Comparision of various ML algorithms

Algorith Name	Library	Function	Calculation Hint	Evaluation Measures
Classification				
KNN	from sklearn.neighbors import KNeighborsClassifier	KNeighborsClassifier(n_neighbors = 2)	Euclidean distance	• Accuracy • Kappa Value • Precision • Recall • Sensitivity • Specificity • F-1 Score • Error rate
Decision Tree	from sklearn.tree import DecisionTreeClassifier	DecisionTreeClassifier()	• Entropy • Information Gain	• Accuracy • Kappa Value • Precision • Recall • Sensitivity • Specificity • F-1 Score • Error rate
Random Forest	from sklearn.ensemble import RandomForestClassifier	RandomForestClassifier(n_estimators=100)	-	• Accuracy • Kappa Value • Precision • Recall • Sensitivity • Specificity • F-1 Score • Error rate
Support Vector Machine	from sklearn import svm	svm.SVC(kernel="linear")	-	• Accuracy • Kappa Value • Precision • Recall • Sensitivity • Specificity • F-1 Score • Error rate
Naïve Bayes	from sklearn.naive_bayes import GaussianNB	GaussianNB()	frequency table, cumulative probability, probability normalization	• Accuracy • Kappa Value • Precision • Recall • Sensitivity • Specificity • F-1 Score

				• Error rate
Supervised Learning: Regression				
Linear	from sklearn.linear_model import LinearRegression	LinearRegression()	-	• R Squared • Mean Absolute Error • Mean Squared
Logistic	from sklearn.linear_model import LogisticRegression	LogisticRegression()	-	• R Squared • Mean Absolute Error • Mean Squared • Accuracy • Kappa Value • Precision • Recall • Sensitivity • Specificity • F-1 Score
Unsupervised Learning : Clustering				
K – means	from sklearn.cluster import KMeans	KMeans(n_clusters=n_clusters, init=init, n_init=n_init)	• Euclidean distance between data points and Centroids • Assign clusters based on minimum distance • Re-compute new clusters by taking the	•

			mean of all the points	
K – Medoids	from sklearn_extra.cluster import KMedoids	KMedoids(n_clusters=2)	• Manhattan distance • Compare new and previous cost • If new cost is lesser, swap otherwise undo swap	•
Hierarchical	• from sklearn.cluster import AgglomerativeClustering • from scipy.cluster import hierarchy	• Data scaling • Normalize data • hierarchy.linkage(principalData, method="ward")	• Euclidean distance • Derieve the distance matrix • Group the items based on smallest distance	•
Apriori (Association rule)	from mlxtend.frequent_patterns import apriori, association_rules	• Identify item set (Unique values) • Identify intersection • apriori(o	• Calculate the support count for each item set	•

		he_df, min_support=0.2, use_colnames=True, verbose=1) • association_rules(freq_items, metric="confidence", min_threshold=0.6)	• Combine the item sets as per provided thresh hold value • Calculate the confidence for all possible combination	