Machine Learning with R

Unsupervised Learning

Clustering: task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups

K-Means

- Randomly choose k
- Group observations by minimizing distance to centroid
- Shift centroids for every classified observation until there are no more shift

R BASE Package

```
kmeans(df, k)
arguments:
          -df: dataset used to run the algorithm
          -k: Number of clusters
          -kmeans(df, k)$cluster returns the
           clusters each row of data belongs to
```

```
> kmeans(df,3)$cluster
 [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3
```

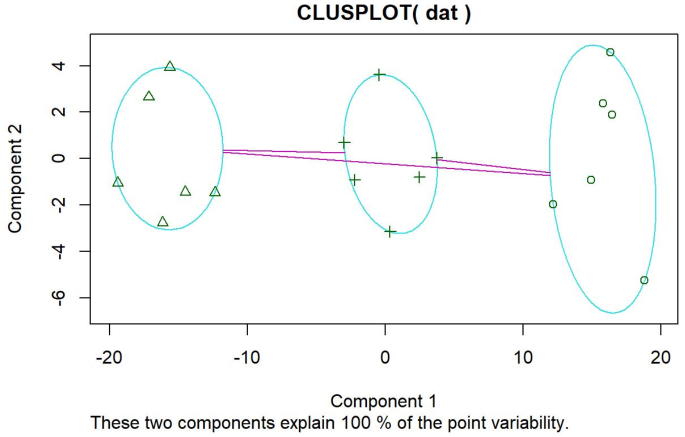
Visualization

cluster Package

- clusplot draws a 2-dimensional plot of clusters using PCA to summarize the variability in two dimensions
- ellipses summarize the covariance within each group

```
clusplot(df,cluster)
          -cluster: the cluster vector; often
           obtained by kmeans(df,k)$cluster
          -create the cluster plot
```

```
> clusplot(df, kmeans(dat,3)$cluster)
```



Ward's Method

- Hierarchical clustering
- Merge pair of clusters at each step based on minimizing the sum of squares error

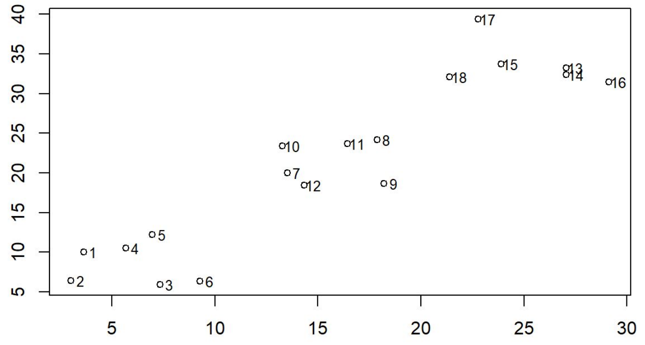
stats Package

```
d_mat=dist(df,method="euclidean")
          -df: dataset used to run the algorithm
          -returns a euclidean distance matrix

tree=hcluster(d_mat,method="ward")
          -returns a tree by fitting Ward's method
           on the previous distance matrix

cutree(tree,k)
          -tree: object returned by hcluster
          -k: number of expected clusters
          -returns the clusters each row of data
           belongs to
```

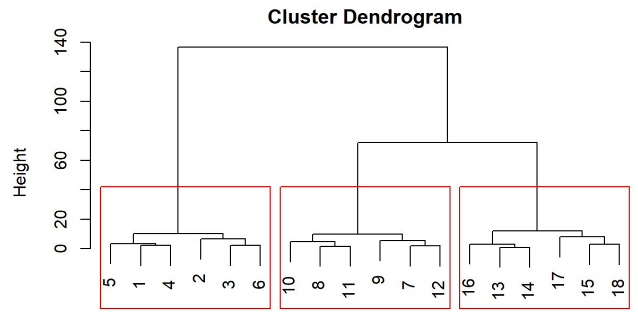
Visualization:

```
plot(tree)
          -plot the cluster dendrogram

rect.hclust(tree,k,border)
          -first 2 arguments are the same as in
           cutree
          -border: border color (e.g. "red")
          -draw the rectangular borders around the
           clusters
```

Suppose our dataset looks like below



```
> cutree(tree,k=3)
 [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3

> plot(tree)
> rect.hclust(tree,k=3,border="red")
```



Supervised Learning

Linear Regression: linear approach for modelling the relationship between a scalar response and one or more explanatory variables

stats Package

```
          lm(formula, data)
          arguments:
          -formula: symbolic description of the
          model to be fit (ex. height~age)
          -data: data points in the form of a
          dataframe
```

Ridge Regression: method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated.

glmnet Package

```
glmnet(x,y,alpha)
arguments:
          -x: input matrix of explanatory variables
          -y: response variable
          -alpha: 1 → Lasso Regression, 0-1 →
          elastic net
```

Random Forests: ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees

randomForest Package

```
randomForest(formula,data,proximity)

          arguments:
          -formula: symbolic description of the
          model to be fit (ex. height~age)
          -data: training data
          -proximity: True/False, used in replacing
          missing data, locating outliers, and
          producing illuminating low-dimensional
          views of the data.
```

Evaluating Random Forest -Confusion Matrix

```
predict(model, data)
          arguments:
          -model: randomForest model object
          -data: test dataset to predict on

confusionMatrix(model_predictions,data)
          -model_predictions: output from predict
           function
          -data: ground truth test dataset
```

```
           setosa versicolor virginica class.error
setosa         35          0         0  0.00000000
versicolor      0         35         1  0.02777778
virginica       0          2        33  0.05714286
```

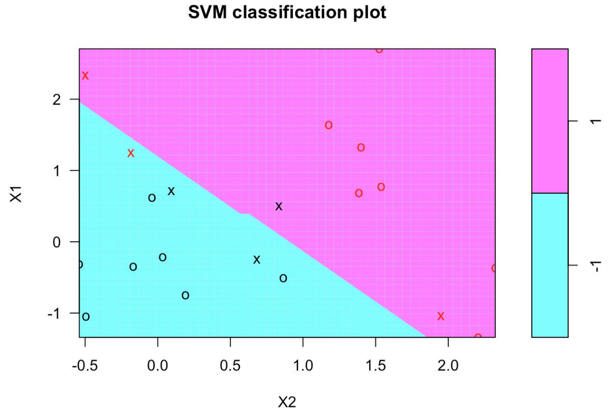
Support Vector Machines: performs supervised learning for classification or regression of data groups

e1071 Package

```
svm(formula, data , kernel, cost, scale)
          -formula: symbolic description of the
           model to be fit
          -data: data frame containing the
           variables in the mode
          -kernel: kernel used in training and
           predicting (linear, polynomial, radial
          basis or sigmoid)
          -cost:  'C'-constant of the regularization
           term in the Lagrange formulation
          -scale: whether to scale the variables
```

Visualizing your Support Vector Machine

```
plot(model, data)
          arguments:
          -model: SVMmodel object
          -data: data frame containing the
          variables in the mode
```



Here you can visualize the support vectors ('x' points) and the the decision boundary. However, there's no control over the colors and breaks with convention since it puts x2 on the horizontal axis and x1 on the vertical axis.

K-Nearest Neighbors: classification algorithm that uses proximity to make classifications or predictions about the grouping of an individual data point

class Package

```
knn(train,test,cl,k)
          arguments:
          -train: data frame of training set cases
          -test: data frame of test set cases
          -cl: factor of true classifications of
           training set
          - k: number of neighbours considered
```

Evaluating KNN-Confusion Matrix

```
table(model,data)
          arguments:
          -model: KNN model object
          -data: test dataset to predict on
```