```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> from matplotlib.colors import ListedColormap
>>> from sklearn import datasets
>>> classification_data, classification_labels = datasets.make_classification(n_features=2,
n_informative=2, n_classes=2, n_redundant=0, n_clusters_per_class=1, random_state=5)
>>> colors = ListedColormap(['red', 'blue'])
>>> light_colors = ListedColormap(['lightcoral', 'lightblue'])
>>> plt.figure(figsize=(8,8))
<Figure size 800x800 with 0 Axes>
>>> plt.scatter(list(map(lambda x: x[0], classification_data)), list(map(lambda x: x[1],
classification_data)),
...             c = classification_labels, cmap=colors)
<matplotlib.collections.PathCollection object at 0x7fb6cee85d90>
>>> plt.show()
>>>
```

```
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> import numpy as np
>>> import pandas as pd
>>> inport scipy
>>> inport sklearn
>>> class Node:
...     def __init__(self, index, t, true_branch, false_branch):
...         self.index = index
...         self.t = t
...         self.true_branch = true_branch
...         self.false_branch = false_branch
...
>>> class Leaf:
...     def __init__(self, data, labels):
...         self.data = data
...         self.labels = labels
...         self.prediction = self.predict()
...
>>> def predict(self):
...     classes = {}
...     for label in self.labels:
...         if label not in classes:
...             classes[label] = 0
...             classes[label] += 1
...     prediction = max(classes, key=classes.get)
...     return prediction
...
>>> def build_tree(data, labels):
...     quality, t, index = find_best_split(data, labels)
...     if quality == 0:
...         return Leaf(data, labels)
...     true_data, false_data, true_labels, false_labels = split(data, labels, index, t)
...     true_branch = build_tree(true_data, true_labels)
...     false_branch = build_tree(false_data, false_labels)
...     return Node(index, t, true_branch, false_branch)
>>> def classify_object(obj, node):
...     if isinstance(node, Leaf):
...         answer = node.prediction
...         return answer
...     if obj[node.index] <= node.t:
...         return classify_object(obj, node.true_branch)
...     else:
...         return classify_object(obj, node.false_branch)
...
>>> def predict(data, tree):
...     classes = []
...     for obj in data:
...         prediction = classify_object(obj, tree)
...         classes.append(prediction)
...     return classes
...
>>>
```