Dimensionality reduction

Overview

- **Big Data and High Dimensionality**
- **Principal Components Analysis (PCA)**
- **PCA for Regression (PCR)**
- **Variable Importance**
- **Evaluating classification methods**
- **Cross Validation**

Big Data and High Dimensionality

What is ‘Big Data’?

- In the world of Data Science, the term *Big Data* gets thrown around a lot. What does *Big Data* mean?
- A rectangular data set has two dimensions: number of observations (*n*) and the number of predictors (*p*). Both can play a part in defining a problem as a *Big Data* problem.

When *n* is big

- When the sample size is large, this is typically not much of an issue from the statistical perspective, just one from the computational perspective.
- Algorithms can take forever to finish. Estimating the coefficients of a regression model, especially one that does not have closed form, can take a while. Wait until we get to Neural Networks!
- If you are tuning a parameter or choosing between models, this exacerbates the problem.
- What can we do to fix this computational issue?
- Perform 'preliminary' steps (model selection, tuning, etc.) on a subset of the training data set. 10% or less can be justified

Keep in mind, big *n* doesn't solve everything

- The era of Big Data (aka, large *n*) can help us answer lots of interesting scientific and application-based questions, but it does not fix everything.

- Remember the old adage: "**crap in = crap out**". That is to say, if the data are not representative of the population, then modeling results can be terrible. Random sampling ensures representative data.

When *p* is big

- When the number of predictors is large (in any form: interactions, polynomial terms, etc.), then lots of issues can occur.
- Matrices may not be invertible (issue in OLS).
- Multicollinearity is likely to be present

- This situation is called *High Dimensionality*, and needs to be accounted for when performing data analysis and modeling.

How Does sklearn handle unidentifiability?

- In a parametric approach: if we have an over-specified model ($p > n$), the parameters are unidentifiable: we only need $n - 1$ predictors to perfectly predict every observation ($n - 1$) because of the intercept.

- So what happens to the 'extra' parameter estimates (the extra β's)?
- the remaining $p - (n - 1)$ predictors' coefficients can be estimated to be anything. Thus there are an infinite number of sets of estimates that will give us identical predictions. There is not one unique set of $\hat{\beta}$'s.

Perfect Multicollinearity

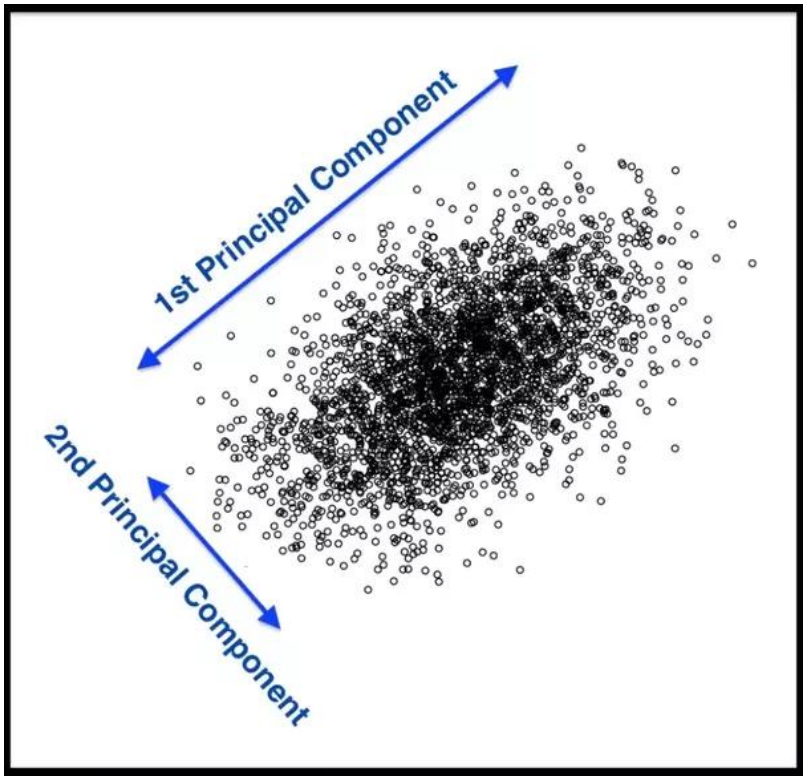
- The $p > n$ situation leads to perfect collinearity of the predictor set. But this can also occur with a redundant predictors (ex: putting X_j twice into a model). Let's see what sklearn in this simplified situation:

```
In [9]: # investigating what happens when two identical predictors are used

        logit1 = LogisticRegression(C=1000000,solver="lbfgs").fit(heart_df[['Age']],y)
        logit2 = LogisticRegression(C=1000000,solver="lbfgs").fit(heart_df[['Age','Age']],y)

        print("The coef estimate for Age (when in the model once):",logit1.coef_)
        print("The coef estimates for Age (when in the model twice):",logit2.coef_)

        The coef estimate for Age (when in the model once): [[0.05198618]]
        The coef estimates for Age (when in the model twice): [[0.02599311 0.02599311]]
```

Principal Components Analysis (PCA)

Principal Components Analysis (PCA)

- *Principal Components Analysis* (PCA) is a method to identify a new set of predictors, as linear combinations of the original ones, that captures the `maximum amount' of variance in the observed data.

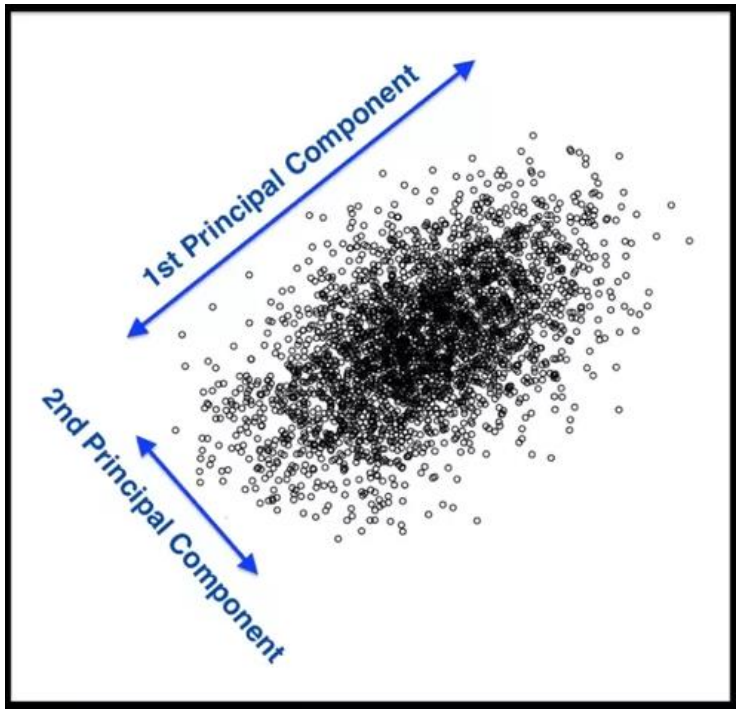


PCA (cont.)

- *Principal Components Analysis* (PCA) produces a list of *p principle components* $Z_1,\ldots, Z_p$ such that
- Each Z_i is a linear combination of the original predictors, and it's vector norm is 1
- The Z_i's are pairwise orthogonal
- The Z_i's are ordered in decreasing order in the amount of captured observed variance.
- That is, the observed data shows more variance in the direction of Z_1 than in the direction of Z_2.
- To perform dimensionality reduction we select the top *m* principle components of PCA as our new predictors and express our observed data in terms of these predictors.

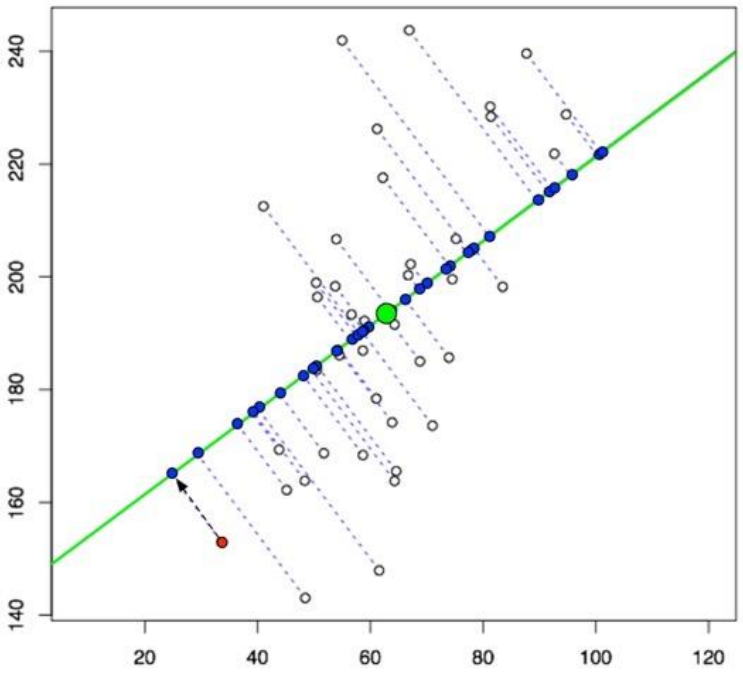
The Intuition Behind PCA

- Top PCA components capture the most of amount of variation (interesting features) of the data.
- Each component is a linear combination of the original predictors - we visualize them as vectors in the feature space.



The Intuition Behind PCA (cont.)

- Transforming our observed data means projecting our dataset onto the space defined by the top *m* PCA components, these components are our new predictors.



PCA for Regression (PCR)

PCA for Regression (PCR)

- PCA is easy to use in Python, so how do we then use it for regression modeling in a real-life problem?

- If we use all p of the new Z_j, then we have not improved the dimensionality. Instead, we select the first $M (<p)$ PCA variables, $Z_1,...,Z_M$, to use as predictors in a regression model.

- The choice of M is important and can vary from application to application. It depends on various things, like how collinear the predictors are, how truly related they are to the response, etc...

A few notes on using PCA

- PCA is an unsupervised algorithm. Meaning? It is done independent of the outcome variable.
  - Note: the components as predictors might not be ordered from best to worst!
- PCA is not so good because:
  1. Direct Interpretation of coefficients in PCR is completely lost. So do not do if interpretation is important.
  2. Will not improve predictive ability of a model.
- PCA is great for:
  1. Reducing dimensionality in very high dimensional settings.
  2. Reducing multicollinearity, and thus may improve the computational time of fitting models.

Variable Importance

Variable Importance for Tree-Based Models

- How does sklearn determine **variable importance** (feature_importance) from a tree-based model?

- It determines the improvement in the loss function every time a predictor is involved in a split.

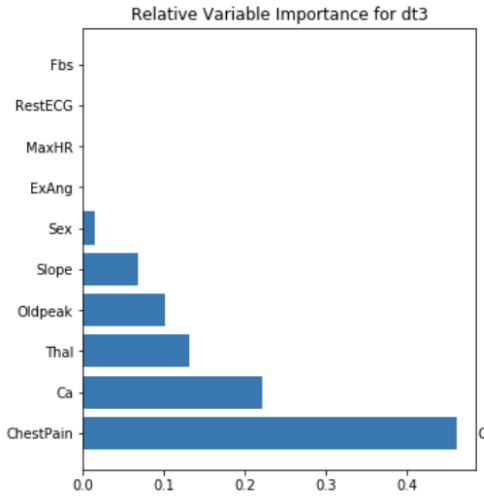
- More specifically, it calculate the total amount that the SSE (for regression) or Gini index (for classification) is improved (decreased) due to splits over a given predictor (averaged over all B trees if a bagged/random forest method).

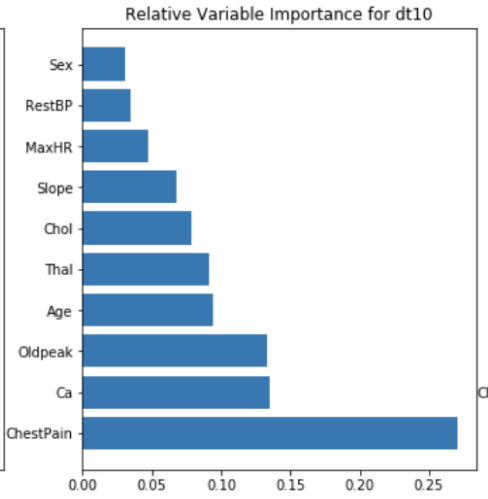
- How should variable importance compare across the various different tree models we've considered (trees, random forests/bagging, and boosting)?

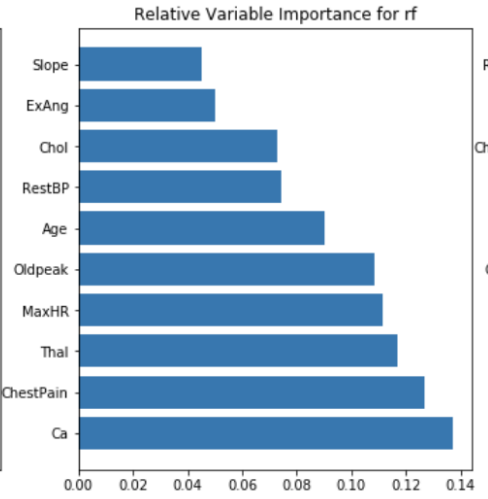
- A picture is worth a thousand words…

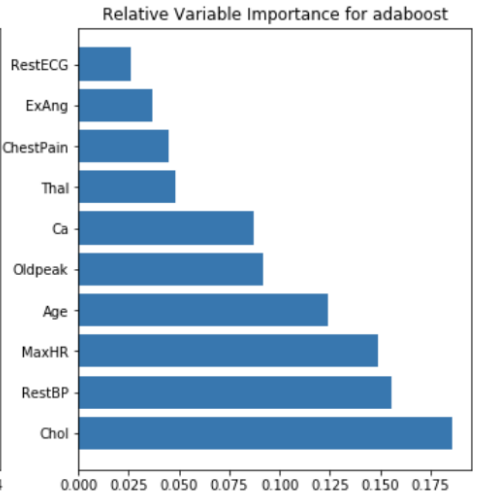
Variable Importance for trees, bags, and boosts

- Below are the variable importance plots for the top 10 predictors for each of a (i) decision tree with maxdepth=3, (ii) decision tree with maxdepth=10, (iii) a random forest, and (iv) an adaboost classifier.









The problem with Variable Importance

- Variable Importance is great! It tells you what features are important in shaping the model.
- But what is missing?
- It does not give any measure for how the predictors are related to the response (positive, negative, quasi-linear, curved, interactions, etc.).

Evaluating classification methods

Evaluating classification methods

- **Predictive accuracy**

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$

- Efficiency
  - time to construct the model
  - time to use the model
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability:
  - understandable and insight provided by the model
- Compactness of the model: size of the tree, or the number of rules.

Evaluation methods

- The available data set *D* is divided into two disjoint subsets,
  - the *training set* D_{train} (for learning a model)
  - the *test set* D_{test} (for testing the model)
- **Important:** training set should not be used in testing and the test set should not be used in learning.
  - Unseen test set provides a unbiased estimate of accuracy.

Evaluation methods (cont…)

- **n-fold cross-validation**: The training data is partitioned into *n* equal-size disjoint subsets.
- Use each subset as the validation set and combine the rest *n*-1 subsets as the training set to learn a classifier.
- The procedure is run *n* times, which give *n* accuracies.
- The final estimated accuracy of learning is the average of the *n* accuracies.
- 10-fold and 5-fold cross-validations are commonly used.
- This method is used when the available data is not large.

Classification measures

- Accuracy is only one measure (error = 1-accuracy).
- **Accuracy is not suitable in some applications**.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, we are interested only in the minority class.
  - High accuracy does not mean any intrusion is detected.
  - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes**.

Precision and recall measures

- Used in information retrieval.
- We use a confusion matrix to introduce them.

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

where

TP: the number of correct classifications of the positive examples (**true positive**),

FN: the number of incorrect classifications of positive examples (**false negative**),

FP: the number of incorrect classifications of negative examples (**false positive**), and

TN: the number of correct classifications of negative examples (**true negative**).

Precision and recall measures (cont...)

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

$$p = \frac{TP}{TP + FP}. \qquad r = \frac{TP}{TP + FN}.$$

- Precision *p* is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
- Recall *r* is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

An example

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

- This confusion matrix gives
  - precision p = 100% and
  - recall r = 1%

  because we only classified one positive example correctly and no negative examples wrongly.
- Note: precision and recall only measure classification on the positive class.

F_1-value (also called F_1-score)

- It is hard to compare two classifiers using two measures. F_1 score combines precision and recall into one measure

$$F_1 = \frac{2pr}{p + r}$$

F_1-score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

- The harmonic mean of two numbers tends to be closer to the smaller of the two.
- For F_1-value to be large, both *p* and *r* much be large.

Receive operating characteristics curve

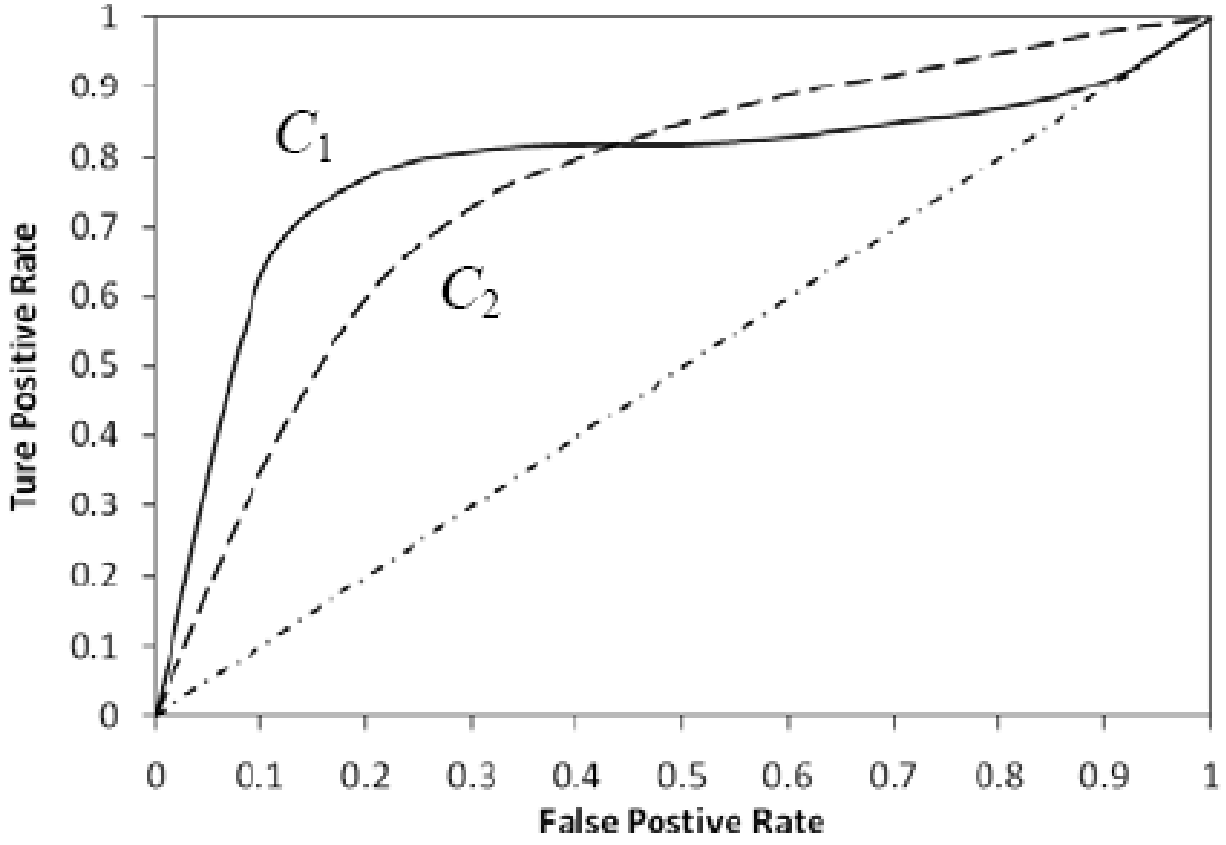
- It is commonly called the ROC curve.
- It is a plot of the true positive rate (TPR) against the false positive rate (FPR).
- True positive rate:

$$TPR = \frac{TP}{TP + FN}$$

- False positive rate:

$$FPR = \frac{FP}{TN + FP}$$

Example ROC curves



ROC curves for two classifiers (C_1 and C_2) on the same data

Area under the curve (AUC)

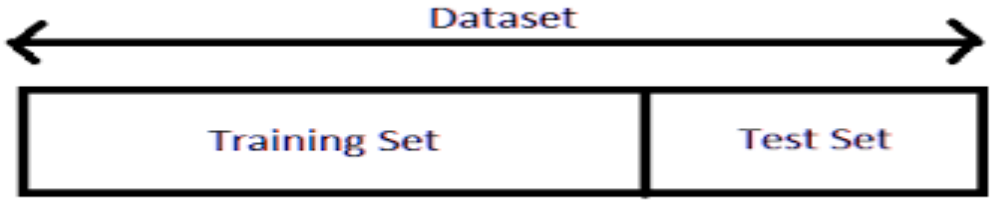
- Which classifier is better, C_1 or C_2?
  - It depends on which region you talk about.
- Can we have one measure?
  - Yes, we compute the area under the curve (AUC)
- If AUC for C_i is greater than that of C_j, it is said that C_i is better than C_j.
  - If a classifier is perfect, its AUC value is 1
  - If a classifier makes all random guesses, its AUC value is 0.5.

Cross Validation

Cross Validation: Motivation

- Using a single validation set to select amongst multiple models can be problematic.
- One solution to the problems raised by using a single validation set is to evaluate each model on **multiple** validation sets and average the validation performance.
- One can randomly split the training set into training and validation multiple times **but** randomly creating these sets can create the scenario where important features of the data never appear in our random draws.

Cross Validation: Illustration



Cross Validation: Illustration