Introduction to AI
Decision Tree with UCI Poker and scikit-learn

Lab 2

21CLC05 - University of Science - VNUHCM

21127135 – Diệp Hữu Phúc

✦

0. Report

This will be a long read so please make use of bookmarks for ease of navigation.

Priority	No.	Task	Status
Required	1	Preparing the data sets	Done
	2	Building the decision tree classifiers	Done
	3	Evaluating the decision tree classifiers • Classification report and confusion matrix • Comments	Done
	4	The depth and accuracy of a decision tree • Trees, tables, and charts • Comments	Done
	5	IPython Notebook	Done

Below is the link to my Github repo,

- https://github.com/kru01/DecisionTree_UCI-Poker

1. Preparing the data sets

Merging the .data files

Merging **poker-hand-training-true.data** and **poker-hand-testing.data** into **poker-hand-data.csv**. Both of **.data** files and the resulting **.csv** are placed in **pokerData** folder.

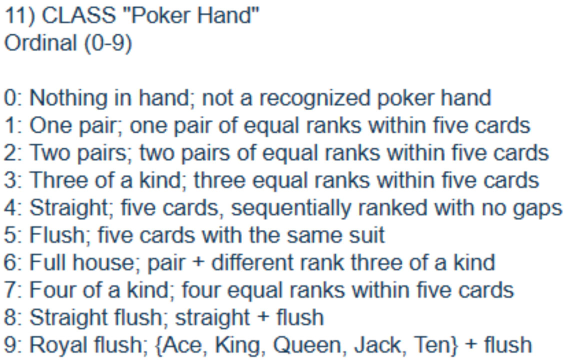
As a merging method was not specified, all training data is written in first then the testing data afterward.

Preparing the subsets

All 16 subsets will be extracted to **pokerSubsets** folder as **.csv** files, their naming format can be explained with an example. For a **train/test** ratio of **40/60**, we will have,

Subset\Ratio	Train	Test
Feature	feature_train_40.csv	feature_test_60.csv
Label	label_train_40.csv	label_test_60.csv

Both the shuffling of the data and stratified-fashion split are handled by **sklearn.model_selection.train_test_split** through the flag $shuffle = True$ and $stratify = labels$, with **labels** here being the list of all elements of the last column, each value represents a poker hand described in the class **Poker Hand**.



11) CLASS "Poker Hand"
Ordinal (0-9)

0: Nothing in hand; not a recognized poker hand
1: One pair; one pair of equal ranks within five cards
2: Two pairs; two pairs of equal ranks within five cards
3: Three of a kind; three equal ranks within five cards
4: Straight; five cards, sequentially ranked with no gaps
5: Flush; five cards with the same suit
6: Full house; pair + different rank three of a kind
7: Four of a kind; four equal ranks within five cards
8: Straight flush; straight + flush
9: Royal flush; {Ace, King, Queen, Jack, Ten} + flush

Additionally, for *a tuple of 42 attribute values*, I assume it refers to setting the flag,

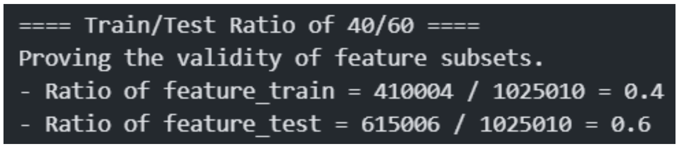
$$random\_state = 42$$

Visualizing the distributions of classes

The visualization for the original set, i.e., **poker-hand-data**, and label subsets, i.e., **label_train** and **label_test**, are straightforward enough as we have access to **Poker Hand**. With each proportion, 3 charts are drawn, those are for **label_train**, **label_test**, and **label_data**.

For **label** subsets, we just need to merge **label_train** and **label_test** into **label_data** and ascertain whether the statistics of this data resemble those of the original set.

However, in the case of **feature** subsets, due to the absence of label (or class), i.e., **Poker Hand**, it is impossible to visualize the distributions. As such, their validity will be confirmed by assuring the content of both **feature_train_X.csv** and **feature_test_Y.csv** obeys their respective ratio. This can be calculated by simply dividing the number of instances presented in each of the files by the total number of instances in the original set.



```
==== Train/Test Ratio of 40/60 ====
Proving the validity of feature subsets.
- Ratio of feature_train = 410004 / 1025010 = 0.4
- Ratio of feature_test = 615006 / 1025010 = 0.6
```

2. Building the decision tree classifiers

- The decision trees drawn by **graphviz** are output to **decisionTreeClassifiers** folder.
- **max_depth=None:** It takes ***an eternity*** for **graphviz** to render this tree so we will only observe its first 10 levels. In other words, the decision tree, i.e., the **model**, is still built with $max\_depth = None$, we will only cut off its plotting at depth 10.

The name template for the tree files is,

$$depth\_ob\{max\_depth\_observable\}\_\{train\_ratio\}\{test\_ratio\}.csv$$

with **ob** here shorts for **observation depth**. For illustration, a decision tree classifier with a **train/test** ratio of **40/60** would have the file **depth_ob10_4060.csv**.

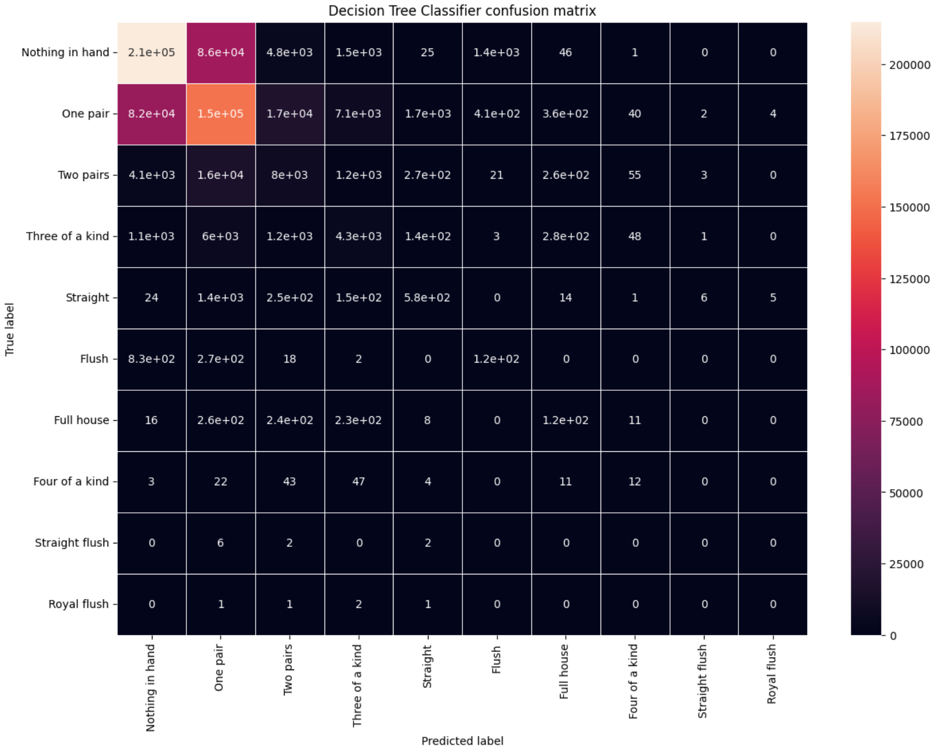
3. Evaluating the decision tree classifiers

We will first attempt to see what kind of information can be gathered from a classification report and confusion matrix before moving on to evaluating all decision tree classifiers.

Interpreting for individual decision tree classifier

Since the underlying principles remain unchanged for every report and matrix, we will only look at the **train/test** ratio of **40/60** as an example.

Confusion matrix



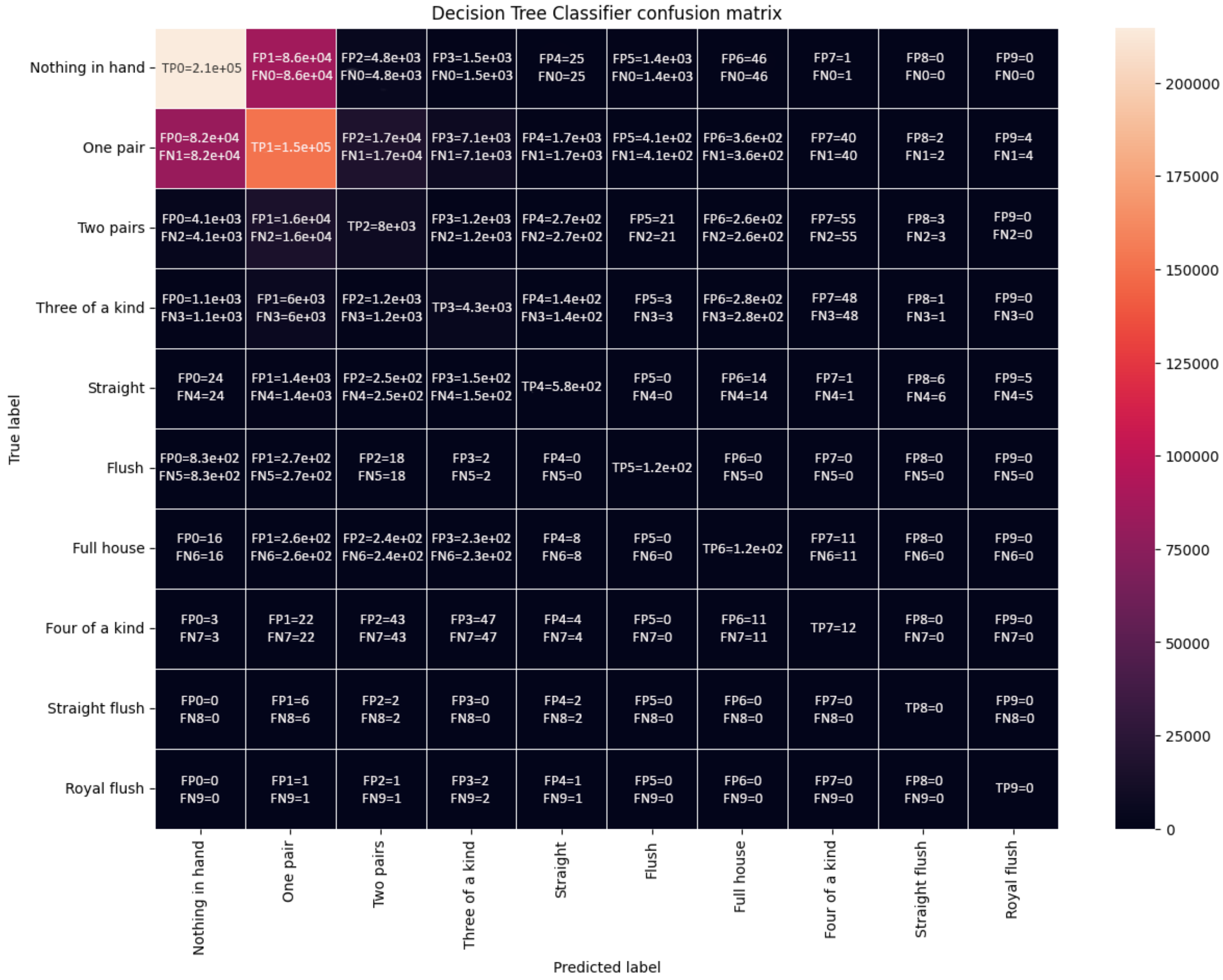
	Nothing in hand	One pair	Two pairs	Three of a kind	Straight	Flush	Full house	Four of a kind	Straight flush	Royal flush
Nothing in hand	2.1e+05	8.6e+04	4.8e+03	1.5e+03	25	1.4e+03	46	1	0	0
One pair	8.2e+04	1.5e+05	1.7e+04	7.1e+03	1.7e+03	4.1e+02	3.6e+02	40	2	4
Two pairs	4.1e+03	1.6e+04	8e+03	1.2e+03	2.7e+02	21	2.6e+02	55	3	0
Three of a kind	1.1e+03	6e+03	1.2e+03	4.3e+03	1.4e+02	3	2.8e+02	48	1	0
Straight	24	1.4e+03	2.5e+02	1.5e+02	5.8e+02	0	14	1	6	5
Flush	8.3e+02	2.7e+02	18	2	0	1.2e+02	0	0	0	0
Full house	16	2.6e+02	2.4e+02	2.3e+02	8	0	1.2e+02	11	0	0
Four of a kind	3	22	43	47	4	0	11	12	0	0
Straight flush	0	6	2	0	2	0	0	0	0	0
Royal flush	0	1	1	2	1	0	0	0	0	0

Firstly, we will explore the confusion matrix for it not only helps us make sense of the data in the classification report but also validate said data. The above matrix will possess the following scheme of TP, FP, and FN.

- **True Positive (TP):** The model predicted positive, and the real value is positive.
- **True Negative (TN):** The model predicted negative, and the real value is negative.
- **False Positive (FP):** The model predicted positive, but the real value is negative.
  - Type I error.
- **False Negative (FN):** The model predicted negative, but the real value is positive.
  - Type II error.

Each row contains one cell with TP and 9 cells with FN, while each column contains one cell with TP and 9 cells with FP. Let us consider **class 1**, which is labeled **One pair** according to **Poker Hand**.

Decision Tree Classifier confusion matrix



	Nothing in hand	One pair	Two pairs	Three of a kind	Straight	Flush	Full house	Four of a kind	Straight flush	Royal flush
Nothing in hand	TP0=2.1e+05	FP1=8.6e+04 FN0=8.6e+04	FP2=4.8e+03 FN0=4.8e+03	FP3=1.5e+03 FN0=1.5e+03	FP4=25 FN0=25	FP5=1.4e+03 FN0=1.4e+03	FP6=46 FN0=46	FP7=1 FN0=1	FP8=0 FN0=0	FP9=0 FN0=0
One pair	FP0=8.2e+04 FN1=8.2e+04	TP1=1.5e+05	FP2=1.7e+04 FN1=1.7e+04	FP3=7.1e+03 FN1=7.1e+03	FP4=1.7e+03 FN1=1.7e+03	FP5=4.1e+02 FN1=4.1e+02	FP6=3.6e+02 FN1=3.6e+02	FP7=40 FN1=40	FP8=2 FN1=2	FP9=4 FN1=4
Two pairs	FP0=4.1e+03 FN2=4.1e+03	FP1=1.6e+04 FN2=1.6e+04	TP2=8e+03	FP3=1.2e+03 FN2=1.2e+03	FP4=2.7e+02 FN2=2.7e+02	FP5=21 FN2=21	FP6=2.6e+02 FN2=2.6e+02	FP7=55 FN2=55	FP8=3 FN2=3	FP9=0 FN2=0
Three of a kind	FP0=1.1e+03 FN3=1.1e+03	FP1=6e+03 FN3=6e+03	FP2=1.2e+03 FN3=1.2e+03	TP3=4.3e+03	FP4=1.4e+02 FN3=1.4e+02	FP5=3 FN3=3	FP6=2.8e+02 FN3=2.8e+02	FP7=48 FN3=48	FP8=1 FN3=1	FP9=0 FN3=0
Straight	FP0=24 FN4=24	FP1=1.4e+03 FN4=1.4e+03	FP2=2.5e+02 FN4=2.5e+02	FP3=1.5e+02 FN4=1.5e+02	TP4=5.8e+02	FP5=0 FN4=0	FP6=14 FN4=14	FP7=1 FN4=1	FP8=6 FN4=6	FP9=5 FN4=5
Flush	FP0=8.3e+02 FN5=8.3e+02	FP1=2.7e+02 FN5=2.7e+02	FP2=18 FN5=18	FP3=2 FN5=2	FP4=0 FN5=0	TP5=1.2e+02	FP6=0 FN5=0	FP7=0 FN5=0	FP8=0 FN5=0	FP9=0 FN5=0
Full house	FP0=16 FN6=16	FP1=2.6e+02 FN6=2.6e+02	FP2=2.4e+02 FN6=2.4e+02	FP3=2.3e+02 FN6=2.3e+02	FP4=8 FN6=8	FP5=0 FN6=0	TP6=1.2e+02	FP7=11 FN6=11	FP8=0 FN6=0	FP9=0 FN6=0
Four of a kind	FP0=3 FN7=3	FP1=22 FN7=22	FP2=43 FN7=43	FP3=47 FN7=47	FP4=4 FN7=4	FP5=0 FN7=0	FP6=11 FN7=11	TP7=12	FP8=0 FN7=0	FP9=0 FN7=0
Straight flush	FP0=0 FN8=0	FP1=6 FN8=6	FP2=2 FN8=2	FP3=0 FN8=0	FP4=2 FN8=2	FP5=0 FN8=0	FP6=0 FN8=0	FP7=0 FN8=0	TP8=0	FP9=0 FN8=0
Royal flush	FP0=0 FN9=0	FP1=1 FN9=1	FP2=1 FN9=1	FP3=2 FN9=2	FP4=1 FN9=1	FP5=0 FN9=0	FP6=0 FN9=0	FP7=0 FN9=0	FP8=0 FN9=0	TP9=0

- **TP** can be taken directly from the matrix.

$$TP_1 = 1.5 \times 10^5$$

- **FP** is the sum of column **One pair**, excluding **TP1** cell.

$$FP_1 = 8.6 \times 10^4 + 1.6 \times 10^4 + 6 \times 10^3 + 1.4 \times 10^3 + 2.7 \times 10^2 + 2.6 \times 10^2 + 22 + 6 + 1 = 109959$$

- **FN** is the sum of row **One pair**, excluding **TP1** cell.

$$FN_1 = 8.2 \times 10^4 + 1.7 \times 10^4 + 7.1 \times 10^3 + 1.7 \times 10^3 + 4.1 \times 10^2 + 3.6 \times 10^2 + 40 + 2 + 4 = 108616$$

- **TN** can be found by eliminating the column and the row of the class of interest from the matrix. The cells that remain contain **TN**.
  - I am not going to calculate TN_1 because we won't be using it, and the number of values going into the formula is absurd.
- **Precision** is the proportion of true positives among all the values predicted as positive.

$$Precision_1 = \frac{TP_1}{TP_1 + FP_1} = \frac{1.5 \times 10^5}{1.5 \times 10^5 + 109959} \approx 0.577014 \approx 0.58$$

- **Recall** returns the proportion of positive values correctly predicted.

$$Recall_1 = \frac{TP_1}{TP_1 + FN_2} = \frac{1.5 \times 10^5}{1.5 \times 10^5 + 108616} \approx 0.580011 \approx 0.58$$

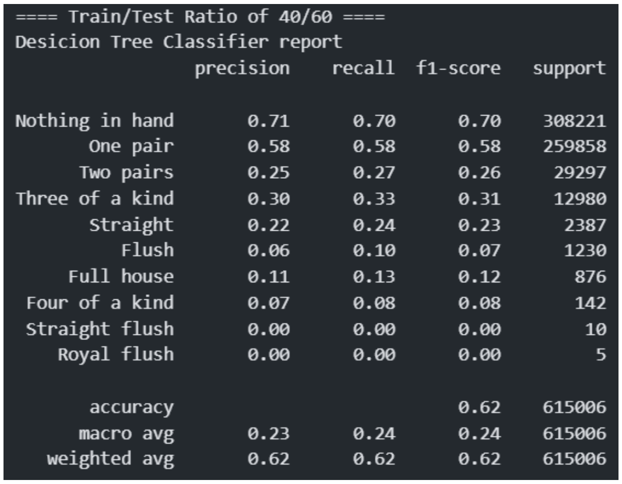
- **Specificity** returns the proportion of negative values correctly predicted.
  - The **classification report** doesn't mention it and we also didn't compute **TN**, so we won't be calculating this. I am mentioning **specificity** purely for completeness.

$$Specificity_1 = \frac{TN_1}{TN_1 + FP_1}$$

- **F1-score** is the harmonic mean of **precision** and **recall**, often used to compare classifiers.

$$\mathrm{F1}score_1 = \frac{2 \times Precision_1 \times Recall_1}{Precision_1 + Recall_1} = \frac{2 \times 0.58 \times 0.58}{0.58 + 0.58} = 0.58$$

Classification report



```
==== Train/Test Ratio of 40/60 ====
Desicion Tree Classifier report
                  precision    recall  f1-score   support

 Nothing in hand       0.71      0.70      0.70    308221
        One pair       0.58      0.58      0.58    259858
       Two pairs       0.25      0.27      0.26     29297
 Three of a kind       0.30      0.33      0.31     12980
        Straight       0.22      0.24      0.23      2387
           Flush       0.06      0.10      0.07      1230
      Full house       0.11      0.13      0.12       876
  Four of a kind       0.07      0.08      0.08       142
  Straight flush       0.00      0.00      0.00        10
     Royal flush       0.00      0.00      0.00         5

        accuracy                           0.62    615006
       macro avg       0.23      0.24      0.24    615006
    weighted avg       0.62      0.62      0.62    615006
```

After getting the hang of some attributes that can be computed for each class, we can finally continue our investigation into the report.

- **Accuracy** returns the proportion of correct predictions.

$$Accuracy = \frac{\sum_{i=0}^{9} TP_i}{\sum_{i=0}^{9} TP_i + \sum_{i=0}^{9} FP_i}$$
$$= \frac{373132}{373132 + (88073 + 109959 + 23554 + 10231 + 2150 + 1834 + 971 + 156 + 12 + 9)}$$
$$\approx 0.611611$$

- **Support** is the number of observations for each class.
- **Macro average** is the arithmetic average of a metric between the two classes.

$$MacroAvg_{precision} = \frac{\sum_{i=0}^{9} precision_i}{10} = \frac{2.3}{10} = 0.23$$

- **Weighted average** is calculated by $\frac{\sum(metric\ of\ interest \times weight)}{\sum weights}$.

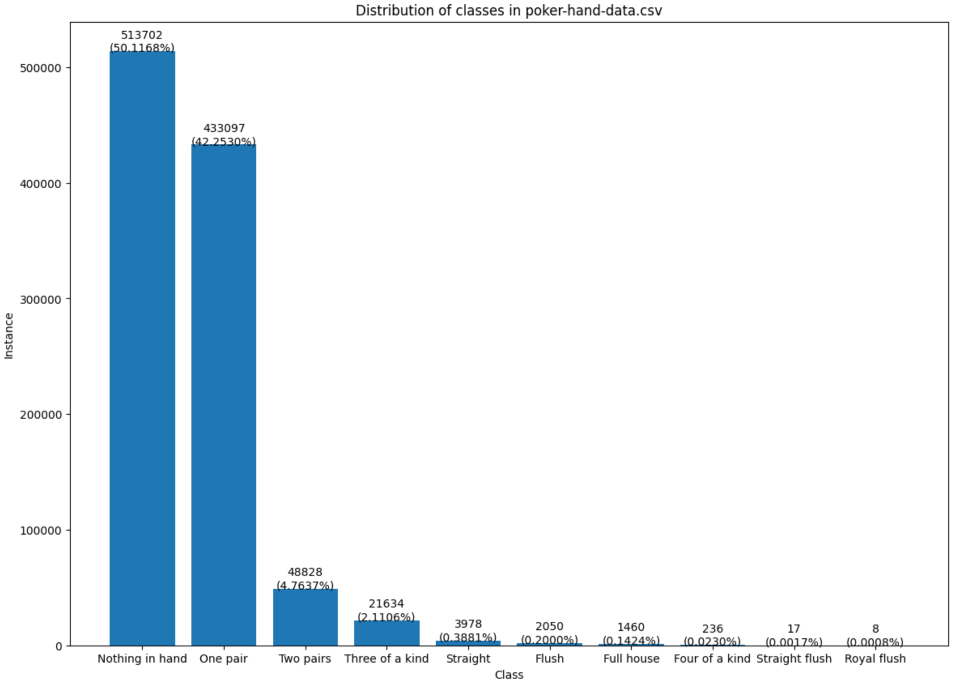
$$WeightedAvg_{precision} = \frac{\sum_{i=0}^{9}(precision_i \times support_i)}{\sum_{i=0}^{9} support_i} = \frac{381478.04}{615006} \approx 0.620283 \approx 0.62$$

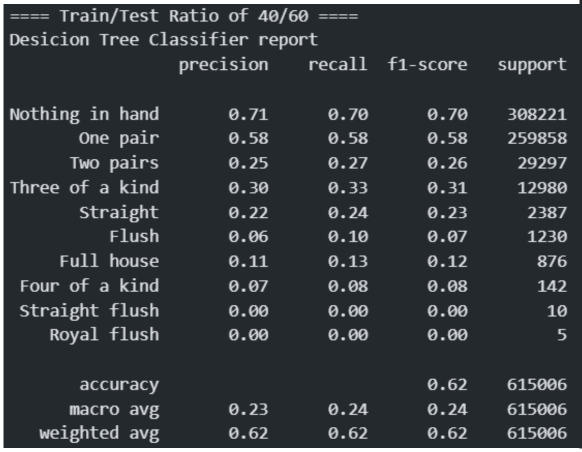
- By comparing the **precision** and **recall** of a class, the relationship between its **FN** and **FP** can be determined. To illustrate,
  - For class 0, e.g., Nothing in hand, **precision** is higher than **recall** ($0.71 > 0.70$), which means that there are more FN0 than FP0 ($93772 > 88073$).
  - For class 1, e.g., One pair, **precision** is the same as **recall** (0.58). However, this doesn't translate to FN1 (108616) and FP1 (109959) having true equality, it only means that their values are very close together (1.1×10^5).
  - For class 2, e.g., Two pairs, **precision** is less than **recall** ($0.25 < 0.27$), which means that there are fewer FN2 than FP2 ($21909 < 23554$).
  - If a class have both precision and recall as 1, it means the model correctly predicted all the observations of that class, and its **F1-score** will also be 1.

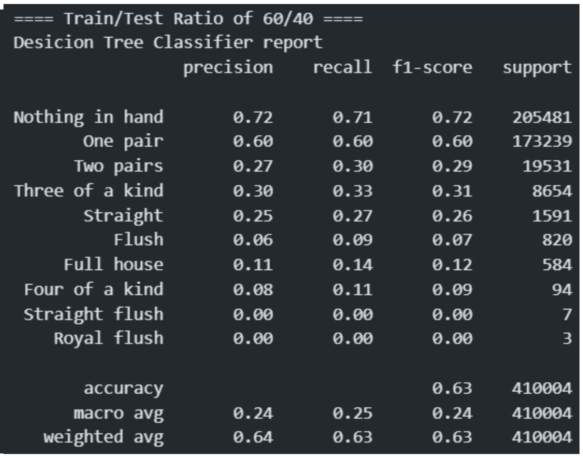
Judging performances

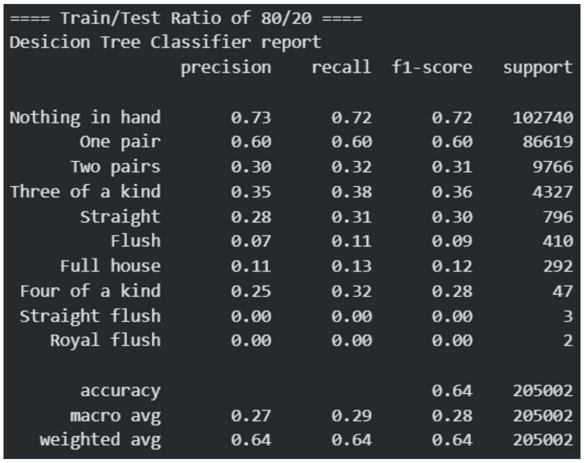
The graph for the original data set we drew in the **Visualizing the distributions of classes** section has shown us that our data set is unbalanced. Consequently, using **Accuracy** to evaluate the classification models can be misleading. Thus, depending on whether **False Positives (type I errors)** or **False Negatives (type II errors)** are preferred, **recall** or **precision**, respectively, is the best metric to employ.

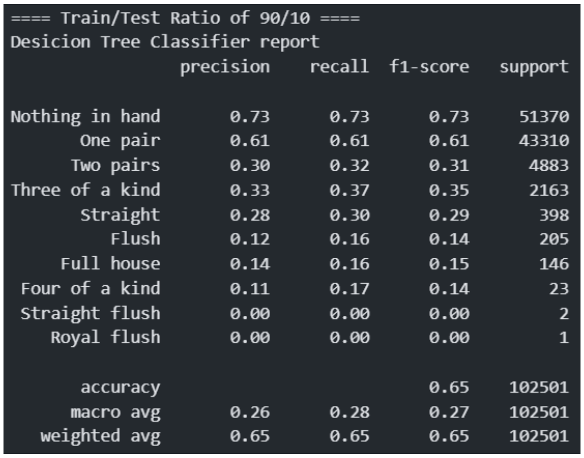
As a side note, I would like to point out that the **Accuracy** seems to follow an upward trend as the **training sets** get larger and the **test sets** smaller.




```
==== Train/Test Ratio of 40/60 ====
Desicion Tree Classifier report
                 precision    recall  f1-score   support

 Nothing in hand       0.71      0.70      0.70    308221
        One pair       0.58      0.58      0.58    259858
       Two pairs       0.25      0.27      0.26     29297
 Three of a kind       0.30      0.33      0.31     12980
        Straight       0.22      0.24      0.23      2387
           Flush       0.06      0.10      0.07      1230
      Full house       0.11      0.13      0.12       876
  Four of a kind       0.07      0.08      0.08       142
  Straight flush       0.00      0.00      0.00        10
     Royal flush       0.00      0.00      0.00         5

        accuracy                           0.62    615006
       macro avg       0.23      0.24      0.24    615006
    weighted avg       0.62      0.62      0.62    615006
```


```
==== Train/Test Ratio of 60/40 ====
Desicion Tree Classifier report
                 precision    recall  f1-score   support

 Nothing in hand       0.72      0.71      0.72    205481
        One pair       0.60      0.60      0.60    173239
       Two pairs       0.27      0.30      0.29     19531
 Three of a kind       0.30      0.33      0.31      8654
        Straight       0.25      0.27      0.26      1591
           Flush       0.06      0.09      0.07       820
      Full house       0.11      0.14      0.12       584
  Four of a kind       0.08      0.11      0.09        94
  Straight flush       0.00      0.00      0.00         7
     Royal flush       0.00      0.00      0.00         3

        accuracy                           0.63    410004
       macro avg       0.24      0.25      0.24    410004
    weighted avg       0.64      0.63      0.63    410004
```


```
==== Train/Test Ratio of 80/20 ====
Desicion Tree Classifier report
                 precision    recall  f1-score   support

 Nothing in hand       0.73      0.72      0.72    102740
        One pair       0.60      0.60      0.60     86619
       Two pairs       0.30      0.32      0.31      9766
 Three of a kind       0.35      0.38      0.36      4327
        Straight       0.28      0.31      0.30       796
           Flush       0.07      0.11      0.09       410
      Full house       0.11      0.13      0.12       292
  Four of a kind       0.25      0.32      0.28        47
  Straight flush       0.00      0.00      0.00         3
     Royal flush       0.00      0.00      0.00         2

        accuracy                           0.64    205002
       macro avg       0.27      0.29      0.28    205002
    weighted avg       0.64      0.64      0.64    205002
```

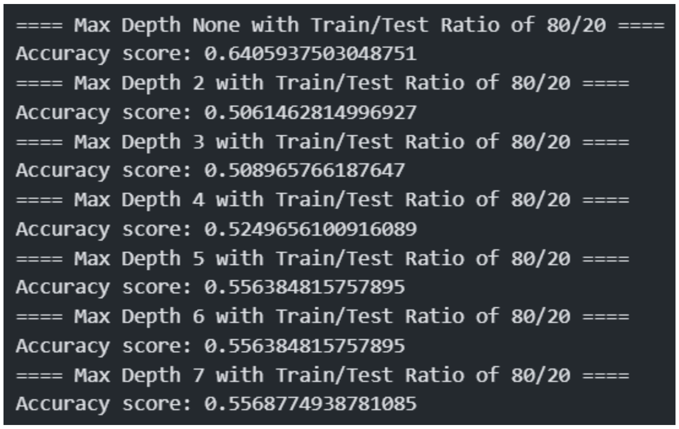

```
==== Train/Test Ratio of 90/10 ====
Desicion Tree Classifier report
                 precision    recall  f1-score   support

 Nothing in hand       0.73      0.73      0.73     51370
        One pair       0.61      0.61      0.61     43310
       Two pairs       0.30      0.32      0.31      4883
 Three of a kind       0.33      0.37      0.35      2163
        Straight       0.28      0.30      0.29       398
           Flush       0.12      0.16      0.14       205
      Full house       0.14      0.16      0.15       146
  Four of a kind       0.11      0.17      0.14        23
  Straight flush       0.00      0.00      0.00         2
     Royal flush       0.00      0.00      0.00         1

        accuracy                           0.65    102501
       macro avg       0.26      0.28      0.27    102501
    weighted avg       0.65      0.65      0.65    102501
```

4. The depth and accuracy of a decision tree

- The decision trees drawn by **graphviz** are output to **treeMaxDepth** folder.
- **max_depth=None:** It takes ***an eternity*** for **graphviz** to render this tree so we will only observe its first 10 levels. In other words, the decision tree, i.e., the **model**, is still built with $max\_depth = None$, we will only cut off its plotting at depth 10.

As we only deal with the **train/test** ratio of **80/20**, the name template for the tree files is,

$$depth\_\{max\_depth\}\_8020.csv$$



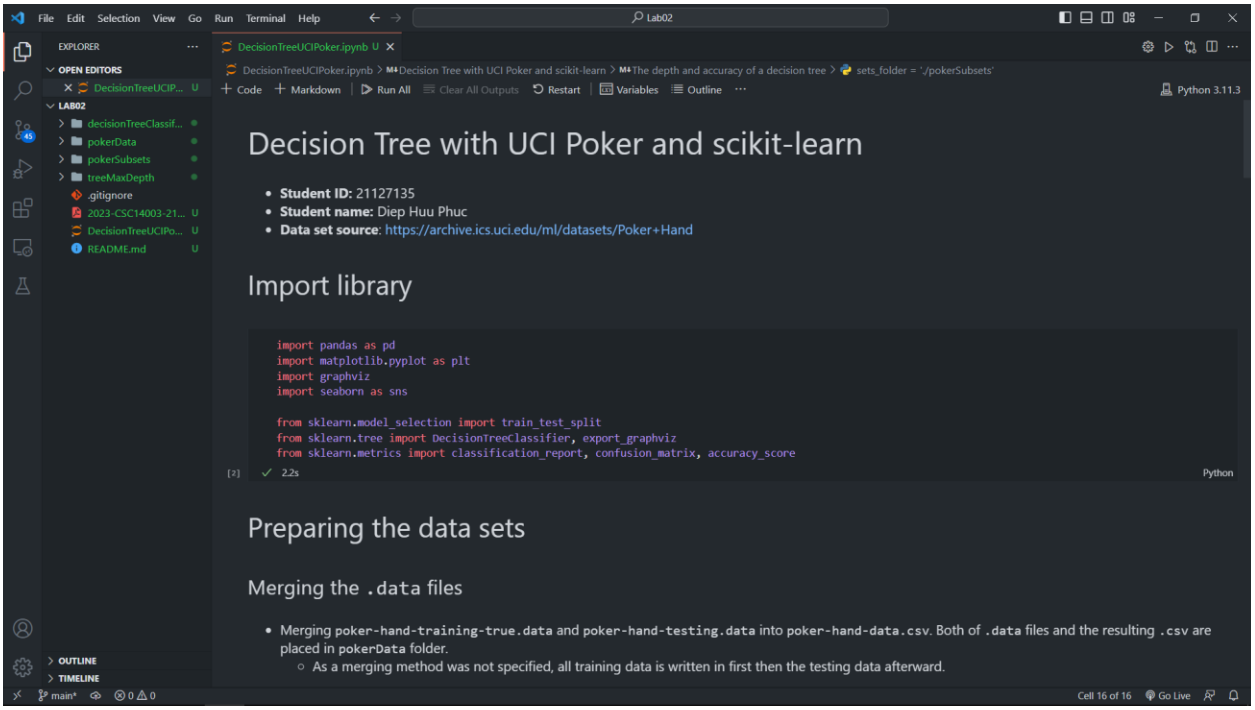
```
==== Max Depth None with Train/Test Ratio of 80/20 ====
Accuracy score: 0.6405937503048751
==== Max Depth 2 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.5061462814996927
==== Max Depth 3 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.508965766187647
==== Max Depth 4 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.5249656100916089
==== Max Depth 5 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.556384815757895
==== Max Depth 6 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.556384815757895
==== Max Depth 7 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.5568774938781085
```

max_depth	**None**	**2**	**3**	**4**	**5**	**6**	**7**
Accuracy	0.640594	0.506146	0.508966	0.524966	0.556385	0.556385	0.556877

Clearly, it can be inferred that **Accuracy** improved steadily as we dived deeper. However, **max_depth** of 5 and 6 sharing seemingly the same score is quite peculiar. When **max_depth** constraint was lifted, i.e., $max\_depth = None$, a drastic gain in **Accuracy** was obtained.

5. IPython Notebook

- Make sure Import Library is run at least once.
- The notebook is written in a way that each task is stand-alone. So long as all libraries are imported, it is not obligated to run through every code block sequentially.


File Edit Selection View Go Run Terminal Help
Lab02
EXPLORER
OPEN EDITORS
DecisionTreeUCIP... U
LAB02
decisionTreeClassif...
pokerData
pokerSubsets
treeMaxDepth
.gitignore
2023-CSC14003-21... U
DecisionTreeUCIPo... U
README.md U
OUTLINE
TIMELINE
DecisionTreeUCIPoker.ipynb U
DecisionTreeUCIPoker.ipynb > M↓Decision Tree with UCI Poker and scikit-learn > M↓The depth and accuracy of a decision tree > sets_folder = './pokerSubsets'
+ Code + Markdown | Run All Clear All Outputs Restart | Variables Outline ...
Python 3.11.3
main* ⊗0 △0
Cell 16 of 16 Go Live

Decision Tree with UCI Poker and scikit-learn

- **Student ID:** 21127135
- **Student name:** Diep Huu Phuc
- **Data set source**: https://archive.ics.uci.edu/ml/datasets/Poker+Hand

Import library

```
import pandas as pd
import matplotlib.pyplot as plt
import graphviz
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

[2] ✓ 2.2s Python

Preparing the data sets

Merging the `.data` files

- Merging `poker-hand-training-true.data` and `poker-hand-testing.data` into `poker-hand-data.csv`. Both of `.data` files and the resulting `.csv` are placed in `pokerData` folder.
  - As a merging method was not specified, all training data is written in first then the testing data afterward.

6. Reference

- https://archive.ics.uci.edu/ml/datasets/Poker+Hand
- https://www.kaggle.com/haimfeld87/analysis-and-classification-of-mushrooms

Preparing the data sets

- **pandas' API reference**
  - https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
  - https://pandas.pydata.org/docs/reference/api/pandas.concat.html
  - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_csv.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- https://matplotlib.org/stable/api/pyplot_summary.html

Building the decision tree classifiers

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://stackoverflow.com/questions/27817994/visualizing-decision-tree-in-scikit-learn
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html

Evaluating the decision tree classifiers

- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
- https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
- https://medium.com/swlh/confusion-matrix-and-classification-report-88105288d48f

The depth and accuracy of a decision tree

- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html