowledge to create new features that help machine learning algorithms discriminate the data.

In geochemistry, elements are used as proxies that give hints to the physical, chemical or biological events that were occurring during rock formation. Ratios of certain elements can indicate the relative strength of various effects. We will focus on three ratios: Si/Zr, Si/Al, and Zr/Al. The Si/Zr ratio is used to indicate relative proportions of biogenic silica and terrestrial detrital inputs and the Si/Al ratio is used as a proxy for biogenic silica to aluminous clay (Croudace and Rothwell, 2015). The Zr/Al ratio is a proxy for terrigenous input; chemical behavior of Zr suggests that this ratio can be used as a proxy for grain size (Calvert and Pederson, 2007).

```
g = geochem_df['Si/Zr'] = geochem_df['SiO2'] / geochem_df['Zr']
g = geochem_df['Si/Al'] = geochem_df['SiO2'] / geochem_df['Al2O3']
g = geochem_df['Zr/Al'] = geochem_df['Zr'] / geochem_df['Al2O3']
```

### Dimensionality Reduction

Not surprisingly, multivariate datasets contain plenty of variables. This richness can be used to explain complex behaviour that can’t be captured with a single observation. Multivariate methods allow us to consider changes in several observations simultaneously. With many observations, it is quite likely that the changes we observe are related to a smaller number of underlying causes. Dimensionality reduction is the process of using the correlations in the observed data to reveal a more parsimonious underlying model that explains the observed variation.

Exploratory factor analysis (EFA) reduces the number of variables by identifying the underlying latent factors present in the dataset. These factors cannot be measured directly; they can be determined only by measuring manifest properties. For example, in the case of our geochemical dataset, a shaliness factor could be associated with high silicon dioxide, calcium and quartz readings. EFA assumes that the observations are a linear combination of the underlying factors, plus some Gaussian noise (Hauck, 2014). A related dimensionality reduction technique is principal component analysis (PCA). PCA determines components that are weighted linear combinations of the observations and makes the assumption that the components themselves account for the all of the observed variance in the data; none of the variance is caused by error (or noise). EFA assumes that there is variance due to error (Rietveld and Van Hout, 1993).

Before applying EFA, the features should be rescaled by standardization. This involves rescaling each feature in the dataset to have zero mean and unit variance. This is necessary for many machine learning algorithms. Consider the case where measurements were made using different scales; one component varies between 10 and 300 [ppm] and another varies between 0.1 and 0.3 [wt%]. The variation in the former seems 1000 times greater than the latter if we don’t account for the scale. This will skew the factor analysis and the weights assigned to each factor. The preprocessing module in the scikit-learn library has tools for rescaling the dataset.

```
from sklearn.preprocessing import scale
data = geochem_df.ix[:, 2:]
data = scale(data)
```

EFA requires that the number of factors to be extracted is specified a priori. Often, it is not immediately obvious how many factors should be specified. Many authors have proposed rules over the years (e.g., Preacher et al, 2013). One simple approach (known as the Kaiser criterion) involves looking at eigenvalues of the data’s covariance matrix and counting the number above a threshold value (typically 1.0). The following code snippet will compute and plot the eigenvalues.

```
covar_matrix = np.cov(data, rowvar=False)
eigenvalues = np.linalg.eig(covar_matrix)[0]
plt.plot(eigenvalues, 'o-')
plt.axhline(y=1.0, color='r', linestyle='--')
```

The resulting plot is shown in Figure 1. There are 6 eigenvalues greater than 1.0 (dashed red line), suggesting there are 6 relevant factors to be extracted.

![Figure 1. Eigenvalues of covariance matrix.](image)

The scikit-learn library contains a factor analysis module that can be used to extract the 6 factors. This is done by creating a factor analysis model and fitting the model to the data.

```
f = FactorAnalysis(n_components = 6)
f = f.fit(data)
factor_data = f.transform(data)
```

### Interpreting the factors

The factors can now be examined to interpret the underlying properties they represent. The factor loadings describe the relationship of each measurement to the underlying factor. Each loading score represents the correlation between the factor and the observed variables. The loading scores vary between -1 and 1. A positive value means that a measurement is associated with the presence of an underlying factor. A negative value suggests that a measurement indicates the absence of a factor. The factor loadings can be easily extracted from the factor model and plotted to show the loadings associated with a given factor.
Continued from Page 27

```python
loading = fa_model.components_
component_names = geochem_df.columns.values[2:]
loading_df = pd.DataFrame(loading,
columns=component_names)
#
plot the 5th factor
loading_df.ix[4].plot(kind='bar')
```

Figure 2 shows the factor loadings associated with the fifth factor. In this case, the factor is associated with high values of plagioclase, illite/smectite/mica, pyrite, and organic material, and the absence of calcite and MgO. We could interpret this factor as the organic-rich clay content. Similar interpretations can be given to the other factors by observing their loading scores.

![Factor loadings associated with the 5th factor.](image)

**Figure 2.** Factor loadings associated with the 5th factor.

### Clustering

The factor analysis has reduced the initial collection of 25 XRF features to a set of 6 underlying factors. A logical next step would be using these factors to group the cuttings samples by their common geochemical traits. These groups can be interpreted to represent geochemical facies. Cluster analysis is a suitable approach for assigning a common facies label to similar samples. Clustering attempts to group samples so that those in the same group (or cluster) are more similar than those in other clusters. Cluster analysis is one class of techniques that fall under the category of unsupervised machine learning. These approaches are used to infer structure from the data itself, without the use of labeled training data to guide the model.

The K-Means algorithm clusters data by trying to separate samples into k groups of equal variance. The algorithm locates the optimal cluster centroids by minimizing the distance between each point in a cluster and the closest centroid. The algorithm has three steps. It initializes by picking locations for the initial k centroids (often random samples from the dataset). Next, each sample is assigned to one of the k groups according to the nearest centroid. New centroids are then calculated by finding the mean values of each sample in each group. This is repeated until the difference between subsequent centroid positions falls below a given threshold.

Similar to EFA, K-Means requires that the number of clusters is specified before running the algorithm. There are a number of approaches to finding the optimal number of clusters. The goal is to choose the minimum number of clusters that accurately partition the dataset. These range from the relatively simple ‘elbow method’ to more rigorous techniques involving the Bayesian information criterion and optimizing the Gaussian nature of each cluster (Hamerly and Elkan, 2003). The following code demonstrates the ‘elbow method’ applied to this dataset. The sum of the squared distance of each point to the nearest cluster centroid (called inertia in scikit-learn) is plotted for an increasing number of clusters. As the number of clusters is increased and better fit the data, error is decreased. The elbow of the curve represents the point of diminishing returns where increasing the number of clusters does not reduce the error appreciably. Figure 3 suggests that about 6 clusters would be adequate for this dataset.

```python
inertias = []
means = []
maxK = 30
for k in range(1, maxK):
kmeans = KMeans(n_clusters=k, random_state=0).fit(factor_data)
means.append(k)
inertias.append(kmeans.inertia_)
plt.plot(means, inertias, 'o-')
```

**Figure 3.** Mean squared error vs. number of clusters for the XRF dataset.

The K-means algorithm in scikit-learn is used to cluster the reduced dataset. Similar to the factor analysis, this is done by creating a K-means model and fitting the factor dataset.

```python
kmeans = KMeans(n_clusters=6, random_state=0)
kmeans.fit(factor_data)
# add the cluster ids to the dataset
geochem_df['Cluster'] = kmeans.labels_ + 1
```

### Interpreting the clusters

Each sample in the dataset has now been assigned to one of six clusters. If we are going to interpret these clusters as geochemical facies, it is useful to inspect the geochemical signature of each cluster. Figure 4 contains a series of box plots that show the distribution of a small subset of measurements across each of the 6 clusters. The box plots depict 5 descriptive statistics; the horizontal lines of the colored rectangle show the first quartile, median, and third quartile. The arms show the minimum and maximum. Outliers are shown as black diamonds. This plot is generated using the statistical visualization library seaborn (https://seaborn.pydata.org/).

```python
sns.boxplot(x='Cluster', y='Calcite', data=geochem_df)
```
Visualizing results

Now the cuttings sample have been organized into 6 geochemical facies (clusters). We can visualize the classification in a well log style plot to better understand how the facies are ordered vertically in a well. The right column of Figure 5 shows the clusters assigned to each sample using a unique color, indexed by measured depth (MD). The other columns show 4 of the corresponding geochemical measurements. Similar plots could be made for the other wells in the dataset and used to identify common intervals.

One application of this analysis is providing data that can be used for geosteering horizontal wells. This is particularly useful in areas that lack a distinctive gamma ray signature. Wellsite classification of cuttings sample could be used to interpret a well’s path through an existing chemo-stratigraphic framework. To build a visualization of this framework, it is helpful to plot the geochemical facies along the well path. Figure 6 shows the trajectory (TVD vs. MD) of the well, with the different facies colored using the same scheme as Figure 5. This can be used to build a pseudo-vertical profile and help identify specific zones as the well porpoises up and down along its length.

This tutorial has demonstrated how dimensionality reduction and unsupervised machine learning can be used to understand and analyze XRF measurements of cuttings to determine geochemical facies. Exploratory factor analysis yields insight into the underlying rock properties that are changing across the reservoir. K-means clustering is used to organize similar samples into a smaller number of groups that can be interpreted as geochemical facies. This can be used to correlate formation tops between wells and provide data necessary for geosteering.

**Figure 4.** Distribution of illite/smectite/mica (a), organic material (b), calcite (c) and MgO (d) across the clusters.

**Figure 5.** Log style plot showing 4 geochemical measurements and cluster assignments.
The code shown in this article demonstrates how to perform most of this analysis using Python and some standard data science and machine learning toolboxes. For the full code used to generate these results and visualizations, please see the GitHub repo at www.github.com/brendonhall/clustering_tutorial.

References


About the Author

Brendon Hall is the director of the Energy Solutions Group at Enthought, focused on applying machine learning and physics based modeling to create scientific software tools for the oil and gas industry. He has over 10 years of experience using computational techniques to solve problems in geoscience. Brendon has a Ph.D. in mechanical engineering from the University of California, Santa Barbara, a B.Eng. in mechanical engineering and a B.Sc. in computer science from Western University in Canada.