```
%run decision_tree.ipynb


# hàm lấy các mẫu dữ liệu ngẫu nhiên trong đó các phần tử có thể lặp lại (trùng nhau)
def bootstrap(X, y): # X là frame, y là series
    n_sample = X.shape[0]
    _id = np.random.choice(n_sample, n_sample, replace = True) # dạng mảng
    return X.iloc[_id], y.iloc[_id] # phải hiểu tại sao iloc cho cả X và y?
    # liên quan đến chỉ số X_train khi dùng train_test_split


# lớp RandomForest
class RandomForest:
    def __init__(self, n_trees = 5, max_depth = 10, min_samples_split = 2, n_features = None):
        self.n_trees =  # số cây để để đưa ra quyết định cho giá trị dự đoán
        self.max_depth =
        self.min_samples_split =
        self.n_features =
        self.trees = []

    def fit(self, X, y):  # X là frame, y là series
        self.trees = [] # tạo list chứa số cây cho dự đoán
        for i in range(self.n_trees):
            # với mỗi giá trị i ta tạo một cây quyết định
            tree = DecisionTreeClass(min_samples_split = self.min_samples_split, max_depth = self.max_depth, n_features = self.n_features)
            X_sample, y_sample = bootstrap(X, y) # tạo mẫu X và y thay đổi qua mỗi lần lặp
            tree.fit(X_sample, y_sample) # tạo cây
            self.trees.append(tree) # thêm cây vào list cây

    def predict(self, X):  # X là frame
        # lấy dự đoán từ từng cây
        arr_pred = np.array([tree.predict(X) for tree in self.trees])
        final_pred = []
        for i in range(arr_pred.shape[1]):
            sample_pred = arr_pred[:, i] # trả loại mảng
            final_pred.append(most_value(pd.Series(sample_pred))) # tham số trong hàm most_value phải ở dạng series
        return np.array(final_pred)  # trả về giá trị dự đoán sau khi vote n cây


Start coding or generate with AI.


Start coding or generate with AI.
```