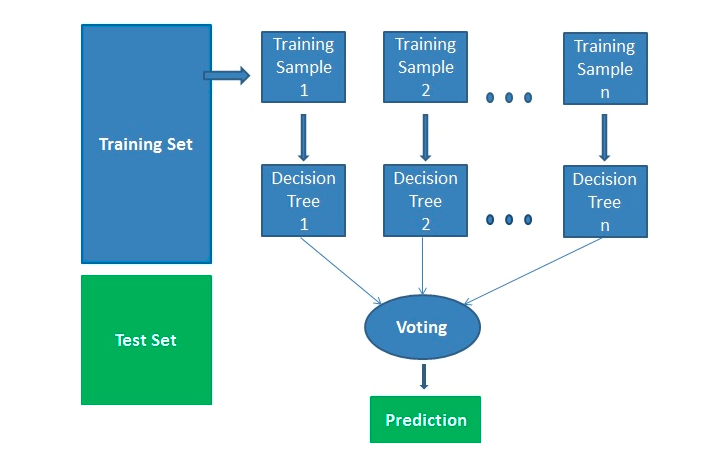
# Random Forest

* Random Forest is a Supervised learning algorithm
* The algorithm can be used in both regression and classification problems
* Random forests creates decision trees on randomly selected data samples, gets prediction from

Each tree and selects the best solution by means of voting.

* The individual decision trees are generated using an attribute selection indicator such as Information Gain, Gain Ratio, and Gini Index for each attribute.
* In Classification problem, each tree votes and the most popular class is chosen as the final result.
* In Regression problem, the average of all the tree output is considered as the final result.



Advantage:

* Random forests is considered as a highly accurate because of the number of decision tress participating in the process.
* It does not suffer from the overfitting problem.
* You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantage:

* Random forest is slow in generating predictions because it has multiple decision trees.
* When it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time consuming.
* Model is difficult to interpret compared to a decision tree.

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (20.0,10.0)

%matplotlib inline

from sklearn.datasets import load\_iris

from sklearn.cross\_validation import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

data = load\_iris()

data.feature\_names

data.target\_names

df=pd.DataFrame({

'sepal length (cm)':data.data[:,0],

'sepal width (cm)':data.data[:,1],

'petal length (cm)':data.data[:,2],

'petal width (cm)':data.data[:,3],

'species':data.target

})

df['species'] = pd.Categorical.from\_codes(data.target, data.target\_names)

df.head()

#X=df.drop("species", axis=1)

X = df[['sepal length (cm)', 'petal length (cm)','petal width (cm)' ]]

Y=df['species']

X\_train,X\_test, Y\_train,Y\_test = train\_test\_split(X,Y,random\_state=1, test\_size=0.3)

model = RandomForestClassifier(n\_estimators=100)

model.fit(X\_train, Y\_train)

Y\_pred = model.predict(X\_test)

print("Accuracy of the model: ", accuracy\_score(Y\_test,Y\_pred)\*100)

#print(model.predict([[3,5,4,2]]))

#feature\_imp = pd.Series(model.feature\_importances\_, index=data.feature\_names).sort\_values(ascending=False)

#feature\_imp

#sns.barplot(y=feature\_imp, x=feature\_imp.index)

#plt.ylabel('Feature Importance Score')

#plt.xlabel('Features')

#plt.title('Important Features')

#plt.show()

#Sepal width is having very very less value of importance, so we can remove this feature in model.