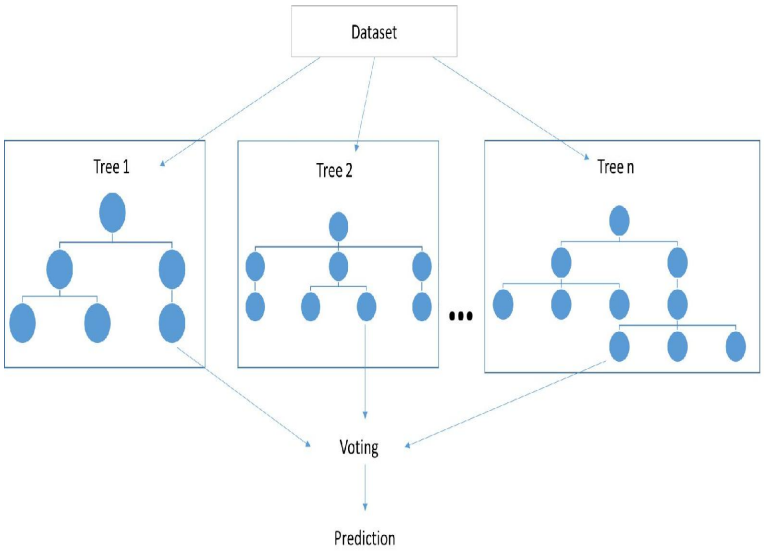
## Random Forests

Random forests are a type of ensemble model. An ensemble model is one which is constructed from other models. This means that it is a combination of several weak learners to form a strong learner. The prediction of an ensemble model may be the average or weighted average of all learners that it is comprised of.

Random forests are an extension of decision trees whereby several decision trees are grown to form a forest. The final prediction of a random forest model is a combination of all component decision trees. For regression it may be a simple average of outputs or a label vote in the case of classification. Though random forest are made of several decision trees, each decision tree is trained on a subset of data that is randomly selected hence the name random forest. The other trick of random forest is that unlike a decision tree where the best attribute is chosen in order to split samples at a decision node from all available attributes, random forest only picks the best attribute from a subset of randomly chosen attributes for each decision node. As a result, each node in a tree is not deterministic, that is for each time we run the algorithm, we are likely to end up with different tree structures. However, the most informative attributes still find their way to trees in the forest and are present across many trees. This makes the results of the random forest algorithm to be less prone to errors due to variations in the input data.

The subset of data on which a decision tree that makes up a random forest is trained on is called bagged data and is usually around 60% of the entire dataset. The remainder on which the performance of individuals trees are tested on is known as the out-of-bag data. Therefore each tree in the forest is trained and evaluated on a different subset of data through the randomization process.




The image above shows a pictorial representation of random forests. It is made up of several trees trained on different instances of the dataset. The attributes in each decision node are also randomized. Finally, the output prediction is an ensemble of the classification of each decision tree.

We would now try out a random forest classifier on the wine dataset and compare its performance on the test set to the decision tree model in the previous section. The beautiful thing about using machine learning models from Scikit-Learn is that the APIs to train and test a model are the same regardless of the algorithm being used. So you would notice that we only need to import the correct classifier, initialize it and all other portions of code would remain unchanged. We are already familiar with how parts of the code works so here is the code for random forest in full.

```
import numpy as np
```

```
import pandas as pd

# load dataset
dataset = pd.read_csv('wine.csv')

# separate features and labels
features = dataset.drop(['Wine'], axis=1)
labels = dataset['Wine']

# split dataset into train and test sets
from sklearn.model_selection import train_test_split
features_train, features_test, labels_train, labels_test = train_test_split(features, labels,
test_size=0.25)

# import random forest classifier from sklearn
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier()

# fit classifier on data
classifier.fit(features_train, labels_train)

# predict classes of test set samples
pred = classifier.predict(features_test)

# evaluate classifier performance using accuracy metric
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(labels_test, pred)
print('Accuracy: {:.2f}'.format(accuracy))
```

```
Accuracy: 0.98
```

We achieve an accuracy of 98% on the test set which is a massive jump from 91% when we used a decision tree classifier. We can see that the randomization approach of random forest enables the algorithm to generalize better hence higher accuracy is recorded on the test set.