ELEC-E7130 Assignment 8. Machine Learning

Markus Peuhkuri Tran Thien Thi
César Iván Olvera Espinosa Yu Fu

Prerequisites

1. At this point, students would be familiar with handling dataframes and arrays in Python or R including how to leverage the libraries available.

   Python is especially recommended as the course staff can best support solutions in Python skeleton codes which are provided in some tasks to give little guidance, but they are optional to use. There are many ways to get the desired result, feel free to use your method.

2. Students should have a basic understanding of Machine Learning. If you are not familiar with Machine Learning, you could refer to the supporting materials for more information:

   a. Machine Learning theory
   b. Traffic classification programming example

Learning outcomes

At the end of this assignment, students should be able to

1. Understand the overall ML traffic classification process before solving tasks.
2. Perform classification tasks with different ML algorithms.
3. Use different performance metrics to evaluate the model.
4. Select and save the best model for future datasets.
5. Load already existing models, which can be helpful in the future when models need to be saved.
6. Reduce the number of features to get more efficient learning.
7. Recognize the utility of stationary time series and supervised learning datasets.

Introduction

During the course, students have learned everything related to network traffic measurements including tools and different parameters, data analysis and

visualization using different techniques, as well as distribution and sampling applications. In addition, it would be beneficial to understand some basic concepts of pre-processing, training, evaluation for ML traffic classification cases. Thereby, the last assignment consists of five tasks as follows:

- Task 1: Traffic classification
- Task 2: Loading pre-defined model and evaluation
- Task 3: Feature selection for classification purposes
- Task 4: Making non-stationary data into stationary
- Task 5: Making time series data into supervised problem

> **Notes:** For this assignment, the files can be found in the directory `/work/courses/unix/T/ELEC/E7130/general/ml-data` or using `MLDATA` environment variable as a path if you have sourced the `use.sh` file.

Please read all instructions before starting because it is helpful to identify common work. Besides that, review the Python skeleton codes to better understand how to develop every task step by step and find out the documentation about the recommended functions for every task.

> **TIPS:**
>
> - Some functions require a dataframe as input, some functions require arrays as input, while for some functions, the type of input does not matter.
> - In Python, you can transform dataframe into array with `to_numpy()/values` and you can transform array into dataframe with `DataFrame()`.
> - In R, you could use `data.matrix()` and `as.data.frame()`, respectively (there are many other functions to use too).
> - Programming overall can be time-consuming and may require trial and errors, use some additional prints in the middle of code to see what is going on or if something looks as desired.

Task 1: Traffic classification

The first task is about training some ML models with different algorithms to predict the traffic classification, analyzing their performance through the accuracy and k-Fold Crossvalidation, and saving the most suitable ML model for future applications.

Download the file `combined_mix.csv` which contains different kind of Internet Traffic.

> **Note:** The dataset is already pre-processed as the task 3 "Data pre-processing for ML purposes" in the previous assignment, and labeled as follows (the label is defined in the last column as `target`):

- OneDrive = `0`
- Dailymotion = `1`
- Twitter = `2`
- Facebook = `3`
- The Guardian = `4`
- Vimeo = `5`
- Web browsing = `6`
- YouTube = `7`

Develop a code to fulfill the following steps. Furthermore, you can use skeleton code `skeleton_ml_1.py` to solve task.

1. Load CSV file.
2. Split the data set into training set (80%) and test set (20%).
3. Define the features and targets both sets (training and test).
4. Train different ML models with the following algorithms:
   - *LinearSVC* algorithm
   - *Naive Bayes* algorithm
   - *KNN* algorithm
   - *Decision Tree Classifier* algorithm
5. Perform classification (prediction) on an array of test vectors X (features) for each model.
6. Plot the distribution of the targets comparing between the test set (real data) and prediction set (each model)
7. Evaluate the performance of each by calculating
   - the *accuracy score* for each model
   - *k-Fold Crossvalidation* with `k=10` splits and compare their average score (using the whole data set)
8. Compare the results, select the most suitable model for the dataset, and save it. Why did you choose this model?

Answer appropriately the next points:

- What are the differences between accuracy score and k-Fold Cross-validation?

  Tips:

  - Useful Python functions could be `fit()`, `predict()`, `accuracy_score()`, `cross_val_score()`, `sns.kdeplot()`, `sns.displot()`, `sns.distplot()` .
  - The desired scores in this task should be usually somewhere *around 25-80% in the cases*.

Report, task 1

- Train, perform classification and evaluate each model successfully.
- Plots for the distribution of the targets.
- Answer the questions appropriately.

Task 2: Loading pre-defined model and evaluation

In this task, the purpose is to load some ML models which are already trained by different algorithms to predict a traffic classification according to the features, and learn different ways to evaluate ML models to determine the most suitable model.

Download the dataset contained in the file `test_dataset.csv` which is intended to be used as a *test set* and contains already pre-processed flow instances which are labeled as Youtube (`=1`) or Web browsing (`=0`).

Download ML models with the `*.pkl` extension to load and evaluate their performance for each one:

- `LinearSVC_model_for_ml_2.pkl`
- `GaussianNB_model_for_ml_2.pkl`
- `KNeighborsClassifier_model_for_ml_2.pkl`
- `DecisionTreeClassifier_for_ml_2.pkl`

Compile a code to perform the points below. Moreover, you can use skeleton code `skeleton_ml_2.py` to solve task.

1. Load CSV file as *test set*.
2. Load the complete ML models
3. Define the features and targets using whole set
4. Perform classification (prediction) of the whole set
5. Evaluate the models with the next performance metrics:
   - Confusion matrix
   - Accuracy score
   - Precision score
   - Recall score
   - F1 score
6. Plot the distribution of the targets comparing the test set (real data) and prediction sets (each model)

Compare the results and answer the following:

- Explain what information is provided by each performance metric.
- Which model seems to be most suitable for the dataset? Explain your reasons.
- Is relying solely on accuracy score sufficient when evaluating model performance? Please explain the reasons and share your insights.

  Tips:

  - Useful Python functions could be `load()` from `pickle` library, `confusion_matrix()`, `accuracy_score()`, `precision_score()`, `recall_score()`, `f1_score()`, `sns.displot()`, `sns.distplot()`, `sns.kdeplot()`.

- The desired accuracy in this case should be *around 0.65 and 0.85*.

Report, task 2

- Load, perform classification and evaluate each model successfully.
- Plots for the distribution of the targets.
- Answer the questions appropriately.

Task 3: Feature selection for classification purposes

Once learned how to train, evaluate, choose, save and load the most suitable models, the third task aims to know the performance in different scenarios mainly by *selecting and reducing the number of features* of the data set (training and test) for the ML models or even *using different ML algorithms* due to their time processing for each one (in this case *KNN algorithm* and *Decision Tree Classifier algorithm*).

The file `combined_mix_nofs.csv` contains a dataset that is mostly preprocessed already but there has not been performed any feature selection yet. At the moment it contains 45 columns (44 features and 1 target). The aim is now to reduce the number of features to 5 and compare its performance results.

Complete the following tasks (you can use the provided skeleton code `skeleton_ml_3.py` as a guide). Seeding is good idea for keeping the results more comparable.

A. For KNN algorithm: - Using *five features*: 1. Load CSV file and define the features and the target 2. Divide the data set into training set (90%) and test set (10%) 3. Perform *feature selection* based on training data choosing 5 most relevant features. Apply them to both training set and test set. 4. Train a ML model with KNN algorithm 5. Evaluate its accuracy score and the running time. - Using *all features*, perform another ML model following the same steps above except step 3, that is, without *feature selection*. Evaluate its accuracy score and running time.

B. For Decision Tree Classification algorithm: - Execute the same idea, as the previous algorithm, using both *five features* and *all features* to evaluate their accuracy score and running time.

Compare the performance of the models with and without feature selection using both algorithms and **draw your conclusions.**

Tips:

- Useful Python function could be `train_test_split()`, `SelectKBest()`, `fit()`, `transform()`. You could use

`f_classif` as feature selection algorithm.
- Regarding running time, it can help the function `time.time()` from the library `time` running it directly with python. In case of working on jupyter notebook, it can help the line `%%time`.

Report, task 3

- Train, perform classification and evaluate its accuracy and running time successfully.
- Draw conclusions appropriately.

Task 4: Making non-stationary data into stationary

The task 4 consists of converting *non-stationary time* series data into *stationary* data observing the differences and importance in terms of Machine Learning.

Use the dataset contained in the file `bytes.csv` with Aalto Brute server's received bytes, received packets, transmitted bytes, and transmitted packets as time series data of each 1h interval. In this task, we will only look at the received bytes.

Develop a code to fulfill the next tasks. 1. Plot the received bytes (*non-stationary* data) 2. Make the received bytes into *stationary* (drop possible NaNs that may occur) and plot them.

Describe briefly their **difference and the usefulness of stationary time series data**.

Tips:

- Useful Python functions could be `diff()`.

Report, task 4

- Plots of both non-stationary and stationary data.
- Discussions about the difference and usefulness.

Task 5: Making time series data into supervised problem

The last task involves developing a function to transform a time series dataset into a supervised learning dataset.

Download the file `rtt.csv` that contains round trip times to distant website whose server is located in Hawaii, collected at 2-hour intervals as time series data.

Complete the following tasks.

1. Make time series into supervised form with window step of 1
2. Make time series into supervised form with window step of 3
3. Make time series into supervised form with window step of 5

   Note:

   Initially, the RTT time series data should look like this:

```
0    220
1    178
2    175
3    168
4    171
```

   The output of the code *in case of window step of 5* should produce something like the following (headers are optional)

```
     t-5    t-4    t-3    t-2    t-1    t
0    NaN    NaN    NaN    NaN    NaN  220
1    NaN    NaN    NaN    NaN  220.0  178
2    NaN    NaN    NaN  220.0  178.0  175
3    NaN    NaN  220.0  178.0  175.0  168
4    NaN  220.0  178.0  175.0  168.0  171
5  220.0  178.0  175.0  168.0  171.0  167
6  178.0  175.0  168.0  171.0  167.0  159
```

Discuss the next points: - Assuming the function is doing one-step-predictions, which column would be features and which column would be targets in all these cases? - What is the importance to convert into supervised learning?

   Tips:

   - Useful function include `shift()` for both Python and R.

Report, task 5

- A function to transform a time series dataset into a supervised learning dataset.
- The output with different window steps.
- Answer the questions appropriately.

Grading standard

To pass this course, you need to achieve at least 15 points in this assignment. And if you submit the assignment late, you can get a maximum of 15 points.

You can get up to 30 points for this assignment:

Task 1

- Train, perform classification and evaluate each model indicated. (4p)
- Plot the distribution of the targets (prediction). (1p)
- Select and save the best model. (1p)

Task 2 - Load, perform classification and evaluate the model appropriately. (4p) - Plot the distribution of the targets (prediction). (1p) - Answer appropriately the points. (4p)

Task 3 - Perform feature selection adequately. (1p) - Train, perform classification and evaluate its accuracy and running time each scenario. (5p) - Draw your conclusions comparing the performance of each scenario. (1p)

Task 4 - Convert non-stationary into stationary data. (1p) - Plot both non-stationary and stationary data. (2p) - Discuss about stationary time series data. (1p)

Task 5 - Develop the function. (3p) - Answer appropriately the points. (1p)

The quality of the report (bonus 2p)

The instruction of assignment

For the assignment, your submission must contain (Please don't contain original data in your submission):

- A zip file that includes your codes and scripts.
- A PDF file as your report.

Regarding the report, your report must have:

- A cover page indicating your name, student ID and your e-mail address.
- The report should include a description of measurements, a summary of the results and conclusions based on the results.
- An explanation of each problem, explain how you solved it and why you did it.