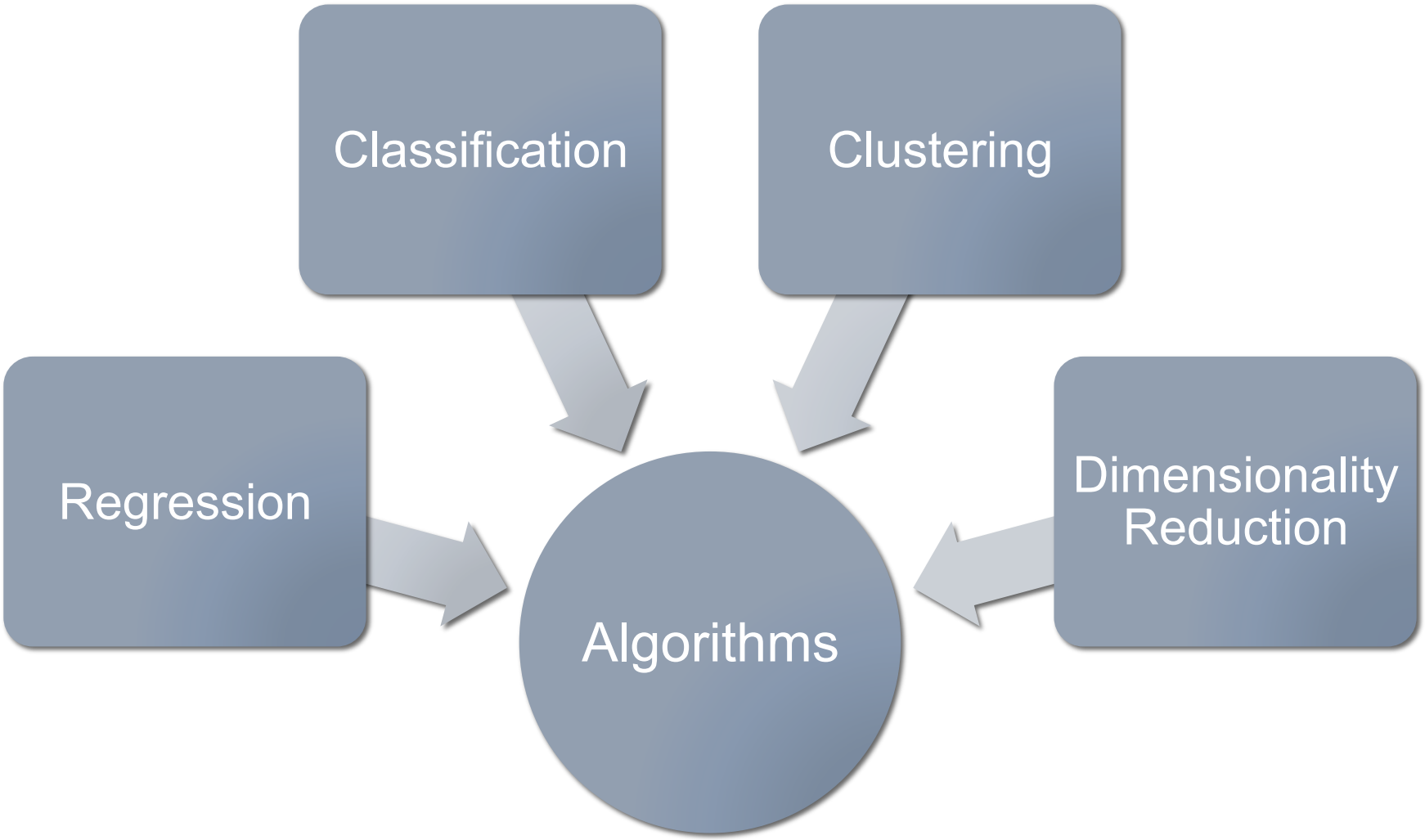
# MACHINE LEARNING USING R

Surbhi

# Machine Learning Tasks



# No Free Lunch

- The “No Free Lunch” theorem states that there is no one model that works best for every problem.
- Example: One model may work well to interpret Latin script, but the same model may not work well to interpret Cyrillic or Hebrew or Greek scripts.
- Assumptions for the model may hold in one situation but may not hold in another.
- Try multiple models and find one that works best for a particular problem.

# REGRESSION

# Regression

- Regression is the supervised learning task for modeling and predicting **continuous, numeric** variables.

- k-Nearest Neighbors
- Linear Regression & Regularized Linear Regression
- Regression Trees (Decision Trees, Random Forests)
- Neural Networks

# k-Nearest Neighbors

- Nearest neighbors algorithms are "instance-based," which means that that save each training observation. They then make predictions for new observations by searching for the most similar training observations and pooling their values.

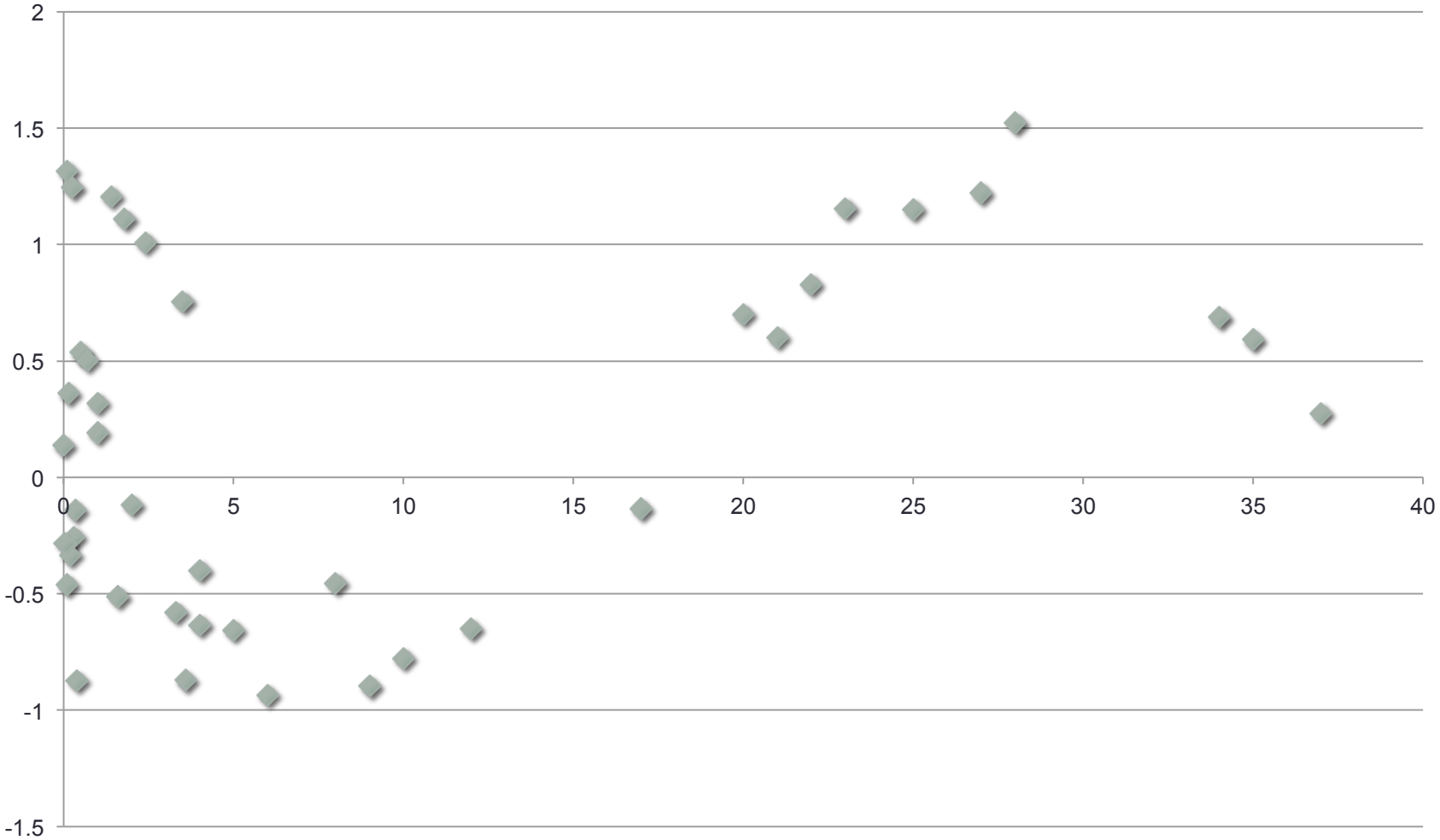
- These algorithms are memory-intensive, perform poorly for high-dimensional data, and require a meaningful distance function to calculate similarity.

# Linear Regression

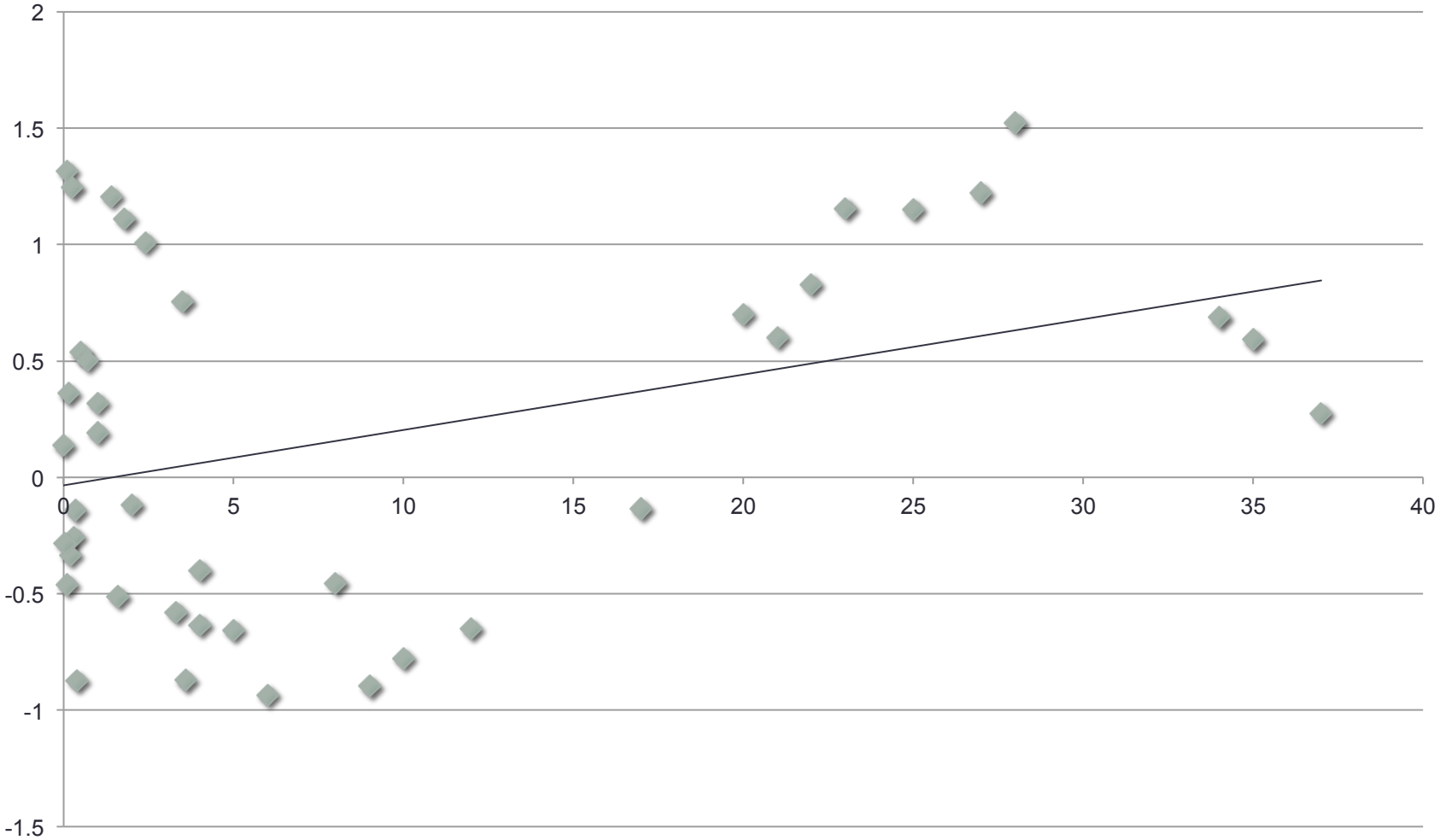
- **Strengths:** Linear regression is straightforward to understand and explain, and can be regularized to avoid overfitting. In addition, linear models can be updated easily with new data using stochastic gradient descent.

- **Weaknesses:** Linear regression performs poorly when there are non-linear relationships. They are not naturally flexible enough to capture more complex patterns, and adding the right interaction terms or polynomials can be tricky and time-consuming.

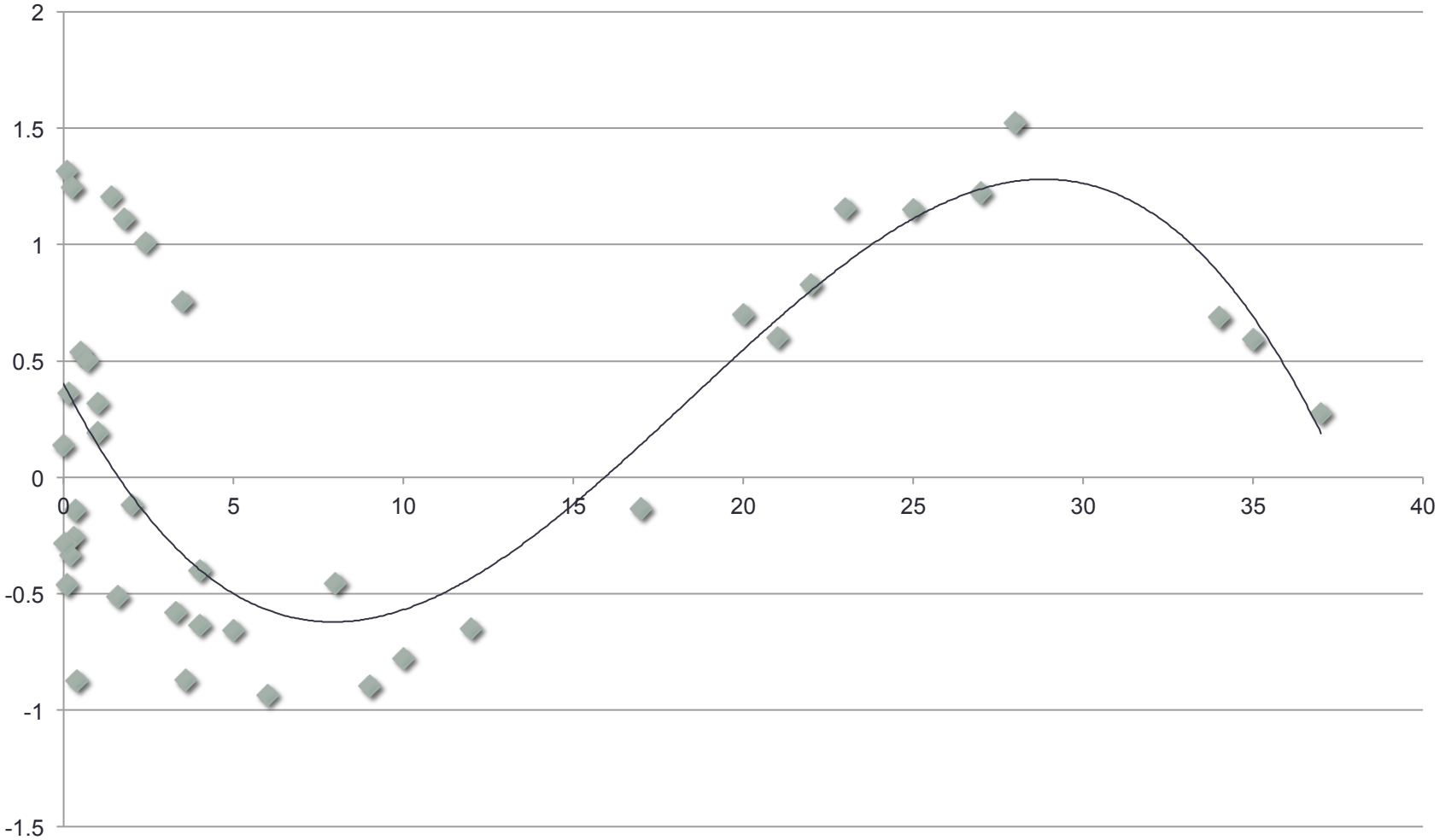
# Data



# Data



# Data



# Regularized Linear regression

- Regularization is a technique for penalizing large coefficients in order to avoid overfitting, and the strength of the penalty should be tuned.
  - LASSO
  - Ridge
  - Elastic-Net

- R Package '`glmnet`'

# Regression Trees

- **Strengths:** Decision trees can learn non-linear relationships, and are fairly robust to outliers. Ensembles perform very well in practice, winning many classical (i.e. non-deep-learning) machine learning competitions.

- **Weaknesses:** Unconstrained, individual trees are prone to overfitting because they can keep branching until they memorize the training data. However, this can be alleviated by using ensembles.

# Neural Networks

- **Strengths:** Deep learning is the current state-of-the-art for certain domains, such as computer vision and speech recognition. Deep neural networks perform very well on image, audio, and text data, and they can be easily updated with new data using batch propagation. Their architectures (i.e. number and structure of layers) can be adapted to many types of problems, and their hidden layers reduce the need for feature engineering.

- **Weaknesses:** Deep learning algorithms are usually not suitable as general-purpose algorithms because they require a very large amount of data. In fact, they are usually outperformed by tree ensembles for classical machine learning problems. In addition, they are computationally intensive to train, and they require much more expertise to tune (i.e. set the architecture and hyperparameters).

# CLASSIFICATION

# Classification

- Classification is the supervised learning task for modeling and predicting **categorical** variables.

- Logistic Regression
- Classification Trees (Decision Trees, Random Forests)
- Deep Learning
- Support Vector Machines (SVM)
- Naïve Bayes

# Logistic regression

- Logistic regression is the classification counterpart to linear regression. Predictions are mapped to be between 0 and 1.

- **Strengths:** Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting. Logistic models can be updated easily with new data using stochastic gradient descent.

- **Weaknesses:** Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

# Classification Tree

- Classification trees are the classification counterparts to regression trees.

- **Strengths:** As with regression, classification tree ensembles also perform very well in practice. They are robust to outliers, scalable, and able to naturally model non-linear decision boundaries thanks to their hierarchical structure.

- **Weaknesses:** Unconstrained, individual trees are prone to overfitting, but this can be alleviated by ensemble methods.

# Deep Learning

- Deep learning is also easily adapted to classification problems. In fact, classification is often the more common use of deep learning, such as in image classification.

- **Strengths:** Deep learning performs very well when classifying for audio, text, and image data.

- **Weaknesses:** As with regression, deep neural networks require very large amounts of data to train, so it's not treated as a general-purpose algorithm.

- R Package '`MXNet`'

# Support Vector Machines (SVM)

- **Strengths:** SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.

- **Weaknesses:** However, SVM's are memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets. Currently in the industry, random forests are usually preferred over SVM's.

- R Package '`kernlab`'

# Naive Bayes

- Probability table based on training data.
- Core assumption of conditional independence (i.e. all input features are independent from one another) rarely holds true in the real world.

- **Strengths:** Even though the conditional independence assumption rarely holds true, Naïve Bayes models actually perform surprisingly well in practice, especially for how simple they are. They are easy to implement and can scale with your dataset.

- **Weaknesses:** Due to their sheer simplicity, NB models are often beaten by models properly trained and tuned using the other algorithms.

# CLUSTERING

# Clustering

- Clustering is an unsupervised learning task for finding natural groupings of observations.
- Because clustering is unsupervised (i.e. there's no "right answer"), data visualization is usually used to evaluate results.
- If you have pre-labeled clusters in your training set, then classification algorithms are typically more appropriate.

- K-Means
- Affinity Propagation
- Hierarchical Clustering
- DBSCAN

# K-Means

- Based on *geometric distances* i.e. distance on a coordinate plane between points.

- **Strengths:** K-Means is hands-down the most popular clustering algorithm because it's fast, simple, and surprisingly flexible if you pre-process your data and engineer useful features.

- **Weaknesses:** The user must specify the number of clusters, which won't always be easy to do. In addition, if the true underlying clusters in your data are not globular, then K-Means will produce poor clusters.

# Affinity Propagation

- Based on *graph distances* between points.
- **Strengths:** The user doesn't need to specify the number of clusters but does need to specify 'sample preference' and 'damping' hyperparameters.
- **Weaknesses:** The main disadvantage of Affinity Propagation is that it's quite slow and memory-heavy, making it difficult to scale to larger datasets. In addition, it also assumes the true underlying clusters are globular.
- R Package '`apcluster`'

# Hierarchical Clustering

- Start with each point in its own cluster. For each cluster, merge it with another based on some criterion. Repeat until only one cluster remains and you are left with a hierarchy of clusters.

- **Strengths:** The main advantage of hierarchical clustering is that the clusters are not assumed to be globular. In addition, it scales well to larger datasets.

- **Weaknesses:** Much like K-Means, the user must choose the number of clusters.

- R Function '`hclust`' in '`stats`' package

# DBSCAN

- DBSCAN is a density based algorithm that makes clusters for dense regions of points. There's also a recent new development called HDBSCAN that allows varying density clusters.

- **Strengths:** DBSCAN does not assume globular clusters, and its performance is scalable. It does not require every point to be assigned to a cluster, reducing the noise of the clusters (this may be a weakness, depending on your use case).

- **Weaknesses:** The user must tune the hyperparameters 'epsilon' and 'min_samples,' which define the density of clusters. DBSCAN is quite sensitive to these hyperparameters.

- R Package '`dbscan`'

# DIMENSIONALITY REDUCTION

# The Curse of Dimensionality

- In machine learning, dimensionality simply refers to the number of features or input variables in your dataset.

- When the number of features is very large relative to the number of observations in your dataset, certain algorithms struggle to train effective models. This is called the Curse of Dimensionality.

- It is especially relevant for clustering algorithms that rely on distance calculations.

# Feature Selection & Feature Extraction

- Feature selection is  filtering irrelevant or redundant features from your dataset.

- Feature extraction is creating a new, smaller set of features that stills captures most of the useful information.

- The key difference between feature selection and extraction is that feature selection keeps a subset of the original features while feature extraction creates brand new ones.

# FEATURE SELECTION

# Feature Selection

- Feature selection is  filtering irrelevant or redundant features from your dataset.

- Variance Thresholds
- Correlation Thresholds
- Genetic algorithms (GA)

# Variance Thresholds

- Variance thresholds remove features whose values don't change much from observation to observation
- Normalize features first as variance is dependent on scale.
- **Strengths:** Applying variance thresholds is based on solid intuition: features that don't change much also don't add much information. This is an easy and relatively safe way to reduce dimensionality at the start of your modeling process.
- **Weaknesses:** If your problem does require dimensionality reduction, applying variance thresholds is rarely sufficient. Furthermore, you must manually set or tune a variance threshold, which could be tricky. We recommend starting with a conservative (i.e. lower) threshold.
- R Function '`nearZeroVar`'

# Correlation Thresholds

- Correlation thresholds remove features that are highly correlated with others.

- **Strengths:** Applying correlation thresholds is also based on solid intuition: similar features provide redundant information. Some algorithms are not robust to correlated features, so removing them can boost performance.

- **Weaknesses:** Again, you must manually set or tune a correlation threshold, which can be tricky to do. Plus, if you set your threshold too low, you risk dropping useful information. Whenever possible, we prefer algorithms with built-in feature selection over correlation thresholds. Even for algorithms without built-in feature selection, Principal Component Analysis (PCA) is often a better alternative.

- R Function '`findCorrelation`'

# Genetic algorithms (GA)

- Broad class of search algorithms that can be adapted to different purposes.
- In machine learning, GA's have two main uses.
  - Optimization - finding the best weights for a neural network.
  - Supervised feature selection
- **Strengths:** Genetic algorithms can efficiently select features from very high dimensional datasets, where exhaustive search is unfeasible. When you need to preprocess data for an algorithm that doesn't have built-in feature selection (e.g. nearest neighbors) and when you must preserve the original features (i.e. no PCA allowed), GA's are likely your best bet.
- **Weaknesses:** GA's add a higher level of complexity to your implementation, and they aren't worth the hassle in most cases. If possible, it's faster and simpler to use PCA or to directly use an algorithm with built-in feature selection.
- R Package '`GA`'

# FEATURE EXTRACTION

# Feature Extraction

- Feature extraction is creating a new, smaller set of features that stills captures most of the useful information.

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Autoencoders

# Principal Component Analysis (PCA)

- Principal component analysis (PCA) is an unsupervised algorithm that creates linear combinations of the original features.
- The new features are orthogonal, which means that they are uncorrelated.
- They are ranked in order of their "explained variance." The first principal component (PC1) explains the most variance in your dataset, PC2 explains the second-most variance, and so on.
- Normalize features first as transformation is dependent on scale.
- **Strengths:** PCA is a versatile technique that works well in practice. It's fast and simple to implement, which means you can easily test algorithms with and without PCA to compare performance. In addition, PCA offers several variations and extensions (i.e. kernel PCA, sparse PCA, etc.) to tackle specific roadblocks.
- **Weaknesses:** The new principal components are not interpretable, which may be a deal-breaker in some settings. In addition, you must still manually set or tune a threshold for cumulative explained variance.
- R Function '`prcomp`' in '`stats`' package

# Linear Discriminant Analysis (LDA)

- Linear discriminant analysis (LDA) creates linear combinations of your original features.
- Unlike PCA, LDA doesn't maximize explained variance. Instead, it maximizes the separability between classes. Therefore, LDA is a supervised method that can only be used with labeled data.
- The LDA transformation is also dependent on scale, so you should normalize your dataset first.
- **Strengths:** LDA is supervised, which can improve the predictive performance of the extracted features. Furthermore, LDA offers variations (i.e. quadratic LDA) to tackle specific roadblocks.
- **Weaknesses:** As with PCA, the new features are not easily interpretable, and you must still manually set or tune the number of components to keep. LDA also requires labeled data, which makes it more situational.
- R Function '`lda`' in '`MASS`' package

# Autoencoders

- Autoencoders are neural networks that are trained to reconstruct their original inputs.
- Input image is the target output, therefore autoencoders are considered unsupervised. They can be used directly (e.g. image compression) or stacked in sequence (e.g. deep learning).
- **Strengths:** Autoencoders are neural networks, which means they perform well for certain types of data, such as image and audio data.
- **Weaknesses:** Autoencoders are neural networks, which means they require more data to train. They are not used as general-purpose dimensionality reduction algorithms.

# SOMETHING FUN…

# Art of Neural Networks

- https://www.youtube.com/watch?v=0qVOUD76JOg

# Example: Rotten Tomatoes

**CRITIC REVIEWS FOR *GUARDIANS OF THE GALAXY VOL. 2***

All Critics (305) | Top Critics (49) | Fresh (249) | Rotten (56) | DVD (1)

In Marvel lingo, Guardians 2 feels like a great six-issue arc, the kind of storytelling that used to be the backbone of superhero comics.

June 16, 2017 | Full Review…

David Sims
The Atlantic
★ Top Critic

Let's hope that Vol. 3 recaptures the fizz of the original, instead of slumping into the most expensive group-therapy session in the universe.

May 8, 2017 | Full Review…

Anthony Lane
New Yorker
★ Top Critic

The reunion of the Guardians cements Pratt's mega-stardom. He has a shaggy young Kurt Russell vibe (think Big Trouble

As Baby Groot's companions battle the tentacular horror in the background, we're treated to the delightful spectacle of the mini-veggie juking his way through "Mr. Blue Sky" in the opening credits.

May 12, 2017 | Full Review…

Christopher Orr
The Atlantic
★ Top Critic

[Yondu's] presence kicks the idling movie into gear, and into a final third act for which all of the previous meandering can be forgiven - and the talk of family finally accrues the weight the film has been trying to put on it.

May 5, 2017 | Full Review…

Alison Willmore
BuzzFeed News
★ Top Critic

# Example: Rotten Tomatoes

Secure | https://www.theatlantic.com/entertainment/archive/2017/05/guardians-of-the-galaxy-vol-2-twice-is-still-the-charm/525513/

Perhaps the finest, funniest moment in *Guardians of the Galaxy Vol. 2* is the first action sequence. Or perhaps I should put quote marks around that: "action sequence." Because for most of its duration, the action is strictly an afterthought. The titular supergroup has been enlisted to defeat a giant star-squid, and its smallest member, Baby Groot (the twig-like offshoot of last installment's arboreal giant), is hooking up some equipment in the foreground as the fight commences behind him. What is Baby Groot fiddling with? Some kind of space cannon?

Of course not. It's a sound system, and no sooner is it plugged in than the Electric Light Orchestra's pop jingle "Mr. Blue Sky" bursts forth in all its giddy, meteorological splendor. As Baby Groot's companions battle the tentacular horror in the background, we're treated to the delightful spectacle of the mini-veggie juking his way through the opening credits. It is, in its way, the perfect deflation of the time-to-save-the-world-again bloat that has grown customary in the superhero genre, and a worthy successor to the loose, goofy vibe of the first *Guardians*: You guys deal with the Latest Threat to All Life over there; us, we're going to hang here and groove to some oldies.

RELATED STORY

Alas, the magic can't quite last. (As even the song warns, *Mr. Blue, you did it right, but soon comes Mr. Night*...) The *Guardians* sequel and latest installment in the Marvel Cinematic Universe certainly has its moments—quite a few in fact—but too often it finds itself weighted down by just the kind of portentous themes and overwrought drama the first film was so careful to avoid.

# References


- http://scikit-learn.org/stable/user_guide.html
- https://elitedatascience.com/blog