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-Pokers

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.csv**, and label subsets, i.e., **label_train** and **label_test**, are straightforward enough as we have access to **Poker Hand**. 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 construct 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

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 construct 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,

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

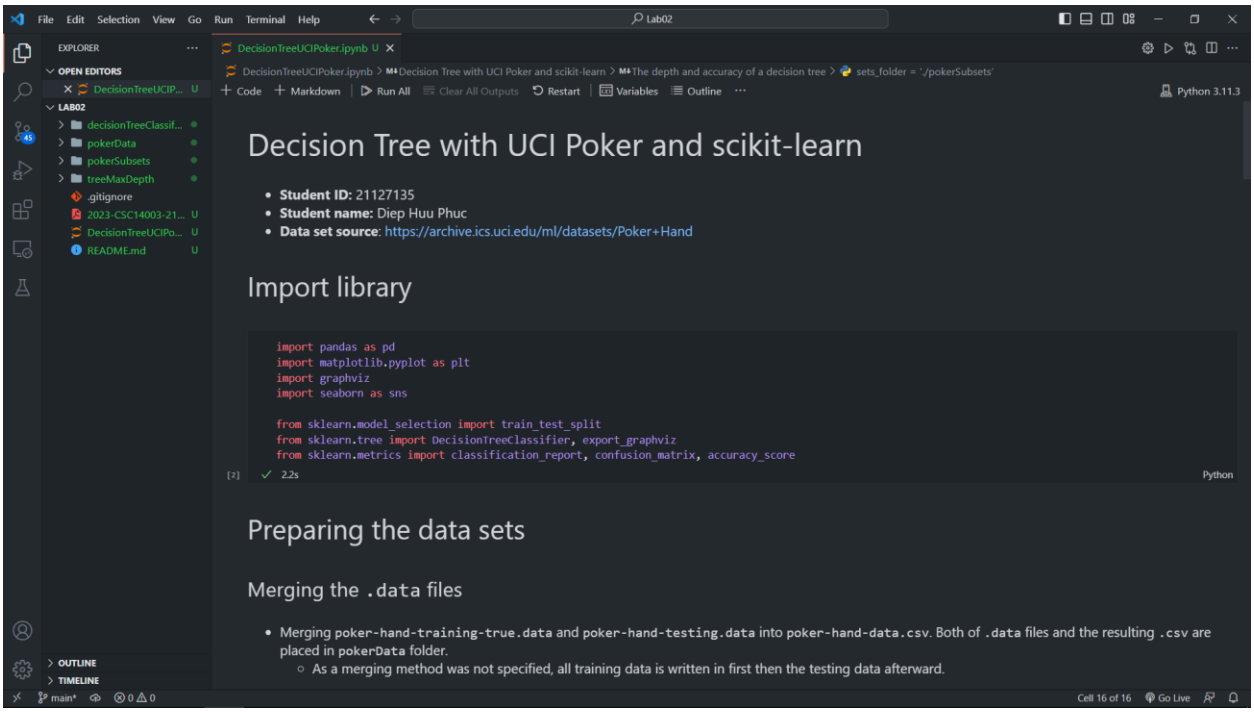
```
==== Max Depth None with Train/Test Ratio of 80/20 ====
Accuracy score: 0.6405937503048751
==== Max Depth 2 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.5061462814996927
==== Max Depth 3 with Train/Test Ratio of 80/20 ====
Accuracy score: 0.508965766187647
==== Max Depth 4 with Train/Test 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 the accuracy improved steadily as we dived deeper. When **max_depth** constraint was lifted, i.e., $max\_depth = None$, a drastic gain in accuracy is obtained.

5. IPython Notebook

- Make sure Import Library is run at least one.
- The notebook is written in the 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.



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

The depth and accuracy of a decision tree

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