Node

Decison nodes

feature index

threshold

left (children) nodes

right (children) nodes

variance reduction

Leaf nodes

predicting value

is used in building a tree →

Decision Tree

root: object (initialized as None)

min_samples_split: int (initialize min_samples_split = 2)

max_depth: int (initialize max_depth = 2)

build_tree (dataset, current_depth)

get_best_split (dataset, num_samples, num_features)

split (dataset, feature_index, threshold)

calculate_leaf_value (y_train)

variance_reduction (parent, left_child, right_child)

fit (X_train, Y_train)

make_prediction (x_test, tree)

predict (X_test)

```
get_best_split
```

1. dataset
2. num features

for each feature index in the dataset

Find all possible values of a feature in the dataset to be used as the threshold value in splitting data

for each threshold value of a feature

- dataset
- feature index
- threshold

split dataset using the threshold

- dataset_left
- dataset_right

compute variance reduction (var) using the dataset_left and right

var: variance reduction

Is the maximum variance reduction ?

True   False

store
best_split with highest variance reduction

best_split: dict
1. best_split ["feature_index"]: int; feature index.
2. best_split ["threshold"]: float; thresholding value associated with the best splitting.
3. best_split ["dataset_left"]: np.array; an array of data samples in the left child node.
4. best_split ["dataset_right"]: np.array; an array of data samples in the right child node
5. best_split ["var_red"]: float; variance reduction associated with the best splitting

check all posible value?

True   False

store
best_split for each feature

check all features?

True   False

outputs: best_spliting

End

```
                          ┌─────────────┐
                          │    Split    │
                          └─────────────┘
                                 │
                                 ▼
                     ╱───────────────────────╲
                    │  1. Feature index        │
                    │  2. Dataset              │
                    │  3. Threshold            │
                     ╲───────────────────────╱
                                 │
                                 ▼
                    ┌───────────────────────────┐
          ┌────────▶│  For all data samples in   │◀──────────┐
          │         │  dataset at feature index  │           │
          │         └───────────────────────────┘           │
          │                      │                           │
          │                      ▼                           │
          │         ╱───────────────────────╲                │
       True        │  If a data sample <= Threshold  │  False │
          │         ╲───────────────────────╱                │
          │                                  │                │
          ▼                                  ▼                │
  ────────────────              ────────────────              │
  store the data sample         store the data sample         │
  in left child node            in right child node           │
  ────────────────              ────────────────              │
          │                                  │                │
          └──────────────┬───────────────────┘                │
                         ▼                                    │
              ╱───────────────────────╲                       │
      True   │      check all           │   False ────────────┘
             │      data samples ?      │
              ╲───────────────────────╱
          │
          ▼
  ┌──────────────────────────────────────────────────┐
  │  Outputs:                                          │
  │   1. dataset_left: np.array; an array of data samples │
  │                    in the right child node         │
  │   2. dataset_right: np.array; an array of data samples │
  │                    in the left child node          │
  └──────────────────────────────────────────────────┘
                         │
                         ▼
                  ┌─────────────┐
                  │     End     │
                  └─────────────┘
```

```
calculate_leaf_value
        │
        ▼
    y_train
        │
        ▼
  mean(y_train)
        │
        ▼
Outputs:
val: float; mean of y_train
        │
        ▼
      End
```

```
variance_reduction
```

1. left_child
2. right_child
3. parent

```
weight_l = len(left_child) / len(parents)
weight_r = len(right_child) / len(parents)
```

```
reduction = np.var(parent) - (weight_l * np.var(left_child) + weight_r np.var(right_child) )
```

Output: reduction: float; variance reduction

```
End
```

```
fit
```

```
1. X_train
2. Y_train
```

```
dataset = np.concatenate((X_train,
        Y_train), axis=1)
```

```
self.root = build_tree(dataset)
```

Decision Tree

root: class node

min_samples_split: int (initialize min_samples_split = 2)

max_depth: int (initialize max_depth = 2)

```
End
```

```
make_prediction
```

1. x
2. root

Is root node
a leaf node ?

True — Output:
prediction: root.value

False

feature_val:= feature value
of x

feature_val <=
root.threshold

True

Predict x using
left child node (root.left).
Thus, set left child node as
new root.

False

Predict x using
right child node (root.right).
Thus, set right child node as
new root.

Output:
1. x
2. new root

End

```
                              ┌─────────────────┐
                              │     predict     │
                              └─────────────────┘
                                       │
                                       ▼
                              ┌─────────────────┐
                              │     X_test      │◄───────────┐
                              └─────────────────┘            │
                                       │                     │
                                       ▼                     │
┌──────────────────────┐      ╭─────────────────╮           │
│    Decision Tree     │      │ For each data    │           │
├──────────────────────┤      │ sample x in      │           │
│ root: class node     │      │ X_test           │           │
│                      │      ╰─────────────────╯           │
│ min_samples_split:int│              │                     │
│                      │              ▼                     │
│ max_depth: int       │      ┌─────────────────┐           │
│                      │─────►│ prediction =     │           │
└──────────────────────┘      │ make_prediction( │           │
                              │ x, self. root)   │           │
   self.root: Trained         └─────────────────┘           │
   decision tree                      │                     │
                                       ▼                     │
                              Collecting prediction of       │
                              each testing sample into       │
                              a list                         │
                                       │                     │
                                       ▼                     │
                                    ◆ Is predict all ◆       │
                            True  ◄ data samples ? ►  False──┘
                                       │
                                       ▼
                              ╱─────────────────╲
                              │ Output:          │
                              │ predictions: list│
                              │ of testing       │
                              │ sample's         │
                              │ prediction       │
                              ╲─────────────────╱
                                       │
                                       ▼
                              ┌─────────────────┐
                              │       End       │
                              └─────────────────┘
```