In [ ]:

```
from sklearn.datasets import load_digits
from sklearn.cross_validation import train_test_split
import numpy as np
np.set_printoptions(suppress=True)

digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target)
```

Removing mean and scaling variance

In [ ]:

```
from sklearn.preprocessing import StandardScaler
```

1) Instantiate the model

In [ ]:

```
scaler = StandardScaler()
```

2) Fit using only the data.

In [ ]:

```
scaler.fit(X_train)
```

3) `transform` the data (not `predict`).

In [ ]:

```
X_train_scaled = scaler.transform(X_train)
```

In [ ]:

```
X_train.shape
```

In [ ]:

```
X_train_scaled.shape
```

The transformed version of the data has the mean removed:

In [ ]:

```
X_train_scaled.mean(axis=0)
```

In [ ]:

```
X_train_scaled.std(axis=0)
```

In [ ]:

```
X_test_transformed = scaler.transform(X_test)
```

Principal Component Analysis

0) Import the model

In [ ]:

```
from sklearn.decomposition import PCA
```

1) Instantiate the model

In [ ]:

```
pca = PCA(n_components=2)
```

2) Fit to training data

In [ ]:

```
pca.fit(X_train)
```

3) Transform to lower-dimensional representation

In [ ]:

```
print(X_train.shape)
X_pca = pca.transform(X_train)
X_pca.shape
```

Visualize

In [ ]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_train)
```

In [ ]:

```

```