# 9-Random_Forest

October 20, 2024

## 1 Random Forest

**Random Forest** is a supervised learning algorithm that is used for both classification and regression tasks. It builds multiple decision trees (hence the term "forest") and combines their outputs to make a final prediction. It is an ensemble method, meaning it aggregates the predictions of many base models (in this case, decision trees) to create a more robust and accurate model.

Each tree in the Random Forest is trained on a different random subset of the data, and the splits within each tree are made using random subsets of features. This randomness helps the model avoid overfitting and improves generalization to unseen data.

- **For classification**, Random Forest uses a majority vote to determine the class label.
- **For regression**, it averages the predictions from all trees.

**When to Use Random Forest?** Random Forest is versatile and can be used in many situations, but it is especially effective when:

- **You need a powerful model that handles both classification and regression problems**.
- **The dataset has many features**: Random Forest can handle high-dimensional datasets and can even provide feature importance, helping identify the most influential variables.
- **The dataset is large**: Unlike a single decision tree, Random Forest can handle large datasets more effectively because it mitigates overfitting by averaging the predictions of many trees.
- **The data has missing values**: Random Forest can handle datasets with some missing values without requiring extensive preprocessing.
- **You need a model that can generalize well to unseen data**: Random Forest is often one of the best choices in terms of accuracy and robustness because it reduces overfitting.

**How Does Random Forest Work?**

1. **Create Multiple Subsets (Bootstrapping)**: Random Forest randomly samples the training data with replacement to create multiple different training subsets (called bootstrap samples). Each decision tree is trained on one of these subsets.
2. **Random Feature Selection (Feature Bagging)**: At each split in a tree, Random Forest only considers a random subset of the features rather than all of them. This ensures that individual trees are diverse and reduces the risk that some features dominate the model.

3. **Build Decision Trees**: Each bootstrap sample is used to build a decision tree. Each tree is grown to its maximum depth (typically unpruned), and the algorithm selects the best split from the subset of features at each node.

4. **Make Predictions**:

    - **For classification**: After training all the trees, Random Forest classifies new data points by making each tree "vote" on the predicted class. The final classification is the class with the most votes.

    - **For regression**: The prediction for each data point is the average of the predictions from all trees.

5. **Aggregate Predictions**: Random Forest aggregates the results of all the trees to make a final prediction. By combining the output of multiple decision trees, Random Forest can achieve better accuracy and stability than any single tree.

**Who Should Use Random Forest?** Random Forest is suitable for:

- **Data scientists and machine learning engineers** who need a strong baseline model.
- **Business analysts and decision-makers** who want to interpret feature importance or detect important decision-making factors.
- **Researchers and practitioners** in fields like finance, healthcare, and bioinformatics, where datasets can be large, complex, and noisy.
- **Beginners in machine learning** who want an easy-to-use and effective algorithm without requiring much fine-tuning.

**Advantages of Random Forest:**

- **Reduces overfitting**: By averaging the predictions of multiple decision trees, Random Forest mitigates overfitting that often occurs with a single decision tree.
- **Handles large datasets well**: Random Forest can scale to large datasets and many features effectively.
- **Works with both classification and regression problems**: It is versatile and can handle different types of machine learning problems.
- **Feature importance**: Random Forest can provide insights into which features are most important for making predictions.
- **Robust to outliers**: Since individual trees are built on random subsets of data, outliers have less impact on the final model.

**Disadvantages of Random Forest:**

- **Slower and more computationally expensive**: Since it trains multiple decision trees, Random Forest can be slower and require more memory compared to simpler models like decision trees or linear models.
- **Less interpretable**: While decision trees are easy to interpret, a Random Forest with hundreds of trees is more difficult to explain and understand.

- **Risk of overfitting in noisy data**: While it generally reduces overfitting, Random Forest can still overfit if there is too much noise in the data or if there are too many trees.

**Real-World Applications:**

- **Credit Scoring and Risk Assessment**: Banks and financial institutions use Random Forest to assess the risk of loan applicants defaulting based on their historical data.
- **Customer Churn Prediction**: Random Forest helps predict which customers are likely to leave a company based on their behavior and interaction history.
- **Fraud Detection**: Random Forest can identify fraudulent transactions by learning patterns from historical financial transaction data.
- **Healthcare and Medical Diagnosis**: Random Forest is used to classify patients based on symptoms, helping doctors make diagnostic decisions.
- **Stock Market Prediction**: Random Forest can be used to predict stock prices based on historical trends and market conditions.

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

df = pd.read_csv('500hits.csv', encoding="latin-1")
df.head()
```

[1]:
```
         PLAYER  YRS     G     AB     R     H   2B   3B   HR   RBI    BB  \
0       Ty Cobb   24  3035  11434  2246  4189  724  295  117   726  1249
1   Stan Musial   22  3026  10972  1949  3630  725  177  475  1951  1599
2  Tris Speaker   22  2789  10195  1882  3514  792  222  117   724  1381
3   Derek Jeter   20  2747  11195  1923  3465  544   66  260  1311  1082
4  Honus Wagner   21  2792  10430  1736  3430  640  252  101     0   963

     SO   SB   CS     BA  HOF
0   357  892  178  0.366    1
1   696   78   31  0.331    1
2   220  432  129  0.345    1
3  1840  358   97  0.310    1
4   327  722   15  0.329    1
```

[2]:
```
df = df.drop(columns=['PLAYER', 'CS'])

X = df.iloc[:, 0:13]
y = df.iloc[:, 13]
```

[3]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=17,␣
↪test_size=0.2)
```

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
y_pred
```

[3]:

```
array([0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0], dtype=int64)
```

[4]:

```
rf.score(X_test, y_test)
```

[4]:

```
0.8279569892473119
```

[5]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87        61
           1       0.79      0.69      0.73        32

    accuracy                           0.83        93
   macro avg       0.82      0.79      0.80        93
weighted avg       0.83      0.83      0.82        93
```

[6]:

```
features = pd.DataFrame(rf.feature_importances_, index=X.columns)
features
```

[6]:

| | 0 |
|---|---|
| YRS | 0.028679 |
| G | 0.082555 |
| AB | 0.081906 |
| R | 0.117477 |
| H | 0.129573 |
| 2B | 0.061198 |
| 3B | 0.047405 |
| HR | 0.055006 |
| RBI | 0.109791 |
| BB | 0.048961 |
| SO | 0.041604 |
| SB | 0.046483 |
| BA | 0.149363 |

[7]:

```
rf2 = RandomForestClassifier(
    n_estimators=1000,
```

```python
    criterion='entropy',
    min_samples_split=10,
    max_depth=14, random_state=42
)

rf2.fit(X_train, y_train)
```

[7]:
```
RandomForestClassifier(criterion='entropy', max_depth=14, min_samples_split=10,
                       n_estimators=1000, random_state=42)
```

[8]:
```python
rf2.score(X_test, y_test)
```

[8]:
```
0.8494623655913979
```

[9]:
```python
y_pred2 = rf2.predict(X_test)
print(classification_report(y_test, y_pred2))
```

```
              precision    recall  f1-score   support

           0       0.85      0.93      0.89        61
           1       0.85      0.69      0.76        32

    accuracy                           0.85        93
   macro avg       0.85      0.81      0.82        93
weighted avg       0.85      0.85      0.85        93
```