Master of Applied Data Science

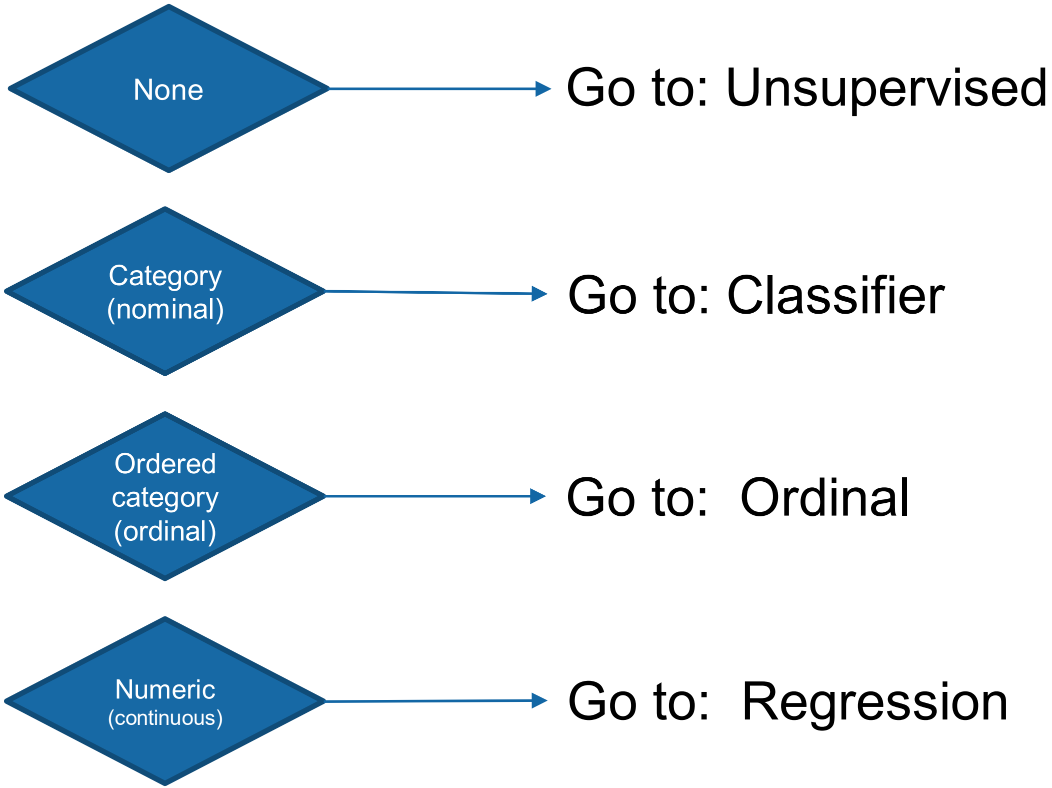
High-level synthesis of machine learning

Supervised and Unsupervised Learning Decision Flowchart: When to use which method?

Kevyn Collins-Thompson & Yumou Wei

Start

What type of target variable are you predicting?



This version v0.9.2022.08.13
Latest version always at:
https://www.umich.edu/~kevynct/mads_ml_synthesis.pdf
This flowchart is a work in progress.

Send feedback to:
Kevyn Collins-Thompson kevynct@umich.edu

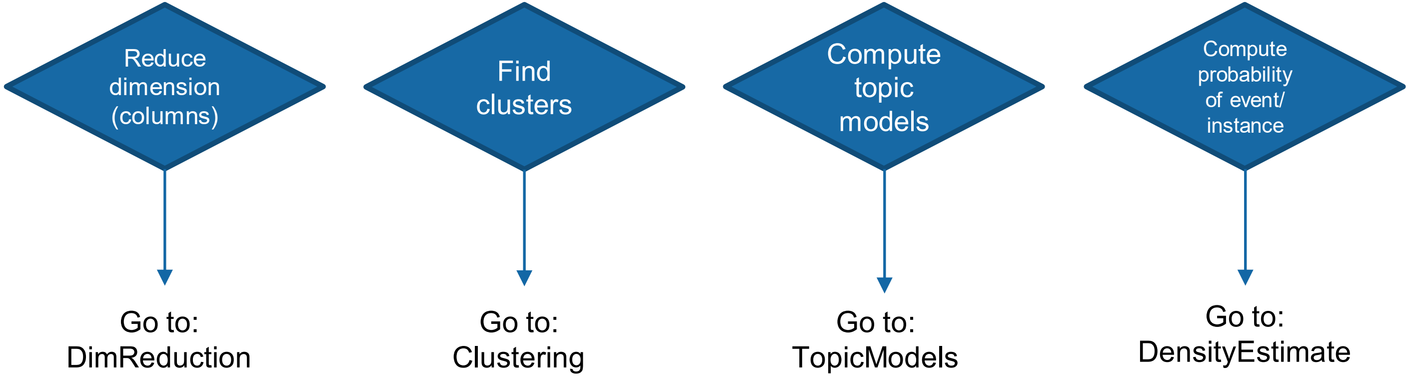
These decision flows are meant as a starting guide only: they should not replace actual thinking about the problem.

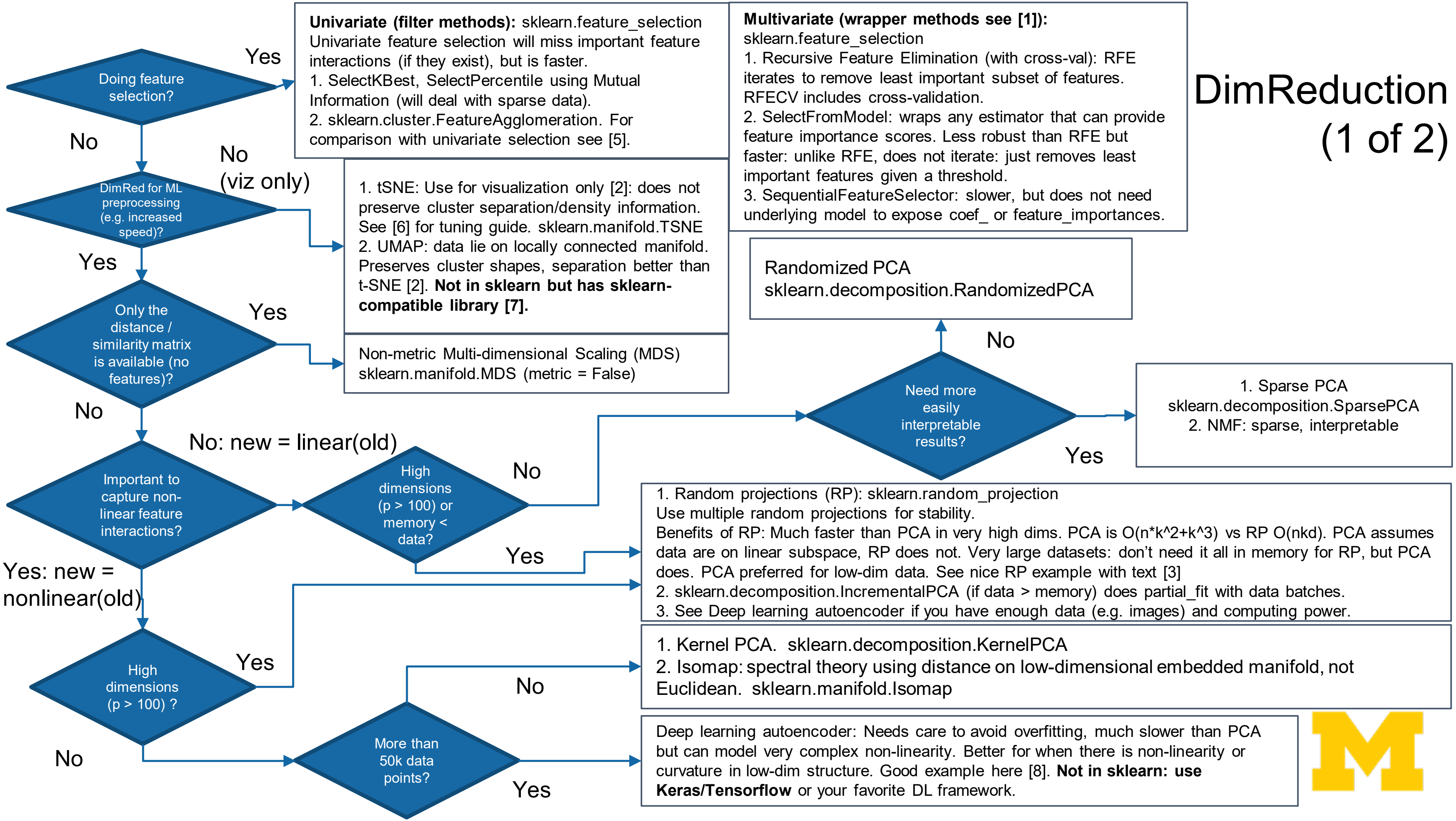
Current assumptions include:
- No missing features or labels
- Roughly balanced classes
- Simple target variable types (e.g. no structures, graphs, permutations, …)

Unsupervised

No target labels

What do you need from your unlabeled data?



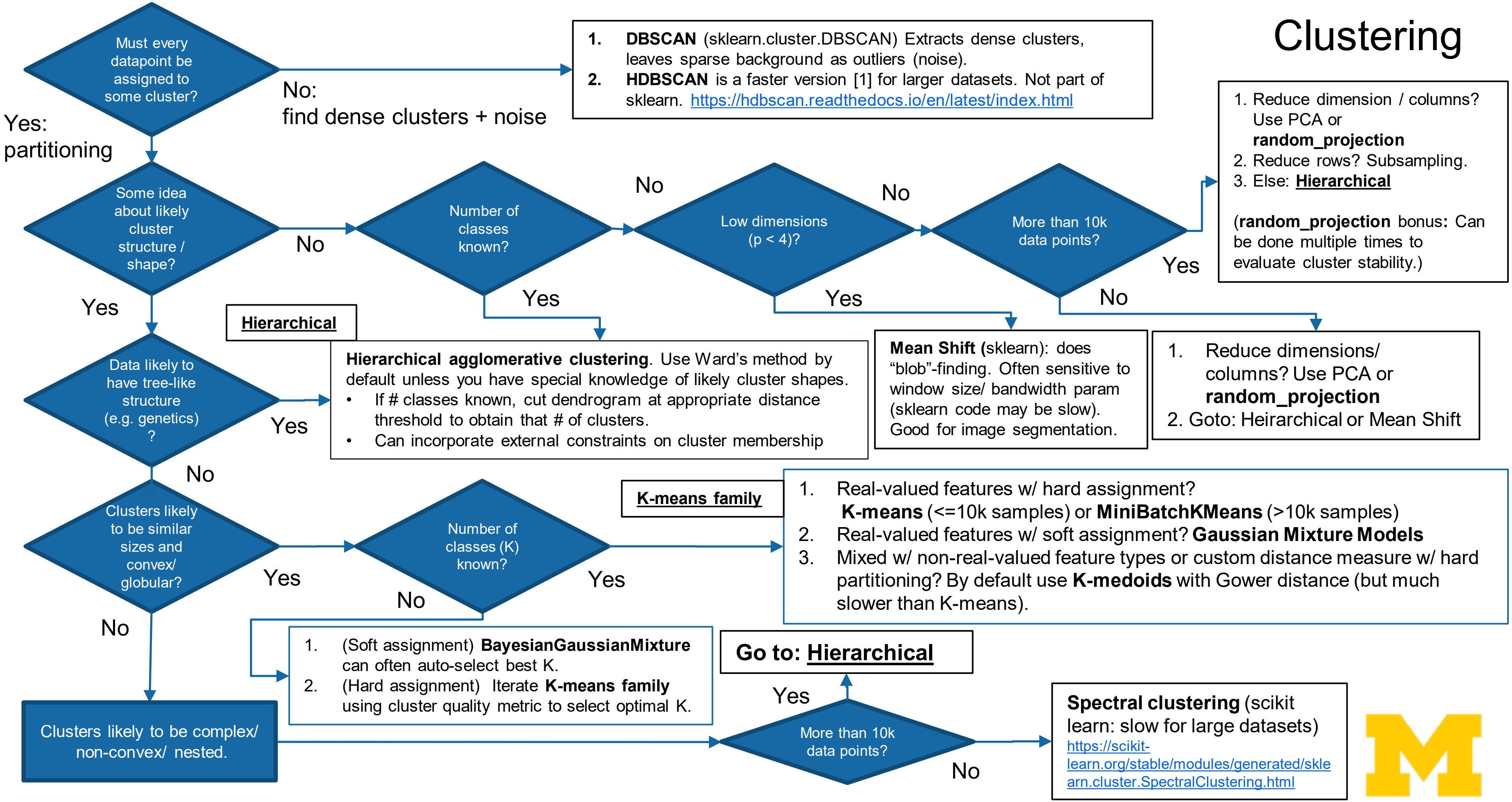


DimReduction
(1 of 2)
Doing feature selection?
Yes
Univariate (filter methods): sklearn.feature_selection
Univariate feature selection will miss important feature interactions (if they exist), but is faster.
1. SelectKBest, SelectPercentile using Mutual Information (will deal with sparse data).
2. sklearn.cluster.FeatureAgglomeration. For comparison with univariate selection see [5].
Multivariate (wrapper methods see [1]):
sklearn.feature_selection
1. Recursive Feature Elimination (with cross-val): RFE iterates to remove least important subset of features. RFECV includes cross-validation.
2. SelectFromModel: wraps any estimator that can provide feature importance scores. Less robust than RFE but faster: unlike RFE, does not iterate: just removes least important features given a threshold.
3. SequentialFeatureSelector: slower, but does not need underlying model to expose coef_ or feature_importances.
No
DimRed for ML preprocessing (e.g. increased speed)?
No
(viz only)
1. tSNE: Use for visualization only [2]: does not preserve cluster separation/density information. See [6] for tuning guide. sklearn.manifold.TSNE
2. UMAP: data lie on locally connected manifold. Preserves cluster shapes, separation better than t-SNE [2]. Not in sklearn but has sklearn-compatible library [7].
Yes
Only the distance / similarity matrix is available (no features)?
Yes
Non-metric Multi-dimensional Scaling (MDS)
sklearn.manifold.MDS (metric = False)
No
Important to capture non-linear feature interactions?
No: new = linear(old)
High dimensions (p > 100) or memory < data?
No
Need more easily interpretable results?
No
Randomized PCA
sklearn.decomposition.RandomizedPCA
Yes
1. Sparse PCA
sklearn.decomposition.SparsePCA
2. NMF: sparse, interpretable
Yes
1. Random projections (RP): sklearn.random_projection
Use multiple random projections for stability.
Benefits of RP: Much faster than PCA in very high dims. PCA is O(n*k^2+k^3) vs RP O(nkd). PCA assumes data are on linear subspace, RP does not. Very large datasets: don't need it all in memory for RP, but PCA does. PCA preferred for low-dim data. See nice RP example with text [3]
2. sklearn.decomposition.IncrementalPCA (if data > memory) does partial_fit with data batches.
3. See Deep learning autoencoder if you have enough data (e.g. images) and computing power.
Yes: new = nonlinear(old)
High dimensions (p > 100) ?
Yes
No
More than 50k data points?
No
1. Kernel PCA. sklearn.decomposition.KernelPCA
2. Isomap: spectral theory using distance on low-dimensional embedded manifold, not Euclidean. sklearn.manifold.Isomap
Yes
Deep learning autoencoder: Needs care to avoid overfitting, much slower than PCA but can model very complex non-linearity. Better for when there is non-linearity or curvature in low-dim structure. Good example here [8]. Not in sklearn: use Keras/Tensorflow or your favorite DL framework.

DimReduction (2 of 2)

[1] R. Kohavi, G.H. John. Wrappers for feature subset selection. https://ai.stanford.edu/~ronnyk/wrappersPrint.pdf
[2] N. Oskolkov. 2020. tSNE vs UMAP: Global Structure https://towardsdatascience.com/tsne-vs-umap-global-structure-4d8045acba17
[3] B. Schmidt. 2018. "Stable Random Projection: Lightweight, General-Purpose Dimensionality Reduction for Digitized Libraries." *Journal of Cultural Analytics* 3 (1). https://doi.org/10.22148/16.025.
[4] L. Nguyen, S. Holmes. 2019. Ten quick tips for effective dimensionality reduction. https://doi.org/10.1371/journal.pcbi.1006907
[5] https://scikit-learn.org/stable/auto_examples/cluster/plot_feature_agglomeration_vs_univariate_selection.html
[6] https://distill.pub/2016/misread-tsne/
[7] https://umap-learn.readthedocs.io/en/latest/basic_usage.html
[8] https://ekamperi.github.io/machine%20learning/2021/01/21/encoder-decoder-model.html

Clustering



[1] https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html

Topic Models

Need probabilistic framework? (input priors, output posterior)

No

Do output topic weights need to be additive/more interpretable (>=0)?

No

Latent Semantic Indexing/Analysis.
a. If fit with N topics, can instantly truncate to K<N topics later.
b. Term and doc similarity matrices are an easy byproduct,
e.g term-term similarity is UU^T, doc-doc similarity is VV^T
c. LSA is best at finding a compact semantic representation. LDA best learns descriptive topics [3].

Yes

1. Non-negative Matrix Factorization ensures the topics/factors contain coherent word groups from the original data.
2. Otherwise, LDA at least has interpretable non-negative probabilities.

Yes

Is the average document length > 100 words?

Yes

Do you have > 3000 documents?

Yes

Do you already know how many topics?

Yes

Use Latent Dirichlet Allocation with specified number of topics.
To update the topics based on new incoming documents use Online LDA for fast model update.

No

No

LDA is bad with short docs or too few docs [1].
For short docs consider specialized version called GSDMM (Gibbs Sampling Dirichlet Multinomial Mixture) [2]. https://github.com/rwalk/gsdmm

No

1. Use Latent Dirichlet Allocation, and tune #topics hyperparameter for minimum perplexity and/or maximum coherence.
2. Hierarchical Dirichlet process topic models infer the number of topics from the data. See Gensim: https://radimrehurek.com/gensim/models/hdpmodel.html

Source:

[1] Understanding the Limiting Factors of Topic Modeling via Posterior Contraction Analysis. ICML 2014. http://proceedings.mlr.press/v32/tang14.pdf
[2] A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering. KDD 2014. https://dl.acm.org/doi/10.1145/2623330.2623715
[3] Exploring Topic Coherence over Many Models and Many Topics. EMNLP 2012. https://aclanthology.org/D12-1087.pdf

Density Estimation

Evaluating density estimators: (1) average test negative log-likelihood across cross-validation folds. Lower is better. (2) task-based evaluation based on metric for downstream task that uses the density estimate.

All features/ variables real-valued with continuous distribution?

No

All features have discrete distribution?

Yes

Use **Histogram** with one bin per possible discrete feature value combination. Bin estimate: compute empirical (observed) probability of joint feature observation in bin. For bins with missing or rare combinations of feature values use smoothing, e.g. with pseudocounts (e.g. Laplace smoothing).

No (mixture of types)

Histograms: simplest form of density estimate. Not in sklearn: use **NumPy histogram.** Bin estimate: compute empirical (observed) probability of events in bin. Highly sensitive to choice of binning scheme.
Beware: some continuous variables may actually follow a discrete distribution, since these frequently arise from quantized measurements, e.g. rounding, truncating decimal places, computing fractional values.

sklearn.mixture.GaussianMixture
Density estimate is superposition of K Gaussian components.
Parameters: Must estimate K (# Gaussians to use) and the means + variances of the K Gaussians.
As with clustering, you should try multiple different initializations [3, p 4]
If data is unlikely to follow known parametric form, **use non-parameteric KernelDensity method**.

Yes

Low dimensions? (p <= 3)

Yes

No

Mid-to-High dimensions (3 < p <= 50)?

No (p > 50)

Naïve Bayes can work surprisingly well for very high dimensions p > 50 (ESL 6.6.3 p211) and also mixtures of column/feature types. Density estimation for NB case is not in sklearn: use **scipy.stats** to get density estimate for each feature (column) then using Bayes Rule, combine log probabilities across all variables. NB makes strong assumptions about conditional independence of features, but can handle mixed variable types, very fast to train and evaluate even in very high dimensions.
Column distribution to use → binary var: Bernoulli; count var: Multinomial; continuous var: Gaussian/Normal.

If important to capture conditional dependence structure between vars, use **probabilistic graphical models** PGMs for approx. multivariate joint distributions.
If doing **exploratory data analysis/visualization** see [4] especially mode-finding with hexagonal plot.

Yes

More than 30k data points?

No

Neighbor-based approach (nonparametric): Kernel Density Estimation. **sklearn.neighbors.KernelDensity**

Kernel type: Gaussian is fine for most applications. Kernel bandwidth: crucial to tune: can use standard cross-validation.
Method: Use Ball tree (instead of K-D tree) for higher dim data (p > 3). Trade off computation time for accuracy using atol and rtol (absolute and relative tolerance) parameters.

Yes

Deep learning: Normalizing Flows (e.g. Real NVP and many other flavors). Transforms a simple density (prior distribution) to a more complex one (posterior distribution) via sequence of invertible transformations [1]. Not available in scikit-learn. One code example using Tensorflow for Real NVP here [2].

ESL refers to Elements of Statistical Learning, 2nd ed. Hastie, Tibshirani, Friedman. Springer.
[1] I. Kobyzev et al. Normalizing Flows: An Intro & Review: https://arxiv.org/pdf/1908.09257.pdf
[2] M. Maria et al. Density estimation using Real NVP. https://keras.io/examples/generative/real_nvp/
[3] Wang & Scott. Nonparametric Density Estimation for High-dim Data. https://arxiv.org/pdf/1904.00176.pdf
[4] Scott, David W. (2004) : Multivariate Density Estimation and Visualization, Papers, No. 2004,16, Humboldt-Universität zu Berlin, Center for Applied Statistics and Economics (CASE), Berlin https://www.econstor.eu/bitstream/10419/22190/1/16_ds.pdf

Classifier

Existing structured features?

No

Got similarity / proximity matrix?

No

Huge amounts of raw data?

No

Go to: Classifier/Deep (few-shot)
Exploit existing pre-trained model in same/similar domain.

Yes

Yes

Yes

Go to: Classifier/Deep
Train new model.

N x N proximity matrix (ESL 18.5)

1. K-NN classifier
2. Kernel-based methods: sklearn.svm.SVC with kernel = ‘precomputed’ and passing Gram matrix to ‘fit’ method instead of dataset.

High-dim features p >> N?

Yes

Classifier/High-Dim: many more features (columns) than data points (rows). See ESL 18.3.

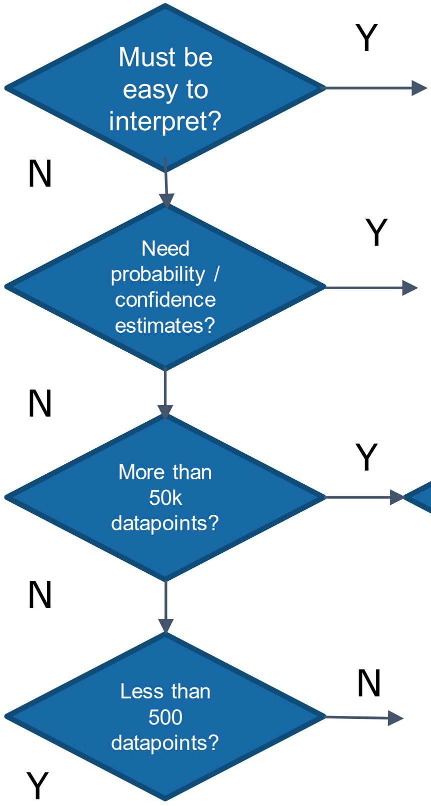
1. Logistic regression w/ L1 or ElasticNet regularization
2. Kernel SVC w/ radial basis kernel
3. Consider feature hashing for the most extreme scenarios. See the excellent vowpal wabbit package https://vowpalwabbit.org/ which is highly scalable both in terms of number of features and training data size.

No

Go to: Classifier/Standard

ESL refers to Elements of Statistical Learning, 2nd ed. Hastie, Tibshirani, Friedman. Springer.

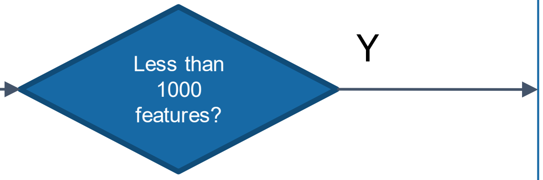
Classifier/Standard



1. Decision tree (pruned w/ max depth)
2. Logistic regression (L1 regularization)
3. Naïve Bayes as first baseline or last resort.

Calibrated classifiers: Fit sigmoid or isotonic regression (calibrator function) to:
1. Logistic regression (L1/L2 regularization)
2. Other existing method, e.g. Random Forests.

scikit-learn: CalibratedClassifierCV
Make sure to use a separate dataset for calibration different from your train, validation, test sets!



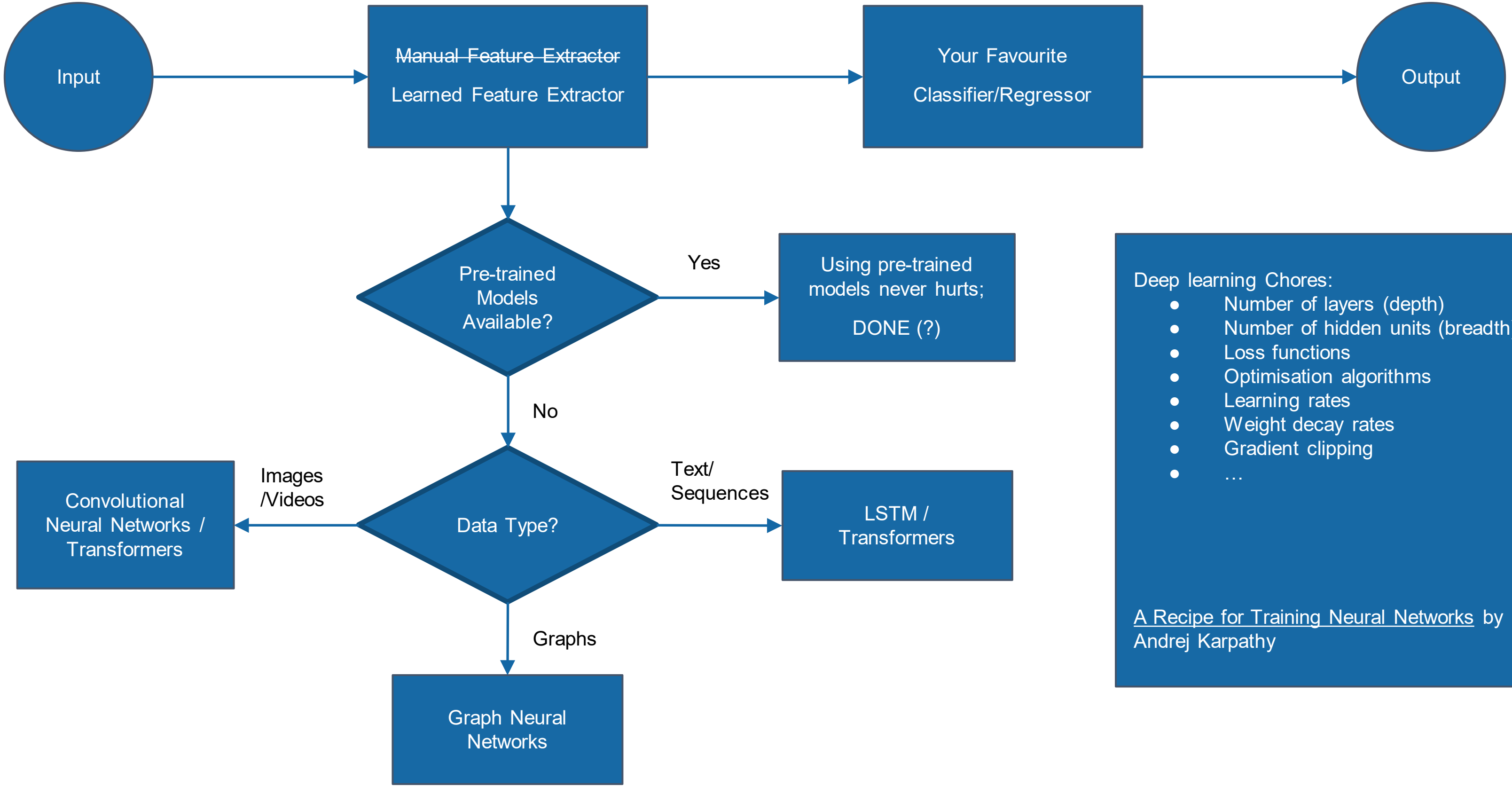
1. Random Forests (parallelized, easier to tune)
2. SGDClassifier
3. XGBoost (high accuracy but tricky/expensive to tune)
4. Logistic regression (tuned L1 + L2)
5. External package vowpal wabbit (see high-dim)
6. Naïve Bayes as fast baseline or last resort.

N: Go to Classifier/High-Dim

Discriminative methods:
1. Logistic regression
2. Random Forests
3. Gradient-boosted decision trees/XGBoost

Generative methods:
Naïve Bayes, GMM, custom.

Classifier/Deep

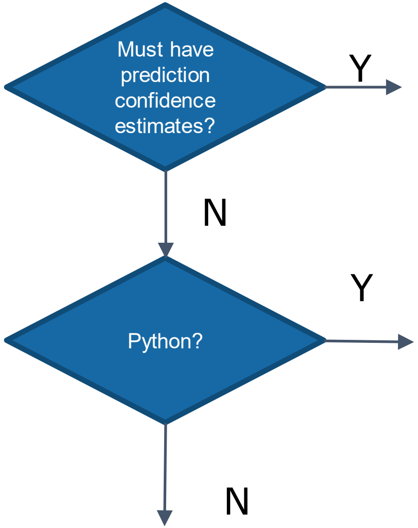


Deep learning Chores:

- Number of layers (depth)
- Number of hidden units (breadth)
- Loss functions
- Optimisation algorithms
- Learning rates
- Weight decay rates
- Gradient clipping
- …

A Recipe for Training Neural Networks by Andrej Karpathy

Ordinal



Gaussian Process Ordinal Regression (separate program, not in scikit-learn)
http://www.gatsby.ucl.ac.uk/~chuwei/ordinalregression.html

mord: Ordinal Regression in Python (not in scikit-learn)
https://pythonhosted.org/mord/

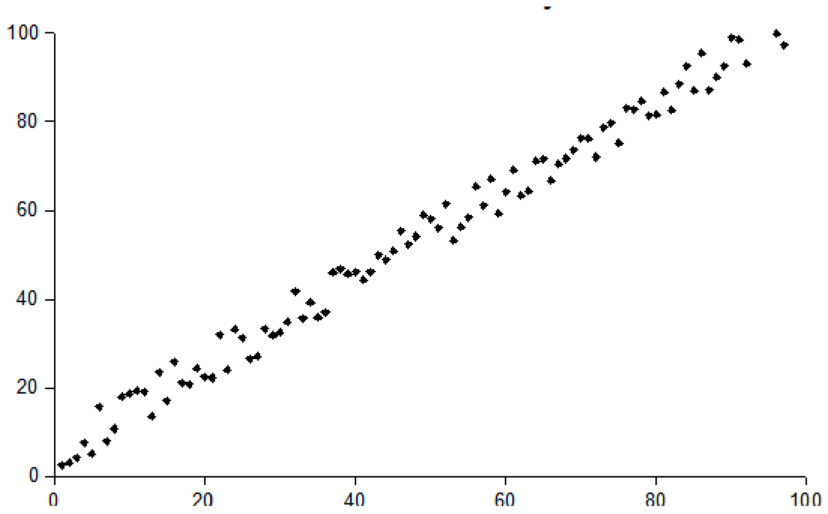
Out of scope..

Regression plan

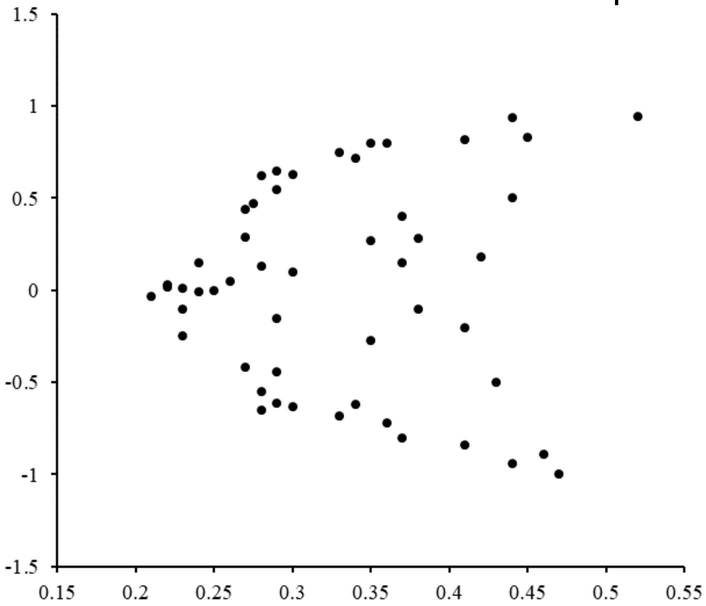
- Background on some regression terms
- How much training data is enough for regression?
- Regression flow chart

Homoscedasticity (say 3x fast)

Yes homoscedastic: noise is even for all x



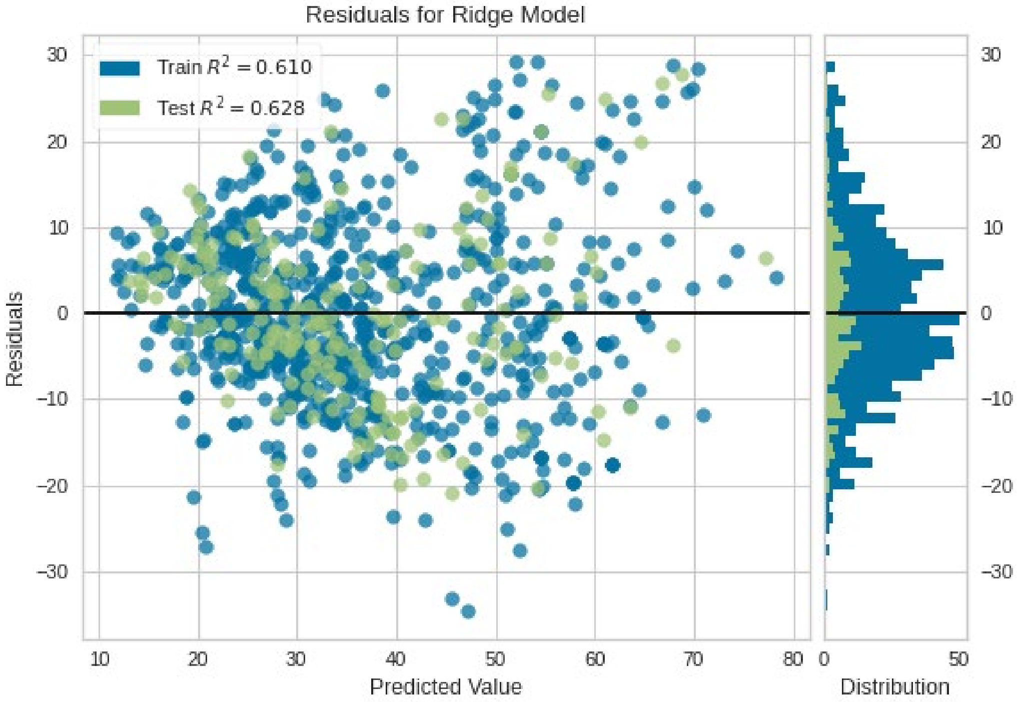
Not homoscedastic: noise depends on x



This is one example that motivates the need for enough training data

Source: https://en.wikipedia.org/wiki/Least_squares

Regression residuals



A <u>residual</u> point shows the error in prediction for a single example.

It's the difference between the true (observed) value and the predicted value of the <u>target variable</u>.

A good linear model fit will have

1. Residual points that are randomly dispersed symmetrically around the horizontal axis.
2. A histogram of the above dispersions that is normally distributed around zero.

Source: https://www.scikit-yb.org/en/latest/api/regressor/residuals.html

How much training data for regression?

- For ordinary linear regression, 30 rows per parameter/feature is a safe bet.
  - e.g. For a one-independent variable linear regression plus a constant you would need 60 examples.
  - You would only need 30 examples if you know the constant (intercept).
- You need enough points to check assumptions about the data and possibly more robust model selection.
  - Linearity plus homoscedasticity (the variance in noise is same for all observations and doesn't depend on the values of the features)
  - Low correlation of input variables/features (check for multi-colinearity)
  - Checking that residuals are random/normally distributed.
  - Capturing possible interaction effects
  - Doing stepwise variable selection? Double it: 60 examples per parameter/features
- The amount of training data depends on:
  - The specific application
  - How well you know the data, noise and missingness patterns, etc.
  - The objective of the modeling: pure prediction vs confident parameter insights.

What if I don't have that much data?

- If you have only 10 to 30 examples per feature, use robust alternatives (see chart) unless you have strong theory about the data or some reason to need least squares.
    - HuberRegressor, RANSACRegressor

- With even fewer data points, say only 5-10 per feature you might still use OLS BUT with caveats. You should consider:

  (a) making the predictions more robust (and reporting confidence intervals!) by using bootstrapping, or use robust regression methods that are less influenced by outliers

  (b) add additional model assumptions by using Bayesian methods like BayesianRidge with specific priors on the parameters.

Regression

Note: highly simplified decision-making here, and no discussion of hyperparameter tuning or computation constraints. For more in-depth coverage of linear regression see ESL Chapter 2.

Text input?

Y → Text Regression: With short n-grams as features, use penalized linear model, e.g. SGDRegressor with L1 penalty OR SVR (support vector regressor). More advanced: see Deep Learning flowchart.

N ↓

Try a linear relationship first (especially with large dataset)
Go to Regression/Linear as a subroutine to try the methods there first.

High R-squared value and random residuals?

Y → If you get very good fit (low RMSE) AND the residuals appear random and normally distributed, you're done.

N ↓

Polynomial regression reasonable?

Y → Compute PolynomialFeatures, retry Linear with sparse/few features option.

N →

1. RandomForestRegressor. Possibly support vector regression (SVR) with RBF kernel with smaller datasets.
2. Use multi-adaptive regression splines package: py-earth https://contrib.scikit-learn.org/py-earth/content.html

Regression/Linear

What are you forecasting about target variable?

1. The median : use mean absolute error (MAE) as the loss metric instead of RMSE.
2. Expected value in some quantile of the target's distribution: use quantile regression.

The mean: use root mean squared error as loss metric

Need probability / confidence estimates?

Y

1. BayesianRidge, especially if you have specific priors on parameters.
2. For general confidence estimates from non-Bayesian regression, use bootstrap estimator with any selected model below (ESL 7.11).

N

Sparse/few features are important?

Y

Use Lasso. Could also use SGDRegressor with either L1 penalty, or ElasticNet penalty, which adds L2 parameter shrinkage like ridge regression, depending on expected nature of sparsity you need and computation speed. To help select the features, you can get the entire Lasso variable selection path using scikit-learn's Lars linear model (least-angle regression: see also ESL 3.4.4)

N

At least 30 examples per feature?

Y

Ordinary least-squares (OLS):
LinearRegression for uncorrelated features.
SGDRegression for possible correlated features and large datasets.

N

At least 15 examples per feature?

Y

Robust least-squares:
e.g. HubertRegression
or RANSACRegression

N

With only 5-10 examples per feature you could still use OLS BUT with caveats. You should consider (a) making the predictions more robust (and reporting confidence intervals!) by using bootstrapping, or use robust regression methods that are less influenced by outliers; (b) add additional model assumptions by using Bayesian methods like BayesianRidge with specific priors on the parameters.

UNIVERSITY OF MICHIGAN

Kevyn Collins-Thompson

kevynct@umich.edu

School of Information