Machine Learning Lab – 20ISL68A

Program 4 - Develop a program to demonstrate the working of the decision tree based CART algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Step 1: Importing Libraries

```
from chefboost import Chefboost as cb
```

Step 2: Loading the Dataset

```
import pandas as pd
data = pd.read_csv(r"C:\Users\kvsuv\OneDrive\Desktop\dataset4.csv")
```

Step 3: Understanding the Dataset

```
data.head()
```

	outlook	temperature	humidity	wind	Decision
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rain	mild	high	weak	yes
4	rain	cool	normal	weak	yes

Step 4: Passing the Data Frame

```
config = {"algorithm": "CART"}
tree = cb.fit(data, config)
```

```
[INFO]:  6 CPU cores will be allocated in parallel running
CART  tree is going to be built...
-------------------------
finished in  1.4239444732666016  seconds
-------------------------
Evaluate  train set
-------------------------
Accuracy:  100.0 % on  14  instances
Labels:  ['no ' 'yes ' 'no']
Confusion matrix:  [[4, 0, 0], [0, 9, 0], [0, 0, 1]]
Decision  no   => Accuray:  100.0 %, Precision:  100.0 %, Recall:  100.0 %, F1:  100.0 %
Decision  yes   => Accuray:  100.0 %, Precision:  100.0 %, Recall:  100.0 %, F1:  100.0 %
Decision  no  => Accuray:  100.0 %, Precision:  100.0 %, Recall:  100.0 %, F1:  100.0 %
```

Step 5: Prediction using instance – Passing Index

```
test_instance = data.iloc[2]
test_instance
```

```
outlook        overcast
temperature         hot
humidity           high
wind               weak
Decision            yes
Name: 2, dtype: object
```

Step 6: Prediction using instance

```
cb.predict(tree,test_instance)
```

```
'yes '
```

Step 7: Prediction using data values

```
moduleName = "outputs/rules/rules"
tree = cb.restoreTree(moduleName)
prediction = tree.findDecision(['sunny', 'hot', 'high', 'weak'])
prediction
```

```
'no '
```

Step 8: Decision Rule

```
df = cb.feature_importance("outputs/rules/rules.py")
df
```

```
Decision rule:  outputs/rules/rules.py
```

	feature	importance
0	outlook	0.8077
3	wind	0.1923
1	temperature	0.0000
2	humidity	0.0000

Built Decision Tree

```
def findDecision(obj): #obj[0]: outlook, obj[1]: temperature, obj[2]: humidity, obj[3]: wind
    # {"feature": "outlook", "instances": 14, "metric_value": 0.3714, "depth": 1}
    if obj[0] == 'sunny':
        # {"feature": "humidity", "instances": 5, "metric_value": 0.0, "depth": 2}
        if obj[2] == 'high':
            return 'no '
        elif obj[2] == 'normal':
            return 'yes '
        else: return 'yes '
    elif obj[0] == 'rain':
        # {"feature": "wind", "instances": 5, "metric_value": 0.2, "depth": 2}
        if obj[3] == 'weak':
            return 'yes '
        elif obj[3] == 'strong':
            # {"feature": "temperature", "instances": 2, "metric_value": 0.0, "depth": 3}
            if obj[1] == 'cool':
                return 'no '
            elif obj[1] == 'mild':
                return 'no'
            else: return 'no'
        else: return 'no '
    elif obj[0] == 'overcast':
        return 'yes '
    else: return 'yes '
```