PPT by:
Sripooja Mallam

Decision Tree using Gini Impurity

Code walkthrough

Neil Gogte
KMIT

Decision Tree using Gini Impurity

PPT by:
Sripooja Mallam

```python
import numpy as np
import pandas as pd

class DecisionTreeClassifierGini:
    def __init__(self, max_depth=None):
        self.max_depth = max_depth
        self.tree = None

    def gini_impurity(self, y):
        classes, counts = np.unique(y, return_counts=True)
        probabilities = counts / len(y)
        gini_impurity_val = 1 - np.sum(probabilities ** 2)
        return gini_impurity_val
```

Initializes the decision tree classifier with a maximum depth.

Calculates the Gini impurity for a given set of labels y.

Decision Tree using Gini Impurity

PPT by:
Sripooja Mallam

```
class DecisionTreeClassifierGini:
    ...

    def split(self, X_column, threshold):
        left_indices = X_column <= threshold
        right_indices = X_column > threshold
        return left_indices, right_indices
```

Splits a column of data into two subsets based on a threshold value

Decision Tree using Gini Impurity

PPT by:
Sripooja Mallam

```
class DecisionTreeClassifierGini:
    ...

    def calculate_gini_split(self, y, left_indices, right_indices):
        n = len(y)
        n_left = np.sum(left_indices)
        n_right = np.sum(right_indices)

        if n_left == 0 or n_right == 0:
            return float('inf')

        gini_left = self.gini_impurity(y[left_indices])
        gini_right = self.gini_impurity(y[right_indices])
        gini = (n_left / n) * gini_left + (n_right / n) * gini_right
        return gini
```

Computes the Gini impurity after splitting the dataset into two subsets (left and right) based on the threshold.

Decision Tree using Gini Impurity

```
class DecisionTreeClassifierGini:
    ...

    def find_best_split(self, X, y):
        best_gini = float('inf')
        best_feature = None
        best_threshold = None

        for feature_idx in range(X.shape[1]):
            X_column = X[:, feature_idx]
            thresholds = np.unique(X_column)
            for threshold in thresholds:
                left_indices, right_indices = self.split(X_column, threshold)
                gini = self.calculate_gini_split(y, left_indices, right_indices)

                if gini < best_gini:
                    best_gini = gini
                    best_feature = feature_idx
                    best_threshold = threshold

        return best_feature, best_threshold, best_gini
```

Finds the best feature and threshold to split the data that results in the least Gini impurity.

PPT by:
Sripooja Mallam

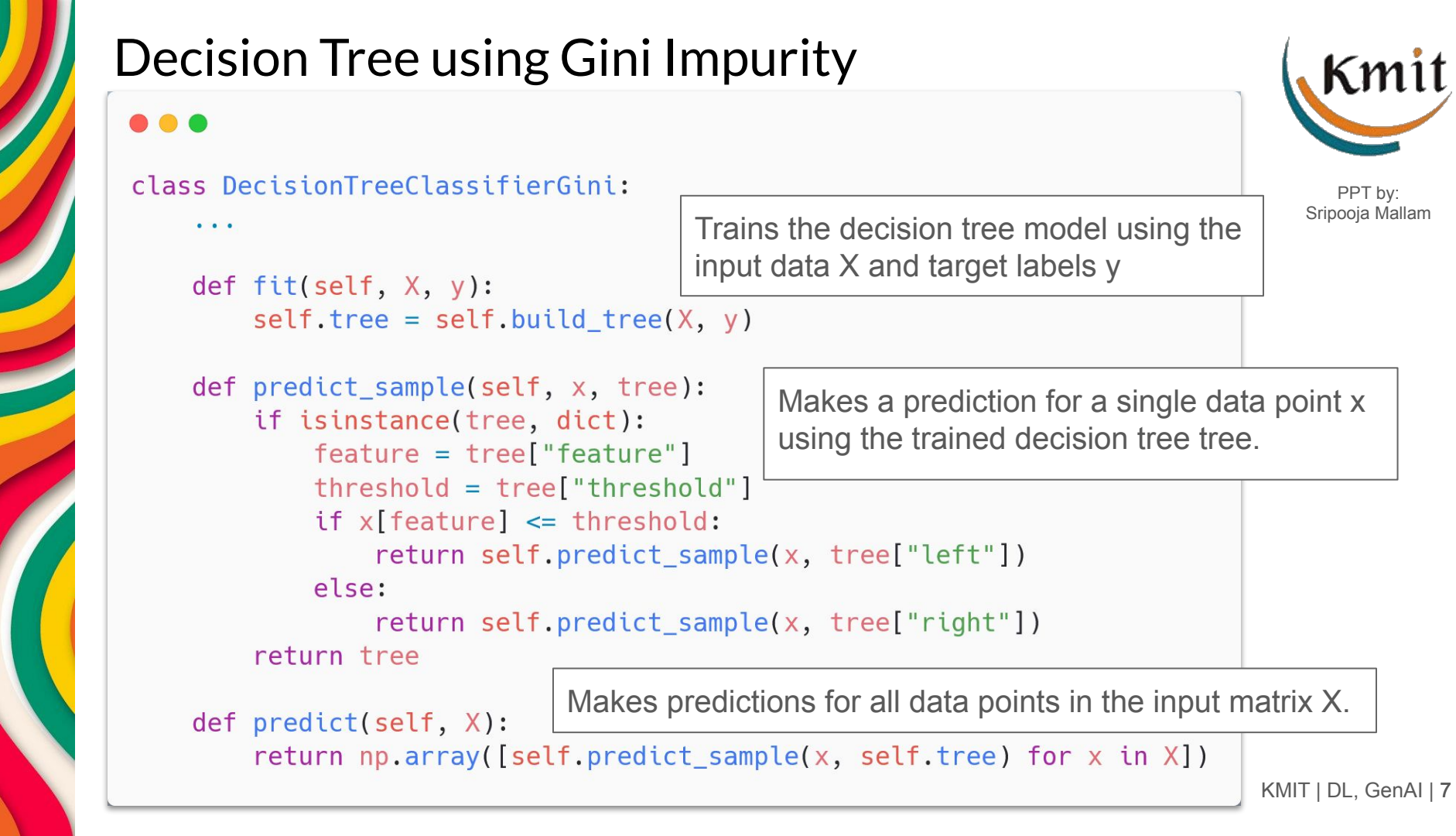
Decision Tree using Gini Impurity

```python
class DecisionTreeClassifierGini:
    ...

    def build_tree(self, X, y, depth=0):
        if len(np.unique(y)) == 1 or (self.max_depth is not None and depth >= self.max_depth):
            return np.argmax(np.bincount(y))

        feature, threshold, gini = self.find_best_split(X, y)
        if feature is None:
            return np.argmax(np.bincount(y))

        left_indices, right_indices = self.split(X[:, feature], threshold)
        left_subtree = self.build_tree(X[left_indices], y[left_indices], depth + 1)
        right_subtree = self.build_tree(X[right_indices], y[right_indices], depth + 1)

        return {
            "feature": feature,
            "threshold": threshold,
            "left": left_subtree,
            "right": right_subtree
        }
```

Recursively builds the decision tree by splitting the data at each node and creating child nodes.

PPT by:
Sripooja Mallam

Decision Tree using Gini Impurity



```
class DecisionTreeClassifierGini:
    ...

    def fit(self, X, y):
        self.tree = self.build_tree(X, y)

    def predict_sample(self, x, tree):
        if isinstance(tree, dict):
            feature = tree["feature"]
            threshold = tree["threshold"]
            if x[feature] <= threshold:
                return self.predict_sample(x, tree["left"])
            else:
                return self.predict_sample(x, tree["right"])
        return tree

    def predict(self, X):
        return np.array([self.predict_sample(x, self.tree) for x in X])
```

Trains the decision tree model using the input data X and target labels y

Makes a prediction for a single data point x using the trained decision tree tree.

Makes predictions for all data points in the input matrix X.

PPT by:
Sripooja Mallam

Decision Tree using Gini Impurity

PPT by:
Sripooja Mallam

```python
# Function to take user input and make predictions
def main():
    # Accept user input for new data
    print("Provide new data (outlook, temperature, humidity, windy) below:")
    new_data = []

    # Collect input data (4 features)
    outlook = input("Outlook (sunny/overcast/rainy): ").strip().lower()
    temperature = input("Temperature (hot/cool/mild): ").strip().lower()
    humidity = input("Humidity (high/normal): ").strip().lower()
    windy = input("Windy (yes/no): ").strip().lower()

    # Encode categorical features
    outlook_map = {'sunny': 0, 'overcast': 1, 'rainy': 2}
    temp_map = {'hot': 0, 'cool': 1, 'mild': 2}
    humidity_map = {'high': 0, 'normal': 1}
    windy_map = {'yes': 0, 'no': 1}
```

The main function that handles user input, encodes categorical values, and uses the decision tree to make predictions.

Decision Tree using Gini Impurity

```
# Function to take user input and make predictions
def main():
    ...
    new_data = [
        [outlook_map[outlook], temp_map[temperature], humidity_map[humidity], windy_map[windy]]
    ]

    new_data = np.array(new_data)

    # Example dataset for training
    data = {
        "outlook": ['sunny', 'overcast', 'rainy', 'sunny', 'sunny', 'overcast', 'rainy',
'overcast', 'sunny', 'rainy'],
        "temperature": ['hot', 'hot', 'mild', 'cool', 'mild', 'cool', 'hot', 'cool', 'mild',
'mild'],
        "humidity": ['high', 'high', 'normal', 'high', 'normal', 'normal', 'high', 'normal',
'high', 'normal'],
        "windy": ['yes', 'yes', 'no', 'no', 'yes', 'no', 'yes', 'yes', 'no', 'yes'],
        "play": ['no', 'yes', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'no', 'yes']
    }

    # Encode categorical data
    outlook_map_train = {'sunny': 0, 'overcast': 1, 'rainy': 2}
    temp_map_train = {'hot': 0, 'cool': 1, 'mild': 2}
    humidity_map_train = {'high': 0, 'normal': 1}
    windy_map_train = {'yes': 0, 'no': 1}
```

PPT by:
Sripooja Mallam

Decision Tree using Gini Impurity

```
# Function to take user input and make predictions
def main():
    ...
    df = pd.DataFrame(data)
    df["outlook"] = df["outlook"].map(outlook_map_train)
    df["temperature"] = df["temperature"].map(temp_map_train)
    df["humidity"] = df["humidity"].map(humidity_map_train)
    df["windy"] = df["windy"].map(windy_map_train)
    df["play"] = df["play"].map({'no': 0, 'yes': 1})

    X = df[["outlook", "temperature", "humidity", "windy"]].values
    y = df["play"].values

    # Train decision tree classifier
    clf = DecisionTreeClassifierGini(max_depth=5)
    clf.fit(X, y)

    # Predict on user input data
    predictions = clf.predict(new_data)
    prediction = predictions[0]

    # Decode prediction (1 -> 'yes', 0 -> 'no')
    return 'yes' if prediction == 1 else 'no'

output = main()
print(f"Prediction: {output}")
```

PPT by:
Sripooja Mallam

For example, if the input is:

Outlook (sunny/overcast/rainy): sunny
Temperature (hot/cool/mild): hot
Humidity (high/normal): high
Windy (yes/no): yes

The expected output would be:

Prediction: yes