Synthetix Machine Learning Technology

Random Forest Algorithm

Introduction

Synthetix includes a Random Forest machine learning technique that is widely used in both academia and industry to solve complex classification and regression problems. It is an ensemble method that combines multiple decision trees to make accurate predictions. Random Forest Algorithm is known for its ability to handle high-dimensional data with a large number of features and has been successfully applied in various fields such as finance, healthcare, and marketing.

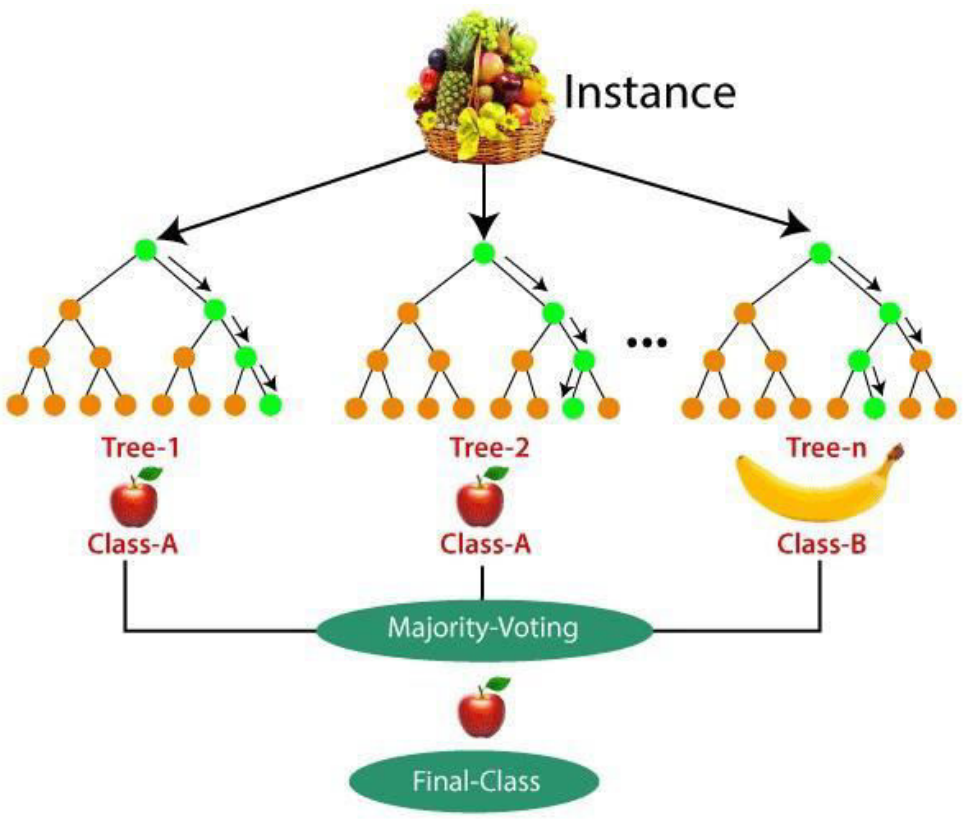
Random Forest Algorithm is a supervised learning technique that belongs to the family of tree-based algorithms. It uses an ensemble of decision trees, each trained on a randomly sampled subset of the training data and a randomly selected subset of features. The algorithm works by constructing a forest of decision trees, where each tree is built using a different subset of features and data points. During the training process, the algorithm selects a random subset of features to split the data at each node, thus reducing the correlation between the trees and improving the accuracy of the model.

The decision trees in the forest are built using a process called recursive partitioning, where the algorithm iteratively splits the data into smaller and smaller subsets based on the values of the input features until a stopping criterion is met. The stopping criterion can be a maximum tree depth, the minimum number of samples per leaf node, or a minimum improvement in the splitting criterion.

The Random Forest Algorithm has several advantages over other machine learning techniques. One of the key advantages is its ability to handle high-dimensional data with a large number of features. The algorithm can select the most important features, reducing the computational cost and improving the accuracy of the model. Moreover, Random Forest Algorithm is less prone to overfitting, which is a common problem in other machine learning algorithms.

Another advantage of the Random Forest Algorithm is its ability to handle missing data and outliers. The algorithm can make accurate predictions even when some data points are missing or when the data contains outliers. This is because the algorithm uses multiple trees, and the final prediction is based on the majority vote of the trees, making the algorithm more robust to outliers and missing data.

Summing up, the Random Forest Algorithm is a powerful machine-learning technique that has been successfully applied in various fields. It is an ensemble method that combines multiple decision trees to make accurate predictions. The algorithm can handle high-dimensional data, missing data, and outliers, making it a robust and reliable machine-learning technique. Its ability to improve the model's accuracy, and reduce the computational cost, makes it a popular choice for solving complex classification and regression problems.



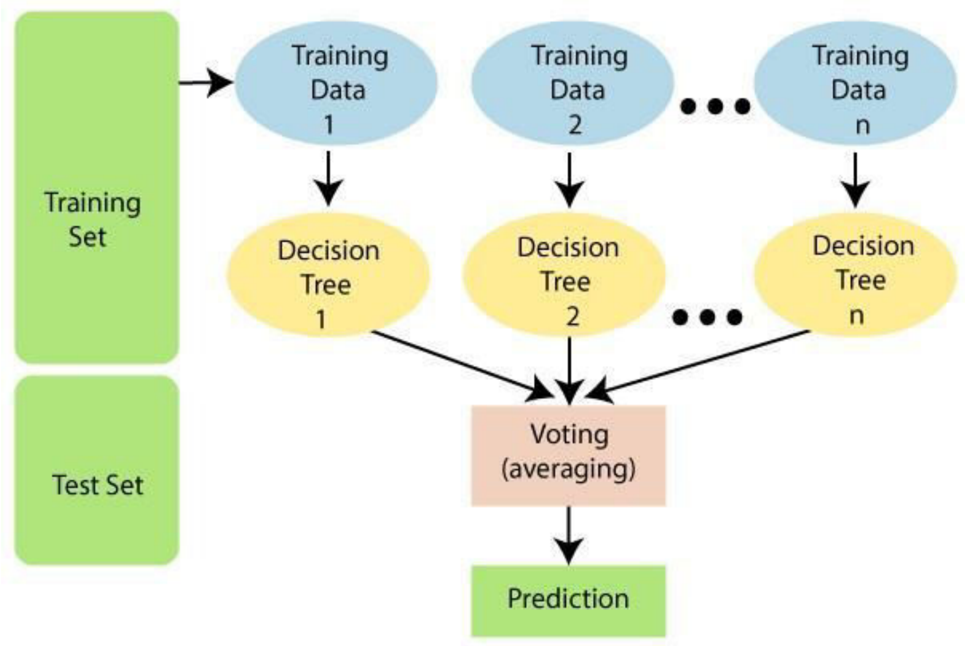
Random Forest Algorithm Example

Image source: JavaPoint.com

The algorithm's Flow

Random Forest algorithm flow builds decision trees using a process called recursive partitioning, where the algorithm iteratively splits the data into smaller and smaller subsets based on the values of the input features until a stopping criterion is met. The Random Forest algorithm requires a dataset with labeled data, which includes both input and features.

The algorithm consists of a random sampling of data and features, constructing decision trees using recursive partitioning, aggregating the predictions of the trees, evaluating the performance of the model, tuning hyperparameters, and making predictions on new data points. This process significantly improves the accuracy and robustness of the model, making it a popular choice for solving complex classification and regression problems.



Random Forest Flow Diagram

Image source: JavaPoint.com

Source Code snippet

C#

This is a demonstration of Synthtix Random Forest algorithm, created using C# source code. Synthetix is using Random Forest algorithm to create a fictitious user's profile and credentials. It creates a "forest" of decision trees, each trained on a randomly sampled subset of the training data (Example: User's name, email, phone number, etc.). In Synthetix we are using a demonstration dataset that contains four input features (Name [First, Last], email, phone #) and a categorical label. We load the dataset into a DataTable object and prepare the data for learning using the ToJagged() and CategoricalEncoder, which are C# functions from the Accord.NET library. We then initialize a Random Forest Classifier with 100 trees and fit it to the data using the Learn() function. To conclude a user's profile, we make predictions on new data using the Decide() function and decode the predicted outputs using the Decode() function. The predicted outputs are a complete, fictitious user profile.

```
// Importing C# required libraries

using System;

using Accord.MachineLearning;

using Accord.MachineLearning.DecisionTrees;

using Accord.MachineLearning.DecisionTrees.Learning;

using Accord.Math.Optimization.Losses;

using Accord.Statistics.Filters;
```

```
// Loading the dataset

var data = new System.Data.DataTable();

data.Columns.Add("FirstName", typeof(string));

data.Columns.Add("LastName", typeof(string));

data.Columns.Add("Email", typeof(string));

data.Columns.Add("Phone", typeof(string));

data.Columns.Add("Label", typeof(string));


data.Rows.Add(“John”, "FirstName");

data.Rows.Add(“Doe”, "LastName");

data.Rows.Add(“syntheticEmail1@synthetix.com, "Email");

data.Rows.Add(“555-555-0001, "Phone");


data.Rows.Add(“Jane”, "FirstName");

data.Rows.Add(“Doe”, "LastName");

data.Rows.Add(“syntheticEmail2@synthetix.com, "Email");

data.Rows.Add(“555-555-0002, "Phone");


// Continuing loading the entire dataset…


// In real case we will load a large synthetic dataset of names, emails, and phone
numbers using For Loop or similar – [ item represents fictitious data like Name,
Email, Phone# ]

// Add data to a DataTable.
```

```
    for(int i = 0; i < 10; i++)
    {
        row = table.NewRow();
        row["id"] = i;
        row["item"] = "item " + i.ToString();
        table.Rows.Add(row);
    }
```

```
// Preparing the data for learning
var features = new int[] { 0, 1 };
var inputs = data.ToJagged<double>(features);
var outputs = data.Columns["Label"].ToStringArray();
var encoder = new CategoricalEncoder(outputs);
outputs = encoder.Encode(outputs);
```

```
// Initializing the Random Forest Classifier
var teacher = new C45Learning();
var forest = new RandomForest<C45Learning>(teacher);
forest.NumberOfTrees = 100;
forest.Learn(inputs, outputs);
```

```
// Making predictions on new data

var newInputs = new string[][] { new string[] { “User1_First”, “FirstName” }, new
string[] { “User2_Last”, “LastName” }, new string[]
{“syntheticEmail5@synthetix.com”, “Email” }, new string[] { “555-555-1000”,
“Phone”} };

var predictedOutputs = forest.Decide(newInputs);


// Decoding the predicted outputs

var decodedOutputs = encoder.Decode(predictedOutputs);


// Displaying the fictitious profile output in the console

for (int i = 0; i < newInputs.Length; i++)

{

   Console.WriteLine("Input: [{0}], Predicted output: {1}", string.Join(", ",
newInputs[i]), decodedOutputs[i]);

}
```

Python

The next is a Python example implementation of Synthetix Random Forest algorithm. We use the Panda DataFrame which is a two-dimensional data structure, like a two-dimensional array, or a table with rows and columns. The dataset contains information about the First Name, Last Name, Email, and Phone #. We split the dataset into training and testing sets using the train_test_split() function from scikit-learn library. We then initialize a Random Forest Classifier with 100 trees and fit it to the training set using the fit() function. Concluding the fictitious profile, we make predictions on the testing set using the predict() function and evaluate the accuracy of the classifier using the accuracy_score() function. The end result is a synthetic profile that contains random data, assembled from the data array provided in the CSV file.

```
# Importing required libraries
import pandas as pd        # Pandas DataFrame
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the dataset – a CSV file that contains: First/Last Name, Email, Phone #
data = pd.read_csv('syntheticDATA.csv')
```

```
# Split the dataset into training and testing sets

X = data.iloc[:, :-1]

y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Initialize the Random Forest Classifier

rfc = RandomForestClassifier(n_estimators=100, random_state=42)


# Train the classifier on the training set

rfc.fit(X_train, y_train)


# Make predictions on the testing set

y_pred = rfc.predict(X_test)


# Evaluate the accuracy of the classifier

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```