Decision Trees and Random Forests

Introduction

Welcome to our module on Decision Trees and Random Forests! Today we're going to learn about two powerful machine learning algorithms that are used to solve many real-world problems. By the end of this module, you'll understand how these algorithms work, when to use them, and why they're so popular.

Building on our previous modules on linear and logistic regression, we'll now explore these **tree-based methods** that can capture more complex relationships in data.

1. What are Decision Trees?

A decision tree is a **flowchart-like model** that makes decisions by following a **series of simple questions**. It's similar to how humans make decisions - we ask a series of yes/no questions until we reach a conclusion.

Imagine you're trying to decide whether to play tennis outside:

- Is it raining? If yes, don't play. If no, continue.
- Is it too hot? If yes, don't play. If no, continue.
- Is it too windy? If yes, don't play. If no, play tennis!

This is exactly how a decision tree works! It breaks down complex decisions into a series of simpler choices.

Parts of a Decision Tree:

- **Root Node:** The starting point that represents the entire dataset
- **Decision Nodes:** Points where the data is split based on feature values
- **Leaf Nodes:** Final outcomes or predictions
- **Branches:** Connections between nodes that represent possible answers

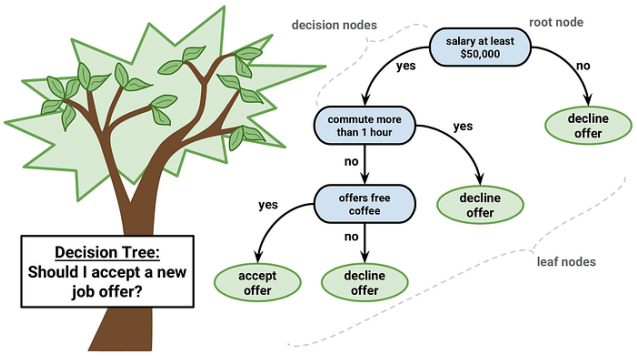

Image Source: https://medium.com/@ambika199820/understanding-decision-trees-df9455f02581

💡 **Interactive Question #1:** In a decision tree, what is a leaf node?

A) The starting point of the tree

B) A point where the data is split into branches

C) A final prediction or outcome where no more splits occur

D) The feature that best separates the data

2. How Decision Trees Work

Let's see how decision trees are built step by step:

2.1 Selecting the Best Feature

The key to creating a good decision tree is **deciding which questions to ask and in what order**. For this, we use concepts like:

- **Gini Impurity:** Measures how **often a randomly chosen element** would be incorrectly labeled
- **Information Gain:** How much a feature helps us **reduce uncertainty**
- **Entropy:** **Measures the disorder or uncertainty** in our data

The algorithm selects the feature that best splits the data at each step.

2.2 Making the Splits

Once we have chosen the best feature, **we split our data and create new branches**. Then we **repeat** the process for each resulting group until we **reach** one of our **stopping conditions**.

2.3 Knowing When to Stop

We can't keep splitting forever! We stop when:

- All samples in a node belong to the same class
- We reach the maximum depth we've set
- The number of samples in a node is less than a minimum value
- The improvement in our splitting measure is below a threshold

💡 **Interactive Question #2:** What metric might be used to decide which feature to split on first in a decision tree?

A) The number of unique values in the feature

B) Information gain or Gini impurity

C) The alphabetical order of feature names

D) The position of the feature in the dataset

AMEYA

3. Types of Decision Trees

Decision trees come in different flavors depending on what we're trying to predict:

3.1 Classification Trees

- Used when we want to **predict categories or classes**
- Examples: Is an email spam or not? Will a customer buy a product?
- Leaf nodes represent the class labels

3.2 Regression Trees

- Used when we want to **predict a numerical value**
- Examples: What will a house sell for? How many units will we sell?
- Leaf nodes represent the average value of the samples in that leaf

💡 **Interactive Question #3:** If you want to predict the price of a car based on its features, which type of decision tree would you use?

A) Classification tree

B) Regression tree

C) Random tree

D) Binary tree

4. Pros and Cons of Decision Trees

4.1 Advantages

- **Easy to understand:** You can explain the logic to non-technical people
- **Minimal data preparation:** No need to normalize or scale features
- **Can handle numerical and categorical data**
- **Can handle missing values**
- **Automatically selects important features**

4.2 Limitations

- **Can create overly complex trees** that don't generalize well (overfitting)
- **Can be unstable:** Small changes in data might result in a very different tree
- **May create biased trees** if some classes dominate
- **Not great for continuous numerical predictions**

AMEYA

5. Pruning Decision Trees

One of the biggest problems with decision trees is that they tend to grow too complex and overfit the training data. Pruning is a technique to **reduce the size of the tree** and improve its ability to generalize to new data.

5.1 What is Pruning?

Pruning means cutting off parts of the tree that provide little value in classification power but add complexity. Think of it like trimming a bush to maintain its shape and health.

5.2 Types of Pruning

There are two main approaches to pruning:

- **Pre-pruning (Early stopping):** Stop growing the tree before it becomes too complex
  - Set limits like maximum depth, minimum samples per leaf, etc.
  - Faster but may stop too early
- **Post-pruning (Reduced Error Pruning):** Grow a full tree, then remove branches that don't improve performance
  - Remove branches and evaluate on a validation set
  - More effective but computationally expensive

💡 **Interactive Question #4:** What is the main purpose of pruning a decision tree?

A) To make the tree grow faster

B) To reduce overfitting and improve generalization

C) To increase the number of leaf nodes

D) To make the tree more complex

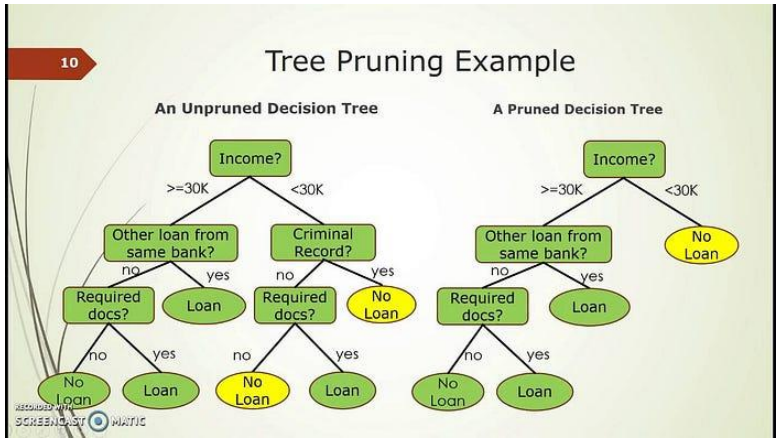

Image Source: https://chirag-sehra.medium.com/decision-trees-explained-easily-28f23241248

6. What are Random Forests?

A random forest is like having a **committee of decision trees that vote on the final prediction**. Instead of relying on just one tree (which might make mistakes), we build **many trees** and **combine their predictions**.

The "random" in random forests comes from two sources of randomness:

1. Each tree is built using a random subset of the data (bootstrap sampling)
2. At each split, only a random subset of features is considered

This randomness helps create diverse trees that make different kinds of mistakes, which **improves the overall accuracy** when we combine them.

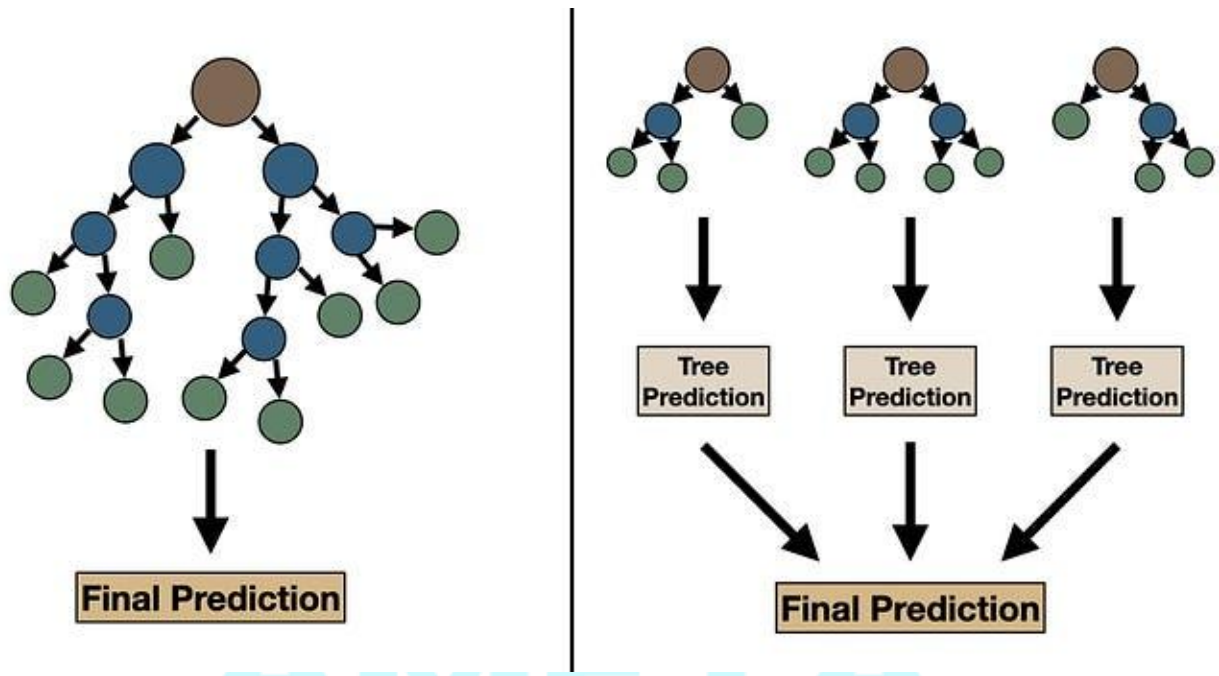

Image Source: https://medium.com/data-science/10-decision-trees-are-better-than-1-719406680564

💡 **Interactive Question #5:** Why do we use random forests instead of a single decision tree?

A) Random forests are easier to interpret

B) Random forests are faster to train

C) Random forests generally give more accurate predictions

D) Random forests require less memory

7. How Random Forests Work

Let's break down the random forest algorithm:

7.1 Bootstrap Sampling

For each tree in our forest:

- **Randomly** select **samples** from our **training data** (with replacement)
- About **2/3 of the original data** is used for **each tree**
- The **remaining 1/3** (called "out-of-bag" samples) can be used to **evaluate the tree**

7.2 Building the Trees

- For **each** bootstrap sample, **grow a decision tree**
- At **each node**, consider only a **random subset of features**
- Typically, we don't prune the trees (let them grow deep)

7.3 Making Predictions

- For **classification**: Each tree **"votes"** for a class, and the **majority wins**
- For **regression**: Take the **average prediction** from all trees

💡 **Interactive Question #6:** What is "bootstrap sampling" in random forests?

A) Testing the model on a small sample before running on the full dataset

B) Randomly selecting samples from the training data with replacement

C) Selecting the most important features for training

D) Removing outliers from the training data

8. Random Forest Parameters

When using random forests, you can adjust several parameters:

- **n_estimators:** The number of trees in the forest
- **max_features:** The number of features to consider at each split
- **max_depth:** How deep each tree can grow
- **min_samples_split:** The minimum number of samples needed to split a node
- **min_samples_leaf:** The minimum number of samples required in a leaf node

AMEYA

9. Implementing Decision Trees and Random Forests

Let's see how to implement these algorithms in Python using scikit-learn:

9.1 Decision Tree Example: Predicting Diabetes

DataSet: *"https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"*

No Copy-Paste, Only Typing

```python
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the Pima Indians Diabetes dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.
column_names = ['pregnancies', 'glucose', 'blood_pressure', 'skin_thickness',
                'insulin', 'bmi', 'diabetes_pedigree', 'age', 'outcome']
diabetes = pd.read_csv(url, names=column_names)

# Prepare features and target
X = diabetes.drop('outcome', axis=1)
y = diabetes['outcome']  # 1 means diabetes, 0 means no diabetes

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and train the model
dt_model = DecisionTreeClassifier(max_depth=3, min_samples_leaf=5)
dt_model.fit(X_train, y_train)

# Make predictions
y_pred = dt_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Decision Tree Accuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Let's see which features were most important
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': dt_model.feature_importances_
}).sort_values('Importance', ascending=False)

print("\nFeature Importance:")
print(feature_importance)
```

No Copy-Paste, Only Typing

When we run this code, we'll see that glucose level, BMI, and age are typically the most important features for predicting diabetes using a decision tree, which aligns with medical knowledge.

9.2 Random Forest Example: Housing Prices

No Copy-Paste, Only Typing

```python
# Import libraries
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load the Boston Housing dataset
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()

# Create a DataFrame for easier handling
housing_df = pd.DataFrame(housing.data, columns=housing.feature_names)
housing_df['PRICE'] = housing.target

# View dataset information
print("Dataset shape:", housing_df.shape)
print("\nFeatures:")
print(housing_df.columns.tolist())
print("\nSample data:")
print(housing_df.head())

# Prepare features and target
X = housing_df.drop('PRICE', axis=1)
y = housing_df['PRICE']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the random forest
rf_model = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)
rf_model.fit(X_train, y_train)
```

```python
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"\nRandom Forest Performance:")
print(f"Mean Squared Error: {mse:.4f}")
print(f"Root Mean Squared Error: {rmse:.4f}")
print(f"R² Score: {r2:.4f}")

# Feature importance
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf_model.feature_importances_
}).sort_values('Importance', ascending=False)

print("\nTop 5 Most Important Features:")
print(feature_importance.head())
```

No Copy-Paste, Only Typing

This example uses the California Housing dataset to predict house prices. Random forests often perform well on this type of regression problem. We'll typically find that location-based features (like median income of the neighborhood) and house characteristics (like average number of rooms) are the most important predictors.

9.3 Visualizing a Decision Tree

No Copy-Paste, Only Typing

```
# For the diabetes dataset example
from sklearn import tree
import matplotlib.pyplot as plt

# Plot the decision tree
plt.figure(figsize=(20, 10))
tree.plot_tree(dt_model,
               feature_names=X.columns,
               class_names=['No Diabetes', 'Diabetes'],
               filled=True,
               rounded=True)
plt.title('Decision Tree for Diabetes Prediction')
plt.savefig('diabetes_decision_tree.png', dpi=300, bbox_inches='tight')
plt.show()

# For a more readable visualization with limited depth
plt.figure(figsize=(15, 10))
tree.plot_tree(dt_model,
               max_depth=2,  # Only show first two levels
               feature_names=X.columns,
               class_names=['No Diabetes', 'Diabetes'],
               filled=True,
               rounded=True,
               fontsize=12)
plt.title('Simplified Decision Tree (Depth=2) for Diabetes Prediction')
plt.show()
```

No Copy-Paste, Only Typing

💡 **Interactive Question #7:** In the scikit-learn code above, what does the parameter max_depth=3 control?

A) The maximum number of features to consider

B) The maximum number of trees in the forest

C) The maximum number of levels in the decision tree

D) The maximum number of samples to use for training

10. Feature Importance

One of the benefits of tree-based methods is that they can tell us which features are most important for making predictions:

No Copy-Paste, Only Typing

```
# Using our housing price random forest model
import pandas as pd
import matplotlib.pyplot as plt

# Get feature importance from the random forest
importances = rf_model.feature_importances_

# Create a DataFrame for better visualization
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
}).sort_values('Importance', ascending=False)

# Print feature importance
print("Feature Importance:")
print(feature_importance)

# Visualize feature importance
plt.figure(figsize=(10, 6))
plt.barh(feature_importance['Feature'][::-1], feature_importance['Importance'][::-1])
plt.xlabel('Importance')
plt.title('Feature Importance in Random Forest Model')
plt.tight_layout()
plt.show()

# For the diabetes dataset with the decision tree model
dt_importance = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': dt_model.feature_importances_
}).sort_values('Importance', ascending=False)

# Visualize decision tree feature importance
plt.figure(figsize=(10, 6))
plt.barh(dt_importance['Feature'][::-1], dt_importance['Importance'][::-1])
plt.xlabel('Importance')
plt.title('Feature Importance in Diabetes Prediction (Decision Tree)')
plt.tight_layout()
plt.show()
```

No Copy-Paste, Only Typing

Feature importance helps us understand which **variables** have the **most predictive power**:

1. For the housing price dataset, we typically find that median income, location, and housing age are the strongest predictors
2. For the diabetes dataset, glucose levels, BMI, and age are usually the most important features

This information can guide feature selection, help us **understand the problem better**, and **provide insights to domain experts**. For example, a healthcare professional would find it valuable to know which factors most strongly predict diabetes risk.

💡 **Interactive Question #8:** What does "feature importance" tell us in a random forest model?

A) Which features take the longest to process

B) Which features have the most unique values

C) How much each feature contributes to the model's predictions

D) Which features were randomly selected the most often

11. Real-World Applications

Decision trees and random forests are used in many fields:

- **Healthcare:** Diagnosing diseases based on symptoms and test results
- **Finance:** Credit scoring and fraud detection
- **E-commerce:** Product recommendations and customer segmentation
- **Environmental science:** Predicting animal habitats and species classification
- **Human resources:** Employee retention prediction

Summary

Decision trees are flowchart-like models that make predictions by asking a series of questions about the features. They're easy to understand but can suffer from overfitting.

Random forests improve on decision trees by combining many trees, each built on random subsets of data and features. This ensemble approach generally gives better performance and reduces overfitting.

Both algorithms are widely used because they're versatile, require minimal data preparation, and can provide insights about feature importance. They work well for both classification and regression problems.

After covering linear and logistic regression in our previous module, we've now added decision trees and random forests to our machine learning toolkit. In the next module, we'll explore clustering algorithms like K-Means and probabilistic approaches like Naive Bayes.

AMEYA

Glossary

- **Bootstrap Sampling:** Randomly selecting samples with replacement from the training data
- **Classification:** Predicting discrete categories or classes
- **Decision Node:** A point in a decision tree where the data is split based on a feature
- **Ensemble Method:** Combining multiple models to improve performance
- **Feature Importance:** A measure of how much each feature contributes to predictions
- **Gini Impurity:** A measure of how often a randomly chosen element would be incorrectly labeled
- **Information Gain:** The reduction in uncertainty achieved by splitting on a feature
- **Leaf Node:** A final node in a decision tree that provides a prediction
- **Out-of-Bag Error:** Error estimate using samples not included in bootstrap samples
- **Overfitting:** When a model performs well on training data but poorly on new data
- **Pruning:** Removing branches of a decision tree to prevent overfitting
- **Regression:** Predicting continuous numerical values
- **Root Node:** The topmost node in a decision tree that represents the entire dataset

Multiple Choice Answers

1. C) A final prediction or outcome where no more splits occur
2. B) Information gain or Gini impurity
3. B) Regression tree
4. B) To reduce overfitting and improve generalization
5. C) Random forests generally give more accurate predictions
6. B) Randomly selecting samples from the training data with replacement
7. C) The maximum number of levels in the decision tree
8. C) How much each feature contributes to the model's predictions

Thank you for completing the Decision Trees and Random Forests module!

AMEYA