Name	Type	Summary	Pros	Cons
kNN	C	The k nearest examples to the one that has to be predicted (nearest in Euclidean distance when plotted across its features) vote on which class the new example should fall in. The example gets 'democratically' assigned to the winning class.	• Simple • Makes no assumptions about underlying data distribution • Fast training phase	• Does not produce a model, limiting the ability to understand how features are related to the class • Requires selection of an appropriate k • Slow classification phase • Nominal features and missing data require additional processing
Naïve Bayes	C	Applied Bayes theorem to the data to predict the probability of an output. Known as Naïve because of its naïve assumptions about the features being equally important and independent. Often used for text classification (spam filters).	• Simple and fast • Does well with noisy and missing data • Requires few examples for training, but still works well with large datasets • Easy to obtain the estimated probability for a prediction	• Relies on the assumption of 'equally important and independent features' • Not ideal if there are many numeric features • Estimated probabilities are less reliable than the predicted classes
Decision Tree	C	Basically, a big flowchart with binary answers at each node up to the leaf node (result).	• Does well on most types of problems. • Does not require the user to specify the model in advance • Excludes unimportant features • Can be used on both small and large datasets	• Often biased toward splits on features having a large number of levels • It is easy to overfit or underfit the model • Can have trouble modelling some relationships due to reliance

			• Model that can be interpreted without a mathematical background	on axis-parallel splits • Small changes in the training data can result in large changes to decision logic • Large trees are hard to interpret
RIPPER Rule Learner	C	Works very similar to a decision tree, but makes rules out of all the possible paths taken from the root node to the output	• Generates easy to understand, human-readable rules • Efficient on large and noisy datasets • Generally, produces a simpler model than a comparable decision tree	• May result in rules that seem to defy common sense or expert knowledge • Not ideal for working on numeric data • Might not perform as well as more complex models
Multiple Linear Regression	R	An equation in terms of the independent variable (features) that fits the training data as good as possible (trying not to overfit). The features of the future variable to predict are simply plugged into the equation (or plotted onto the regression line graph) to predict its dependent variable.	• Most common approach for modelling numeric data • Can be adapted to model almost any modelling task • Provides estimates of both the strength and size of the relationships among features and the outcome	• Makes strong assumptions about data • The model's form must be specified by the user in advance • Does not handle missing data • Only works with numeric features, so categorical data requires extra processing • Requires some knowledge of statistics to understand the model
Regression Trees	R	Exactly like a decision tree, but to get a numerical prediction, they make predictions based on the average value of the	• Combines the strengths of decision trees with the ability to model numerical data	• Requires a large amount of training data • Difficult to determine the overall net effect of

		examples that reach a leaf.	• Does not require the user to specify the model in advance • Uses automatic feature selection, which allows the approach to be used with a very large number of features • May fit some types of data much better than linear regression • Does not require knowledge of statistics to interpret the model	individual features on the outcome • Large trees can become more difficult to interpret than a regression model
Model Trees	R	Just like Regression Trees, but at each leaf they build a multiple linear regression model from the examples reaching that node.	• All the above • More powerful and more accurate predictions than regression trees	• All the above • Much more complicated to interpret than regression trees
Neural Networks	C/R	Inputs (from nodes) are weighted per their importance (usually calculated with backpropagation) and summed into a new node. The sum is then fed into an activation function (sigmoid usually) that passes on the signal if activated (sum is > than a threshold). Multiple nodes can be layered to model more	• Capable of modelling more complex patterns than nearly any other algorithm • Makes few assumptions about the data’s underlying relationships	• Extremely computationally intensive and slow to train, particularly if the network topology is complex • Very prone to overfitting training data • Results impossible to interpret

		complex relationships.		
Support Vector Machines	C/R	Uses flat hyperplanes to separate the data plotted in multi-dimensional space. The partitions created by the hyperplanes tend to be, more or less, homogeneous. Kernel functions are used to ‘add’ calculated features that were not present in the raw data. This helps redistribute the data in a larger dimensional space to find a better-fit flat hyperplane to partition it.	• Not too much affected by noisy data • Not prone to overfitting • Known to obtain very good results	• Various Kernel functions must be tested to find the best (trial and error) • Very lengthy training for large datasets • Results impossible to interpret
Random Forest	C/R	An ensemble of trees (500 by default in R). After bootstrap sampling the training data it selects random features to train each individual tree to increase diversity. The trees then ‘vote’ the prediction and one final prediction is made from these votes (average for numerical predictions). The votes can be weighted based on previous model performance.	• Ensemble • Performs well on most problems. • Can handle noisy and missing data as well as categorical or continuous features. • ‘Automatically’ selects the most important features. • Can be used on data with extremely large number of features or examples.	• Not easy to interpret. • Requires a lot of work to tune the model to the data.

Gradient Tree Boosting	C/R	A sort of gradient descent algorithm is applied to many different ‘weak’ decision trees built one by one on top of each other. It is an ensemble of trees just like a Random Forest, but instead of working with many powerful models with low bias to decrease the overall variance, it works with many weak model with low variance to decrease the overall bias. Different implementations act slightly different. A common one is XGBoost (Extreme Gradient Boosting)	• Lot of flexibility with the choice of loss function: it can be adapted to the characteristics of the problem at hand • Known for achieving very good results	• Sensitive to small differences: not robust against outliers • Not easy to interpret • Computationally expensive
K-Means	Clu	The data will be divided into K clusters. K centroids are chosen randomly and each point is assigned to the closest centroid’s cluster. The average position between all points in a cluster is then computed and the centroid is moved to the centre of the cluster. Points are reassigned to the new closest centroids. Iterates until all clusters are final. Mahalanobis distance is usually used to get rid of Euclidean bias.	• Running time is O(nkdi) (n = samples, k = clusters, d = dimensions, i = number of iterations until convergence) • Uses simple principles that can be explained in non-statistical terms • Highly flexible: can be easily adapted to any situation • Performs quite well for its simplicity	• Assumes all clusters are normally distributed in all dimensions. • Less sophisticated than modern advanced clustering algorithms • Randomness may not guarantee the cost function reaches a global minimum • Requires a guess to how many clusters are there • Not ideal for non-spherical clusters

CURE	Clu	Hierarchical clustering algorithm that adopts a middle ground between the centroid based and all point extremes. In CURE, a constant number of well scattered points of a cluster are chosen and they are shrunk towards the centroid of the cluster by a fixed amount. The scattered points after shrinking are used as representatives of the cluster. The clusters with the closest pair of representatives are the clusters that are merged. Repeats until k clusters left.	• Allows clusters to assume any shape • It handles well noise in the data set • Can deal well with outliers	• Running time is $O(n^2 \log n)$ • Requires a guess to how many clusters are there
DBSCAN	Clu	Density based clustering algorithm. Clusters are made of point which are density-reachable between each other (a 'dense' area of points close to each other). The distance ε with which to calculate density-reachability is given by the user	• The number of clusters is not required • It can handle a large amount of noise in the data set • It produces arbitrary shaped clusters • Time complexity $O(n \log n)$. • Robust to outliers • Allows clusters to assume any shape	• The user needs to specify the distance ε • Not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed. • Cannot cluster data sets well with large differences in densities