<https://www.techsparks.co.in/hot-topic-for-project-and-thesis-machine-learning/>

|  |  |
| --- | --- |
| K-Nearest Neighbors | |
| * Computes classification form majority or weighted votes of the “K” nearest observation of each data point. | |
| Advantages | Disadvantages |
| * simple to implement * good for large training data * good for noisy training data | * CPU intensive * needs to compute “K” |
| Dependencies | |
| * Eculean Distance | |
| Pseudo | |
| 1. Load the training and test data  2. Choose the value of K  3. For each point in test data:  - find the Euclidean distance to all training data points  - store the Euclidean distances in a list and sort it  - choose the first k points  - assign a class to the test point based on the majority of classes present in the chosen points  4. End | |

|  |  |
| --- | --- |
| Decision Tree | |
| * Applies a sequence of rules to classify the data. | |
| Advantages | Disadvantages |
| * easy to understand * easy to visualize * can handle numeric & categorical data | * can be to complex * small variations have a huge effects |
| Dependencies | |
| * Gini * Entropy | |
| Algorithms | |
| ID3 (Iterative Dichotomiser 3)  C4.5 (successor of ID3)  CART (Classification And Regression Tree)  CHAID (CHi-squared Automatic Interaction Detector) | |

|  |  |
| --- | --- |
| Naïve Bayes | |
| * Bayes’ theorem with the assumption that each pair of features are independent. | |
| Advantages | Disadvantages |
| * Needs less training data * Fast * Simple | * Bad estimator |
| Dependences | |
| Bayes’ Theorem   * Consider each attribute Ai and class label C as random variables. * P(A): probability of observing input attributes A * P(C): probability of class label C (prior probability) * P(A|C): probability of observing A given that C is class label (class-conditional probability) * P(C|A): probability of C as class label given that A was observed (posterior probability) | |
| Pseudo: Gaussian | |
| 1. Read training data 2. Calculate mean and standard deviation for target variable. 3. Repeat    1. Calculate probability with [v1,v2 .. vn] with gauss density equation 4. Calculate class probability 5. Return greatest probability. | |
| Models Types: Gaussian, Multinomial, Bernoulli | |

|  |  |
| --- | --- |
| Feed Forward Neural Network | |
| Characteristics:   * Perceptron are arranged in layers * Layers:   + First -> Input   + Middle -> Hidden   + Last -> Output * Perceptron form the one layer connected to all perceptron of the next layer (reason for name) * Perceptron in the same layer has no connections. | |
| Terms for Diagram | |
| Node | The basic unit of computation (represented by a single circle) |
| Layer | A collection of nodes of the same type and index (i.e. input, hidden, outer layer) |
| Connection | A weighted relationship between a node of one layer to the node of another layer |
| W | The weight of a connection |
| I | Input node (the neural network input) |
| H | Hidden node (a weighted sum of input layers or previous hidden layers) |
| HA | Hidden node activated (the value of the hidden node passed to a predefined function) |
| O | Output node (A weighted sum of the last hidden layer) |
| OA | Output node activated (the neural network output, the value of an output node passed to a predefined function) |
| B | Bias node (always a contract, typically set equal to 1.0) |
| *e* | Total difference between the output of the network and the desired value(s) (total error is typically measured by estimators such as mean squared error, entropy, etc.) |
|  | |
| Pseudo | |
| Initialization   1. Initialize all weights W1 through W12 with a random number from a normal distribution, i.e. ~N(0, 1). 2. Set all bias nodes   Feed Forward   1. Set the values of all input nodes. 2. Calculate hidden node values. 3. Select an activation function for the hidden layer; for example, the Sigmoid function: 4. Calculate hidden node activation values: 5. Calculate output node values: 6. Select an activation function for the output layer; for example, the linear function: 7. Calculate output node activation values: 8. Calculate the total error; if OAi is the obtained output value for node i, then let yi be the desired output.   Algorithms: select one   * Backpropagation * Conjugate gradient * Genetic algorithms * Etc.… | |

|  |
| --- |
| Apriori Algorithm |
| * Use for Association Analysis * Mining frequent item sets for binary association rules * It improves the Apriori |
| Pseudo-code |
| a: Candidate item-set of size k  b : frequent item-set of size k  c = {frequent items};  For ( k = 1; b != null; k++) do begin  a+1 = candidates generated from b;  for each transaction t in database do  increment the count of all candidates in a+1 that are contained in t  b+1 = candidates in a+1 with min support  end  Return ∪kb; |

|  |
| --- |
| FP-Growth Algorithm |
| * Use for Association Analysis * Find frequent item-sets without using candidate generations. |
| Pseudo-code |
| if Tree contains a single path P then  For each combination do generate pattern (B U a) with support = minimum support of nodes in (B).  Else For each header (ai) in the header of Tree do {  Generate pattern (B = ai U a) with support = ai.support;  Construct B.s conditional pattern base and then B.s conditional FP-tree Tree B  If Tree B = null Then call FP-growth (Tree B, B)  }  End  End |
| Steps |
| 5.1 Create FP Tree by drawing value counts to each other. |

|  |
| --- |
| K-Means Clustering Algorithm |
| * Used for Cluster Analysis * Find k number of clusters * K is user-specified |
| Pseudo-code |
| Select K points as the initial centroid  Repeat  From K clusters by assigning all points to the closest centroid  Re-compute the centroid of each cluster  Until The centroid don’t change |
| Features:   * Initial centroids are chosen at random. * Clusters are specified. eg.“K = 2” “2 clusters” * Sum of Squared Error (SSE) is used to evaluate k-mean clusters. * If SSE is high model is bad else if SSE low model is good |

|  |
| --- |
| Hierarchical Clustering Algorithm |
| * Used for Cluster Analysis * Produce clusters in a hierarchical format * Visualized as a dedrogram * Types: [Agglomerative (bottom-up), Divisive (top-down)] |
| Pseudo-code |
| Compute proximity matrix (Euclidean distance matrix)  Initialize by considering each data point be a cluster  Repeat  Merge two closest clusters  Update proximity matrix  Until desired number of clusters remains |
| Features: Proximity Measures   * Single link (MIN): distance between their two closest members. * Complete link (MAX): distance between their two farthest members. * Centroid = based: distance between their centroids. * Group Average: average distance between all members of the clusters. * Ward’s Method: based on increase in squared error when the clusters are merged. |

|  |
| --- |
| K-Folds Cross-Validation |
| * Cross-Validation is a resampling procedure used to evaluate machine learning models on a limited data sample * Segment data into k disjoint partitions. * During each iteration, one partition is used for testing. * Repeat process k times. Each partition used for testing exactly once. * Overall error estimate is average of error rates for k iterations. |
| Pseudo-code |
| Shuffle the dataset randomly.  Split the dataset into k groups  For each unique group:  Take the group as a hold out or test data set  Take the remaining groups as a training data set  Fit a model on the training set and evaluate it on the test set  Retain the evaluation score and discard the model  Summarize the skill of the model using the sample of model evaluation scores |

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
| * The Confusion matrix shows actual output vs. predicted output for classifier. | | |
| Features | | |
| Accuracy | summary of information | (TP + TN)/(TP + TN + FP + FN) |
| Error Rate | model performance | (FP + FN) / (TP + TN + FP + FN) |
| Sensitivity (TPR) | proportion of positive records correctly identified | TP / (TP + FN) |
| Specificity (TNR) | proportion of negative examples correctly identified | TN / (TN + FP) |
| Precision | percentage of samples labeled as positive are actually in positive class | TP / (TP + FP) |
| Recall | percentage of positive samples correctly identified as positive | TP / (TP + FN) |
| Abbreviations definitions   * T = True * F = False * P = Positive * N = Negative * R = Rate | | |
| Other Calculations   |  |  |  |  | | --- | --- | --- | --- | |  |  | **Predicted** | | |  |  | class = yes | class = no | | **Observed** | class = yes | TP | FN | | class = no | FP | TN | |  |  |  |  | |  | # Obs | TP + TN + FP + FN | | |  | # Obs in Yes class | TP + FN | | |  | # Obs in No class | FP + TN | | |  | # Pred Yes | TP + FP | | |  | # Pred No | TN + FN | | | | |

|  |
| --- |
| ROC Curve and AUC |
| * receiver operating characteristic * shows trade-off between TPR and FPR for classifier * Only work when class has 2 levels * AUC = Area Under Curve * Roc curve is better if the AUC is higher |
| Math Stuff |
| TPR = TP / (TP + FN)  FPR = FP / (FP + TN) |
| Sample ROC curve   * Diagonal line = Random guessing * Below diagonal line = Prediction is opposite of true class     M1 is better than M2 because more AUC. |
|  |

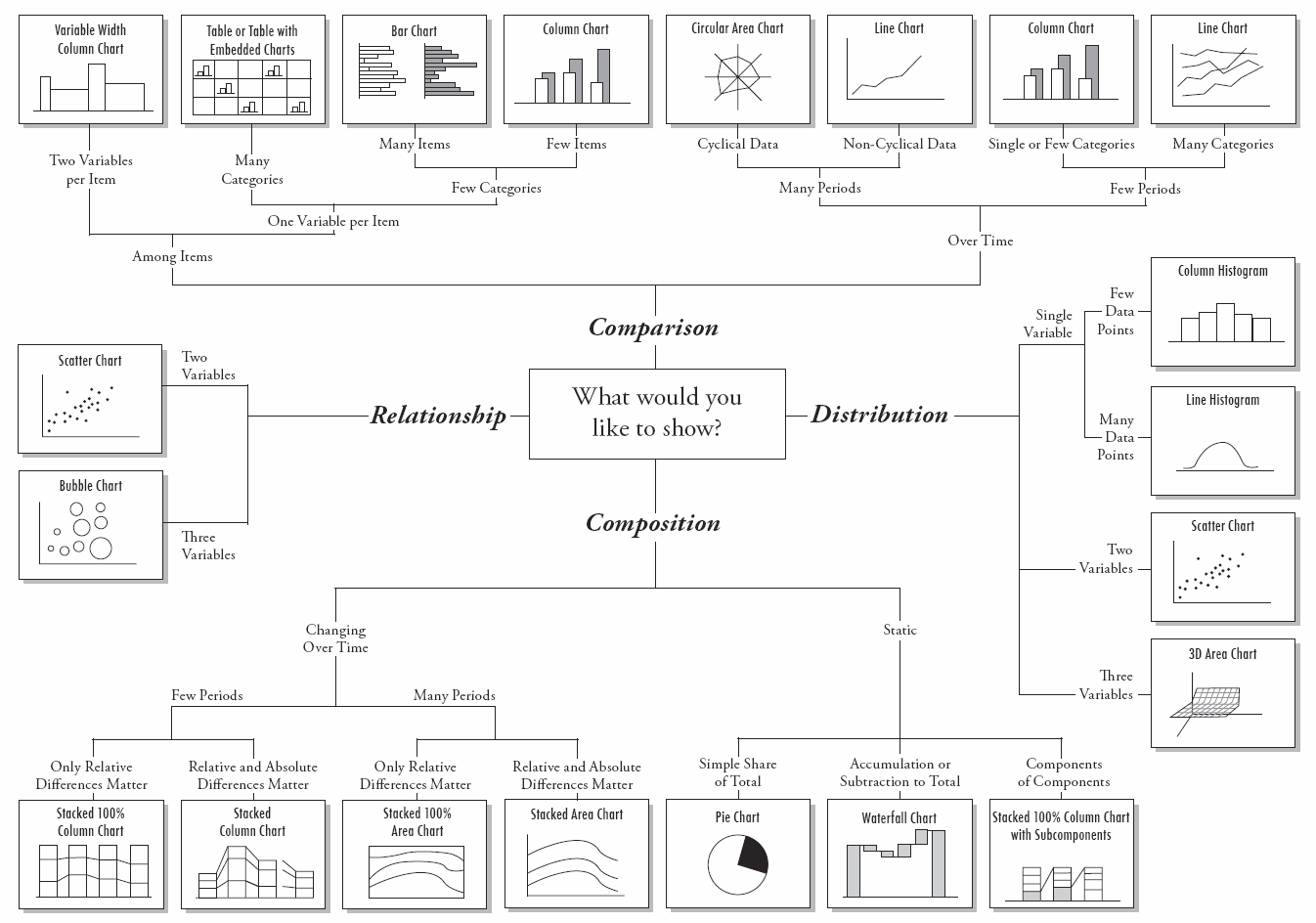
|  |
| --- |
| Ensemble Methods |
| * Improve model performance by combining predictions of several models |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Ensemble Methods** | **Characteristics** | **Final Predictions** | **Pro** | **Con** | | **Randomization** | * Create training data with features (replacement) * Separate classifiers * Classifiers predict label of unknown records * Sample with replacement | Majority vote | reduce correlation errors | difficult to perform | | **Bagging** | * Create training data through instances (replacement) * Separate classifiers * Classifiers predict label of unknown records * Classify data that are sensitive to variation | Majority vote | reduce overfitting | increase bias | | **Boosting** | * Create training data iteratively * Build classifier based on previous misclassified records * Flag important records * Able to classify difficult samples. | Weighted | reduce bias | increase overfitting | |
| Pseudo-code |
| Construct a set of classifiers from training data.  Prediction is made by combining outputs of the multiple classifiers {  Classification: Combine votes of classifiers  Regression: Calculate average of outputs from classifiers  } |

|  |  |
| --- | --- |
| Random Forest | |
| * Ensemble method specifically designed for decision tree classifiers * Built by using a subset attributes to perform splits.   + Randomly determine   + Can use bagging * Final prediction done with majority vote * Reduce correlation error * Improve generalization error | |
| Types | |
| * Forest-RI * Forest-RC * Randomly select best split | |
| Pros and Cons | |
| Pro | Con |
| * Improve Generalization error * Can be parallelized | * Takes longer to train * Difficult to understand |

|  |
| --- |
| Regression |
| How to select the appropriate procedure |

|  |
| --- |
| Linear Regression Model |
| Procedure |

When to use a Chart



Key

|  |  |  |
| --- | --- | --- |
| Term | Definition | Example |
| Data Type | | |
| Quantitative | Measured value or count which are expressed in numbers. | Height, weight, amount |
| Qualitative | Measure of types and represented as symbols. | Race, sex, age group, educational level |
| Numerical | Data that is measurable, and it is represented as numbers. | Height, weight, amount |
| Categorical | Any data that has a meaning assigned to it and is not measureable. | Race, sex, age group, educational level |
| Ordinal | (Ordered) Data where the order of the values are important. | Pain level: 1-Bad, 2-Okay, 3-Great |
| Binary | Data can only take two possible values | Yes or No, True or False, Crazy or Sane |
| Nominal | (Un-ordered) Values used as labels including | Names, id, symbols, alphabets |
| Discrete | Has a finite number of values. | Number of: students, parts in car. |
| Continuous | Has an infinite number of values. (negative to positive) | Temperature, human height, frequency |
| Models | | |
| Training Data Set | The data set used to train the model | |
| Validation Data Set | Used to selected model with best performance while training. | |
| Test Data Set | Used to verify the accuracy of the model | |
| OverFitting | Model does well on the training data set but fail at the test data set | |  |  |  |  | | --- | --- | --- | --- | |  | Testing Data | | | | Training Data |  | Low Error | High Error | | Low Error | Good Fit | Over-fit | | High Error | Not Possible | Under-fit | |
| UnderFitting | Model does badly on training and test data set. |
| Supervised | Has outcome variable. | |
| Unsupervised | No outcome variable. | |
|  |  |  |