Non-negative matrix factorization (NMF)

This is a dimension reduction technique, but unlike PCA, but models are interpretable

It is required that sample features are non-negative

NMF expresses images as combinations of patterns, and documents as combination of common themes

Uses fit/transform, but number of components must be specified

This has use cases for images, audio, etc.

```
from sklearn.decomposition import NMF
model = NMF(n_components=2)
model.fit(samples)
nmf_features=model.transform(samples)

# if you multiply features and components, you can reconstruct the original
samples as a product of matrices
```

NMF learns interpretable parts

tf-idf identifies word-frequency (tf is for word frequency, idf reduces common works like "the")

Grayscale images have only shades of gray, represented by a value between 0 (black) and 1 (white)

2D arrays of grayscale images values can be flattened to 1D

So a collection of images of the same size can be encoded as a 2D array, each row corresponds to an image, and each column corresponds to a pixel

We can recover a flattened array

```
print(sample)
bitmap = sample.reshape((2,3))
print(bitmap)

from matplotlib import pyplot as plt
plt.imshow(bitmap, cmap='gray', interpolation='nearest')
plt.show()
```

Building recommender systems using NMF

Compare NMF feature values between articles

We can compare documents with cosine similarity, to compare vectors assigned to articles

Higher values means more similarity

```
from sklearn.preprocessing import normalize
norm_features = normalize(nmf_features)
# if has index 23
current_article = norm_features[23, :]
similarities = norm_features.dot(current_article)

import pandas as pd
df = pd.DataFrame(norm_features, index=titles)
current_articles = df.loc['Dog bites man']
similarities = df.dot(current_article)

# find articles with highest cosine similarity
print(similarities.nlargest())
```