Content

Gini Impurity

Code Walkthrough

```
[ ] import pandas as pd
    import numpy as np
```

```
[ ] !gdown  1l53Fgkg1G1ekCxxgaDQ80EXrnSMTeJj-
```

```
Downloading...
From: https://drive.google.com/uc?id=1l53Fgkg1G1ekCxxgaDQ80EXrnSMTeJj-
To: /content/sample_data.csv
100% 32.5k/32.5k [00:00<00:00, 24.8MB/s]
```

```
[ ] sample_data = pd.read_csv('sample_data.csv')
```

```
[ ] sample_data
```

	Gender	Age_less_35	JobRole	Attrition
0	Male	True	Laboratory Technician	0
1	Male	False	Sales Executive	1
2	Male	True	Sales Representative	1
3	Female	False	Healthcare Representative	0
4	Male	True	Sales Executive	0
...	...	...	...	...
995	Male	False	Laboratory Technician	1
996	Female	False	Manufacturing Director	0
997	Female	True	Sales Executive	0
998	Male	False	Manager	0
999	Female	True	Laboratory Technician	0

1000 rows × 4 columns

```
[ ] sample_data.Attrition.value_counts()
```

```
0    831
1    169
Name: Attrition, dtype: int64
```

```
[ ] def gini_impurity(y):

      if isinstance(y, pd.Series):
        p = y.value_counts()/y.shape[0]
        gini = 1-np.sum(p**2)
        return gini

      else:
        raise('Object must be a Pandas Series.')
```

```
[ ] gini_impurity(sample_data.Attrition)
```

```
0.28087799999999996
```

Weighted Gini impurity for child node

```
[ ] def calculate_weighted_gini(feature, y):
        categories = feature.unique()

        weighted_gini_impurity = 0

        for category in categories:
            y_category = y[feature == category]
            gini_impurity_category = gini_impurity(y_category)
            # print(category)
            # print(gini_impurity_category)
            weighted_gini_impurity += y_category.shape[0]/y.shape[0]*gini_impurity_category


        return weighted_gini_impurity
```

```
[ ] calculate_weighted_gini(sample_data.Age_less_35, sample_data.Attrition)
```

```
0.2724771918985819
```

Information Gain

```
[ ] def information_gain(feature,y):
        parent_gini = gini_impurity(y)

        child_gini = calculate_weighted_gini(feature,y)

        ig = parent_gini - child_gini

        return ig
```

```
[ ] information_gain(sample_data.Age_less_35, sample_data.Attrition)
```

```
0.008400808101418078
```

```
[ ] for feature in sample_data.columns[:-1]:
        print(f'Information Gain for feature {feature} is {information_gain(sample_data[feature],sample_data.Attrition)}')
```

```
Information Gain for feature Gender is 1.2832567979348397e-06
Information Gain for feature Age_less_35 is 0.008400808101418078
Information Gain for feature JobRole is 0.020654039636781696
```

Splitting Numerical Feature

```
[ ] !gdown 19L3rYatfhbBL1r5MHrv-p_oM2wlvrhqk
    !gdown 1N7O_fWCTJLu8SIa_paKcDEzllgpMk8sK
```

```
Downloading...
From: https://drive.google.com/uc?id=19L3rYatfhbBL1r5MHrv-p_oM2wlvrhqk
To: /content/preprocessed_X_sm.pickle
100% 534k/534k [00:00<00:00, 118MB/s]
Downloading...
From: https://drive.google.com/uc?id=1N7O_fWCTJLu8SIa_paKcDEzllgpMk8sK
To: /content/y_sm.pickle
100% 15.4k/15.4k [00:00<00:00, 22.0MB/s]
```

```
[ ] import pickle
    # Load data (deserialize)
    with open('preprocessed_X_sm.pickle', 'rb') as handle:
        X_sm = pickle.load(handle)

    with open('y_sm.pickle', 'rb') as handle:
        target = pickle.load(handle)
```

Code walkthrough

Let's split the Age feature and find which threshold is best to split age along with its information gain

```
[ ] age = X_sm.Age
```

Sorting the age

```
[ ] thresholds = age.sort_values().unique()
    thresholds
```

```
array([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60])
```

```
[ ] thresholds.shape
```

```
(43,)
```

Calculating information gain for each threshold

```
[ ] def information_gain(y, mask):
        left_node_count = sum(mask)
        total = mask.shape[0]
        right_node_count = total - left_node_count


        parent_gini = gini_impurity(y)

        child_gini = left_node_count/total*gini_impurity(y[mask]) + right_node_count/total*gini_impurity(y[~mask])

        ig = parent_gini - child_gini
        return ig
```

```
[ ] ig_list = []

    for thr in thresholds:
        mask =   age <= thr

        ig = information_gain(target, mask)
        ig_list.append(ig)
```

```
[ ] ig_list = np.array(ig_list)

    ig_list.shape
```

```
(43,)
```

Finding threshold with maximum IG

```
[ ] print(f'Best threshold for Age with maximum IG is {thresholds[ig_list.argmax()]} with IG: {ig_list.max()}')
```

```
Best threshold for Age with maximum IG is 33 with IG: 0.027621195039458812
```

```
[ ] Start coding or generate with AI.
```