# Advanced Data Analysis

DATA 71200

Class 7: Unsupervised Learning (Principal Component Analysis, Non-Negative Matrix Factorization, and Manifold Learning)

# Schedule

| | |
|---|---|
| **24-Jun** | **Unsupervised Learning (Dimensionality Reduction & Feature Extraction, and Manifold Learning)**<br>**Ethics (Part One)**<br>***Project 2 Due*** |
| 25-Jun | DataCamp: AI Ethics |
| 26-Jun | Unsupervised Learning (Clustering)<br>Ethics (Part Two) |
| 2-Jul | *Project 3 Due* |
| 12-Jul | *Final Paper Due*<br>*Data Camp Due* |

# Project 2 (Due Today)

## Project 2

Application of two supervised learning techniques on the dataset you created in Project 1. This assignment should be completed as a Jupyter notebook in your GitHub repository.

To **submit** the assignment, submit a link to your project Jupyter notebook on Blackboard.

The **goal** for this assignment is to apply different types of supervised learning algorithms with a range of parameter settings to your data and to observe which performs better.

**Step 1: Load your data, including testing/training split from Project 1.**

- Your testing and training split should be balanced
- Your data should be clean and missing data should be addressed

**Step 2: (If not already done in Project 1) Prepare your data**

- Make sure that your all appropriate variables are converted to categorical variables (as ordinal or one hot)
- Perform any necessary feature scaling

**Step 3: Examine your target attribute**. Based on the data exploration you did in Project 1, confirm and examine the attribute you are going to predict.

- Examine and plot the distribution of the target attribute in your training set (e.g., is it Gaussian, uniform, logarithmic). This will help you interpret the performance of different algorithms on your data.

**Step 4: Selected two of the following supervised learning algorithms, ideally one from the first half of the list and one from the second half of the list**

- K-Nearest Neighbor
- Linear Models
- Naïve Bayes
- Decision Trees
    - Single tree
    - Random Forest
    - Gradient Descent decision trees
- Support Vectors Machines

**Step 5: For each of your selected models**

- Run with the default parameters using cross-validation
    - Calculate precision, recall, and F1
- (Where possible) adjust 2-3 parameters for each model using grid search
    - Report evaluation metrics for the best and worst-performing parameter settings

**Tip:** You should make notes on what worked well and what didn't. Such notes will be useful when you write up the paper for your final project.

# Project 2 Rubric

| | Missing | Fair | Good | Excellent |
|---|---|---|---|---|
| **Data Prepatation** | **0** (0.00%) | **2** (13.33333%)<br>Missing saling and encoding, where appropriate, or data is inaccessible | **2.5** (16.66666%)<br>Missing saling or encoding, where appropriate, or data is inaccessible | **3** (20.00%)<br>Data is accessible, scaling and encoding done, where appropriate |
| **Testing/Training** | **0** (0.00%) | **0.5** (3.33333%)<br>Not properly balanced or not used properly | **0** (0.00%) | **1** (6.66666%)<br>Properly balanced and used |
| **Target variable** | **0** (0.00%) | **0.5** (3.33333%)<br>Selected but not appropriate | **0** (0.00%) | **1** (6.66666%)<br>Selected and appropriate |
| **Algorithm 1- Default** | **0** (0.00%) | **1.5** (10.00%)<br>Run without cross-validation or appropriate evaluation metrics | **1.75** (11.66666%)<br>Run without either cross-validation or appropriate evaluation metrics | **2** (13.33333%)<br>Run with both cross-validation and appropriate evaluation metrics |
| **Algorithm 1 - Parameters** | **0** (0.00%) | **1.5** (10.00%)<br>Run without 2 parameter adjustments and appropriate evaluation metrics | **1.75** (11.66666%)<br>Run with either at least 2 parameter adjustments and appropriate evaluation metrics | **2** (13.33333%)<br>Run with at least 2 parameter adjustments and appropriate evaluation metrics |
| **Algorithm 2 - Default** | **0** (0.00%) | **1.5** (10.00%)<br>Run without cross-validation or appropriate evaluation metrics | **1.75** (11.66666%)<br>Run with either cross-validation or appropriate evaluation metrics | **2** (13.33333%)<br>Run with both cross-validation and appropriate evaluation metrics |
| **Algorithm 2 - Parameters** | **0** (0.00%) | **1.5** (10.00%)<br>Run with neither 2 parameter adjustments and appropriate evaluation metrics | **1.75** (11.66666%)<br>Run with either at least 2 parameter adjustments and appropriate evaluation metrics | **2** (13.33333%)<br>Run with at least 2 parameter adjustments and appropriate evaluation metrics |
| **Comments** | **0** (0.00%) | **1.5** (10.00%)<br>Present but not detailed enough | **1.75** (11.66666%)<br>Largely complete and/or some detail missing | **2** (13.33333%)<br>Complete with sufficient detail |

# Unsupervised Learning

- **A set of algorithms that learn representations or partitions of unlabeled data**
  - In the absence of labels, evaluation is challenging
    - Often performed through visualization
- **Useful for**
  - Exploratory data analysis
  - Pre-processing data

# Curse of Dimensionality

- **Large number of features makes training slow and it is hard to find robust solutions**
- **High-dimensional representations tend to be sparser than low-dimensional representations**
  - Meaning that new data tends to be further away than existing data — increasing the risk of overfitting

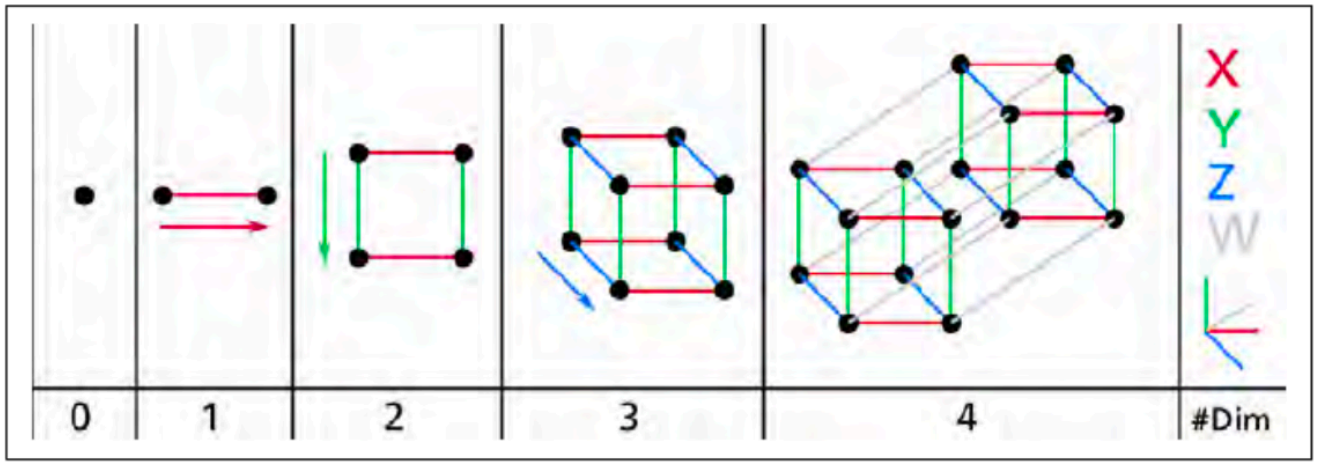


*Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)*[2]

Géron, Aurélien. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow* O'Reilly Media, Inc.

# Principal Component Analysis (PCA)

- **PCA can be used for *projection***
  - *Projection* takes advantage of the fact that most data is not uniformly distributed across the dimensions
    - Rather correlations exist
- **PCA exploits these correlations by finding directions that capture the maximum amount of variance in the data**
    - These directions, or principle components, are orthogonal eigenvectors

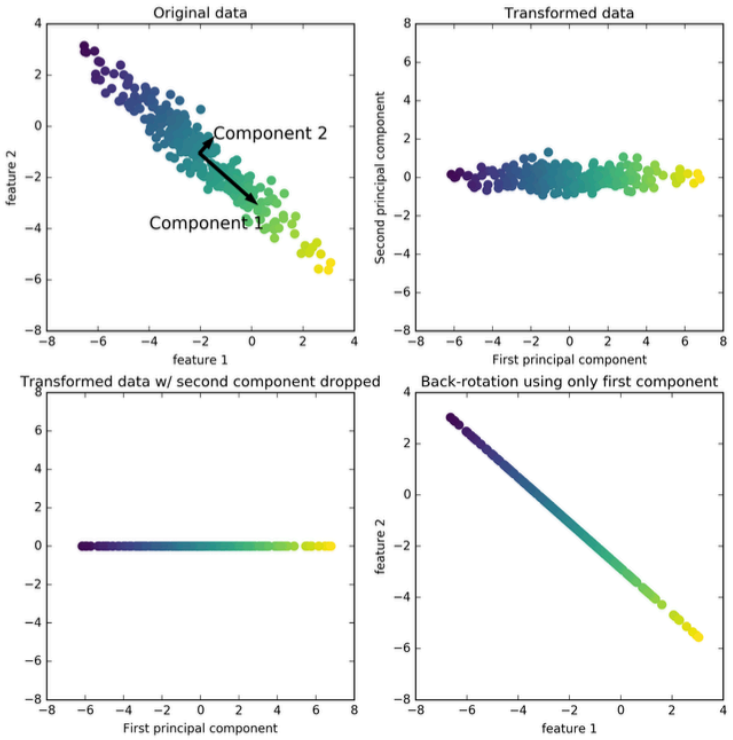


*Figure 3-3. Transformation of data with PCA*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Principal Component Analysis (PCA)

- **PCA can be used for *visualization***

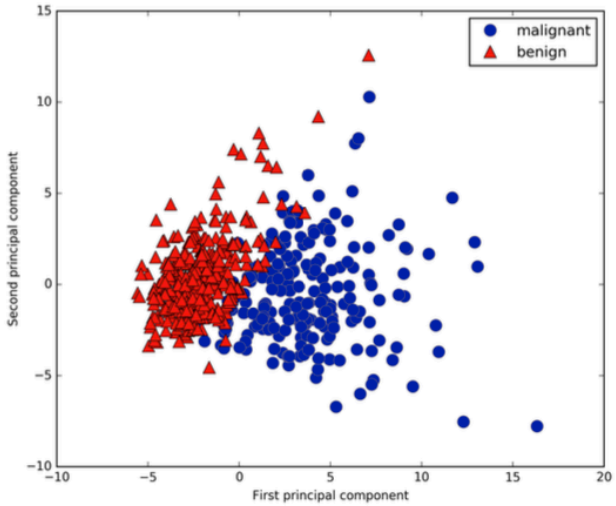


*Figure 3-5. Two-dimensional scatter plot of the Breast Cancer dataset using the first two principal components*

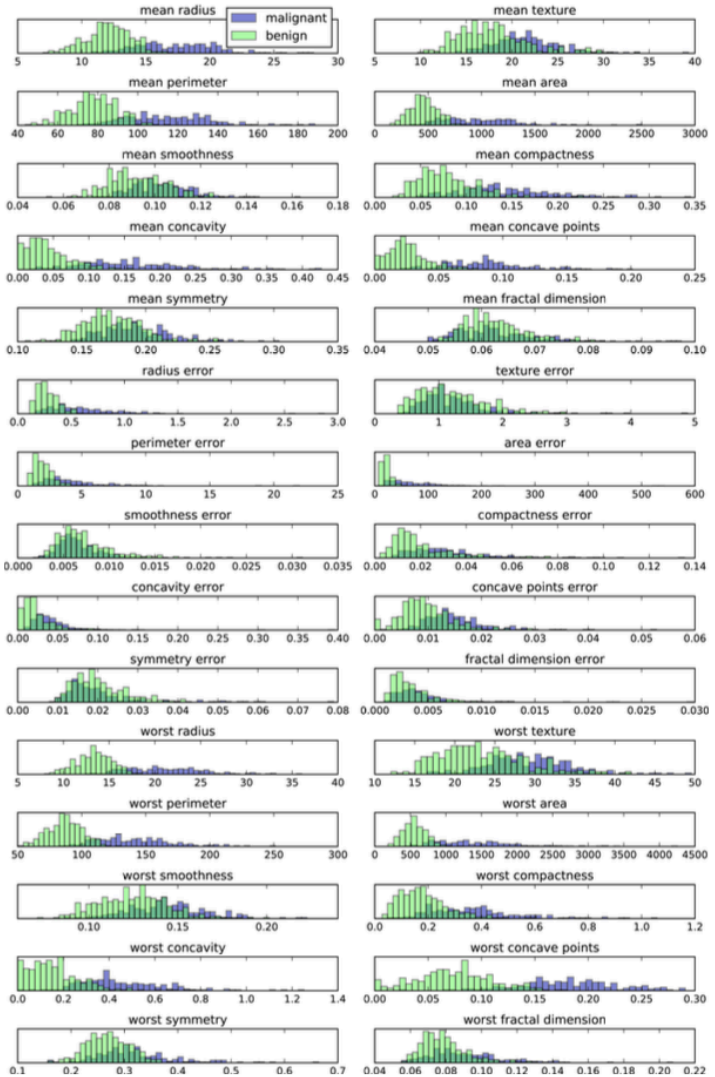


*Figure 3-4. Per-class feature histograms on the Breast Cancer dataset*

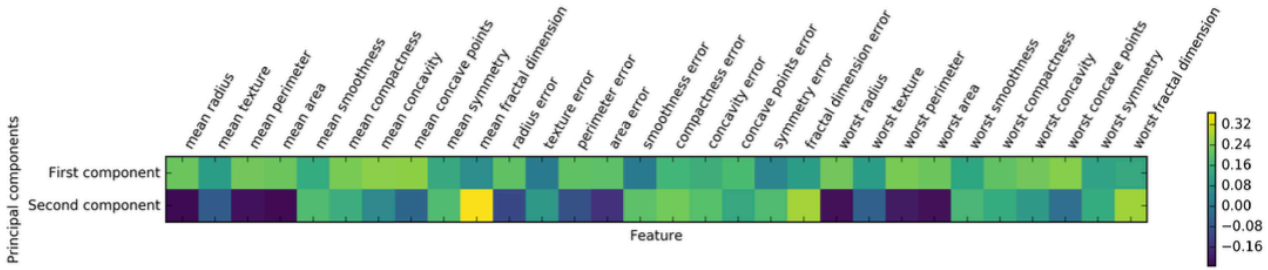


*Figure 3-6. Heat map of the first two principal components on the Breast Cancer dataset*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Principal Component Analysis (PCA)

- **PCA can be used for *feature extraction***
- **Whitening transforms the data to the same scale**
  - also de-correlates the data

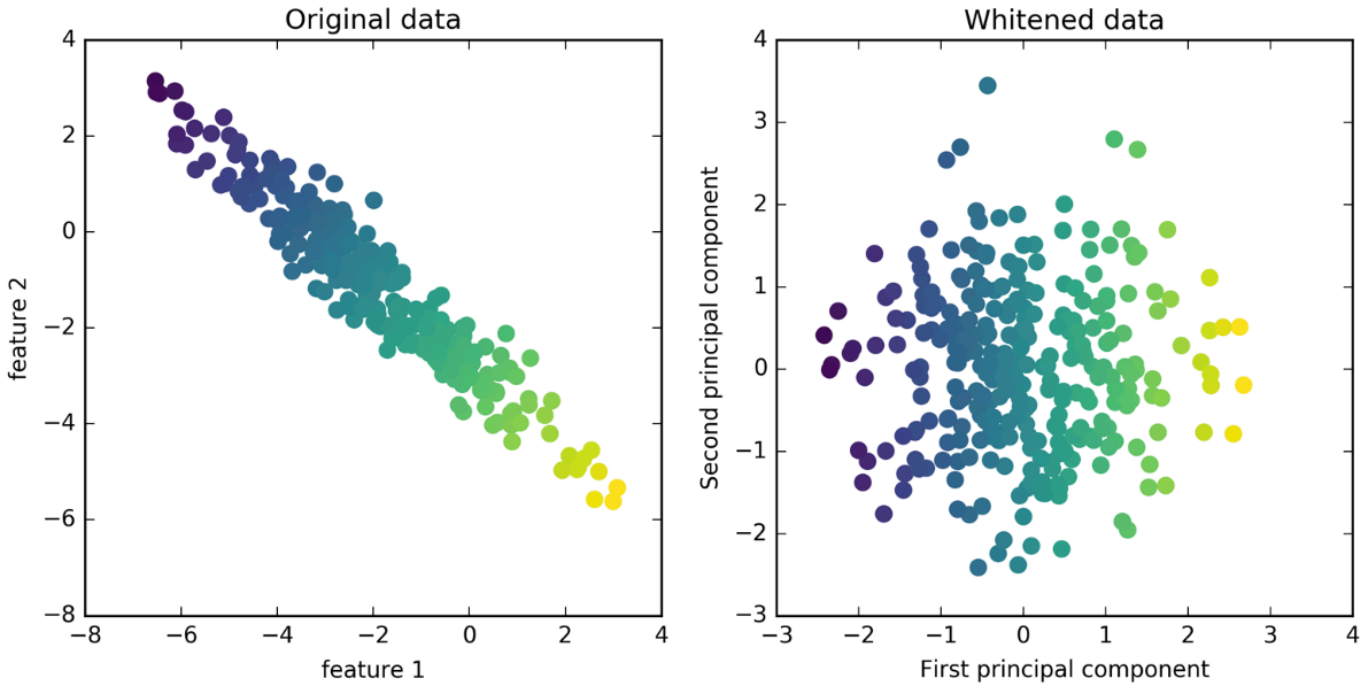


*Figure 3-8. Transformation of data with PCA using whitening*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Principal Component Analysis (PCA)


*Figure 3-7. Some images from the Labeled Faces in the Wild dataset*

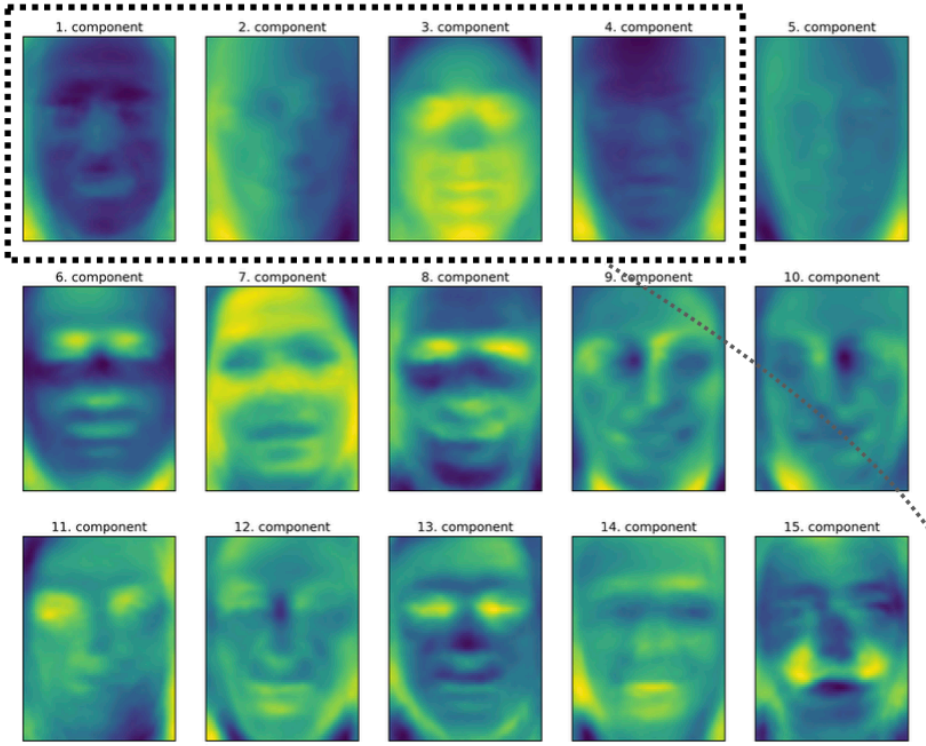


*Figure 3-9. Component vectors of the first 15 principal components of the faces dataset*

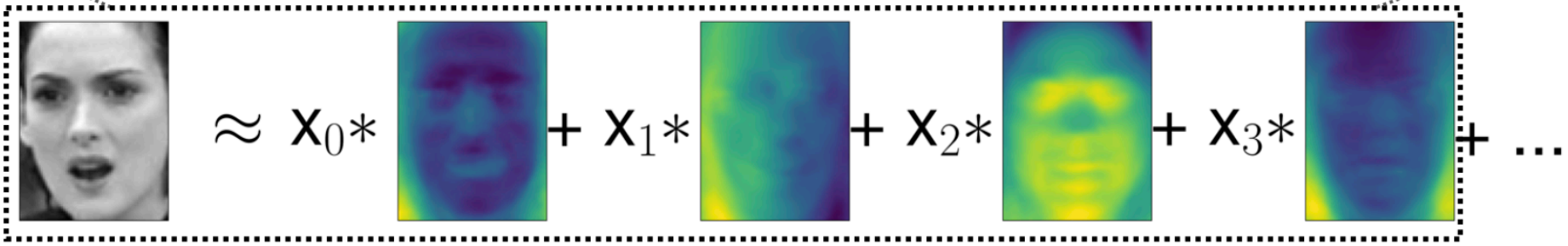


*Figure 3-10. Schematic view of PCA as decomposing an image into a weighted sum of components*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Principal Component Analysis (PCA)


*Figure 3-11. Reconstructing three face images using increasing numbers of principal components*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Principal Component Analysis (PCA)

- **Variants**
  - Incremental PCA
    - split the data into mini-batches and feed an IPCA algorithm one mini-batch at a time
  - Randomized PCA
    - quick approximation of the first *d* components
  - Kernel PCA
    - complex nonlinear projections for dimensionality reduction (same kernel trick as SVM)

# Principal Component Analysis (PCA)

- **Best Practices**
  - Scale data to have unit variance
  - Dimensionality reduction: choose the number of dimensions that add up to a sufficiently large portion of the variance (e.g., 95%)
  - Visualization: limit to 2 or 3 dimensions
- **Strengths**
  - Removes correlations from data
  - Reduces overfitting
  - Speeds up processing time
- **Weaknesses**
  - Hard to interpret
  - Assumes linearity

# PCA Activity

- **Import the wine dataset from Scikit-learn**
  - from sklearn.datasets import load_wine
  - data = load_wine()
- **Based on the code in data71200class7.ipynb**
  - Split into a testing and training set
  - Import PCA
  - Calculate how many features you need to capture 95% of the variance in the training set features and visualize the components
  - Compare the results with a Decision Tree Classifier of the pre-processed data and PCA 95% variance versions of the training set
  - Use another model (logistic regression, SVM, etc) if you have time

# Non-Negative Matrix Factorization (NMF)

- **Only works on zero or positively valued data**
  - Unlike PCA which can work on negative data as well
  - Makes the NMF bases more interpretable than PCA's
- **No ordering of the bases, as there is in PCA**

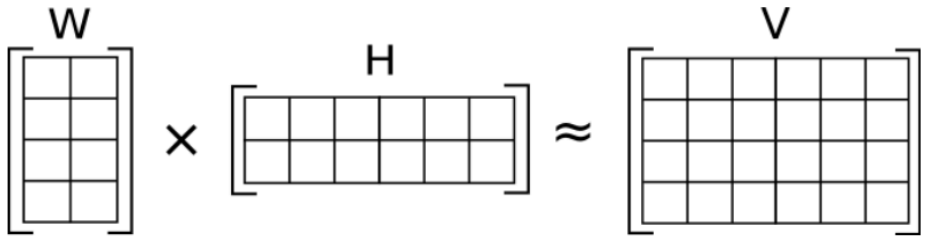


*Illustration of approximate non-negative matrix factorization: the matrix **V** is represented by the two smaller matrices **W** and **H**, which, when multiplied, approximately reconstruct **V***

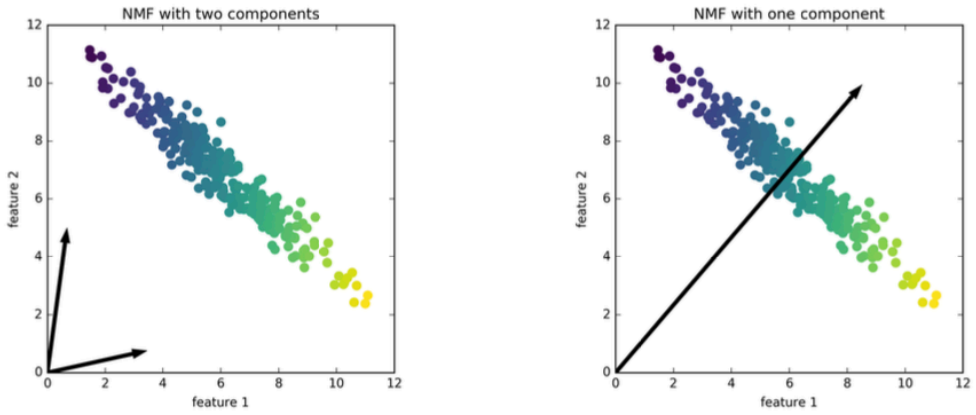


*Figure 3-13. Components found by non-negative matrix factorization with two components (left) and one component (right)*

https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Non-Negative Matrix Factorization (NMF)

## PCA


*Figure 3-11. Reconstructing three face images using increasing numbers of principal components*

## NMF


*Figure 3-14. Reconstructing three face images using increasing numbers of components found by NMF*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Non-Negative Matrix Factorization (NMF)

## PCA

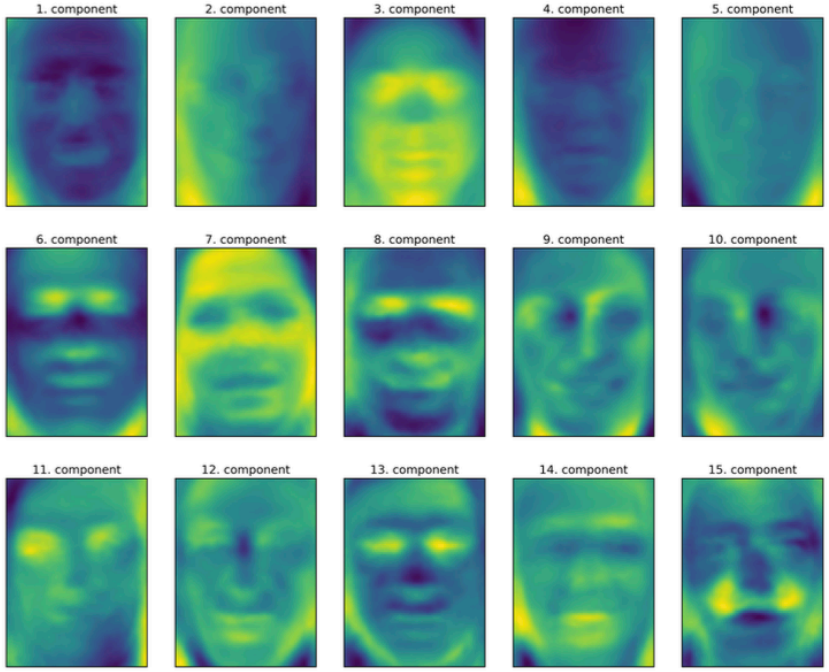


*Figure 3-9. Component vectors of the first 15 principal components of the faces dataset*

*Figure 3-16. Faces that have a large coefficient for component 3*

## NMF

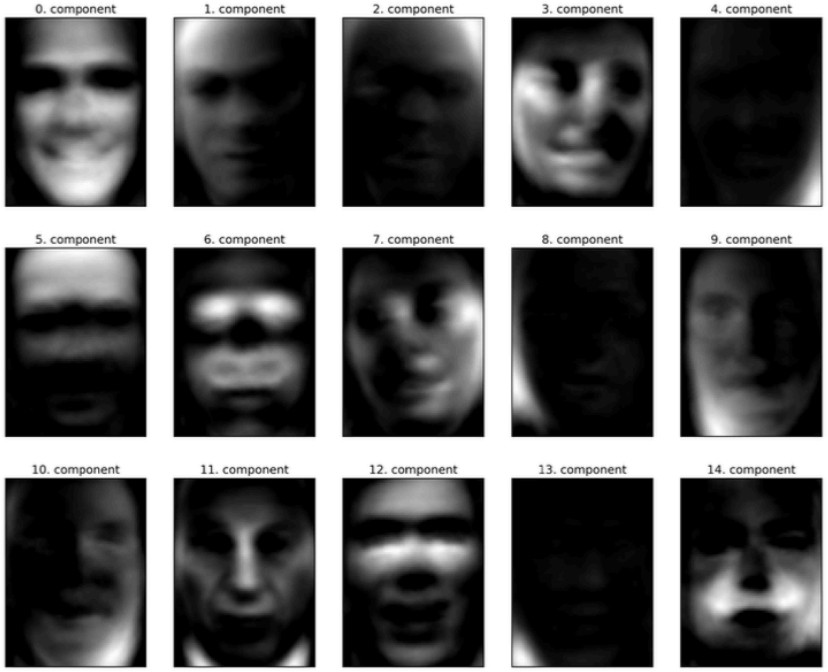


*Figure 3-15. The components found by NMF on the faces dataset when using 15 components*

*Figure 3-17. Faces that have a large coefficient for component 7*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Non-Negative Matrix Factorization (NMF)

- **NMF works better than PCA for decomposing data into its constituent parts**

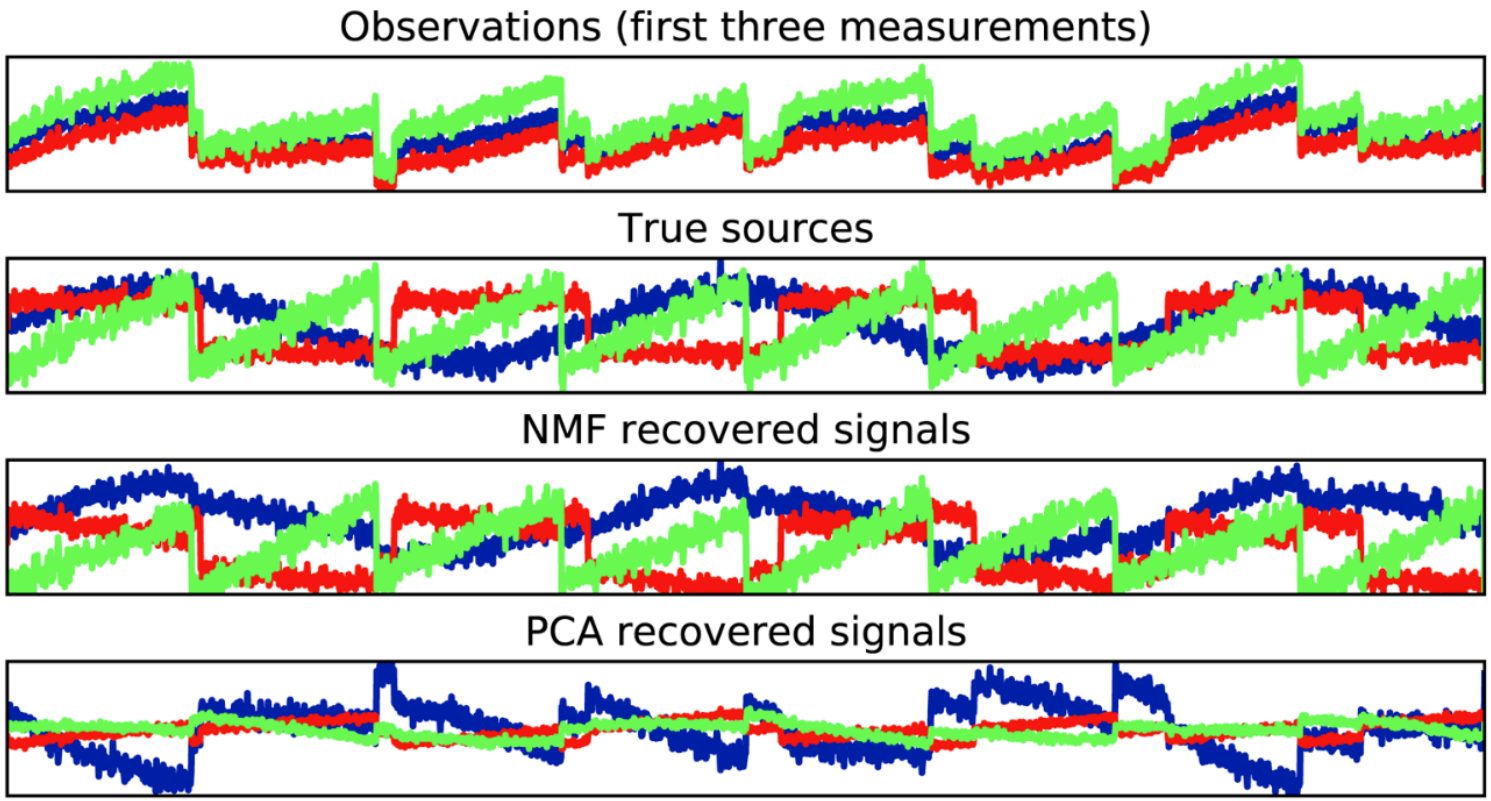


*Figure 3-19. Recovering mixed sources using NMF and PCA*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Non-Negative Matrix Factorization (NMF)

- **Best Practices**
  - Specify initialization if you want reproducibility
- **Strengths**
  - Positive bases are more interpretable than negative
  - Decomposes data into its constituent parts
- **Weaknesses**
  - Bases values can vary based on starting point and number of dimensions
  - Doesn’t have a closed form solution

# Scaling and NMF Activity

- **Scale the data from the wine dataset**
- **Run PCA with 2 components (and 95% of variance if you have time) on the unscaled and scaled data**
  - Compare performance differences with an SVM
  - Visualize the differences in .components_ with plt.matshow
- **Run NMF with 2 components on the unscaled and scaled data**
  - Compare performance differences with an SVM
  - Visualize the differences in .components_ with plt.matshow

# Manifold Learning

- **Manifold assumption - that most high-dimensional representations are close to a related low-dimensional representation**
- **t-SNE (t-Distributed Stochastic Neighbor Embedding) finds a 2D representation of the data that attempts to preserve distances between data points**
  - Particularly for data points that are close to one another

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Manifold Learning

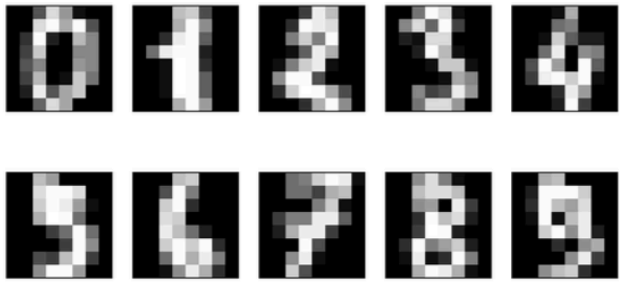

*Figure 3-20. Example images from the digits dataset*

## PCA

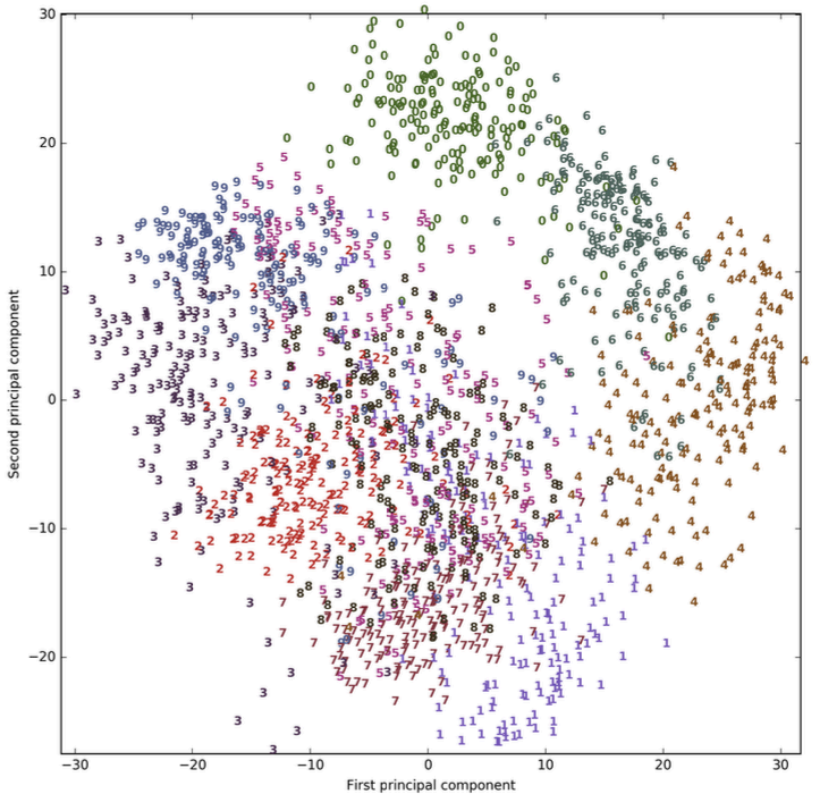


*Figure 3-21. Scatter plot of the digits dataset using the first two principal components*

## t-SNE

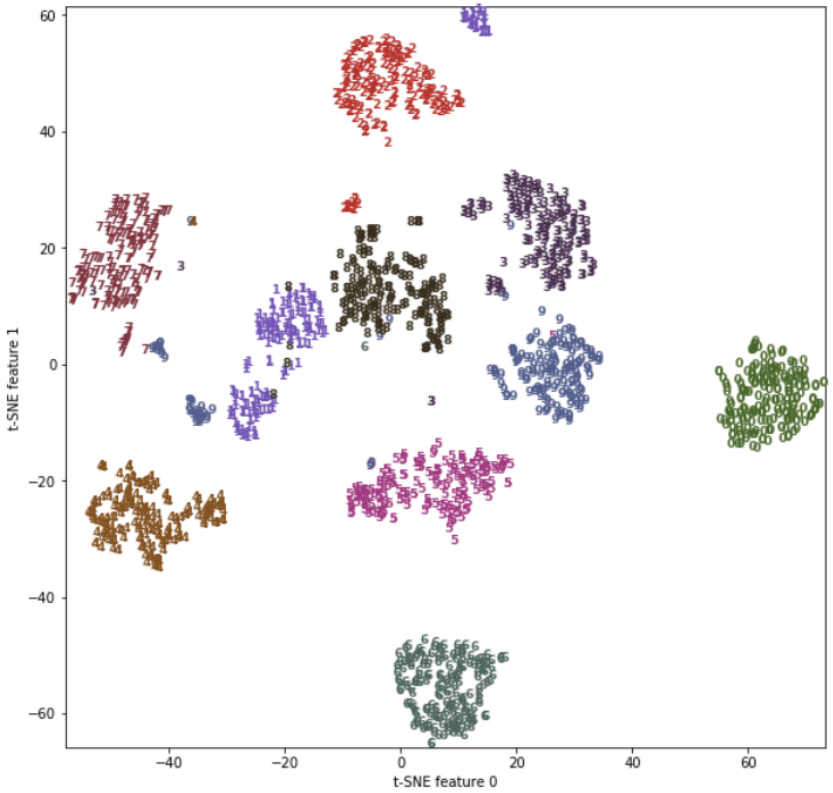


*Figure 3-22. Scatter plot of the digits dataset using two components found by t-SNE*

Muller, Andreas C. and Sarah Guido. (2016). Introduction to Machine Learning with Python, O'Reilly Media, Inc.

# Manifold Learning

- **Best Practices**
  - If the number of dimensions is > 50, reduce to 50 dimensions by first running PCA
- **Parameters**
  - Perplexity - number of nearest neighbors considered, generally larger data sets require larger perplexity
- **Strengths**
  - Retains local distance relationships in the data
- **Weaknesses**
  - Can’t be applied new data - won’t work for transforming a training set and then applying the same transformation to the testing set
  - Bases values can vary based on starting point and number of dimensions
  - No closed form solution

# TSNE Activity

- **Compare the visualizations of PCA (2 components) and TSNE for both the unscaled and scaled versions of the Wine training set**

# Upcoming Work

- **Project 2 due today**
- **Videos for June 26 (password: data71200)**
  - Unsupervised Learning 2: https://vimeo.com/415374034
- **Reading for June 26**
  - Ch 3: “Unsupervised Learning” in Guido, Sarah and Andreas C. Muller. (2016). Introduction to Machine Learning with Python, O’Reilly Media, Inc. 170–211
  - West, Sarah Myers, Meredith Whittaker, and Kate Crawford. (2019). “Discriminating systems: Gender, race and power in AI.” AI Now Institute, 1–33
- **DataCamp**
  - *AI Ethics*
- **Project 3 due July 2**

# Project 3 (Due July 2)

## Project 3

To **submit** the assignment, submit a link to your project Jupyter notebook on Blackboard.

The **goal** for this assignment is two apply different types of unsupervised learning techniques on the dataset you created in Project 1.

**Step 1: Load your data, including testing/training split from Project 1.**

- Your testing and training split should be balanced
- Your data should be clean and missing data should be addressed
- All appropriate variables are converted to categorical variables (as ordinal or one hot)
- Any necessary feature scaling should be performed
- YOU SHOULD ONLY WORK ON YOUR TRAINING SET

# Project 3 (Due July 2)

**Step 2: PCA for feature selection**

- Show how many features do you need to retain to capture 95% of the variance
- Evaluate whether this improves your best-performing model from Project 2

**Step 3: Apply 3-types of clustering on your data and visualize the output of each *both with and without PCA run on it first*. Calculate both ARI and Silhouette Coefficient for all six of the combinations.**

- k-Means (use an elbow visualization to determine the optimal numbers of clusters)
- Aggolmerate/Hierarchical
- DBSCAN

*If your data from projects 1 and 2 really doesn't lend itself to clustering,*
*you can use the breast_cancer dataset from scikit-learn.*

*Still submit your attempts on your own data in the notebook.*

**Tip:** You should make notes on what worked well and what didn't. Such notes will be useful when you write up the paper for your final project.

# Project 3 (Due July 2)

| | Missing | Fair | Good | Excellent |
|---|---|---|---|---|
| **Data** | **0** (0.00%) | **1.33** (8.86666%)<br>One of: Balanced testing/training split. Only using training set. Scaling and encoding done, where appropriate. | **1.67** (11.13333%)<br>Two of: Balanced testing/training split. Only using training set. Scaling and encoding done, where appropriate. | **2** (13.33333%)<br>Balanced testing/training split. Only using training set. Scaling and encoding done, where appropriate. |
| **PCA (95% variance)** | **0** (0.00%) | **0.5** (3.33333%)<br>PCA performed but captures a different amount of variance captured. | **0** (0.00%) | **1** (6.66666%)<br>PCA captures 95% of variance |
| **PCA on supervised learning** | **0** (0.00%) | **0** (0.00%) | **0** (0.00%)<br>PCA feature selection performed with some issue as a pre-processing step for supervised learning. | **1** (6.66666%)<br>PCA feature selection accurately performed as a pre-processing step for supervised learning. |
| **k-Means** | **0** (0.00%) | **1.33** (8.86666%)<br>Run and neither visualized or with and without PCA | **1.67** (11.13333%)<br>Run and either visualized or with and without PCA | **2** (13.33333%)<br>Run and visualized with and without PCA |
| **k-Means elbow visualization** | **0** (0.00%) | **0.5** (3.33333%)<br>Run and results not used to select number of clusters | **0** (0.00%) | **1** (6.66666%)<br>Run and results used to select number of clusters |
| **Agglomerative/Hierarchical** | **0** (0.00%) | **2** (13.33333%)<br>Run and neither visualized or with and without PCA | **2.5** (16.66666%)<br>Run and either visualized or with and without PCA | **3** (20.00%)<br>Run and visualized with and without PCA |
| **DBSCAN** | **0** (0.00%) | **2** (13.33333%)<br>Run and neither visualized or with and without PCA | **2.5** (16.66666%)<br>Run and either visualized or with and without PCA | **3** (20.00%)<br>Run and visualized with and without PCA |
| **Comments** | **0** (0.00%) | **1.33** (8.86666%)<br>Present but not detailed enough | **1.67** (11.13333%)<br>Largely complete | **2** (13.33333%)<br>Complete with good detail |