OSS Project #2

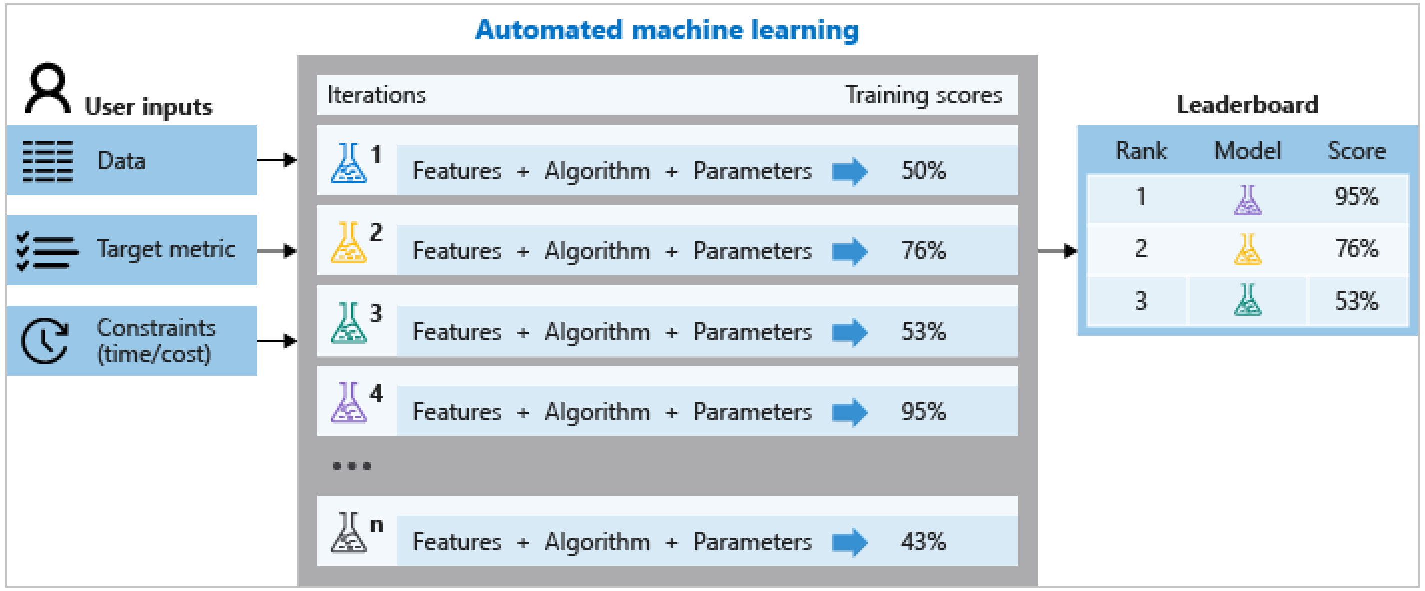
Automated Binary Classification

Prof. Do-Guk Kim

dgkim@inha.ac.kr

Automated Machine Learning (AutoML)

- AutoML is the process of automating the time-consuming, iterative tasks of machine learning model development
- In this project, students will implement simple script about AutoML



Automated machine learning

User inputs

- Data
- Target metric
- Constraints (time/cost)

Iterations	Training scores
1 Features + Algorithm + Parameters	50%
2 Features + Algorithm + Parameters	76%
3 Features + Algorithm + Parameters	53%
4 Features + Algorithm + Parameters	95%
...	
n Features + Algorithm + Parameters	43%

Leaderboard

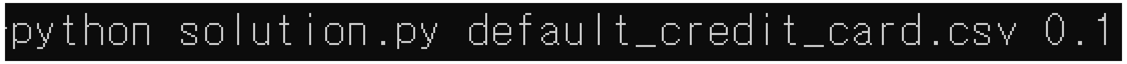
Rank	Model	Score
1	⚗	95%
2	⚗	76%
3	⚗	53%

Image from: https://learn.microsoft.com/en-us/azure/machine-learning/concept-automated-ml

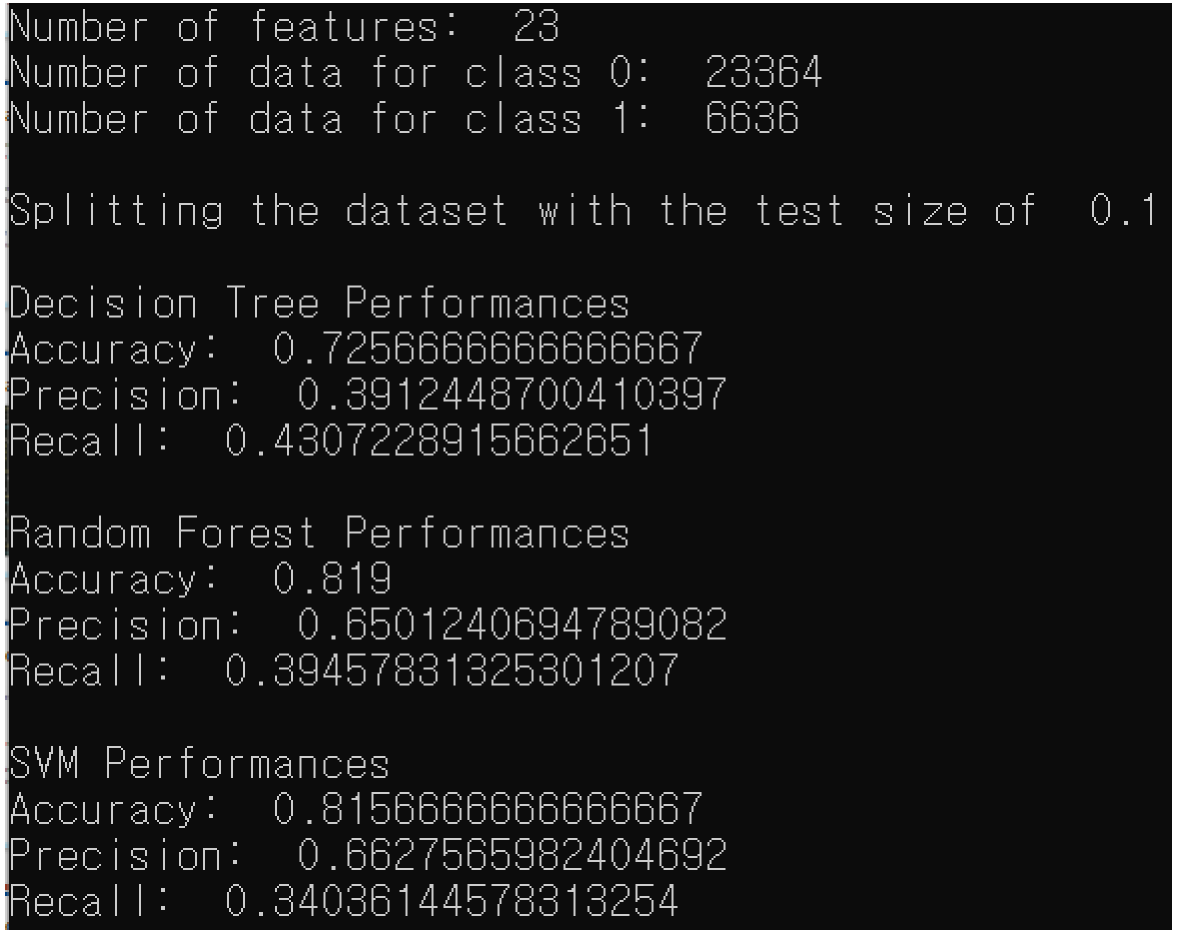
Project Goals

- The implemented script should automatically load data, split train dataset and test dataset, train various models and report performances
- Usage : **python template.py DATA_CSV_PATH TEST_SIZE**
  - TEST_SIZE in the form of ratio
- In this project, datasets must satisfy following requirements
  ① Binary classification
  ② All values are numeric
  ③ No missing values
  ④ The name of the label column is “target”

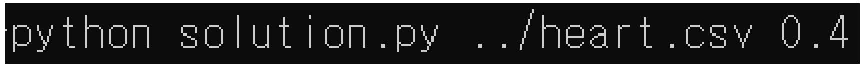
Example Commands & Results



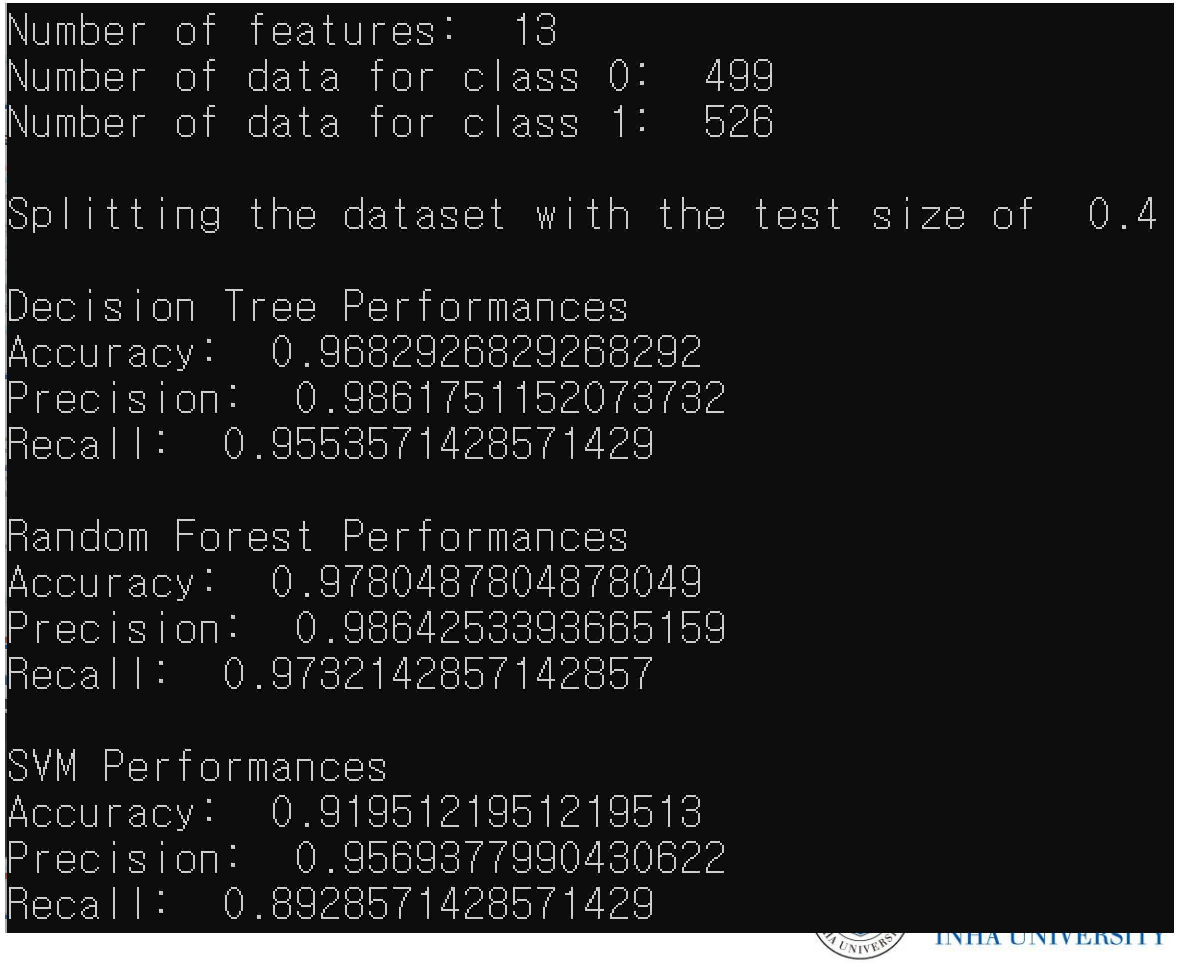
```
python solution.py default_credit_card.csv 0.1
```



```
Number of features:  23
Number of data for class 0:  23364
Number of data for class 1:  6636

Splitting the dataset with the test size of  0.1

Decision Tree Performances
Accuracy:  0.7256666666666667
Precision:  0.39124487004103967
Recall:  0.43072289156626551

Random Forest Performances
Accuracy:  0.819
Precision:  0.6501240694789082
Recall:  0.39457831325301207

SVM Performances
Accuracy:  0.8156666666666667
Precision:  0.6627565982404692
Recall:  0.34036144578313254
```



```
python solution.py ../heart.csv 0.4
```



```
Number of features:  13
Number of data for class 0:  499
Number of data for class 1:  526

Splitting the dataset with the test size of  0.4

Decision Tree Performances
Accuracy:  0.9682926829268292
Precision:  0.9861751152073732
Recall:  0.9553571428571429

Random Forest Performances
Accuracy:  0.9780487804878049
Precision:  0.9864253393665159
Recall:  0.9732142857142857

SVM Performances
Accuracy:  0.9195121951219513
Precision:  0.9569377990430622
Recall:  0.8928571428571429
```

Project Requirements

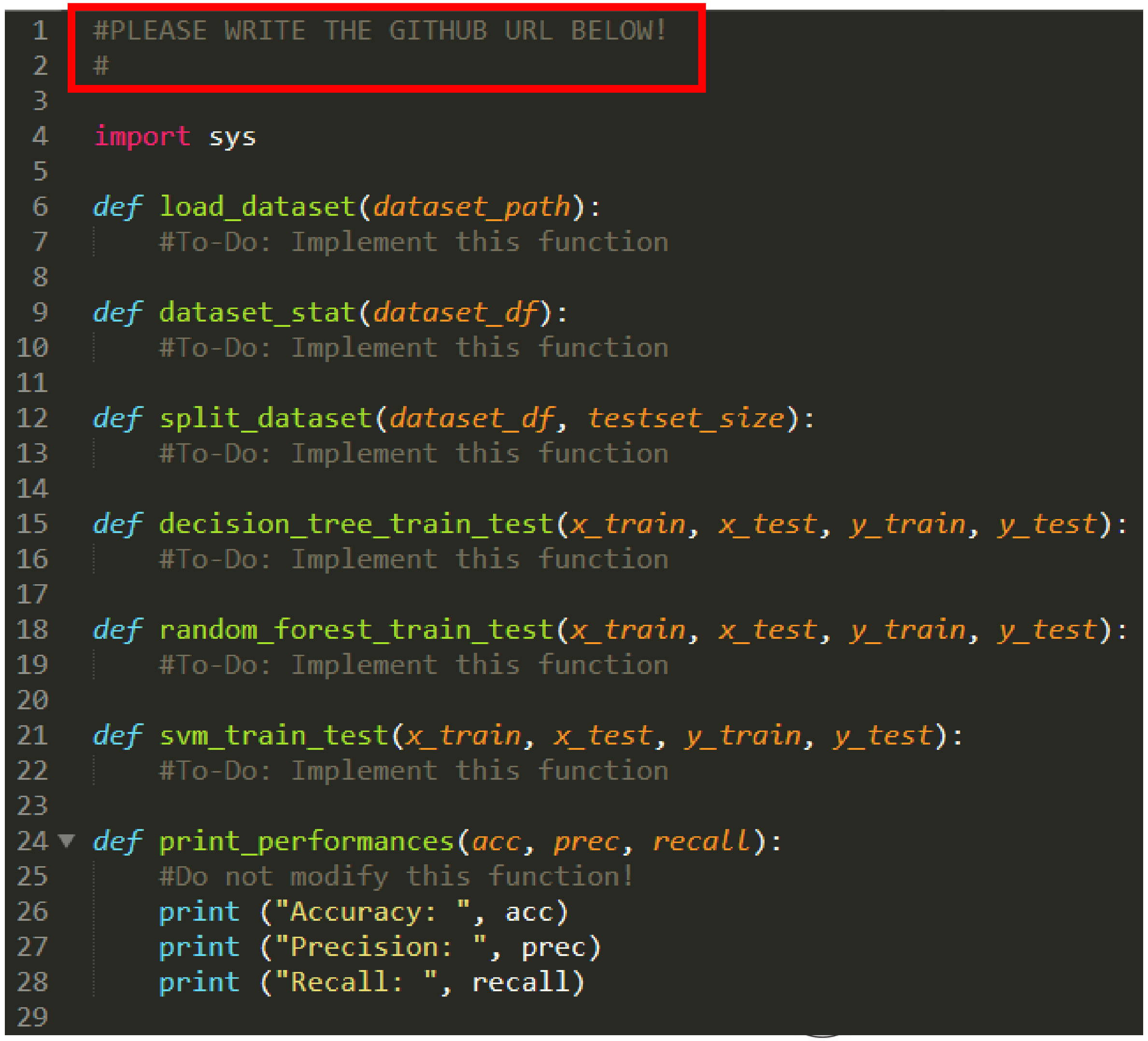
- All requirements and corresponding scores are represented in the below
  - Upload the completed code on the GitHub (10 points)
  - Complete the function **load_dataset** (10 points)
  - Complete the function **dataset_stat** (15 points)
  - Complete the function **split_dataset** (20 points)
  - Complete the function **decision_tree_train_test** (15 points)
  - Complete the function **random_forest_train_test** (15 points)
  - Complete the function **svm_train_test** (15 points)

  - A total of 100 points

Project template

- The template for the code is provided on the I-Class and you must implement functions in the template and submit the completed code

- Please upload the completed code on your personal GitHub, and write the URL as the comment in the second line of the script!
    - If you missed writing URL or the script is not on the written URL, you cannot acquire **10 points about GitHub uploading**



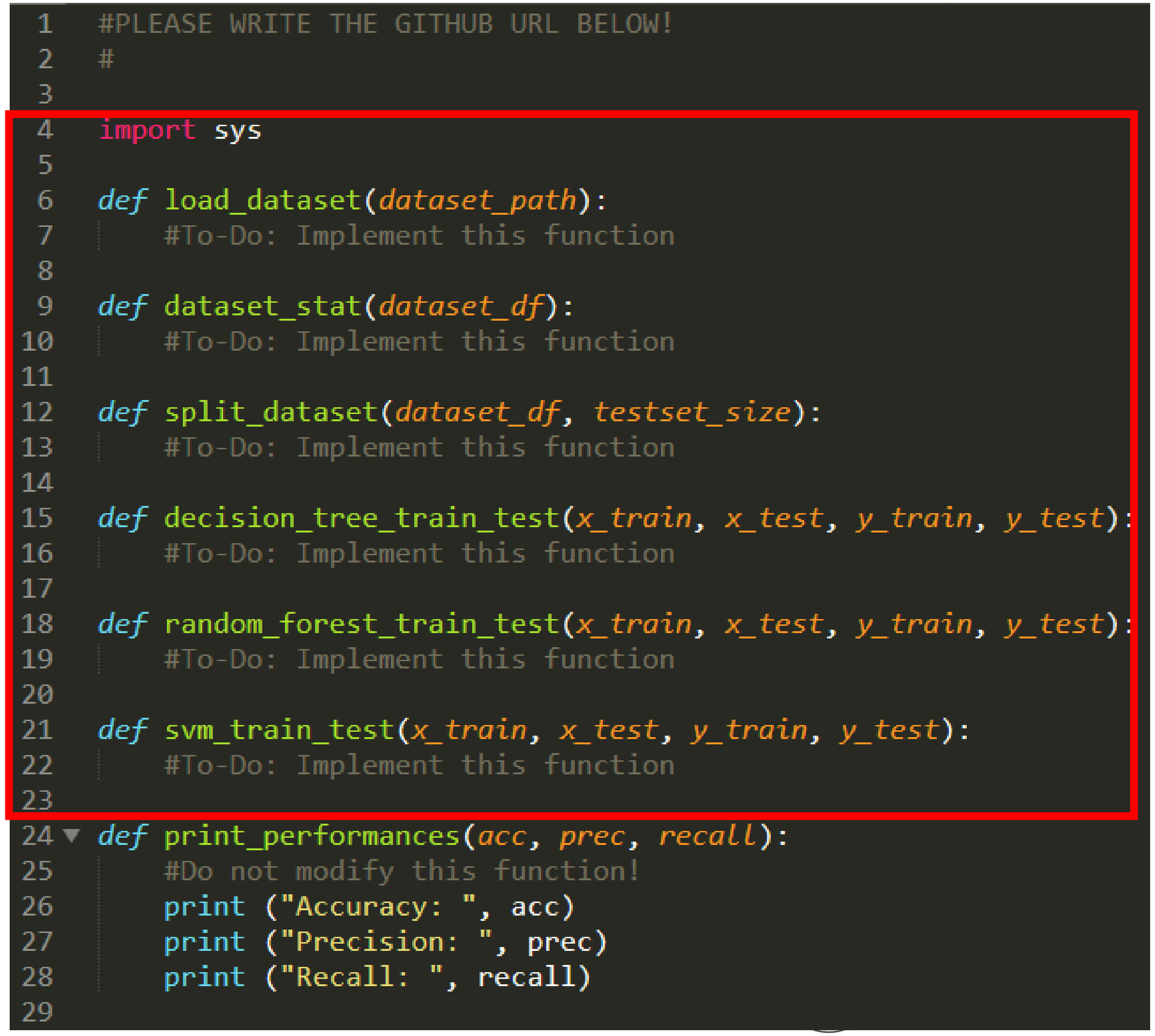
```
1   #PLEASE WRITE THE GITHUB URL BELOW!
2   #
3
4   import sys
5
6   def load_dataset(dataset_path):
7       #To-Do: Implement this function
8
9   def dataset_stat(dataset_df):
10      #To-Do: Implement this function
11
12  def split_dataset(dataset_df, testset_size):
13      #To-Do: Implement this function
14
15  def decision_tree_train_test(x_train, x_test, y_train, y_test):
16      #To-Do: Implement this function
17
18  def random_forest_train_test(x_train, x_test, y_train, y_test):
19      #To-Do: Implement this function
20
21  def svm_train_test(x_train, x_test, y_train, y_test):
22      #To-Do: Implement this function
23
24  def print_performances(acc, prec, recall):
25      #Do not modify this function!
26      print ("Accuracy: ", acc)
27      print ("Precision: ", prec)
28      print ("Recall: ", recall)
29
```

Project template

- You can import additional modules that you need to implement each function

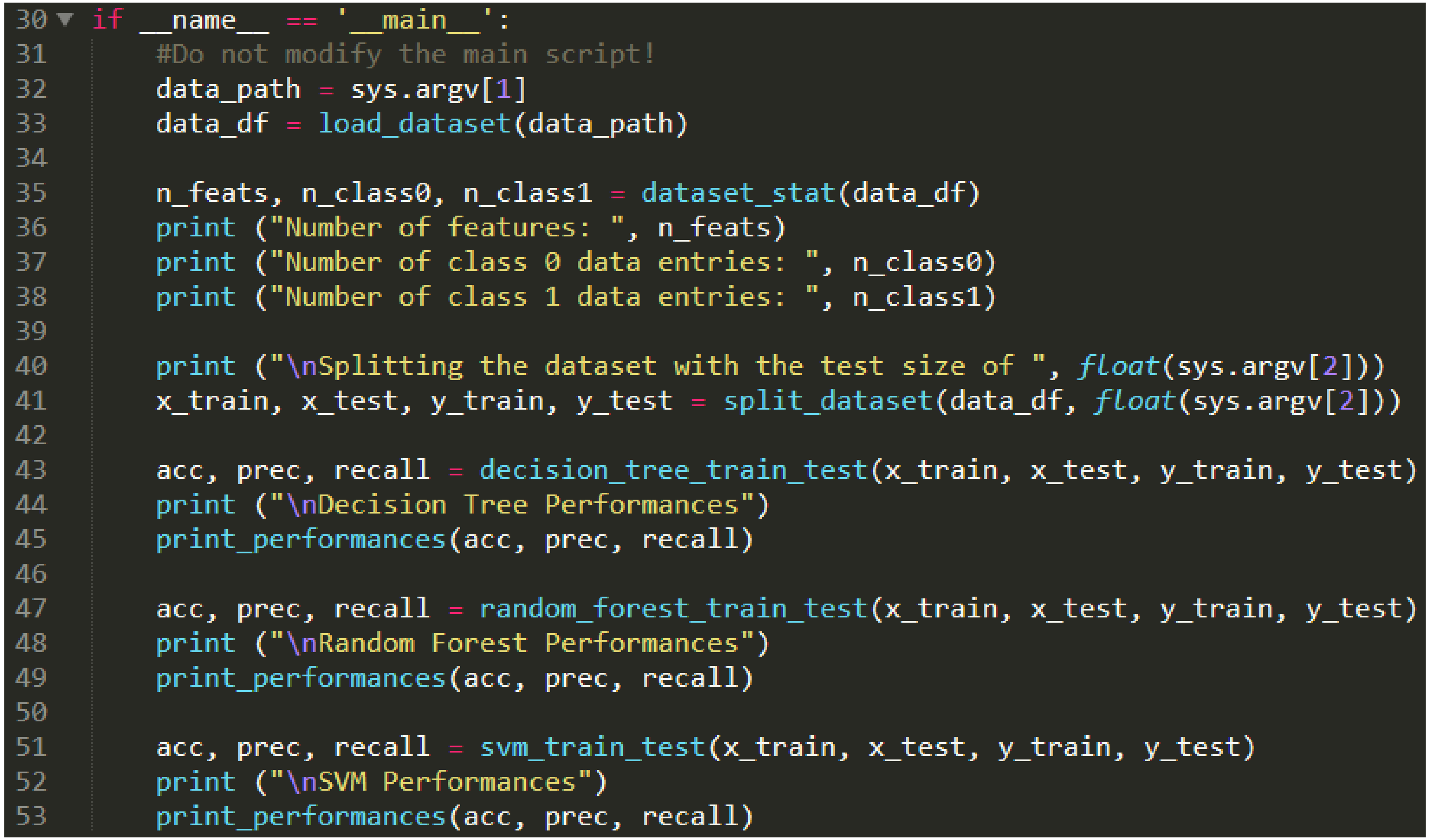
- Do not modify the function header (function name and parameter names)

- Please implement within the functions with “To-Do” comments. Do not edit other function and scripts



```python
#PLEASE WRITE THE GITHUB URL BELOW!
#

import sys

def load_dataset(dataset_path):
	#To-Do: Implement this function

def dataset_stat(dataset_df):
	#To-Do: Implement this function

def split_dataset(dataset_df, testset_size):
	#To-Do: Implement this function

def decision_tree_train_test(x_train, x_test, y_train, y_test):
	#To-Do: Implement this function

def random_forest_train_test(x_train, x_test, y_train, y_test):
	#To-Do: Implement this function

def svm_train_test(x_train, x_test, y_train, y_test):
	#To-Do: Implement this function

def print_performances(acc, prec, recall):
	#Do not modify this function!
	print ("Accuracy: ", acc)
	print ("Precision: ", prec)
	print ("Recall: ", recall)

```

Project template



```python
if __name__ == '__main__':
    #Do not modify the main script!
    data_path = sys.argv[1]
    data_df = load_dataset(data_path)

    n_feats, n_class0, n_class1 = dataset_stat(data_df)
    print ("Number of features: ", n_feats)
    print ("Number of class 0 data entries: ", n_class0)
    print ("Number of class 1 data entries: ", n_class1)

    print ("\nSplitting the dataset with the test size of ", float(sys.argv[2]))
    x_train, x_test, y_train, y_test = split_dataset(data_df, float(sys.argv[2]))

    acc, prec, recall = decision_tree_train_test(x_train, x_test, y_train, y_test)
    print ("\nDecision Tree Performances")
    print_performances(acc, prec, recall)

    acc, prec, recall = random_forest_train_test(x_train, x_test, y_train, y_test)
    print ("\nRandom Forest Performances")
    print_performances(acc, prec, recall)

    acc, prec, recall = svm_train_test(x_train, x_test, y_train, y_test)
    print ("\nSVM Performances")
    print_performances(acc, prec, recall)
```

Function Descriptions

- load_dataset
  - Load the csv file at the given path into the pandas DataFrame and return the DataFrame

- dataset_stat
  - For the given DataFrame, return the following statistical analysis results in order
    - Number of features
    - Number of data for class 0
    - Number of data for class 1

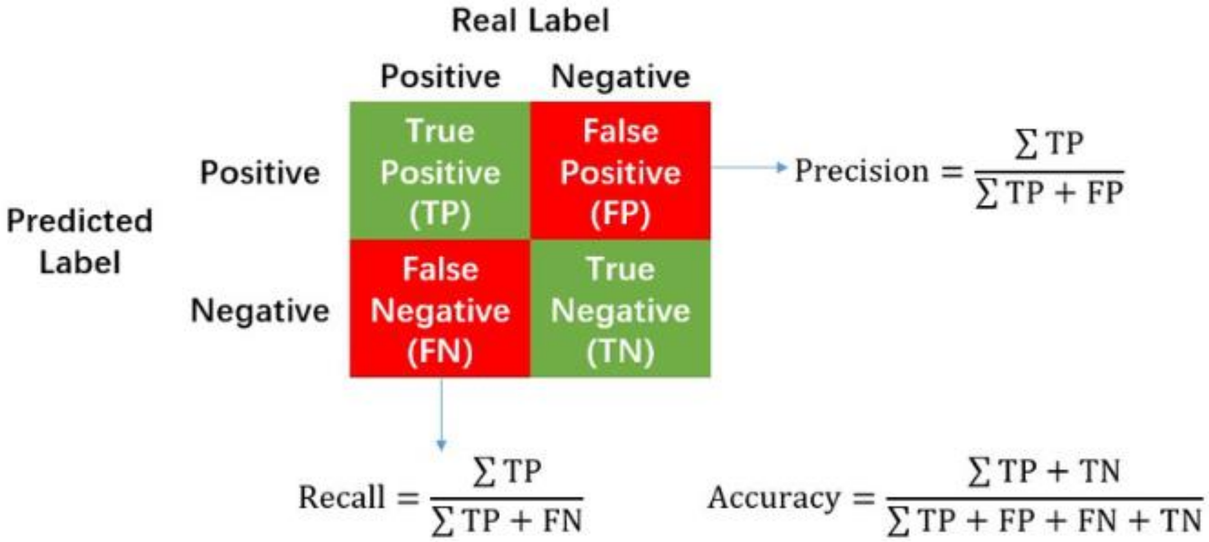
Function Descriptions

- split_dataset
  - Splitting the given DataFrame and return **train data, test data, train label, and test label** in order
  - You must split the data using the given test size

- decision_tree_train_test
  - Using the given train dataset, train the decision tree model
    - You can implement with default arguments
  - After the training, evaluate the performances of the model using the given test dataset
  - Return three performance metrics (accuracy, precision, recall) in order

Function Descriptions

- As described in the below figure, the **precision** is defined as **tp / (tp + fp)** in the confusion matrix. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative
- The **recall** is defined as **tp / (tp + fn)**. The recall is intuitively the ability of the classifier to find all the positive samples
- Accuracy is not a good metric for the imbalanced dataset, therefore we analyze the model performance with additional metrics



		Real Label	
		Positive	Negative
Predicted Label	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

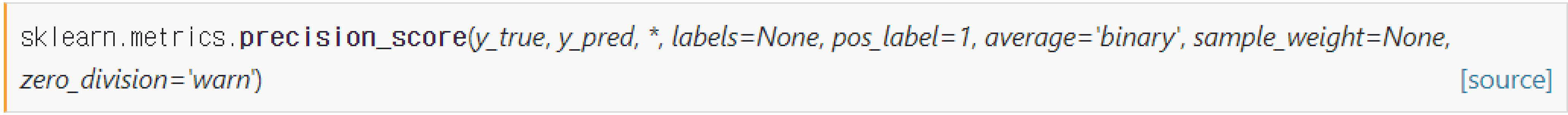
$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$$

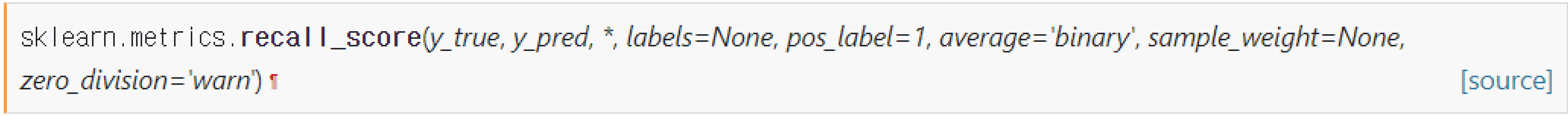
$$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Function Descriptions

- Please refer below sklearn functions



```
sklearn.metrics.precision_score(y_true, y_pred, *, labels=None, pos_label=1, average='binary', sample_weight=None,
zero_division='warn')                                                                                    [source]
```



```
sklearn.metrics.recall_score(y_true, y_pred, *, labels=None, pos_label=1, average='binary', sample_weight=None,
zero_division='warn') ¶                                                                                  [source]
```

Function Descriptions

- random_forest_train_test
  - Using the given train dataset, train the random forest model
    - You can implement with default arguments
  - After the training, evaluate the performances of the model using the given test dataset
  - Return three performance metrics (accuracy, precision, recall) in order
- svm_train_test
  - Using the given train dataset, train **the pipeline consists of a standard scaler and SVM**
    - You can implement with default arguments
  - After the training, evaluate the performances of the model using the given test dataset
  - Return three performance metrics (accuracy, precision, recall) in order

Submission

- Submission requires only a completed **template.py** file
  - Don’t forget to write GitHub URL and upload the template.py file on the GitHub
- Please submit your file on the I-Class
- Due date is 11/27(Sun) 23:59
- TA will verify your submissions using the another auto-script and copy checking tools

- If you have any questions, don’t hesitate to send an e-mail (dgkim@inha.ac.kr) or an I-Class message