DECISION TREE REGRESSOR FROM SCRATCH

```python
import numpy as np
class Node:
    def __init__(self,features=None,threshold=None,left=None,right=None,*,value=None):
        self.features=features
        self.threshold=threshold
        self.left=left
        self.right=right
        self.value=value
    def is_leaf_node(self):
        return self.value is not None
```

```python
class DecisionTreeRegressor:
    def __init__(self,max_depth=100,min_sample_split=2,n_features=None) -> None:
        self.root=None
        self.max_depth=max_depth
        self.min_sample_split=min_sample_split
        self.n_features=n_features
        self.d=-1

    def fit(self,X,y):
        self.n_features=X.shape[1] if not self.n_features else min(self.n_features,X.shape[1])
        self.root=self._Tree(X,y)

    def _split(self,X_column,threshold):
        left_idxs=np.where(X_column<=threshold)[0]
        right_idxs=np.where(X_column>threshold)[0]
        return left_idxs,right_idxs
```

```
    def _Tree(self,X,y,depth=0):
        samples,features=X.shape
        if (depth>=self.max_depth or samples<self.min_sample_split):
            return Node(value=np.mean(y))
        feat_idxs=np.random.choice(self.n_features,features)
        best_feat,best_threshold=self._best_split(X,y,feat_idxs)

        if best_threshold!=None:
            left_idxs,right_idxs=self._split(X[:,best_feat],best_threshold)

            left=self._Tree(X[left_idxs],y[left_idxs],depth+1)
            right=self._Tree(X[right_idxs],y[right_idxs],depth+1)
            if depth>self.d:
                self.d=depth
            return Node(best_feat,best_threshold,left,right)
```

```
    def _best_split(self,X,y,feat_idxs):
        best_var=-1
        split_idx,split_thres=None,None
        for feature in feat_idxs:
            thresholds=(np.unique(X[:,feature]))

            for threshold in thresholds:
                if threshold!=None:
                    var=self._variance_reduction(X[:,feature],y,threshold)

                    if var >best_var:
                        best_var=var
                        split_idx,split_thres=feature,threshold
    return split_idx,split_thres
```

```
    def _variance_reduction(self,X,y,threshold):
        parent_var=np.var(y)
        left_idxs,right_idxs=self._split(X,threshold)
        if len(left_idxs)==0 and len(right_idxs)==0:
            return 0
        n=len(y)
        n_l,n_r=len(y[left_idxs]),len(y[right_idxs])
        left_var,right_var=np.var(y[left_idxs]),np.var(y[right_idxs])
        child_var=((n_l/n)*left_var)+((n_r/n)*right_var)
        return parent_var-child_var
```

```python
from sklearn.datasets import load_boston
dataset=load_boston()
X=dataset['data']
Y=dataset['target']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.1,random_state=1234)
```

```python
from DecisionTreeRegressor import DecisionTreeRegressor
DR=DecisionTreeRegressor(max_depth=15)
DR.fit(X_train,y_train)
y_pred=DR.predict(X_test)
from sklearn.metrics import r2_score
print(r2_score(y_test,y_pred))

#output:
0.8311416
```

Click for Github

LINK