What You Will Learn

By reading this document and using the provided code,
you will gain the following insights:

By scanning this QR code, you will gain access to all the files mentioned in this document.

Understanding Gini Impurity:

- Learn what Gini Impurity is and its significance in data analysis and machine learning.
- Understand the mathematical formula and its interpretation.

Feature Impact Analysis:

- Discover how to calculate Gini Impurity for dataset features.
- Identify which features have the most influence on the target variable based on their Gini Impurity scores.

Practical Application:

- Get hands-on experience with Python code that implements Gini Impurity calculations.
- Learn to process and analyze real-world datasets stored in Excel format.

Foundation for Decision Trees:

- Understand how Gini Impurity is used in building decision trees for classification problems.
- Build a foundation for more advanced machine learning concepts.

Relation Between Gini Impurity, Decision Trees and Entropy

Gini Impurity and Entropy are metrics used in decision trees to measure the quality of splits:

- **Gini Impurity:** Measures the probability of misclassification. It's computationally efficient and commonly used in CART (Classification and Regression Trees).
- **Entropy:** Based on information theory, it measures uncertainty. Decision trees like ID3 and C4.5 use entropy to calculate information gain.

Key Differences:

- Gini is faster to compute since it avoids logarithmic calculations.
- Both metrics aim to create purer subsets, often resulting in similar tree structures.

In short, Gini and Entropy guide decision trees to select the best splits for accurate classification.

Gini index and Gini impurity

The Gini Index (or Gini Impurity) is a widely employed metric used for splitting a classification decision tree or, in macroeconomics, to measure income inequality within a country. The Gini Index is also known as Gini Impurity. Both names are correct, but since we are working about ‘data mining’ and ‘decision trees’, we will use Gini Impurity. in continue, we provide a brief explanation of both names for the same formula.

It derives its name from the Italian mathematician and statistician **Corrado Gini**. The Gini index was developed in 1912 by him. In 1910, he acceded to the Chair of Statistics in the University of Cagliari and then at Padua in 1913.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

“GINI IN DECISION”

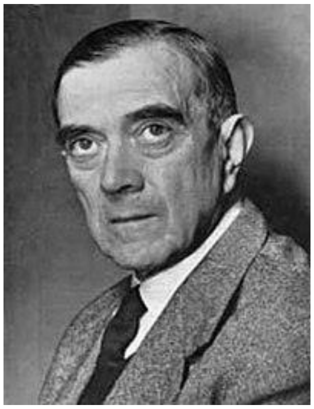
Gini impurity is a measure of how mixed or impure a dataset is. The Gini impurity ranges between 0 and 1(or 0% to 100%), where 0 represents a pure dataset and 1 represents a completely impure dataset. This means that an attribute with a lower Gini index should be preferred. In other words, the lower the Gini impurity, the better and stronger the feature is for influencing decisions and splitting the dataset in decision trees.

from machinelearning-basics.com

“GINI IN ECONOMY”

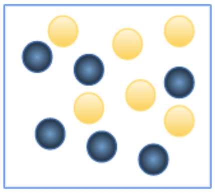
The Gini index in macroeconomics determines a nation's level of income inequality by measuring the income or wealth distribution across its population. The Gini index is a measure of the distribution of income across a population. A higher Gini index indicates greater inequality, with high-income individuals receiving much larger percentages of the populations total income. It means a country that everyone resident the same income would have an income Gini coefficient of 0.

from investopedia.com



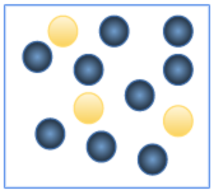
Gini.C 1848-1965
from wikipedia.org

Highly Impuree



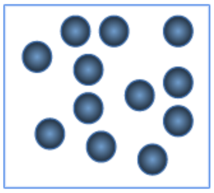
High Gini Index

Medium Impure



Medium Gini Index

Pure



Low Gini Index
(even=0)

Mathematical Formula

The computation of the Gini index is as follows:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Where:

- C: The number of classes included in the subset
- i: The class i, with i {1, 2, 3, …, C}
- p(i) is the probability of a specific class and the summation is done for all classes present in the dataset.

remember, the lower the value of the Gini Index, the purer the dataset and the lower the entropy.

Another commonly used version of the formula:

$$Gini\ impurity = 1 - \Sigma\left(p(i) * \left(1 - p(i)\right)\right)$$

In this form actually p(i) is probability of our target and 1 – p(i) is probability of opposite of target.

Both of these formulae are equivalent, and the result you get from these formulae would be the same. The first formula is more computationally efficient, therefore it is more commonly used.

Practical example on paper

I have designed an imaginary table for going on a walk based on weather conditions to illustrate how the Gini index formula works and its components. Later, we will use this same table as the first example in the Python program.

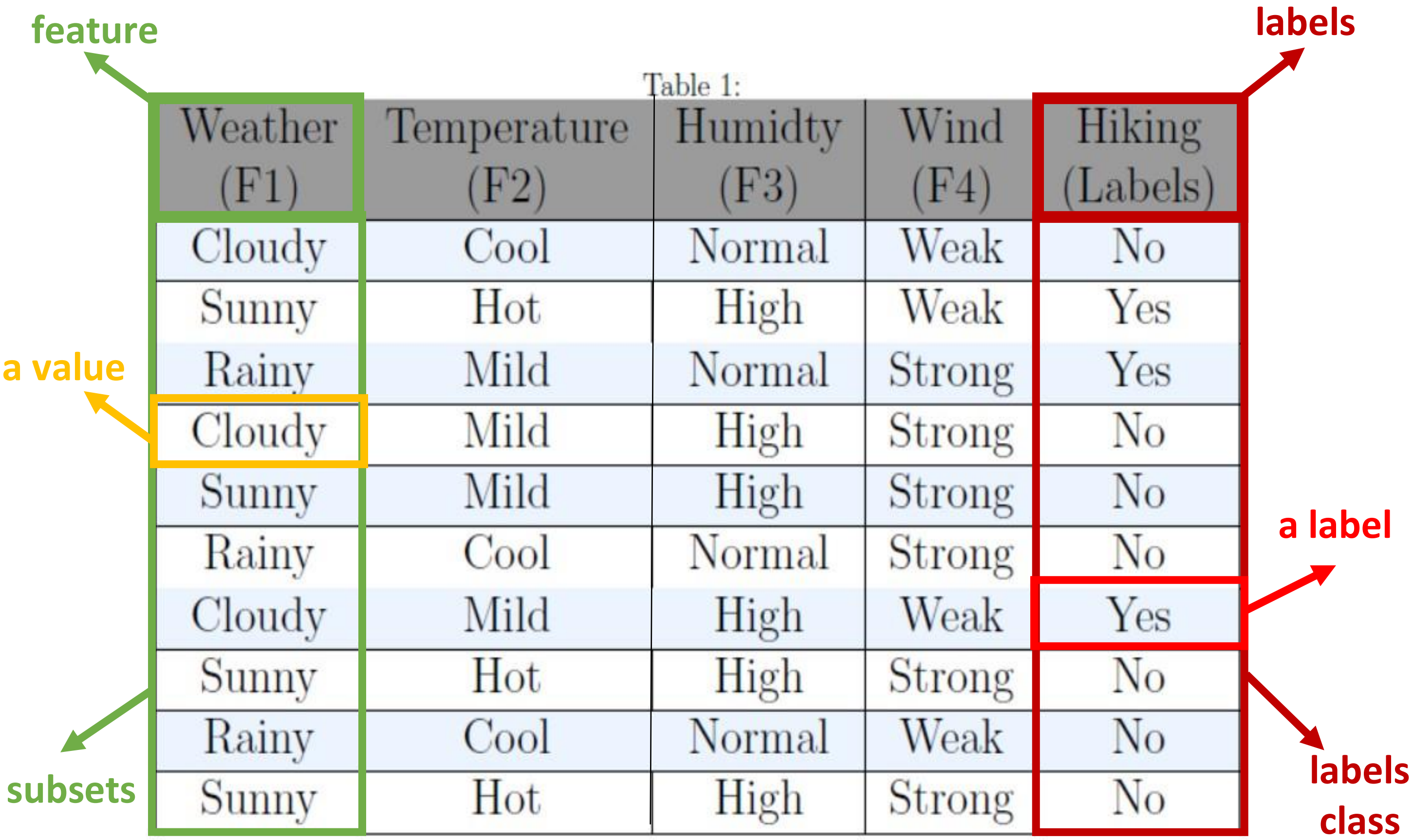

feature
labels
a value
a label
subsets
labels class

Table 1:

Weather (F1)	Temperature (F2)	Humidty (F3)	Wind (F4)	Hiking (Labels)
Cloudy	Cool	Normal	Weak	No
Sunny	Hot	High	Weak	Yes
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Rainy	Cool	Normal	Strong	No
Cloudy	Mild	High	Weak	Yes
Sunny	Hot	High	Strong	No
Rainy	Cool	Normal	Weak	No
Sunny	Hot	High	Strong	No

Download this table from here

1. Calculate P(Hiking-labels)

P (Yes) = 3/10 , P (No) = 7/10

2. Calculate P(~) (All features)

3. Calculate Gini impurity for all features

(next page)

Weather (F1)	Temperature (F2)	Humidty (F3)	Wind(F4)
P(F1 = Cloudy) = $\frac{3}{10}$	P(F2 = Cool) = $\frac{3}{10}$	P(F3=Normal) = $\frac{4}{10}$	P(F4 = Weak) = $\frac{4}{10}$
P(F1 = Sunny) = $\frac{4}{10}$	P(F2 = Hot) = $\frac{3}{10}$	P(F3 = High) = $\frac{6}{10}$	P(F4 = Strong) = $\frac{6}{10}$
P(F1 = Rainy) = $\frac{3}{10}$	P(F2 = Mild) = $\frac{4}{10}$		

Gini Index of Weather (F1) >>>

Gini Index of Cloudy = 1 - $\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2$ = 0.44

Gini Index of Sunny = 1 - $\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2$ = 0.375

Gini Index of Rainy = 1 - $\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2$ = 0.44

Gini Index of Weather (F1) =

$$\frac{3}{10} \times 0.44 + \frac{4}{10} \times 0.375 + \frac{3}{10} \times 0.44 = 0.414$$

Gini Index of Temperature (F2) >>>

Gini Index of Cool = 1 - $\left(\frac{0}{3}\right)^2 + \left(\frac{3}{3}\right)^2$ = 0

Gini Index of Hot = 1 - $\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2$ = 0.44

Gini Index of Mild = 1 - $\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2$ = 0.5

Gini Index of Temperature (F2) =

$$\frac{3}{10} \times 0 + \frac{4}{10} \times 0.44 + \frac{3}{10} \times 0.5 = 0.332$$

Gini Index of Humidity (F3) >>>

Gini Index of Normal = 1 - $\left(\frac{2}{4}\right)^2 + \left(\frac{3}{4}\right)^2$ = 0.375

Gini Index of High = 1 - $\left(\frac{6}{6}\right)^2 + \left(\frac{0}{6}\right)^2$ = 0.44

Gini Index of Humidity (F3) =

$$\frac{4}{10} \times 0.375 + \frac{6}{10} \times 0.44 = 0.414$$

Gini Index of Wind (F4) >>>

Gini Index of Weak = 1 - $\left(\frac{4}{8}\right)^2 + \left(\frac{4}{8}\right)^2$ = 0.5

Gini Index of Strong = 1 - $\left(\frac{6}{12}\right)^2 + \left(\frac{6}{12}\right)^2$ = 0.278

Gini Index of Wind (F4) =

$$\frac{4}{10} \times 0.5 + \frac{6}{10} \times 0.278 = 0.367$$

As a result, by combining the outcomes of each feature, you should obtain a result similar to the table shown below. The lowest Gini value corresponds to the "Temperature" feature, It means that temperature has the greatest impact on the decision to going for a walk compared to the other features.

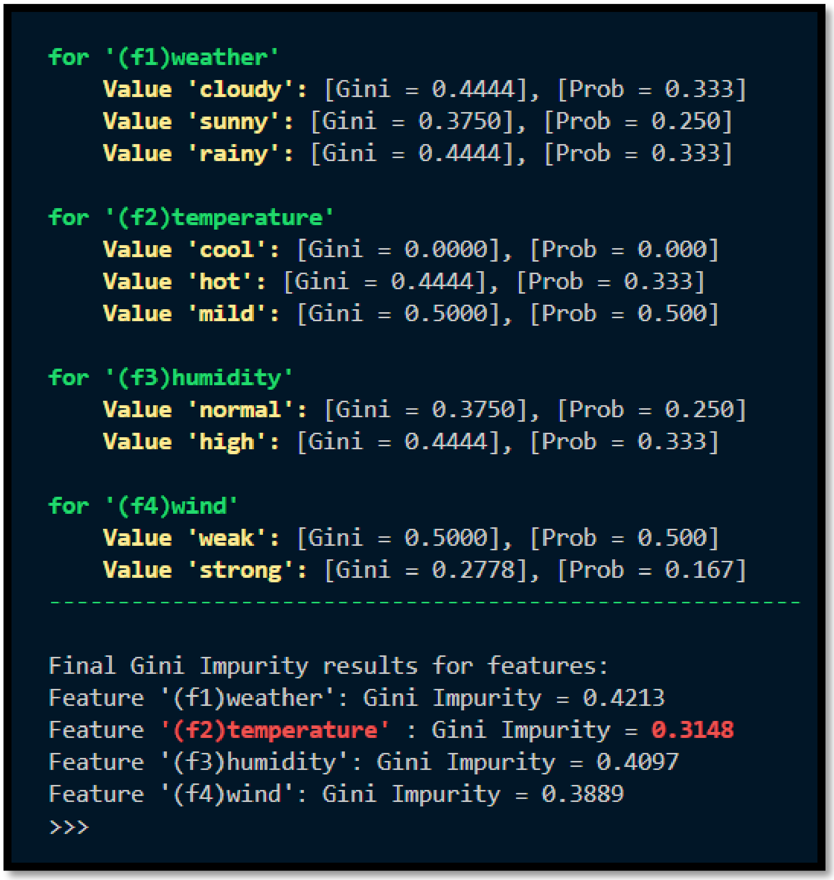
(F1) Weather	0.414
(F2) Temperature	0.332
(F3) Humidity	0.414
(F4) Wind	0.367

In next, I will introduce a Python script that, with a same input in Excel format, provides the corresponding output in a fraction of a second.

gini_impurity.py

First of all, you can download the file from the provided link. Then, ensure that you had installed Python and Pandas library. Next, you need to replace the path "X:/Your/Excel/file/path/here.xlsx" with the actual file path in Your system. The program will then run and display the result in the terminal.

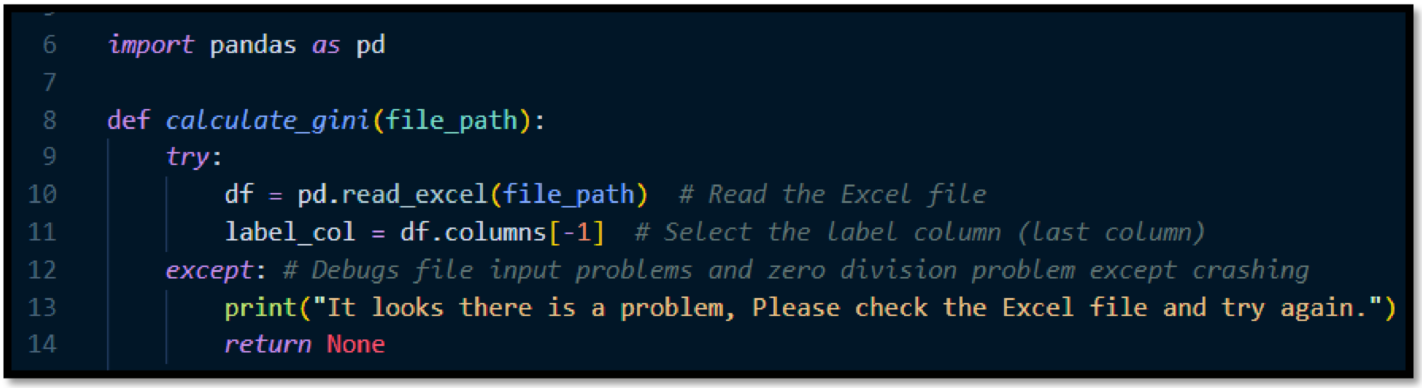
Pay attention the last column of the Excel file must contain 'Yes' or 'No' and empty cells will makes issues.



```
for '(f1)weather'
    Value 'cloudy': [Gini = 0.4444], [Prob = 0.333]
    Value 'sunny': [Gini = 0.3750], [Prob = 0.250]
    Value 'rainy': [Gini = 0.4444], [Prob = 0.333]

for '(f2)temperature'
    Value 'cool': [Gini = 0.0000], [Prob = 0.000]
    Value 'hot': [Gini = 0.4444], [Prob = 0.333]
    Value 'mild': [Gini = 0.5000], [Prob = 0.500]

for '(f3)humidity'
    Value 'normal': [Gini = 0.3750], [Prob = 0.250]
    Value 'high': [Gini = 0.4444], [Prob = 0.333]

for '(f4)wind'
    Value 'weak': [Gini = 0.5000], [Prob = 0.500]
    Value 'strong': [Gini = 0.2778], [Prob = 0.167]
------------------------------------------------------

Final Gini Impurity results for features:
Feature '(f1)weather': Gini Impurity = 0.4213
Feature '(f2)temperature' : Gini Impurity = 0.3148
Feature '(f3)humidity': Gini Impurity = 0.4097
Feature '(f4)wind': Gini Impurity = 0.3889
>>>
```

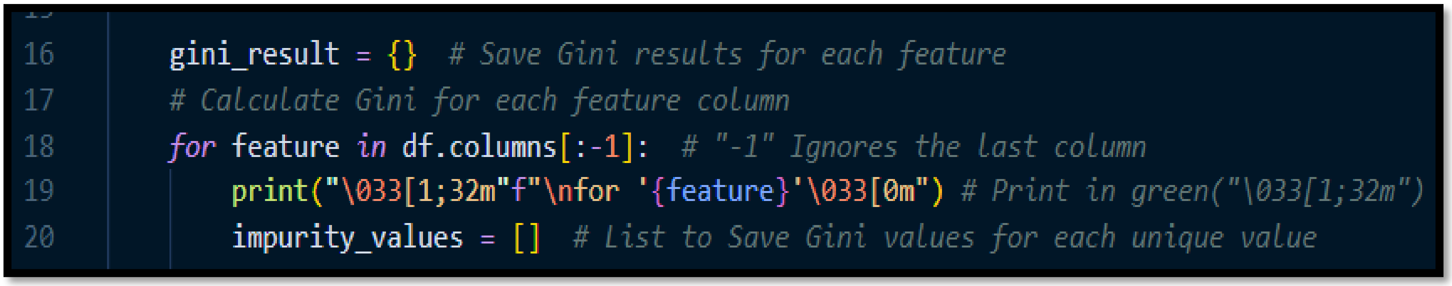
[Download all requirements from here]

Here is a line-by-line explanation of My code with line numbers, simple explanations, and references to the relevant functions used from the libraries. I've also included links to the official documentation for each the library functions.



```
 6  import pandas as pd
 7
 8  def calculate_gini(file_path):
 9      try:
10          df = pd.read_excel(file_path)  # Read the Excel file
11          label_col = df.columns[-1]  # Select the label column (last column)
12      except: # Debugs file input problems and zero division problem except crashing
13          print("It looks there is a problem, Please check the Excel file and try again.")
14          return None
```

- **Line 6**: This imports the **pandas** library, which is essential for handling and analyzing data in Python, especially data in tabular form (like Excel files).
- **Reference**: pandas documentation
- **Line 8**: This defines a function named calculate_gini, which accepts the file path of an Excel file as an argument.
- **Line 9**: The `try` block begins to handle errors that might occur while reading the file.
- **Line 10**: `pd.read_excel()` reads an Excel file and loads it into a pandas DataFrame.
- **Reference**: pandas.read_excel
- **Line11**: `df.columns[-1]` selects the last column in the DataFrame, which is assumed to be the label column (the target, typically 'yes' or 'no').
- **Lines 12-14**: The `except` block catches any error (like file reading issues or zero division errors) and prints a user-friendly error message.
- **Hint**: It is better to specify the exact errors to catch (e.g., `except Exception as e`), but this will catch all errors in this case. Also I tried to write the codes simple as possible to easy underastanding.



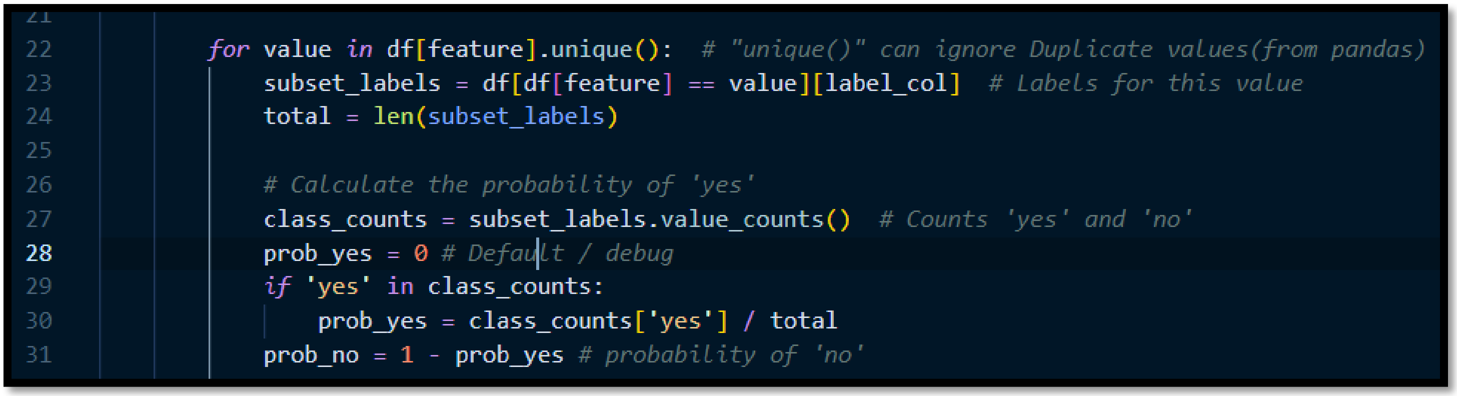
```
16      gini_result = {}  # Save Gini results for each feature
17      # Calculate Gini for each feature column
18      for feature in df.columns[:-1]:  # "-1" Ignores the last column
19          print("\033[1;32m"f"\nfor '{feature}'\033[0m") # Print in green("\033[1;32m")
20          impurity_values = []  # List to Save Gini values for each unique value
```

Line 16: `gini_result` is initialized as an empty dictionary, which will store the Gini impurity for each feature column in next.

Line 18: This `for` loop iterates through all the feature columns except the last column (which is the label column).

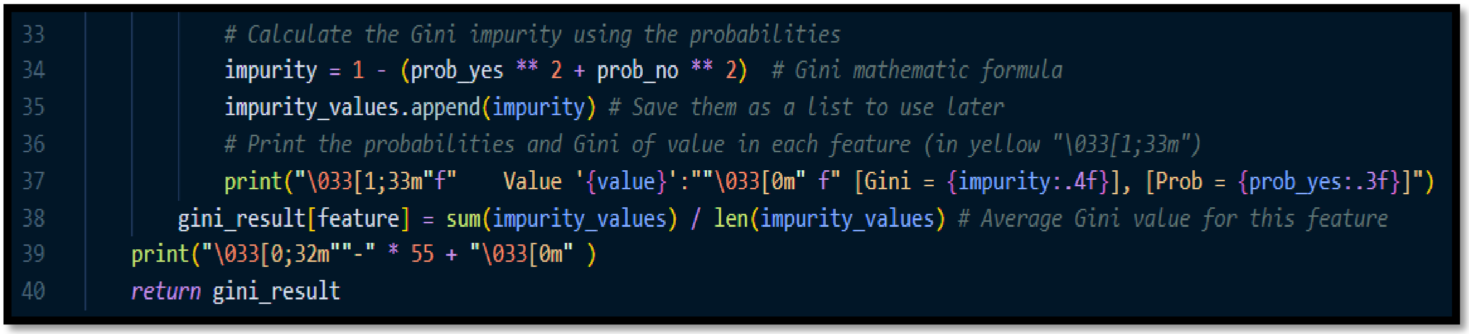
- **Line 19**: This prints the name of the current feature in green using ANSI escape codes for color formatting.
- **Hint**: My goal was to make it as easy to understand as possible. It will make visual enhancement in the terminal.
- **Reference**: ANSI escape codes

Line 20: `impurity_values` is an empty list that will hold the Gini impurity values for each unique value of the feature.



```
22          for value in df[feature].unique():  # "unique()" can ignore Duplicate values(from pandas)
23              subset_labels = df[df[feature] == value][label_col]  # Labels for this value
24              total = len(subset_labels)
25
26              # Calculate the probability of 'yes'
27              class_counts = subset_labels.value_counts()  # Counts 'yes' and 'no'
28              prob_yes = 0 # Default / debug
29              if 'yes' in class_counts:
30                  prob_yes = class_counts['yes'] / total
31              prob_no = 1 - prob_yes # probability of 'no'
```

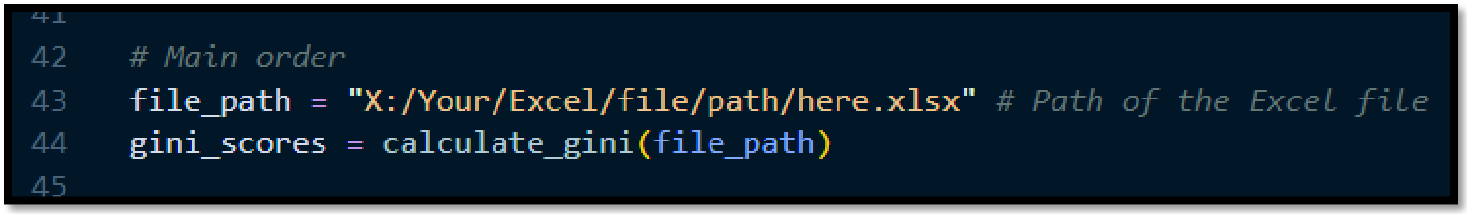
- **Line 22**: This `for` loop iterates over each unique value of the current feature column. The `unique()` function from pandas ensures that duplicate values are ignored.
- **Reference**: pandas.Series.unique
- **Line 23**: This filters the rows where the feature column matches the current value, then selects the label column.
- **Line 24**: `len(subset_labels)` counts the number of rows for the current value.
- **Line 27**: `value_counts()` counts the occurrences of each unique value in the `subset_labels` (which are either 'yes' or 'no').
- **Reference**: pandas.Series.value_counts
- **Line 28-31**: The probabilities of 'yes' and 'no' are calculated based on the counts of each label. If 'yes' exists, its probability is calculated, and the probability of 'no' is simply (`1 - prob_yes`).



```
33              # Calculate the Gini impurity using the probabilities
34              impurity = 1 - (prob_yes ** 2 + prob_no ** 2)  # Gini mathematic formula
35              impurity_values.append(impurity) # Save them as a list to use later
36              # Print the probabilities and Gini of value in each feature (in yellow "\033[1;33m")
37              print("\033[1;33m"f"    Value '{value}':""\033[0m" f" [Gini = {impurity:.4f}], [Prob = {prob_yes:.3f}]")
38          gini_result[feature] = sum(impurity_values) / len(impurity_values) # Average Gini value for this feature
39      print("\033[0;32m""-" * 55 + "\033[0m" )
40      return gini_result
```

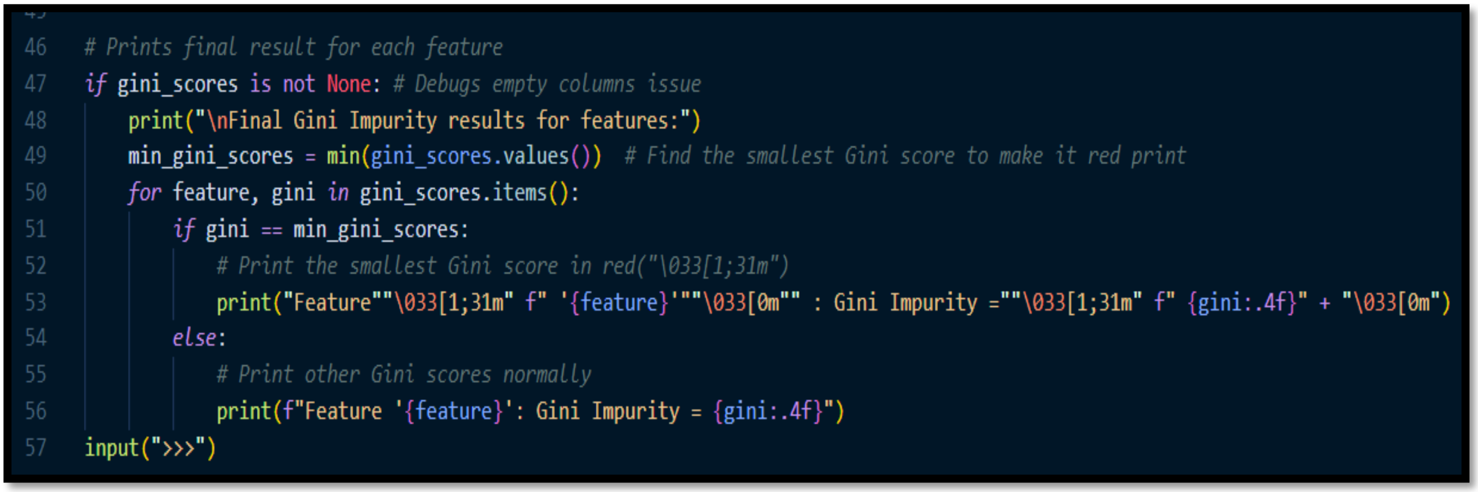
- **Line 34**: do just like **Gini impurity** second formula, 1 - (prob_yes^2 + prob_no^2).
- **Line 35**: The impurity value is appended to `impurity_values`.
- **Line 37**: This prints the Gini impurity and the probability of 'yes' for the current value of the feature, in yellow color and rounded.
- **Line 38**: The **average** Gini impurity for the current feature is calculated and stored in the `gini_result` dictionary.
- **Line 39**: its will print a green line in terminal, to separate the results of features and values.
- **Line 40:** finally This returns the dictionary `gini_result` that contains the Gini impurity values for each feature, dict file because I need to Iterating on key and values Separately.

```
gini_result = {
    "feature1": 0.414,
    " feature2": 0.332
}
```



```
42    # Main order
43    file_path = "X:/Your/Excel/file/path/here.xlsx" # Path of the Excel file
44    gini_scores = calculate_gini(file_path)
45
```

- **Line 43-44**: These lines define the file path and call the `calculate_gini()` function, passing the file path as an argument.
- **Hint**: you should replace the path `"X:/Your/Excel/file/path/here.xlsx"` with the actual file path in Your system.



```
46  # Prints final result for each feature
47  if gini_scores is not None: # Debugs empty columns issue
48      print("\nFinal Gini Impurity results for features:")
49      min_gini_scores = min(gini_scores.values())  # Find the smallest Gini score to make it red print
50      for feature, gini in gini_scores.items():
51          if gini == min_gini_scores:
52              # Print the smallest Gini score in red("\033[1;31m")
53              print("Feature""\033[1;31m" f" '{feature}'""\033[0m"" : Gini Impurity =""\033[1;31m" f" {gini:.4f}" + "\033[0m")
54          else:
55              # Print other Gini scores normally
56              print(f"Feature '{feature}': Gini Impurity = {gini:.4f}")
57  input(">>>")
```

- **Line 47:** This checks that result of the function should not be none(not none). It is another part of debugging empty entries and more possible problems in lines 9 to 14, also this trick improves the codes performance.
- **Line 48**: This prints a header message where the final Gini results will be shown.
- **Line 49:** This finds the minimum Gini score (the most important feature) using the `min()` function and puts in `min_gini_scores`.
- **Line 50**: This iterates over each feature and its corresponding Gini score in the `gini_scores` dictionary.
- **Line 51-52**: If the current feature has the smallest Gini(`min_gini_scores`) score, it is printed in red and rounded.
- **Line 54-56:** and prints other results of features ruonded in next step.
- **Line 57:** it is a simple trick because when you are running the script in a terminal doesn't close after finished.