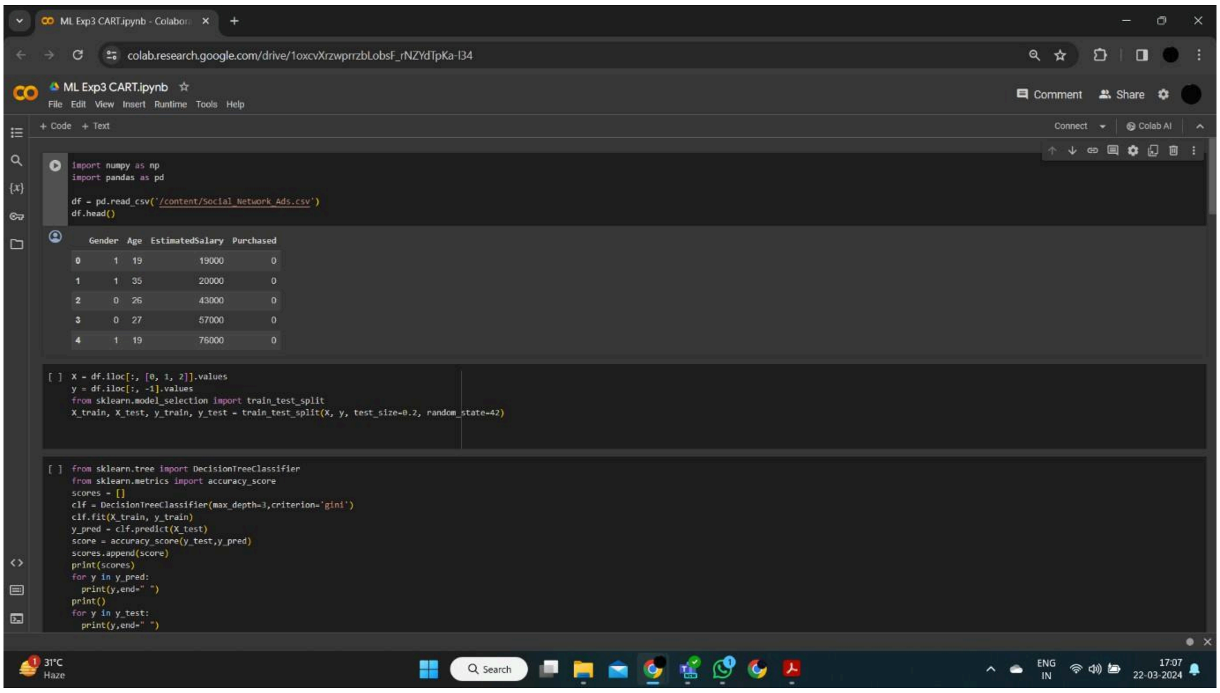
ML Experiment 3

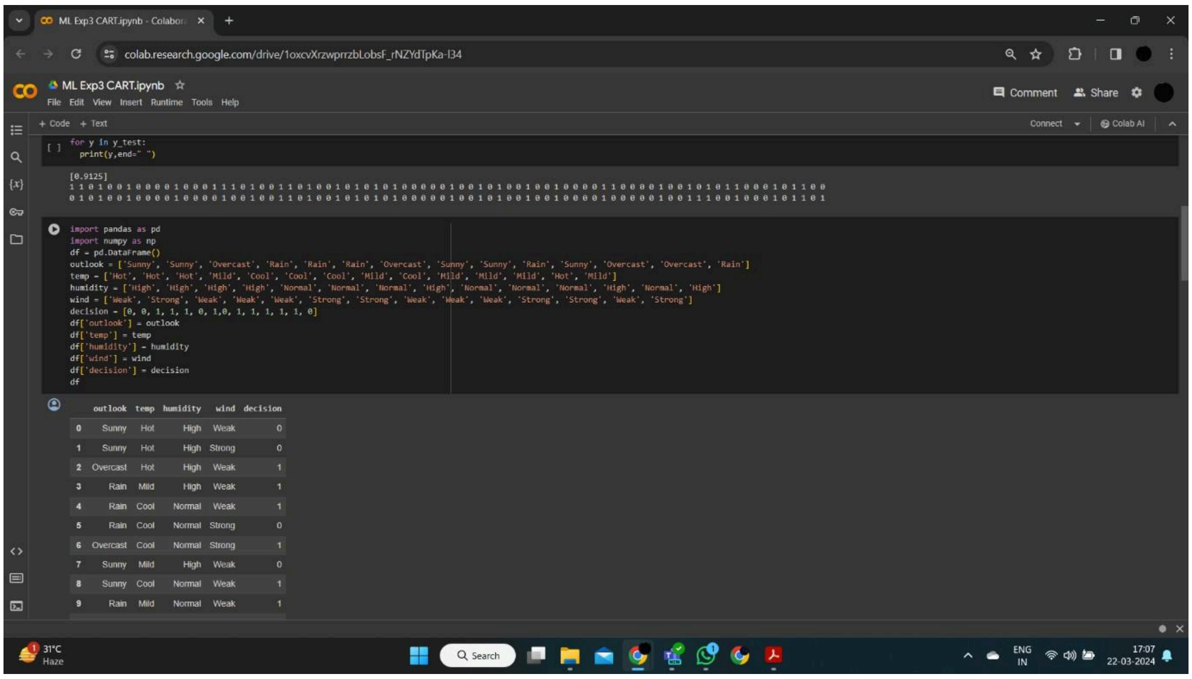


```python
import numpy as np
import pandas as pd

df = pd.read_csv('/content/Social_Network_Ads.csv')
df.head()
```

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0

```python
X = df.iloc[:, [0, 1, 2]].values
y = df.iloc[:, -1].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
scores = []
clf = DecisionTreeClassifier(max_depth=3,criterion='gini')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
score = accuracy_score(y_test,y_pred)
scores.append(score)
print(scores)
for y in y_pred:
  print(y,end=" ")
print()
for y in y_test:
  print(y,end=" ")
```

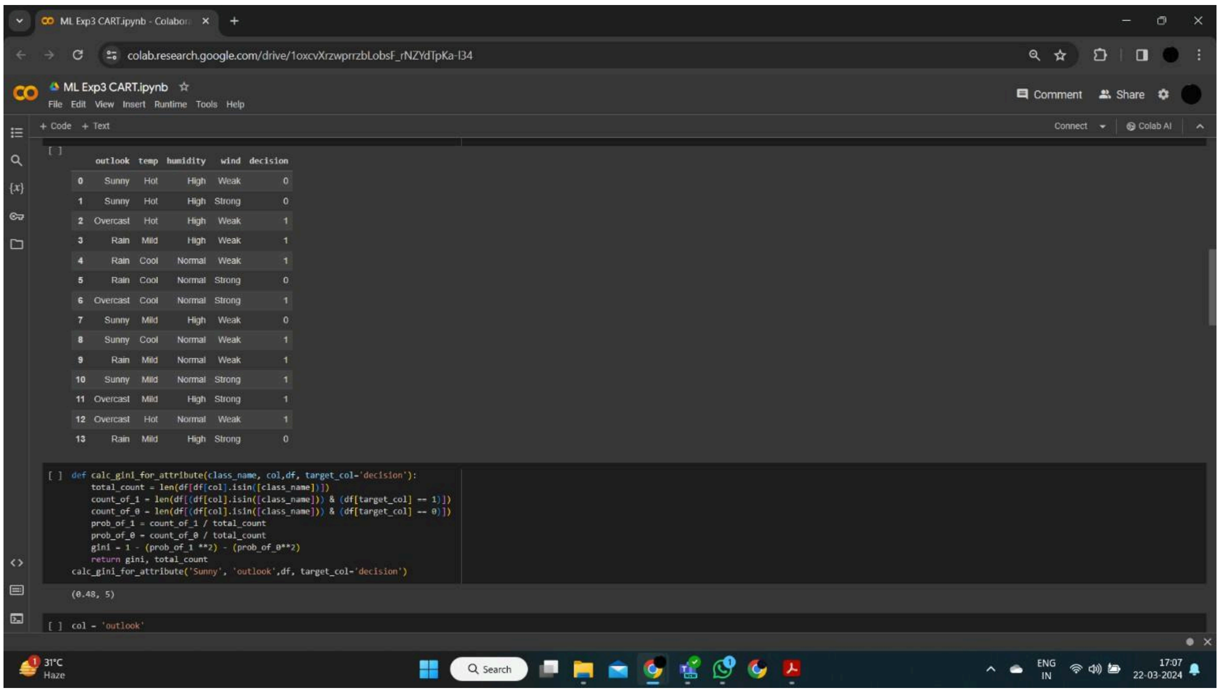


```python
for y in y_test:
  print(y,end=" ")
```

```
[0.9125]
1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 1 0 1 1 0 0
0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 0 1 0 1 1 0 1
```

```python
import pandas as pd
import numpy as np
df = pd.DataFrame()
outlook = ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast', 'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast', 'Overcast', 'Rain']
temp = ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild']
humidity = ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'High']
wind = ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Strong']
decision = [0, 0, 1, 1, 1, 0, 1,0, 1, 1, 1, 1, 1, 0]
df['outlook'] = outlook
df['temp'] = temp
df['humidity'] = humidity
df['wind'] = wind
df['decision'] = decision
df
```

	outlook	temp	humidity	wind	decision
0	Sunny	Hot	High	Weak	0
1	Sunny	Hot	High	Strong	0
2	Overcast	Hot	High	Weak	1
3	Rain	Mild	High	Weak	1
4	Rain	Cool	Normal	Weak	1
5	Rain	Cool	Normal	Strong	0
6	Overcast	Cool	Normal	Strong	1
7	Sunny	Mild	High	Weak	0
8	Sunny	Cool	Normal	Weak	1
9	Rain	Mild	Normal	Weak	1

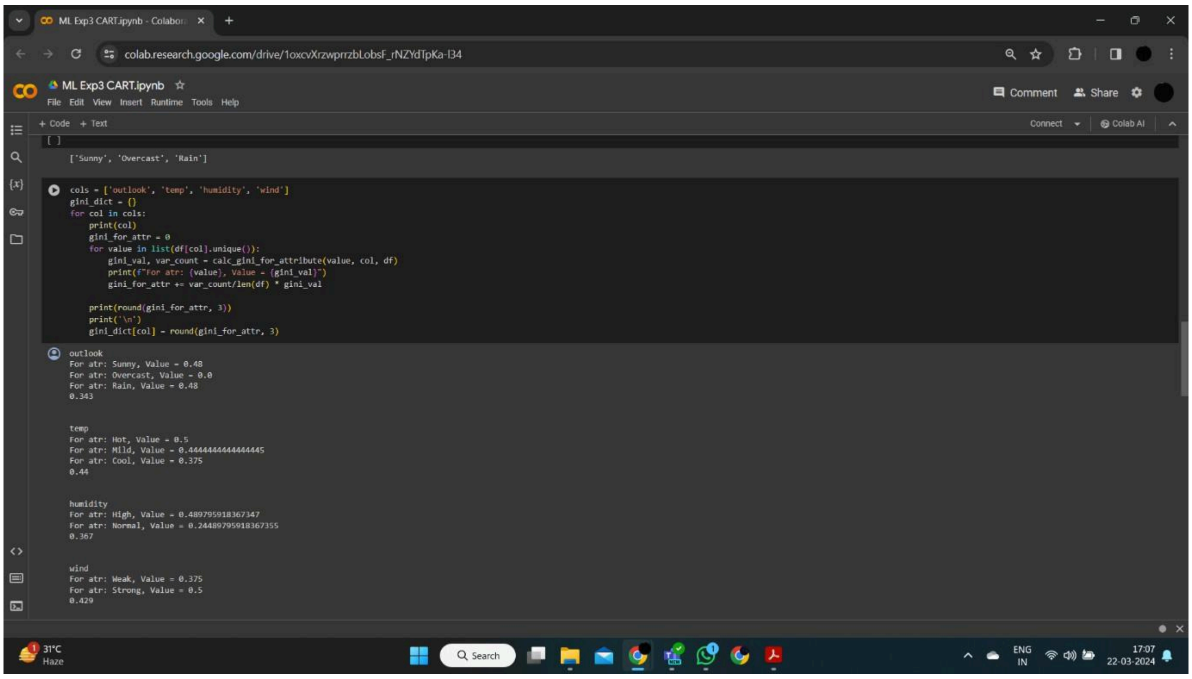


	outlook	temp	humidity	wind	decision
0	Sunny	Hot	High	Weak	0
1	Sunny	Hot	High	Strong	0
2	Overcast	Hot	High	Weak	1
3	Rain	Mild	High	Weak	1
4	Rain	Cool	Normal	Weak	1
5	Rain	Cool	Normal	Strong	0
6	Overcast	Cool	Normal	Strong	1
7	Sunny	Mild	High	Weak	0
8	Sunny	Cool	Normal	Weak	1
9	Rain	Mild	Normal	Weak	1
10	Sunny	Mild	Normal	Strong	1
11	Overcast	Mild	High	Strong	1
12	Overcast	Hot	Normal	Weak	1
13	Rain	Mild	High	Strong	0

```
def calc_gini_for_attribute(class_name, col,df, target_col='decision'):
    total_count = len(df[df[col].isin([class_name])])
    count_of_1 = len(df[(df[col].isin([class_name])) & (df[target_col] == 1)])
    count_of_0 = len(df[(df[col].isin([class_name])) & (df[target_col] == 0)])
    prob_of_1 = count_of_1 / total_count
    prob_of_0 = count_of_0 / total_count
    gini = 1 - (prob_of_1 **2) - (prob_of_0**2)
    return gini, total_count
calc_gini_for_attribute('Sunny', 'outlook',df, target_col='decision')
```

```
(0.48, 5)
```

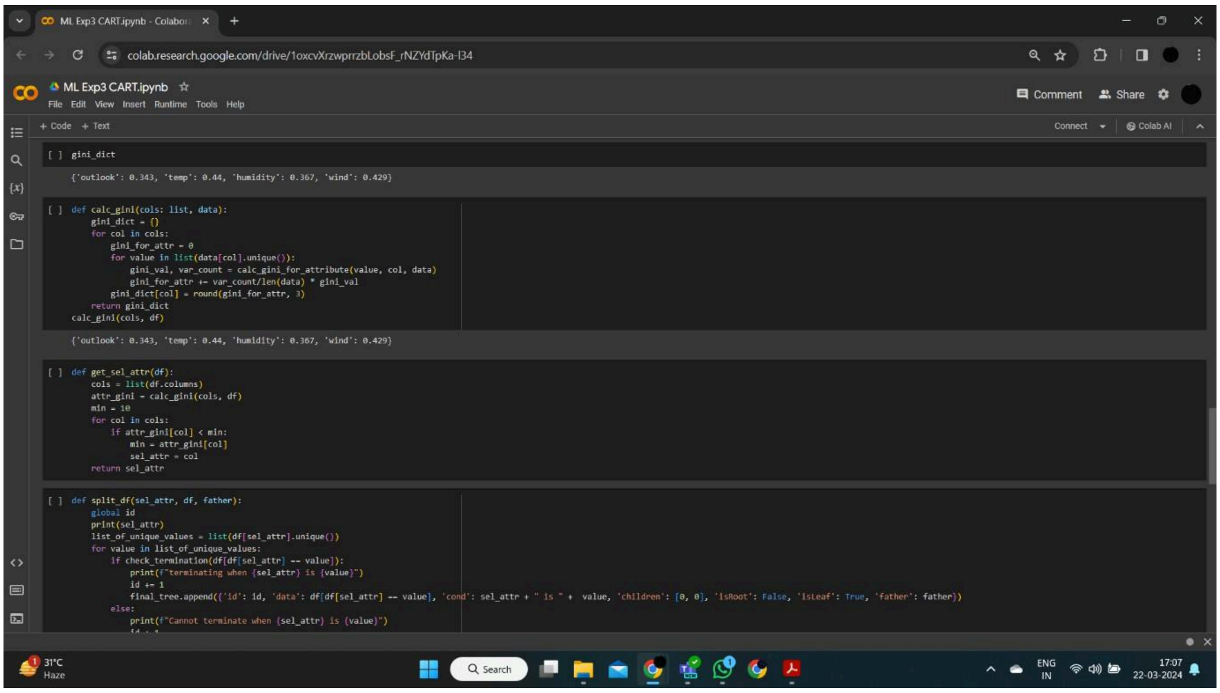
```
col = 'outlook'
```



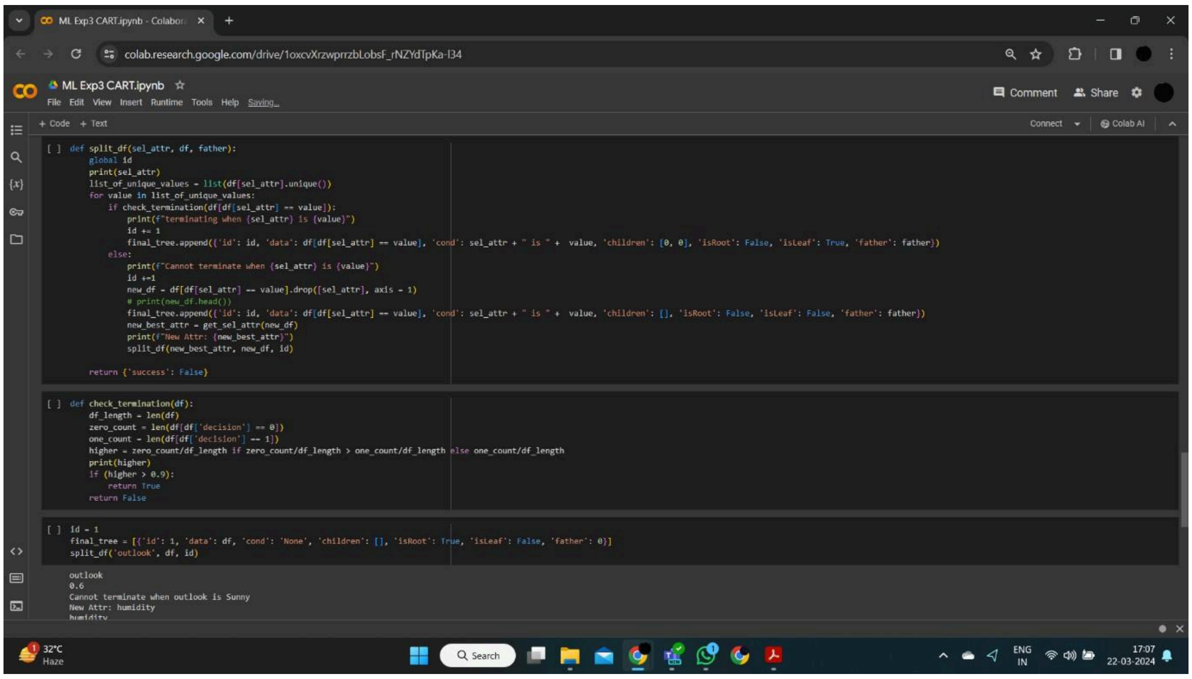
```
['Sunny', 'Overcast', 'Rain']
```

```
cols = ['outlook', 'temp', 'humidity', 'wind']
gini_dict = {}
for col in cols:
    print(col)
    gini_for_attr = 0
    for value in list(df[col].unique()):
        gini_val, var_count = calc_gini_for_attribute(value, col, df)
        print(f"For atr: {value}, Value = {gini_val}")
        gini_for_attr += var_count/len(df) * gini_val

    print(round(gini_for_attr, 3))
    print('\n')
    gini_dict[col] = round(gini_for_attr, 3)
```

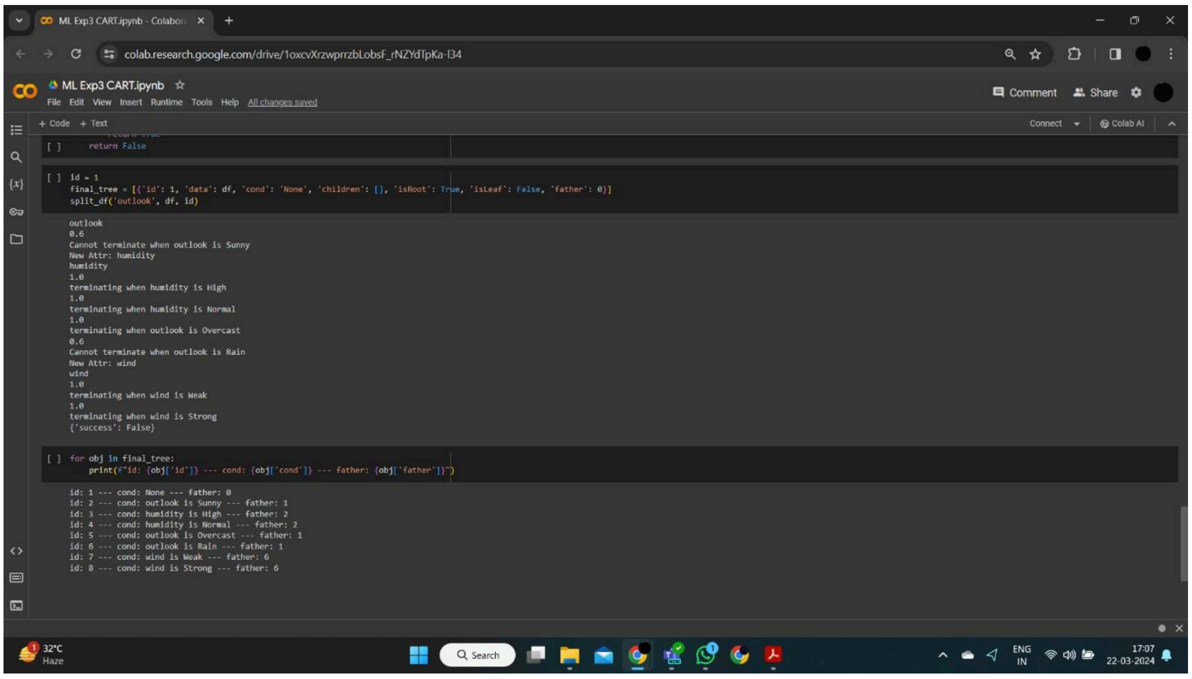
```
outlook
For atr: Sunny, Value = 0.48
For atr: Overcast, Value = 0.0
For atr: Rain, Value = 0.48
0.343


temp
For atr: Hot, Value = 0.5
For atr: Mild, Value = 0.4444444444444445
For atr: Cool, Value = 0.375
0.44


humidity
For atr: High, Value = 0.489795918367347
For atr: Normal, Value = 0.24489795918367355
0.367


wind
For atr: Weak, Value = 0.375
For atr: Strong, Value = 0.5
0.429
```


ML Exp3 CART.ipynb - Colabor
colab.research.google.com/drive/1oxcvXrzwprrzbLobsF_rNZYdTpKa-I34
ML Exp3 CART.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
Colab AI
31°C
Haze
Search
ENG
IN
17:07
22-03-2024

```
[ ] gini_dict

    {'outlook': 0.343, 'temp': 0.44, 'humidity': 0.367, 'wind': 0.429}

[ ] def calc_gini(cols: list, data):
        gini_dict = {}
        for col in cols:
            gini_for_attr = 0
            for value in list(data[col].unique()):
                gini_val, var_count = calc_gini_for_attribute(value, col, data)
                gini_for_attr += var_count/len(data) * gini_val
            gini_dict[col] = round(gini_for_attr, 3)
        return gini_dict
    calc_gini(cols, df)

    {'outlook': 0.343, 'temp': 0.44, 'humidity': 0.367, 'wind': 0.429}

[ ] def get_sel_attr(df):
        cols = list(df.columns)
        attr_gini = calc_gini(cols, df)
        min = 10
        for col in cols:
            if attr_gini[col] < min:
                min = attr_gini[col]
                sel_attr = col
        return sel_attr

[ ] def split_df(sel_attr, df, father):
        global id
        print(sel_attr)
        list_of_unique_values = list(df[sel_attr].unique())
        for value in list_of_unique_values:
            if check_termination(df[df[sel_attr] == value]):
                print(f"terminating when {sel_attr} is {value}")
                id += 1
                final_tree.append({'id': id, 'data': df[df[sel_attr] == value], 'cond': sel_attr + " is " +  value, 'children': [0, 0], 'isRoot': False, 'isLeaf': True, 'father': father})
            else:
                print(f"Cannot terminate when {sel_attr} is {value}")
```


ML Exp3 CART.ipynb - Colabor
colab.research.google.com/drive/1oxcvXrzwprrzbLobsF_rNZYdTpKa-I34
ML Exp3 CART.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
Connect
Colab AI
32°C
Haze
Search
ENG
IN
17:07
22-03-2024

```
[ ] def split_df(sel_attr, df, father):
        global id
        print(sel_attr)
        list_of_unique_values = list(df[sel_attr].unique())
        for value in list_of_unique_values:
            if check_termination(df[df[sel_attr] == value]):
                print(f"terminating when {sel_attr} is {value}")
                id += 1
                final_tree.append({'id': id, 'data': df[df[sel_attr] == value], 'cond': sel_attr + " is " +  value, 'children': [0, 0], 'isRoot': False, 'isLeaf': True, 'father': father})
            else:
                print(f"Cannot terminate when {sel_attr} is {value}")
                id +=1
                new_df = df[df[sel_attr] == value].drop([sel_attr], axis = 1)
                # print(new_df.head())
                final_tree.append({'id': id, 'data': df[df[sel_attr] == value], 'cond': sel_attr + " is " +  value, 'children': [], 'isRoot': False, 'isLeaf': False, 'father': father})
                new_best_attr = get_sel_attr(new_df)
                print(f"New Attr: {new_best_attr}")
                split_df(new_best_attr, new_df, id)

        return {'success': False}

[ ] def check_termination(df):
        df_length = len(df)
        zero_count = len(df[df['decision'] == 0])
        one_count = len(df[df['decision'] == 1])
        higher = zero_count/df_length if zero_count/df_length > one_count/df_length else one_count/df_length
        print(higher)
        if (higher > 0.9):
            return True
        return False

[ ] id = 1
    final_tree = [{'id': 1, 'data': df, 'cond': 'None', 'children': [], 'isRoot': True, 'isLeaf': False, 'father': 0}]
    split_df('outlook', df, id)

    outlook
    0.6
    Cannot terminate when outlook is Sunny
    New Attr: humidity
```



```
    return False
```

```
id = 1
final_tree = [{'id': 1, 'data': df, 'cond': 'None', 'children': [], 'isRoot': True, 'isLeaf': False, 'father': 0}]
split_df('outlook', df, id)
```

```
outlook
0.6
Cannot terminate when outlook is Sunny
New Attr: humidity
humidity
1.0
terminating when humidity is High
1.0
terminating when humidity is Normal
1.0
terminating when outlook is Overcast
0.6
Cannot terminate when outlook is Rain
New Attr: wind
wind
1.0
terminating when wind is Weak
1.0
terminating when wind is Strong
{'success': False}
```

```
for obj in final_tree:
    print(f"id: {obj['id']} --- cond: {obj['cond']} --- father: {obj['father']}")
```

```
id: 1 --- cond: None --- father: 0
id: 2 --- cond: outlook is Sunny --- father: 1
id: 3 --- cond: humidity is High --- father: 2
id: 4 --- cond: humidity is Normal --- father: 2
id: 5 --- cond: outlook is Overcast --- father: 1
id: 6 --- cond: outlook is Rain --- father: 1
id: 7 --- cond: wind is Weak --- father: 6
id: 8 --- cond: wind is Strong --- father: 6
```

ML Experiment 3 DATE:

Kreena Shah
60004210243
C'32

27Feb23

Aim : To implement CART algorithm

Theory :
CART (Classification & Regression Trees) is a variation of the decision tree algorithm
It can handle both classification & regression tasks
It is a predictive algorithm used in ML & it explains how the target variables values can be predicted based on other parameters

CART Algorithm
(1) Tree structure
(2) Splitting criteria

$$Gini = 1 - \sum_{i=1}^{n} (P_i)^2$$

(3) Pruning

Advantages
(1) Results are simplistic
(2) Trees are non parametric & non-linear
(3) Trees implicitly perform feature selection

Disadvantages
(1) Overfitting
(2) High Variance
(3) Low bias

DATE:

Conclusion : Thus, we implemented CART algorithm