Foundations of Statistical and Machine Learning for Actuaries -

Big Data, Dimension Reduction and Non-Supervised Learning

Edward (Jed) Frees, University of Wisconsin - Madison
Andrés Villegas Ramirez, University of New South Wales

July 2025

Schedule

Day and Time	Presenter	Topics	Notebooks for Participant Activity
Monday Morning	Jed	Welcome and Foundations Hello to Google Colab	Auto Liability Claims
	Jed	Classical Regression Modeling	Medical Expenditures (MEPS)
Monday Afternoon	Andrés	Regularization, Resampling, Cross-Validation	Seattle House Sales
	Andrés	Classification	Victoria road crash data
Tuesday Morning	Andrés	Trees, Boosting, Bagging	
	Jed	Big Data, Dimension Reduction and Non-Supervised Learning	Big Data, Dimension Reduction, and Non-Supervised Learning
Tuesday Afternoon	Jed	Neural Networks	Seattle House Prices Claim Counts
	Jed	Graphic Data Neural Networks	MNIST Digits Data
Tuesday 4 pm	Fei	Fei Huang Thoughts on Ethics	
Wednesday Morning	Jed	Recurrent Neural Networks, Text Data	Insurer Stock Returns
	Jed	Artificial Intelligence, Natural Language Processing, and ChatGPT	
Wednesday After Lunch	Dani	Dani Bauer Insights	
Wednesday Afternoon	Andrés	Applications and Wrap-Up	

Tuesday Morning 3B. Big Data, Dimension Reduction and Non-Supervised Learning

This module covers:

- flexible and nonparametric variations of regression,
- motivations for big data in insurance,
- graphic data and the curse of dimensionality,
- introduction to principal components, and
- two unsupervised learning techniques.

In addition:

- During lecture and lab, participants may follow the notebook *BigDataDimReduction*.

Top Python Packages for Deep Learning

- 1. **TensorFlow / Keras**
  - Website: https://www.tensorflow.org/
  - *Use*: Industry-grade deep learning framework. Keras is the high-level API that makes it easy to build models.
  - Why it matters: Simple syntax, great documentation, many tutorials.
- 2. **PyTorch**
  - Website: https://pytorch.org/
  - *Use*: Flexible, research-friendly deep learning framework.
  - Why it matters: More "pythonic," dynamic computation graphs, lots of community support.

Python Packages that support Deep Learning

- 3 **Scikit-learn**
  - *Use*: Classical machine learning (SVMs, trees, regression), sometimes used for preprocessing deep learning input.
  - Why it matters: Excellent for pipeline construction, feature engineering.
- 4 **Matplotlib / Seaborn**
  - *Use*: Visualization — crucial for understanding training curves, data, and model behavior.
- 5 **Pandas + NumPy**
  - *Use*: Data manipulation (Pandas), numerical operations (NumPy).
  - Why it matters: These are essential for preparing and processing data for deep learning models.

Many observations (n), few features (p)

- This is the traditional set-up in statistics, based on experimental data from agriculture, medicine, and engineering
- The traditional emphasis is on **linear** functions of the mean of the form

$$\mathrm{E}(y) = f(\mathbf{X}) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

- Naturally, scientific relationships need not be linear.
  - It is less difficult to think about *known* non-linear functions, such as through GLMs.
  - It is more difficult to think about *unknown* non-linear functions,
- In principle, we would like to "Let the data speak for themselves."

Polynomial Regression

- One would like to build non-linear models that well approximate a function locally.
  - Thinking about a Taylor series expansion, one can approximate a function

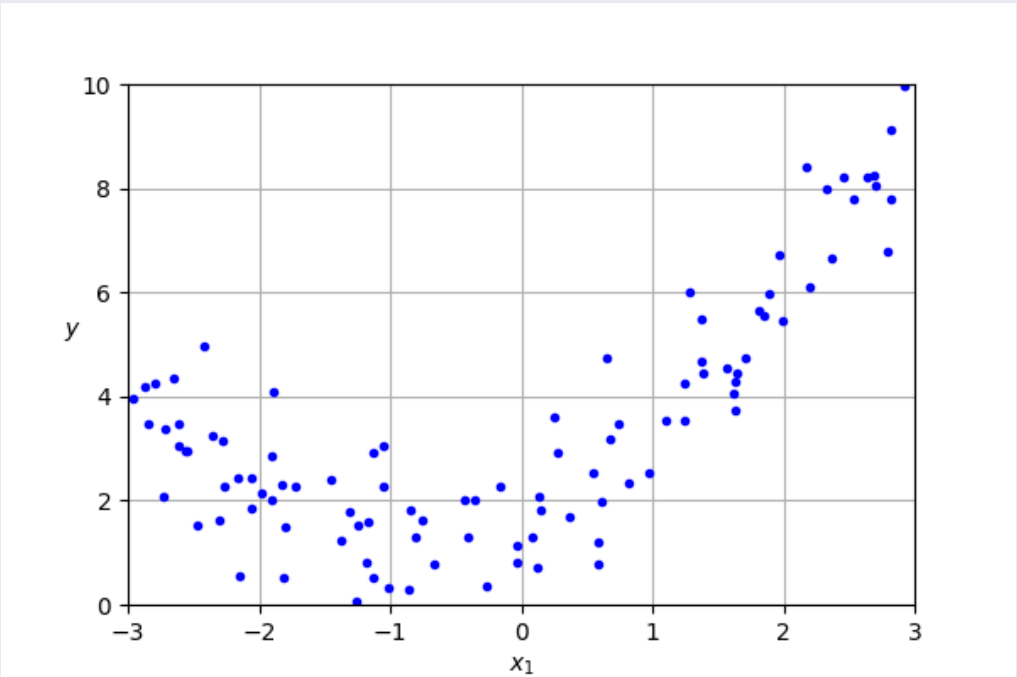
$$f(x) = f(x_0) + (x - x_0)f'(x_0) + (x - x_0)^2 \frac{f''(x_0)}{2} + \cdots$$

  - So, for a single variable, this suggests a polynomial regression fit of the form

$$\mathrm{E}(y) = f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p.$$

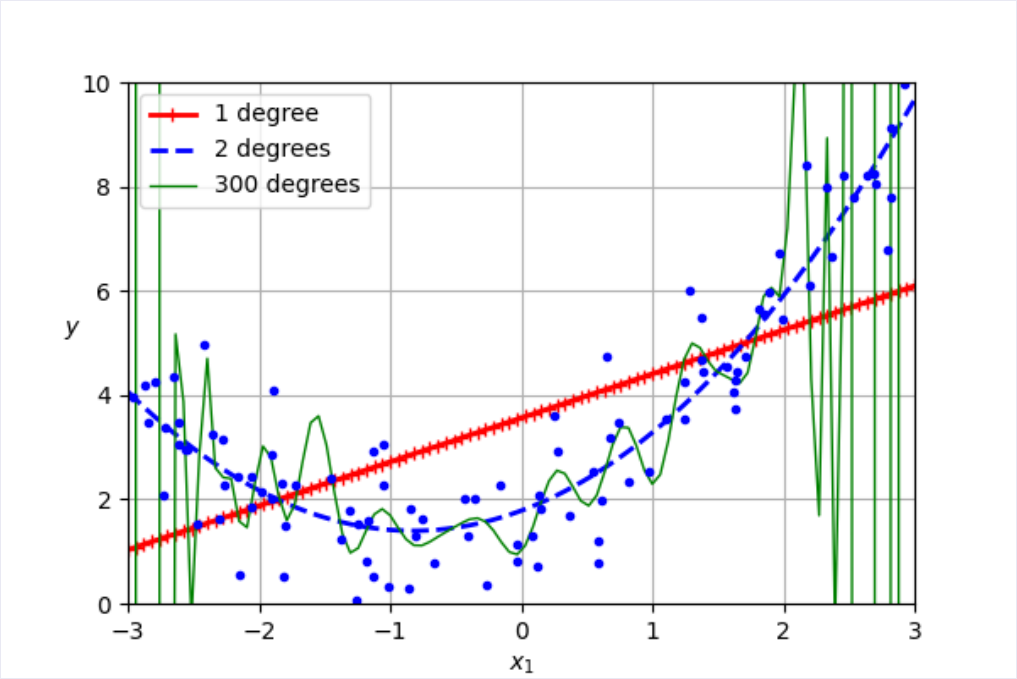
  - How many degrees to include? Traditionally, this has been limited to 3 or 4.
- With the aid of regularization procedures, we can let the data decide.

A Non-linear (Quadratic) Relationship



{ *Credit*: This is an example from Chapter 4 of Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media. }

Polynomial Regression Fits



Nonparametric Regression

- If the number of covariates (p) is very small (such as 2 or 3), one can take a fully nonparametric approach and estimate the model,

$$y_i = f(X_{i1}, \ldots, X_{ip}) + \epsilon_i$$

.

  - Here, $f(\cdot)$ is a nonlinear function to be estimated from the data.
  - The idea is to estimate the function using averages of observations in a local neighborhood.
- As we emphasize soon, this approach fares poorly with an even moderate number of covariates.

Nonparametric (Local) Regression Fits

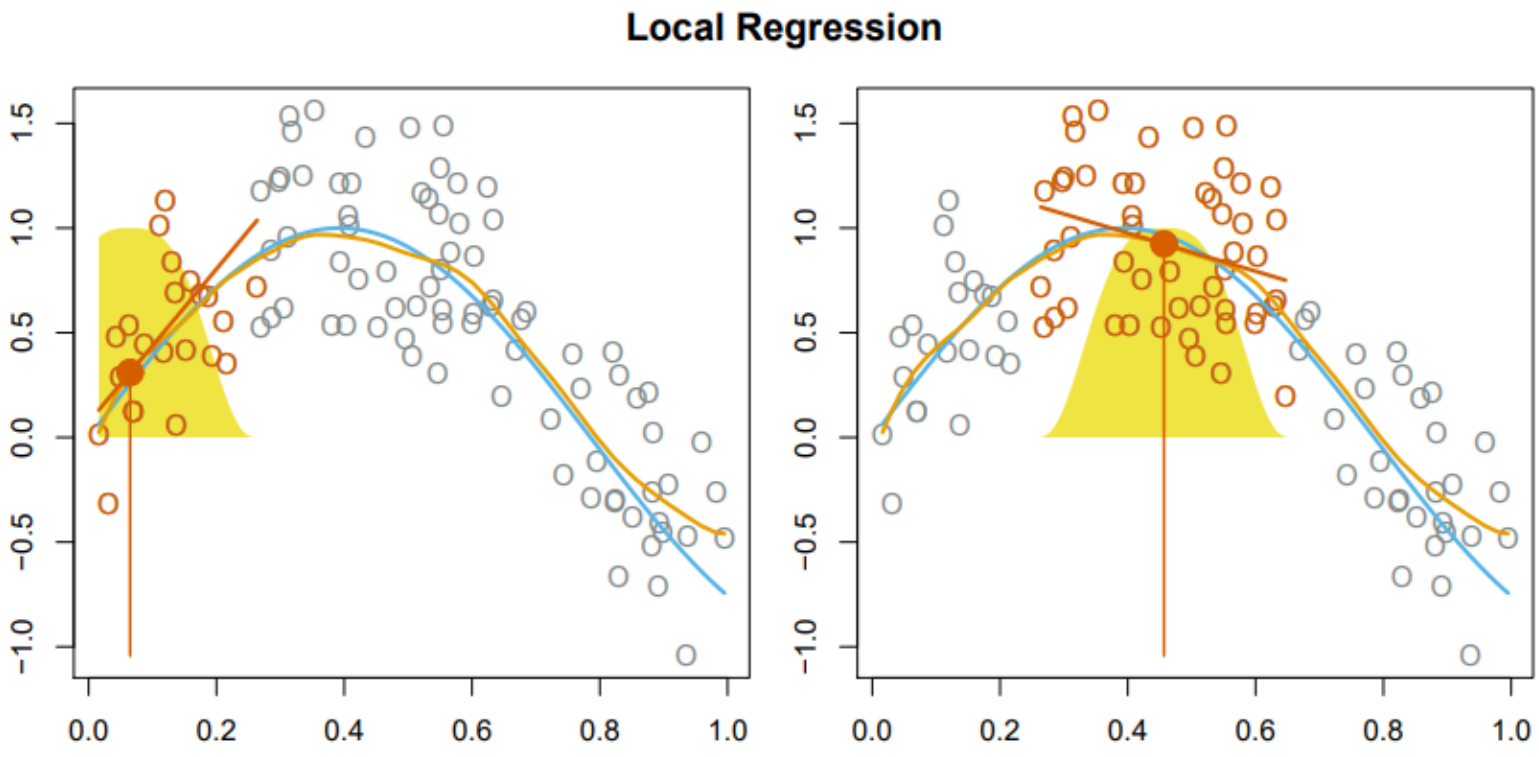


FIGURE 7.9. *Local regression illustrated on some simulated data, where the*

{ *Credit*: From Chapter 7 of - James et al. (2023), An Introduction to Statistical Learning with Applications in Python }

Generalized Additive Models

- A popular approach for extending the basic linear model

$$y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \epsilon_i$$

  is to use the expression

$$y_i = \beta_0 + \beta_1 f_1(X_{i1}) + \cdots + \beta_p f_p(X_{ip}) + \epsilon_i.$$

  - Here, $f_j(\cdot)$ are general nonlinear (unknown) functions that are estimated from the data.
  - The main restriction of this model is that covariate effects are *additive*.
    - You can learn more about this model in Section 7.7 of ISLP.
- GAMs provide a useful compromise between linear and fully nonparametric models.
- For other flexible approaches, we have already seen random forests and boosting.

Many Features (Large p)

There is a growing class of **insurance problems involving high-dimensional feature spaces**, often driven by IoT, telematics, medical devices, or behavioral data.

- *Telematics-Based Auto Insurance (Usage-Based Insurance, UBI)*
  - Modern vehicles generate gigabytes of data per day.
  - Features include: speed, acceleration, braking, cornering, location, time of day, trip duration, phone usage while driving, road type, weather, etc.
- *Health Insurance with Wearables and EMRs*
  - Features from **wearable devices**: heart rate, sleep cycles, steps, glucose levels, etc.
  - Also: Electronic Medical Records (EMRs), genomics, prescription data.
- *Cyber Insurance for Businesses*
  - Collect logs from thousands of endpoints, firewall alerts, patch records, employee behavior.

Many Features 2 (Large p)

- *Property Insurance with Satellite and IoT Sensors*
  - Satellite images, drone imagery, LIDAR scans
  - Smart home sensors: water leaks, temperature, humidity, motion
- *Fraud Detection in Claims Processing*
  - Use of **text mining** on claim narratives, call transcripts, claimant history, provider patterns, etc.
- *Life Insurance with Genetic and Behavioral Risk Scores*
  - Polygenic risk scores (PRS), detailed health behaviors, family history, financial behavior.
- *Customer Churn and Cross-Sell Modeling*
  - Marketing and behavior data: web/app clicks, agent interactions, past purchases, complaints, NPS.

Some References on Telematics-based Auto Insurance

- *Analyzing the Influence of Telematics-Based Pricing Strategies...* — Discusses using **non-negative sparse PCA** to reduce hundreds of telematics features while maintaining interpretability and predictive power
- *Unravelling the predictive power of telematics data in car insurance* — Uses rich telematics data from young drivers to extract meaningful features and model claim risk
- *Integration of traditional and telematics data for efficient insurance claims prediction* — Integrates large telematics feature sets with classic underwriting, handling dimensionality via calibration techniques
- *Claim Prediction and Premium Pricing for Telematics Auto Insurance* — Compares penalized Poisson models (e.g., adaptive Lasso) for high-dimensional telematics feature selection

Some References on Health Insurance + Wearables & EMRs

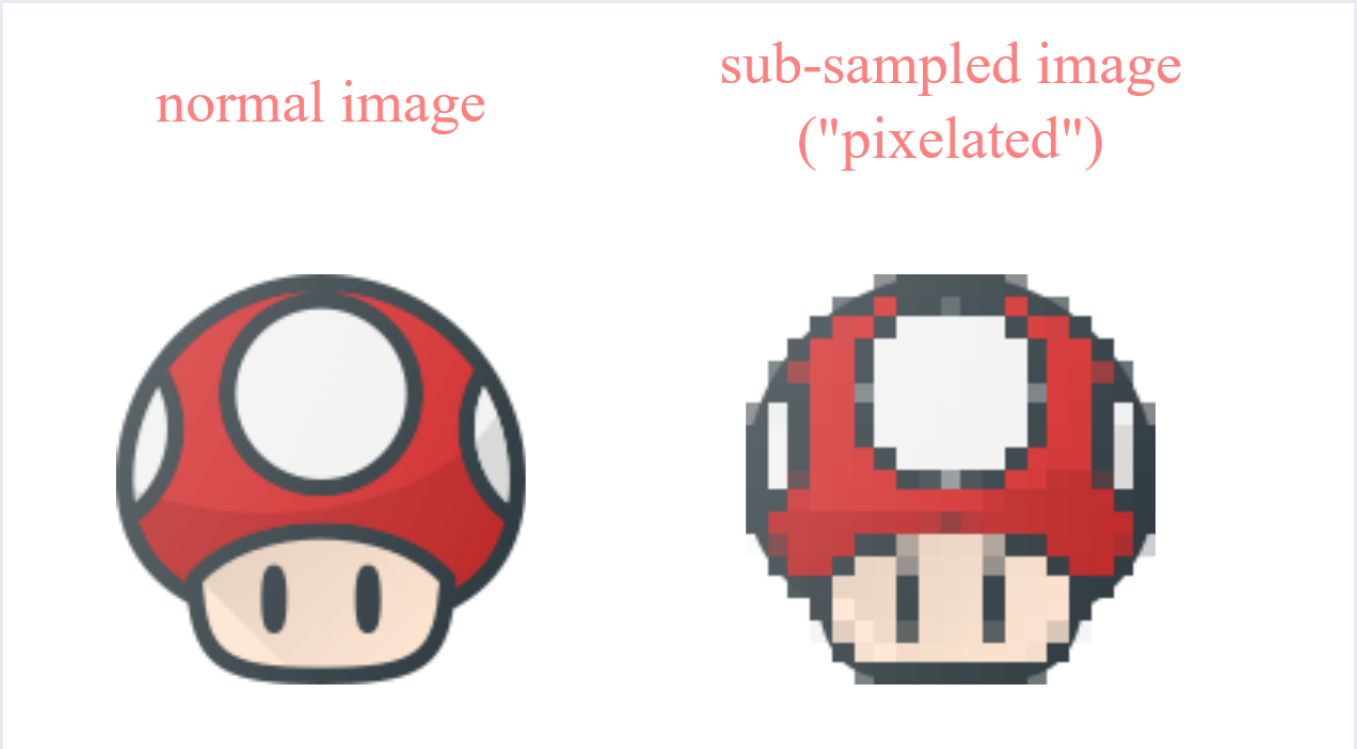
- *Deep learning for prediction of population health costs* — Trains neural networks on claims data with over 1.4 million records containing hundreds or thousands of features (diagnoses, procedures, demographics), demonstrating superior performance
- *Digital medicine and the curse of dimensionality* — Reviews challenges of high-dimensional wearable and medical sensor data in healthcare analytics, highlighting variable selection and regularization

Image / Graphic Data

Images provide a handy, easy to understand, use case to motivate why we care about high-dimensional features.

- Images are stored in a computer as a matrix of numbers known as **pixel** values.
  - These pixel values represent the intensity of each pixel.
  - In grayscale images, a pixel value of 0 represents black, and 255 represents white.
- A **channel** is a matrix of pixel values, and we have only one channel in the case of a grayscale image.
  - The dimension of this image will be 24 x 16, we mean it would be 24 pixels across the height and 16 pixels across the width.

Pixelated Image



{ *Source*: Introduction to Neural Network Models of Cognition, by Pablo Caceres }

Image / Graphic Data

- Almost all colors can be generated from the three primary colors – Red, Green, and Blue.
  - Therefore, we can say that each colored image is a unique composition of these three colors or 3 channels – Red, Green, and Blue.

{ *Source*: https://www.analyticsvidhya.com/blog/ }

Common Image Resolutions:

- 320×240 pixels – Suitable for small-screen devices.
- 1024×768 pixels – Standard computer monitors.
- 720×576 pixels – Used in standard-definition (SD) TVs (4:3 aspect ratio).
- 1280×720 pixels – Ideal for widescreen monitors (HD).
- 1280×1024 pixels – Fits full-screen LCD monitors (5:4 aspect ratio).
- 1920×1080 pixels – Common for HD TVs.
- 4K (3840×2160 pixels) – Supported by ultra HD monitors and TVs.
- 5K (5120×2880 pixels) – Found in high-end displays.
- 8K (7680×4320 pixels) – Used in ultra high-definition screens.

An Approach

- So, for a small screen, we would have $76,800 = 320 \times 240$ pixels that could be used as covariates for a black and white image. And 230,400 for a color image.
- Suppose there are a large number of features and a limited number of observations ($p > n$).
  - Then we can implement traditional methods for determining which features are important (variable selection)
  - This could include regularization techniques such as ridge regression and LASSO
- Suppose there are a large number of features and observations (big p and n)
  - we have an opportunity to fit complex models that can learn from the data

Curse of Dimensionality

Volume grows exponentially with the number of features

- Not a lot of "local" information
- Observations tend to be close to an edge - **extreme**
- Observations can be far from one another

Example. Consider 100,000 points that are uniformly distributed in a p-dimensional unit cube, $[0,1]^p$. Consider a cell where each side has width - 0.5.

- If $p = 1$, expect about $50,000 = 100000 \times \frac{1}{2}$ points in the cell.
- If $p = 2$, expect about $25,000 = 100000 \times \frac{1}{2^2}$ points in the cell.
- If $p = 10$,expect about $98 = 100000 \times \frac{1}{2^{10}}$ points in the cell.
- If $p = 25$, expect about $0.1 = 100000 \times \frac{1}{2^{25}}$ points in the cell.

Not a lot of local information in large dimensions.

Example. As a continuation, what is the probability that a point is within 0.001 of an edge?

- If $p = 2$, the probability is $0.003996 = 1 - (1 - 0.001 \times 2)^2$.
- If $p = 20$, the probability is $0.0392 = 1 - (1 - 0.001 \times 2)^{20}$.
- If $p = 2000$, the probability is $0.98175 = 1 - (1 - 0.001 \times 2)^{2000}$.
- If $p = 20000$, the probability is $1 \approx 1 - 4.081556 \times 10^{-12} = 1 - (1 - 0.001 \times 2)^{20000}$.

Observations tend to be close to an edge - ***extreme***

Example. As a continuation, what is the approximate expected distance between points?

This is less obvious: a handy approximation is

$$\sqrt{\frac{p}{6}} \times \left(1 - \frac{1}{4p}\right).$$

- If $p = 2$, the expected distance is 0.50518.
- If $p = 5$, the expected distance is 0.867.
- If $p = 200$, the expected distance is 5.766.
- If $p = 20000$, the expected distance is 57.734.

Observations can be far from one another.

Challenge: Curse of dimensionality

- **Volume grows exponentially with the number of features**
  - Not a lot of "local" information
  - Observations tend to be close to an edge - **extreme**
  - Observations can be far from one another
- An equation for f that is "too flexible" (e.g., number of parameters grows as our number of observation grows) is problematic
  - In order to be learnable, we cannot localize too much
  - Because distances in high dimensions are long, "connecting" the dots to get an obvious relationship as in a single dimension typically not possible
- **Fully nonparametric is not possible, our models need to impose a structure**

Dimension Reduction via Principal Component Analysis

- Principal component analysis (PCA) seeks a low-dimensional representation of data.
  - Very traditional, and still a good place to start, for reducing the dimensionality of our data.
- Consider p features $X_1, \ldots, X_p$ with n observations

$$\text{data} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

where each column has been centered to have a mean of zero.

- The first principal component is

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p$$

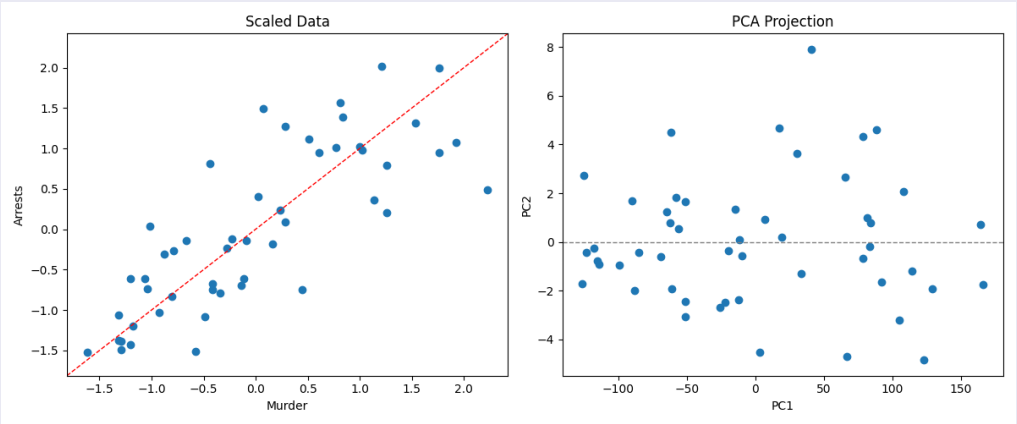
  that has the largest variance, subject to a normalization constraint, $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- That is, the loadings $\{\phi_{j1}\}$ solve the problem

$$\text{maximize}_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

- The second principal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance out of all linear combinations that are *uncorrelated* with Z_1.
- The third principal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance out of all linear combinations that are *uncorrelated* with Z_1 and Z_2. And so forth.

US Arrests Example

- This data set is built into R.
- It contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973.
- Also given is the percent of the population living in urban areas.



Principal components provide low-dimensional linear surfaces that are closest to the observations.

Proportion of Variance Explained

The total variance is

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m} \mathrm{Var}(Z_m)$$

The variance explained by the mth principal component is

$$\mathrm{Var}(Z_m) = \frac{1}{n}\sum_{i=1}^{n} z_{im}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2$$

The proportion of variance explained by the mth principal component is

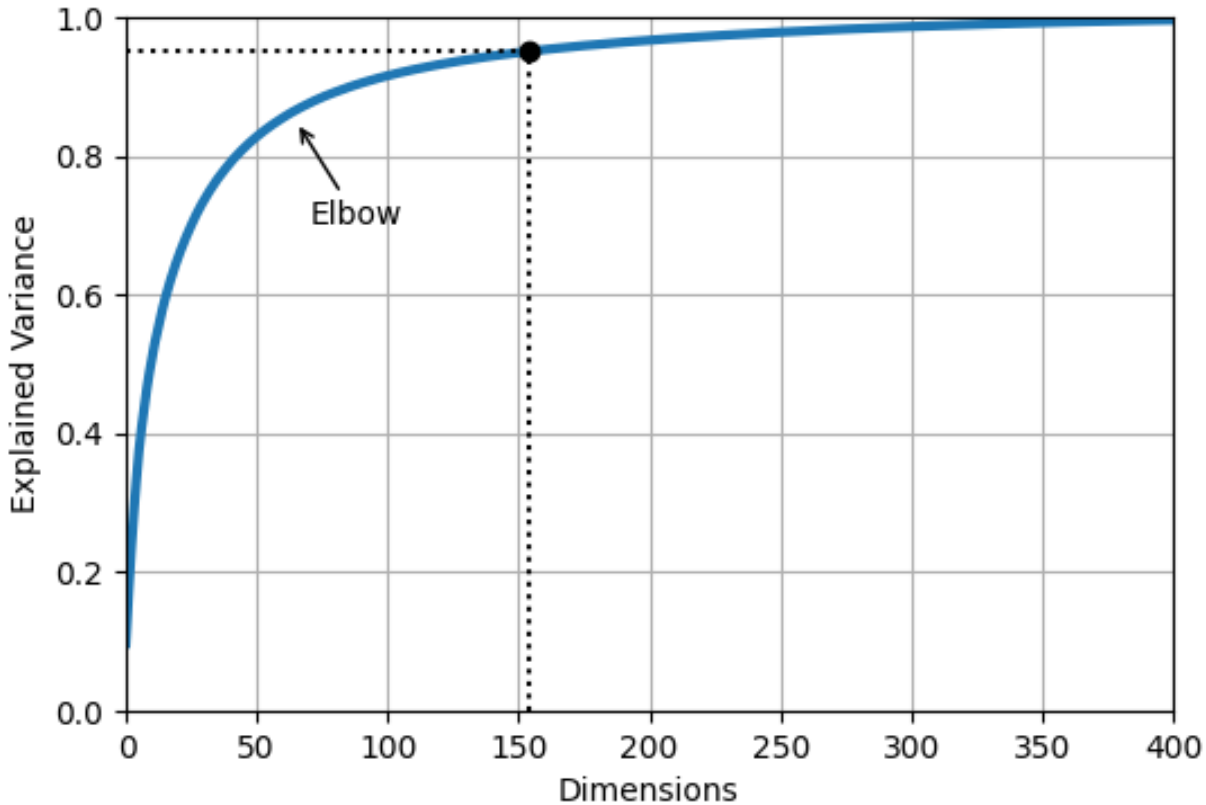
$$\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{i=1}^{n}\sum_{j=1}^{p}x_{ij}^2}.$$

PCA for the MNIST Data

- The MNIST dataset is a classic one in the machine learning community.
- It is a set of 60,000 training images, plus 10,000 test images, assembled by the National Institute of Standards and Technology (the NIST in MNIST) in the 1980s.
- The problem is to classify grayscale images of handwritten digits (28×28 pixels) into their 10 categories (0 through 9).
- However, to begin, let us think about ways that we can reduce the complexity of the 784 ($= 28 \times 28$) features.
  - By providing a less complex version of the image, we **compress** it which is very useful when sending/receiving images.

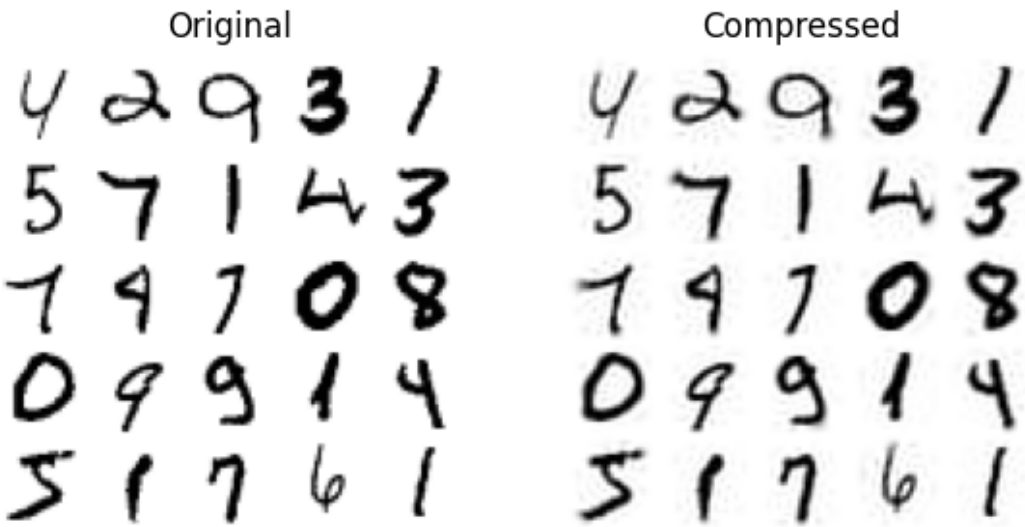
Here are a few selected images.

The code performs PCA, then computes the minimum number of dimensions required to preserve 95% of the training set's variance. This gives us a way to select the number of principal components.



{ *Credit*: This example is from Chapter 8 of Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.. }

And here is a comparison of the uncompressed versus compressed images.



Pretty good!

Non-Supervised Learning

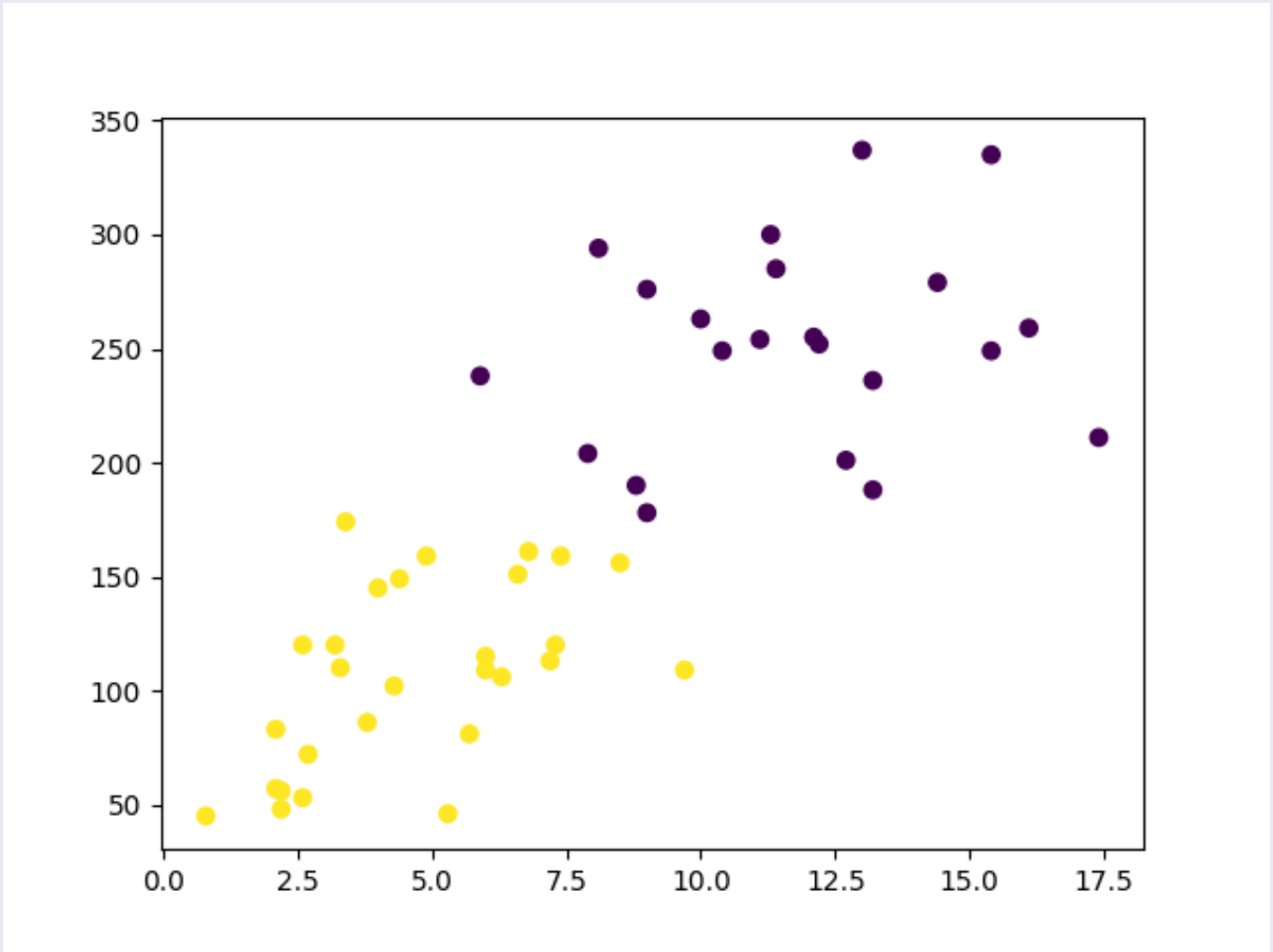
- Recall that for non-supervised learning methods, the data are treated the same and there is no artificial divide between "inputs" and "outputs."
- Later, we will get an introduction to *autoencoding*, a non-supervised technique based on neural networks. It learns a compressed (encoded) representation and then reconstructs the original data from that encoding.
- For now, let us focus on *clustering*, a very broad set of techniques for finding subgroups, or *clusters*, in a data set.

K-Means

K-means clustering is a simple and elegant approach for partitioning a data set into K distinct, non-overlapping clusters.
Here is the algorithm for K-means clustering:

1. Randomly assign a number, from 1 to K, to each of the observations.
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).

US Arrests Example: Arrest versus Murder



Hierarchical Clustering

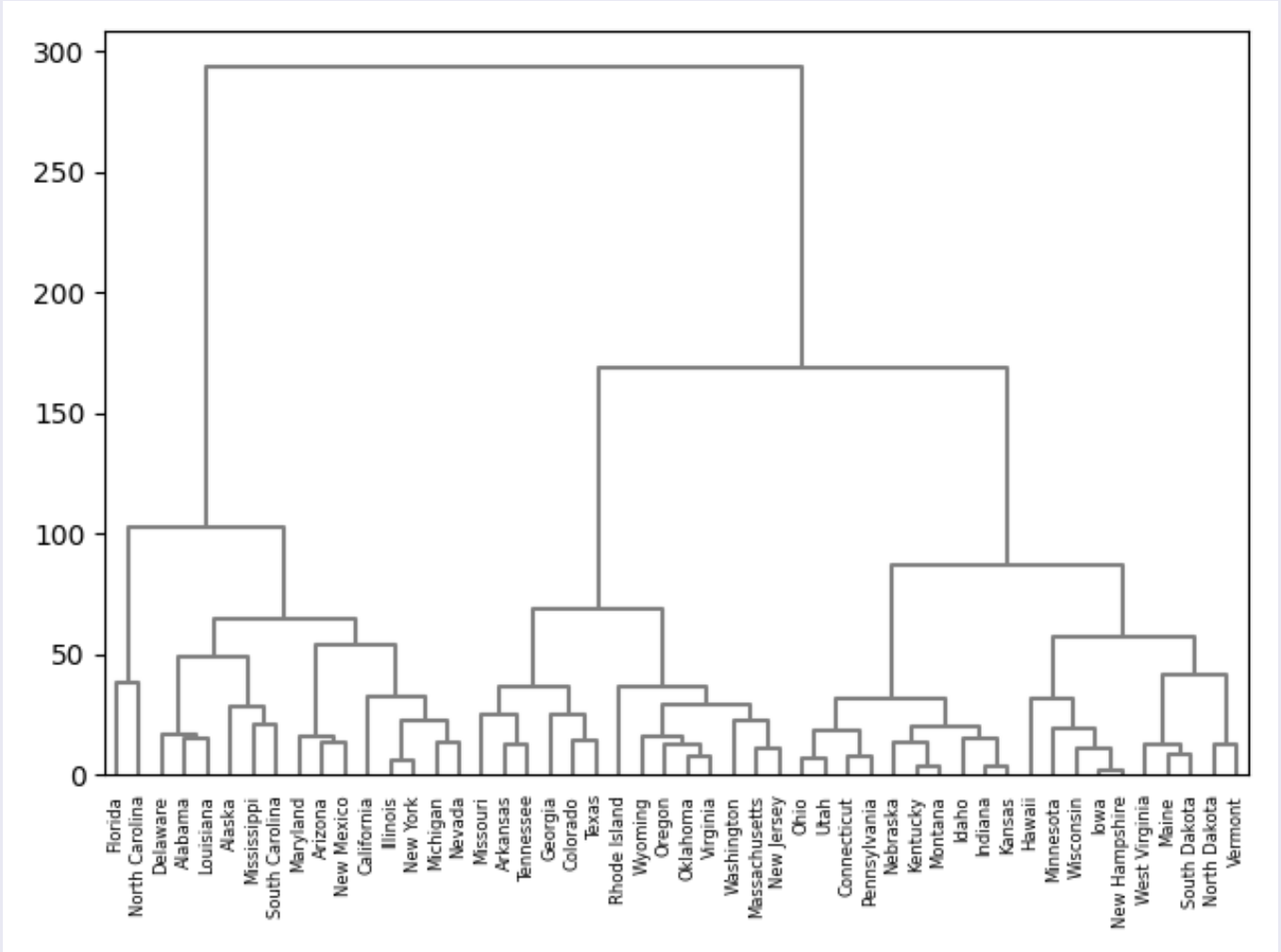
- One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters K.
  - *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of K.
- Hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a *dendrogram*.

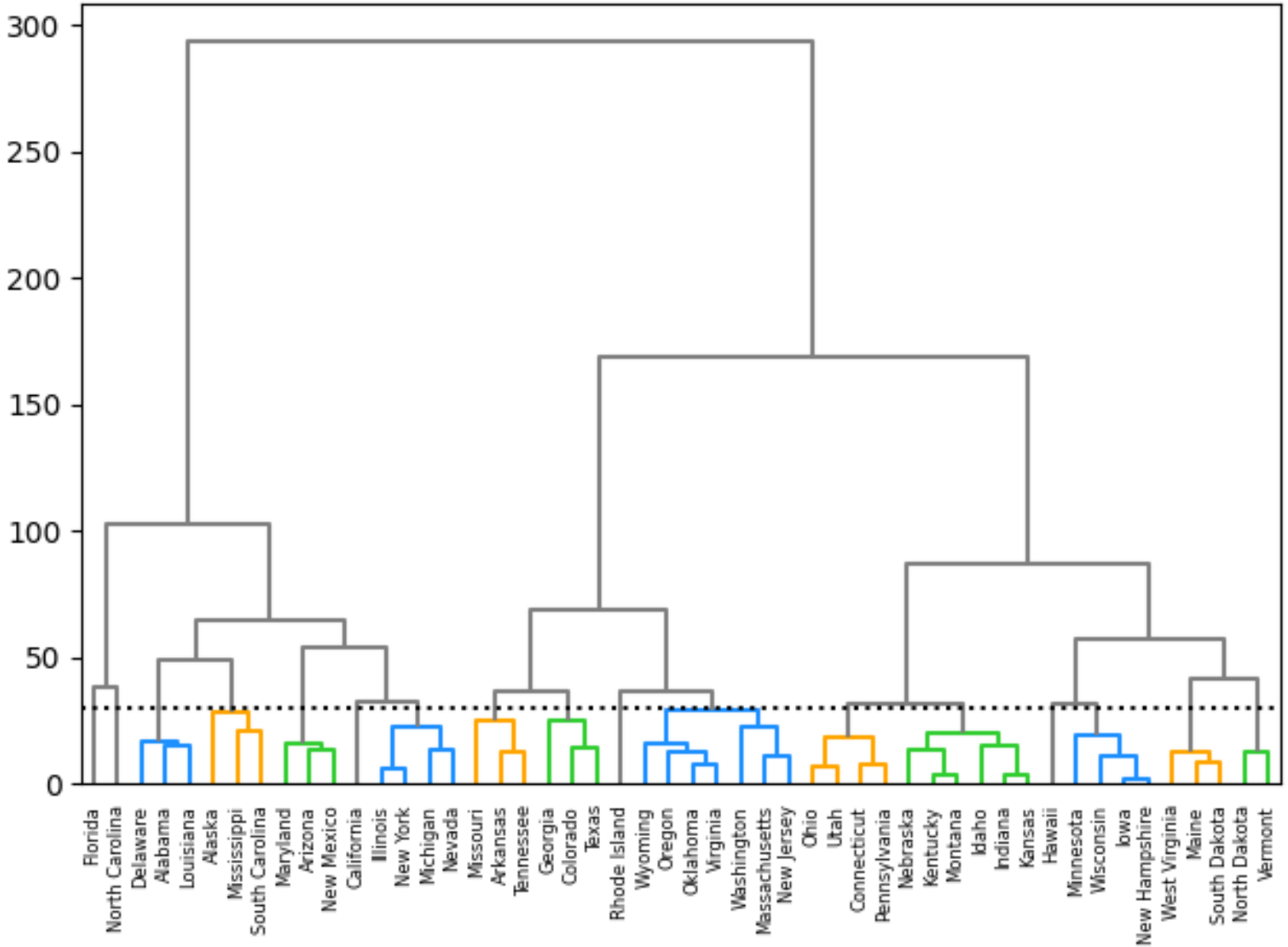
Algorithm Hierarchical Clustering.

1. Begin with n observations and a measure (such as Euclidean distance) of all the $n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For $i = n, n-1, \ldots, 2$:
   a. Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
   b. Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

US Arrests Example

To begin, it looks like Iowa and New Hampshire are the most similar, ...




300
250
200
150
100
50
0
Florida
North Carolina
Delaware
Alabama
Louisiana
Alaska
Mississippi
South Carolina
Maryland
Arizona
New Mexico
California
Illinois
New York
Michigan
Nevada
Missouri
Arkansas
Tennessee
Georgia
Colorado
Texas
Rhode Island
Wyoming
Oregon
Oklahoma
Virginia
Washington
Massachusetts
New Jersey
Ohio
Utah
Connecticut
Pennsylvania
Nebraska
Kentucky
Montana
Idaho
Indiana
Kansas
Hawaii
Minnesota
Wisconsin
Iowa
New Hampshire
West Virginia
Maine
South Dakota
North Dakota
Vermont

Session 3B. Big Data, Dimension Reduction and Non-Supervised Learning - Summary

This module covered:

- flexible and nonparametric variations of regression,
- motivations for big data in insurance,
- graphic data and the curse of dimensionality,
- introduction to principal components, and
- two unsupervised learning techniques.

In addition:

- During lab, participants may follow the notebook Big Data, Dimension Reduction and Non-Supervised Learning