# EP2300 Project Tasks Report

Yuan Fan<yfan@kth.se>

October 10, 2018

## 1 Introduction

This project is to estimating the Service Level Agreements conformance of a VoD services using metrics on both client and server side. During this stage, I learned to write and test the code with useful package tools in python that involves data processing and machine learning. And the code were written and tested on *jupyter notebook*.
Task1 is a simple warm up about **Data Exploration** using python-based data exploration tools, while task2 is leading us to the basic estimating service metrics in machine learning concept. There are several sub-tasks in both tasks.

## 2 Task I Results

### 2.1 task1.1

This sub-task mainly evaluate the statistic value of different features of *.csv* file X and Y. The code includes useful functions in **pandas**. The Figure 1 show the result of the program:

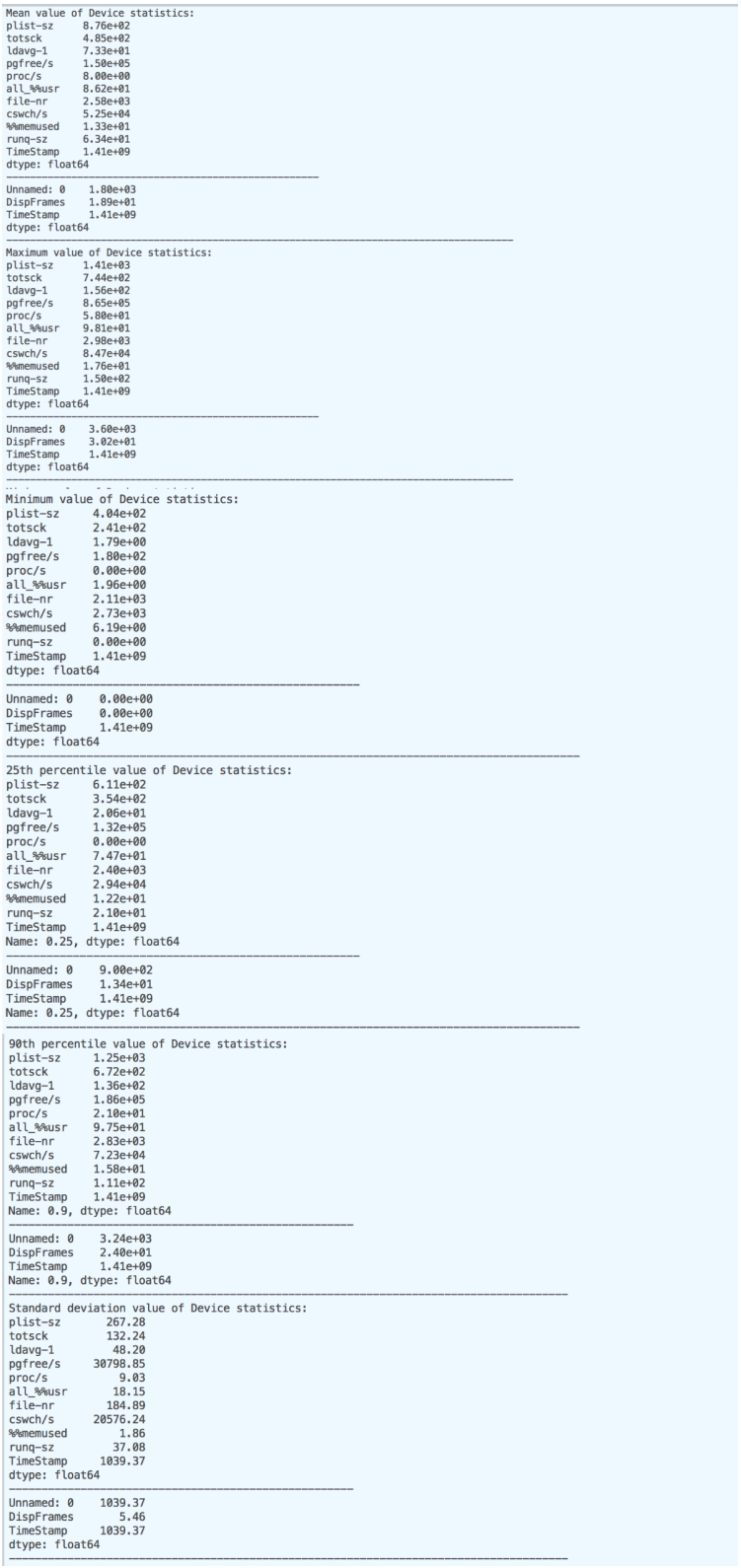


Figure 1: Statistic evaluation of X&Y

## 2.2 task1.2

Task requires us to finding the data volume between a certain range and the average number of one feature relies on another scope. The result shows in Figure 2:

```
The number of observations with CPU utilization smaller than 90% and memory utilization smaller than 50% is: 1114
The average number of used sockets for observations with less than 60000 context switches per seconds is 356.12
```

Figure 2: Quantities computations

## 2.3 task1.3

Applying the *matplotlib* is very important in this part. I used the **DataFrame** in pandas to generate some of the plots. The task requires the analytical approach and plot the figure in time series based, the box version, density plot and histograms plot for both memory usage and CPU utilization features. In this part, I also use the package *seaborn* to form a density plot in a simple, fast way. Following Figures3&4&5 are the outcomes:

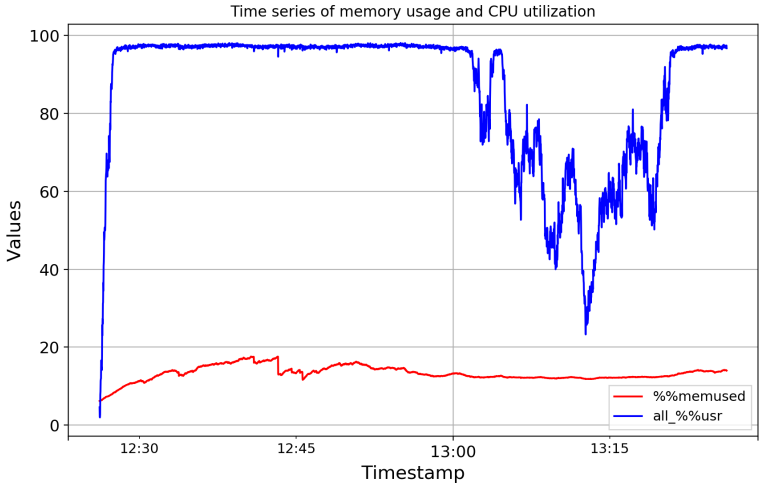


Figure 3: Time series plot

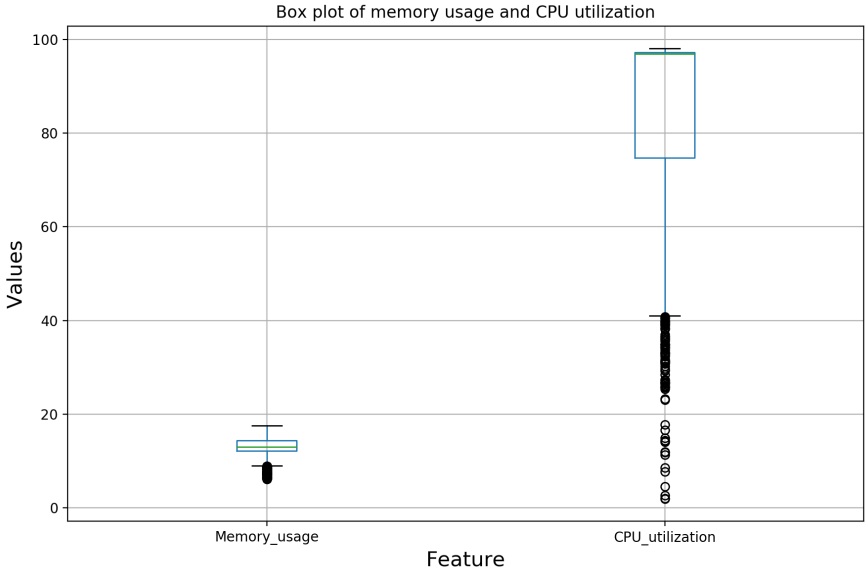


Figure 4: Box plot

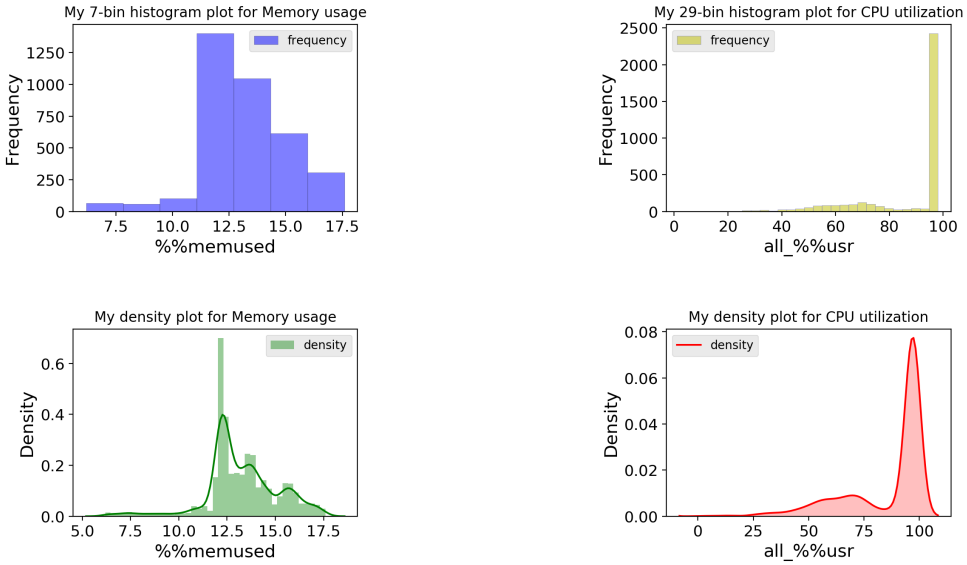


Figure 5: Density and Histograms plot

# 3 Task II Results

## 3.1 task2.1 Evaluate the Accuracy of Service Metric Estimation

In this section, I equipped myself with the basic concepts of machine learning(1) since this is new to me. The concept of training set and test set as well as the different regression evaluation indexes such as MSE, MAE and so on interests me. And my work is to follow steps to train a model to give the related parameters (functions involves set splits and prediction), analyze the accuracy of this model, also, draw the feature plots of them.
In the model training part, the results :

- `Coefficients` are `[-1.57e-02 1.66e-03 6.40e-03 -3.43e-06 -5.26e-03 8.54e-02 -3.42e-03 -7.79e-05 3.74e-01 -1.66e-02]`.
- The `offset` is `33.56`.

To evaluate the accuracy of model, **NMAE(normalized mean absolute error)** is defined in the instruction. What I did is to divided the functions into several parts in a stupid way and then combine them.
Overall,

- The estimation error the `Average of training set in naive method` is `18.95`
- `Normalized Mean Absolute Error(NMAE)` :`0.11(0.10505375967359024)`
- `Normalized Mean Absolute Error(NMAE) in naive method`:`0.27(0.2673974072428627)`

The time series plot of prediction in a naive method, model estimations and measurements as required are shown in Figure 6

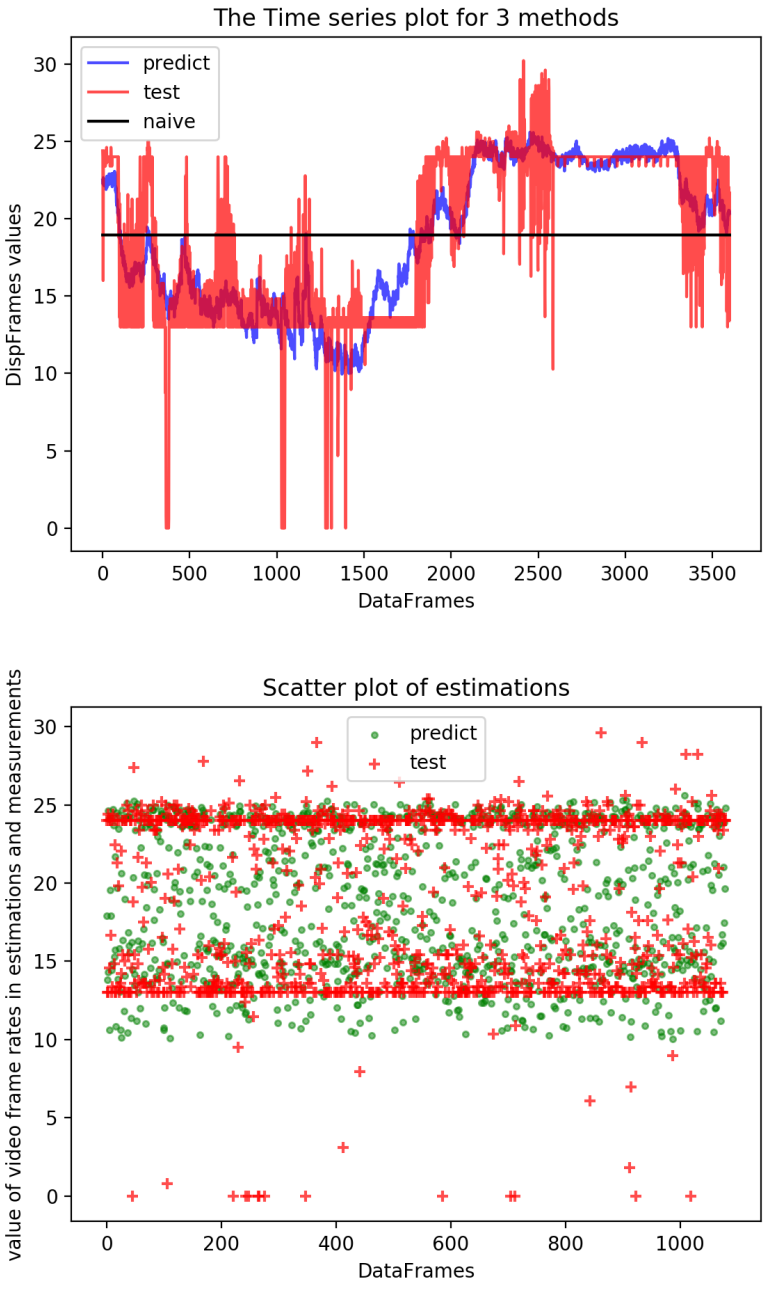


Figure 6: Time series plot and Scatter plot in task2

Figure 7& 8 show the density and histogram plot for Video Frame Rate and the prediction error in separate plots :

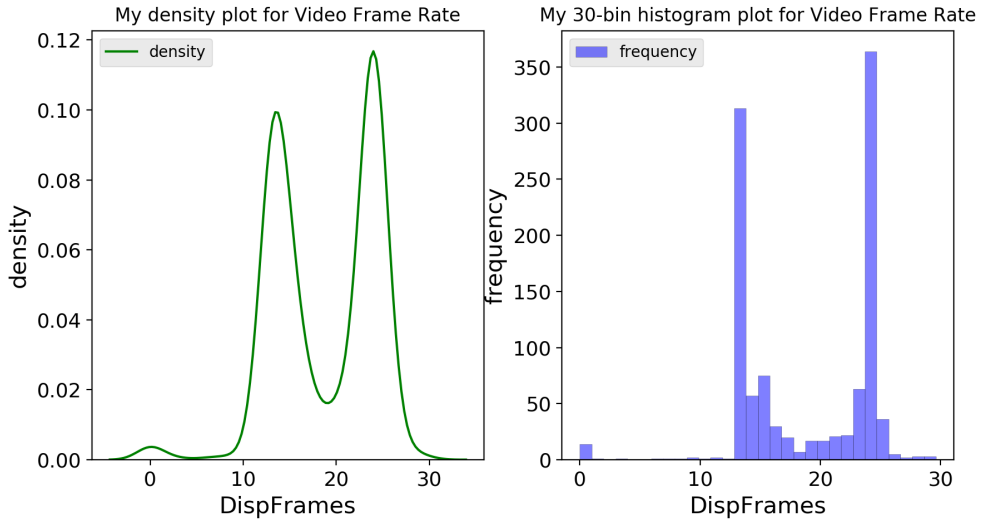


Figure 7: Density and Histogram plot in task2

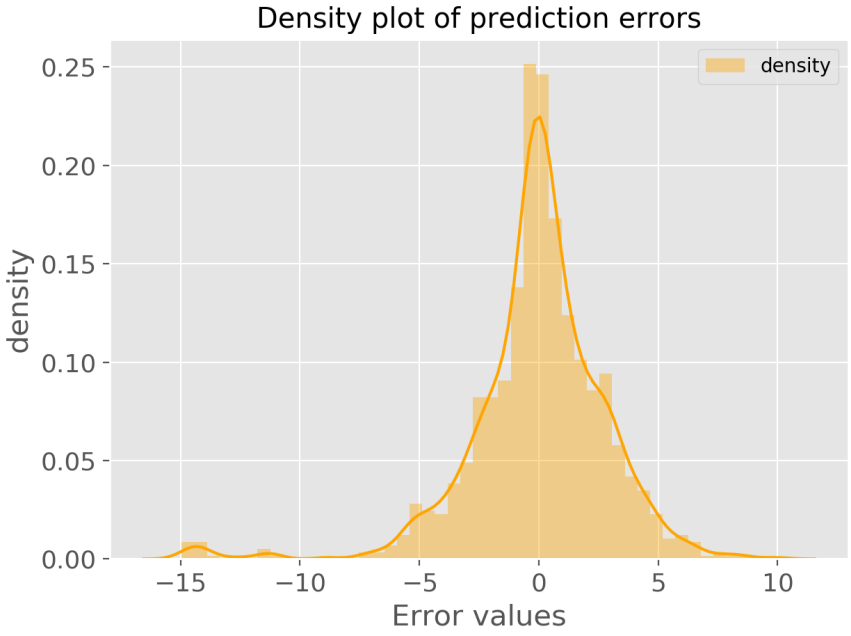


Figure 8: Density plot of prediction error

Based on the above figures, the conclusion has been drawn that the accuracy of this model training is relatively high and the prediction error plot is close to normal distri-

bution indicated the error range within +-5 dataframes are high.

### 3.2 task2.2 Study the Relationship between Estimation Accuracy and the Size of the Training Set

This part divide the training set observations into 6 subsets, each of them has their own NMAE towards 1080 test set. So after I form the model, the NMAE in different subset are shown as:

1. Normalized Mean Absolute Error(NMAE) for train size: 50 is : 0.11 (0.11173685334845512)
2. Normalized Mean Absolute Error(NMAE) for train size: 100 is : 0.11 (0.10911477760618228)
3. Normalized Mean Absolute Error(NMAE) for train size: 200 is : 0.11 (0.10807893264714397)
4. Normalized Mean Absolute Error(NMAE) for train size: 500 is : 0.11 (0.10636910508542098)
5. Normalized Mean Absolute Error(NMAE) for train size: 1000 is : 0.11 (0.10550005779900312)
6. Normalized Mean Absolute Error(NMAE) for train size: 2520 is : 0.11 (0.10505375967359017)

After performing the model in 50 times in order to train the model more accurate, the box plot of the different range of NMAE are produced as shown in Figure 9

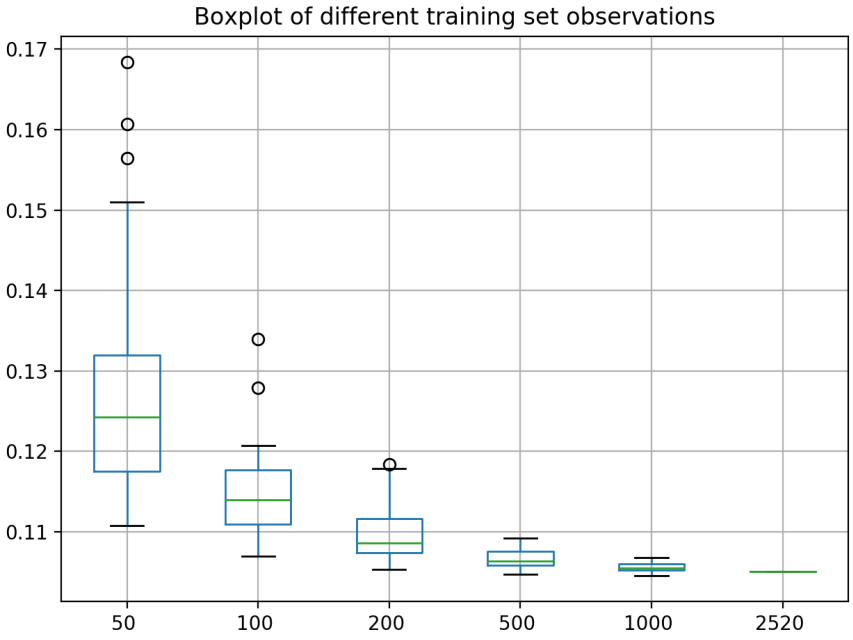


Figure 9: Box plot of different training set observations

As clearly shown in the result, the training set size is a key value when take the estimation error into consideration, when we use greater portion of training set, the accuracy increases.

# 4 Task III Results

## 4.1 task3.1 Build the logistic regression model

Estimating SLA conformance and violation is the main purpose of performing a logistic regression in machine learning, the output range of the hypothetical function of logistic regression is 0 1 in our case is that we need to pre-process the data in Y.csv according to the requirement of conformance SLA, and form a new column of output, read the data to the target ones. During this step, I used a class called `StandardScaler` the advantage of using this class is that you can save the parameters (mean, variance) in the training set directly using its object transformation test set data. In the model training part, the results :

- `Coefficients` are `[-0.581 -0.61 0.028 -0.113 -0.02 0.788 -0.217 -0.568 0.181 -0.801]`.
- The `offset` is `0.191`.

## 4.2 task3.2 Accuracy of the classifier C

To estimate the model accuracy, the project introduced a concept named classifier error, which defined by a certain equation. The variables can be extracted from the *confusion matrix*. In the field of machine learning, the confusion matrix is also called the probability table or the error matrix. It is a specific matrix used to visualize the performance of the algorithm. Each column in this matrix represents a predicted value, and each row represents the actual category.
As the Figure 10 plot the confusion matrix with its metrics, and the result of `classifier error is 0.08 (0.07685185185185184)`:

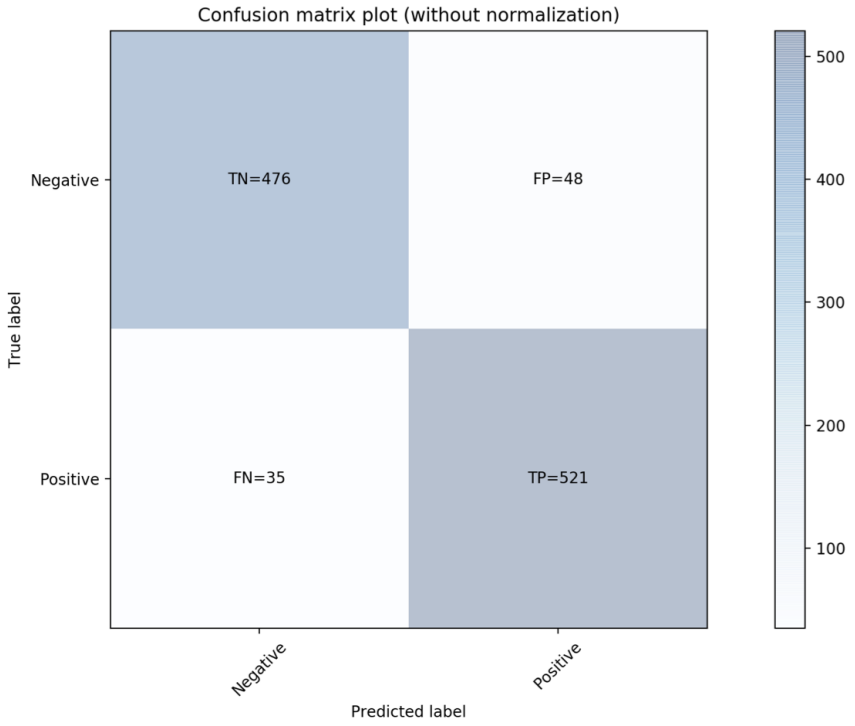


Figure 10: Confusion matrix plot

### 4.3 task3.3 Applying the naive method to compute classifier error

Similar to task 2, this part we need to use naive method to calculate the probability of Y values that conform with the SLA, and this method is only relies on Y values only, in the code, I performed the random test size number between 0 and 1, and recalculate the Y data under which conform with the probability I got from earlier.
The results show that:

- `The fraction of Y values that conform with the SLA is 0.531944444444444`.
- The `classification error for the naive classifier on the test set is 0.51 (0.5083333333333333)`.

It's pretty clear to see that the error has increased a lot once we use the naive method, thus the model has higher accuracy.

### 4.4 task3.4 Compute the classifier error based on SLA threshold

At this time, the SLA threshold was decided by Linear regression and its predict value, the goal is to compute the predict value and the test value to the data set that either 0 or 1. Then the data can be used in confusion matrix and numerate the classifier error. This operation can be treated as an extension of Lineat regression and my output of this step is:

```
The extended new classifier has classification error of :  0.08
(0.08148148148148149)
```

As a comparison to the traditional Logistic regression, this method brings a small differences in the classification error, the result is a little bit higher than the Logistic regression but not to much.

## 5 Task IV Results

The main target of this task is to reduce the number of device statistics of X that needed for estimating Y accurately. Since a set of size N has $2^N$ subsets, in our data set, the number of features are too large, so finding a minimal subset of X that predicts Y with the smallest error is quite challenging. In the instruction, we are provided with 3 methods and will be discussed individually in following sections:

### 5.1 Method 1 Optimal method

As introduced, this will bring better accuracy output when the data has small number of features. To find out all subsets X has, I use the function of iterator with subset length from 1 to 10. After the subsets are calculated, I modified the NMAE finding function learned from task2 to fit with each subset of train data then calculated the NMAE and put them in to a list. The produced histogram plot of these NMAEs are shown in Figure 11:

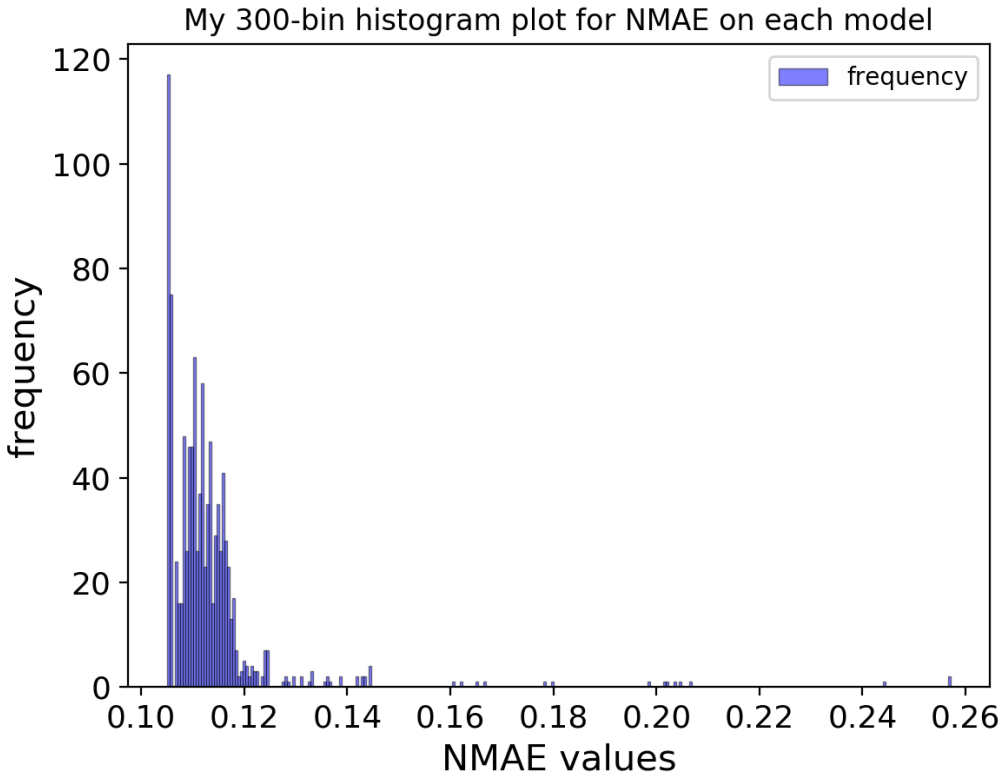


Figure 11: Histogram plot for NMAE on each model

Finding the smallest NMAE in the whole big sets takes the function of *index*, the result is :

```
The features subset model with smallest NMAE is :
['runq-sz' 'proc/s' 'cswch/s' 'all_%%usr' 'ldavg-1' 'totsck' 'pgfree/s'
'plist-sz' 'file-nr']
```

Figure 12: Smallest NMAE feature set

The biggest obstacle of this task showed up when I tried to plot the box plot for NMAE of models contain 1-10 number of features, I have the trouble to slice NMAE set in a functional way, so I did the naive method to split the sample. Figure 13 shows the output:

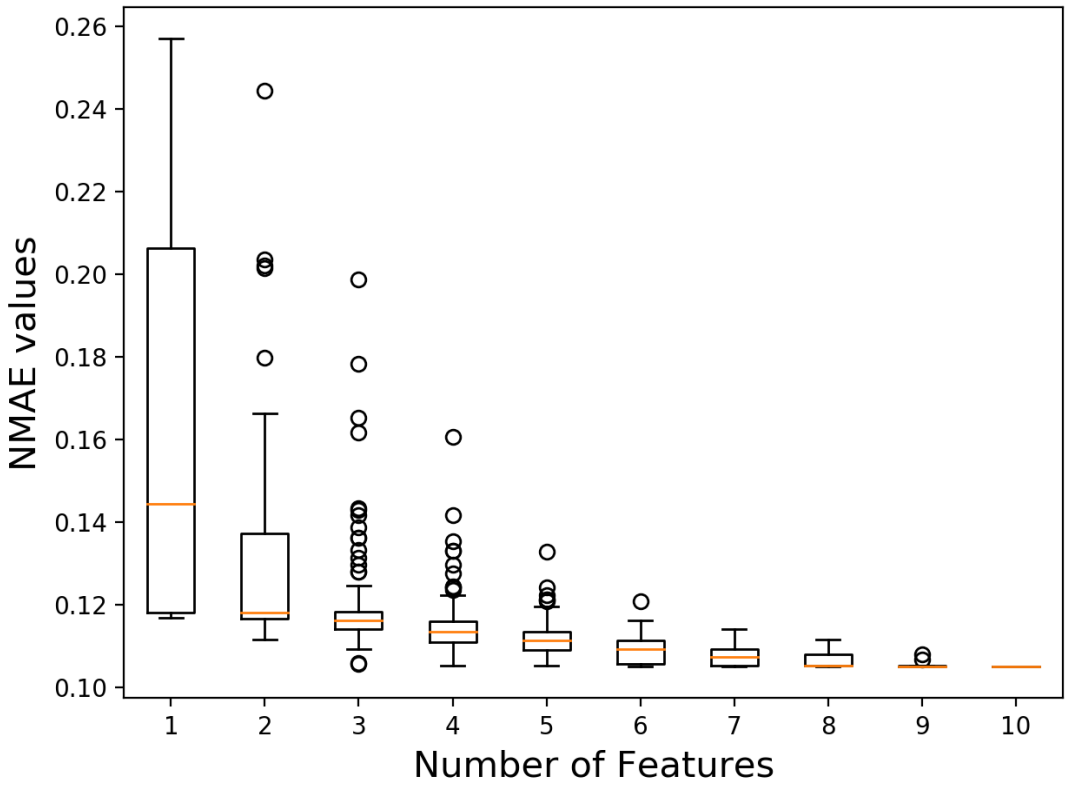


Figure 13: Box plot of NMAE subsets

## 5.2 Method 2 Heuristic method

This method is using linear uni-variate feature selection(2), according to the requirements, I apply a lambda function to calculate the correlation coefficients with train target and rank the feature after they are squared the output are:

| features | square_corr |
|---|---|
| plist-sz | 0.696309 |
| totsck | 0.694316 |
| ldavg-1 | 0.664014 |
| runq-sz | 0.647281 |
| file-nr | 0.576660 |
| cswch/s | 0.539877 |
| all_%%usr | 0.317872 |
| %%memused | 0.304389 |
| proc/s | 0.053881 |
| pgfree/s | 0.052447 |

Table 1: Rank of squared correlation coefficients

Then we are asked to plot the relationship between NMAE values and the number of features in train set as in Figure 14 and the heat map of correlation matrix contains both features for X train data and Y train data. At first, I overthink this and performed circulation of correlation coefficients, with the help of `pandas.concat()` function, it simplified to only one line code or two, result see in Figure 15

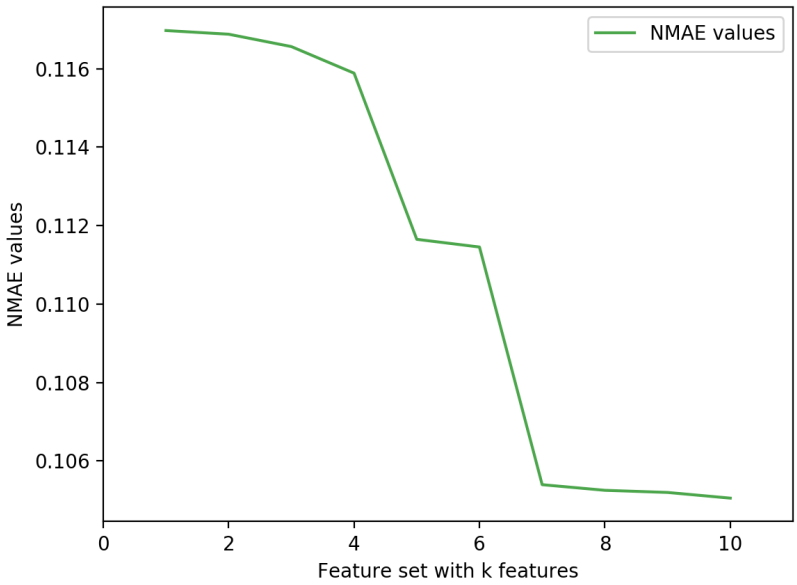


Figure 14: NMAE with k features

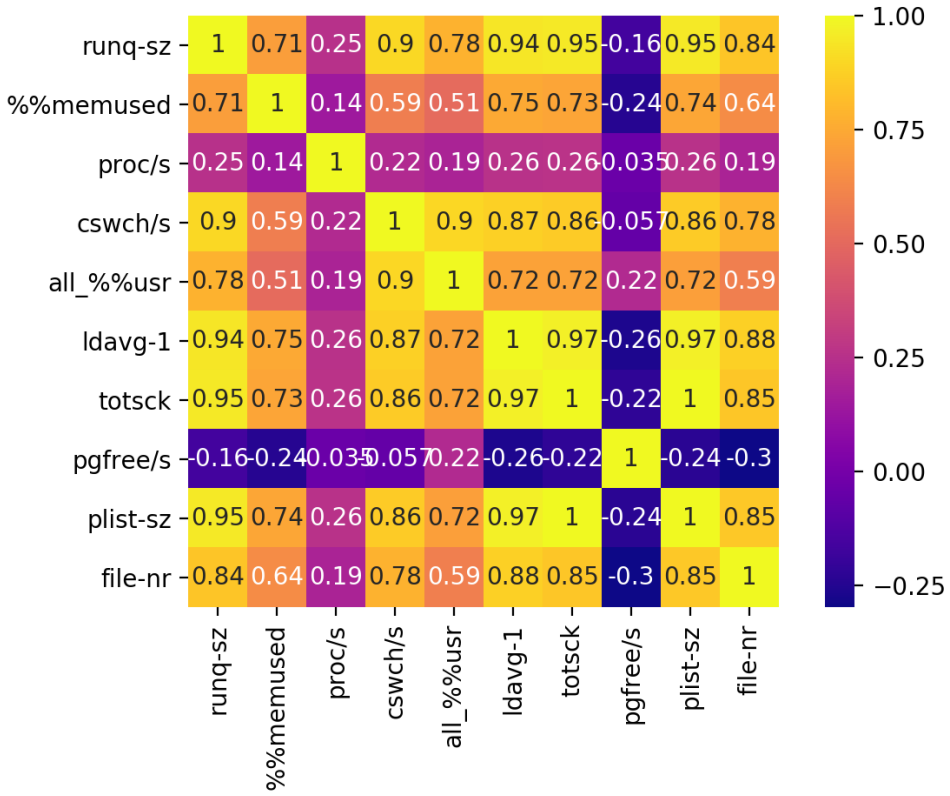


Figure 15: Heat map of different features

### 5.3 Method 3 Optional

This method taught us to perform the Lasso regression, which is a linear model that estimates sparse coefficients. Understanding the algorithm, I formed a list of alphas ranging from $10^{-1}$ all the way to $10^{3}$. Then used its feature to calculate the NMAE for each alpha and sum up the number of none zero value features and formed into one Dataframe. Then to find out the connection between alphas and the number of none zero value features shown in Figure 16

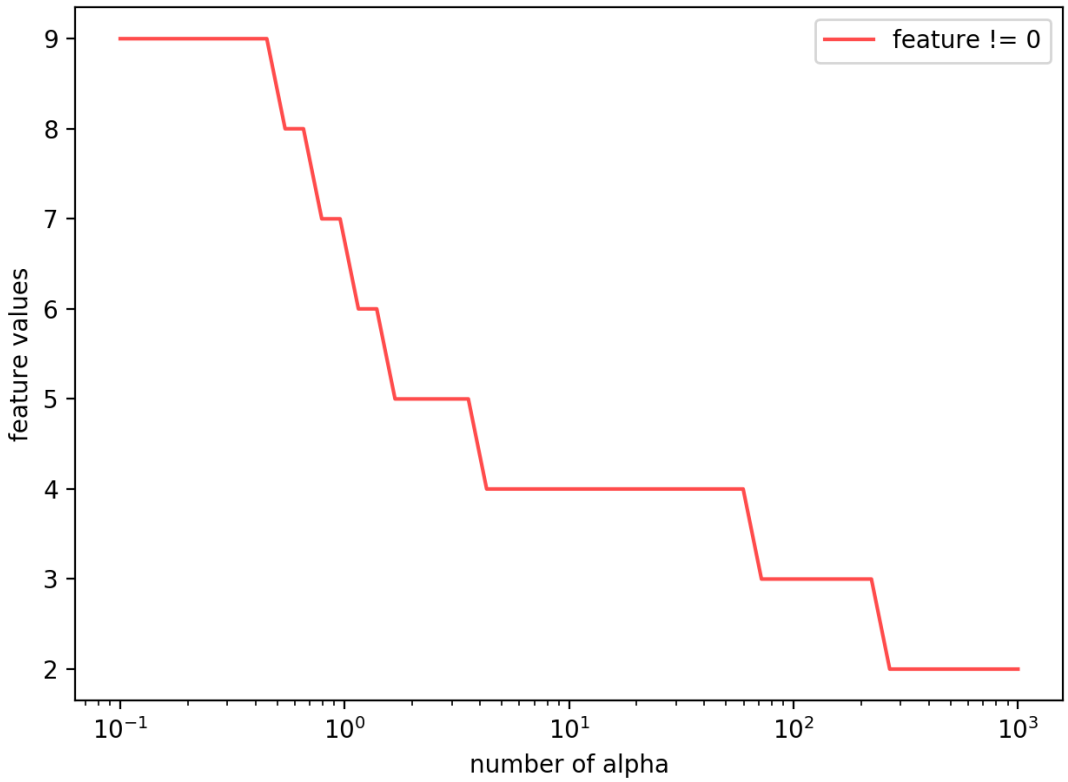


Figure 16: Relationship of alpha and features≠0

### 5.4 Comparison of 3 methods

As we can easily draw the conclusion from above outputs that each method has its pros and cons. For example, method 1 has the advantage that we obtained the lowest NMAE in a smaller number of feature. However, it takes a long period of time to run the code for it need to calculate 1024 subsets, the execution time as shown in our output,

- execute time of Method 1 is : 2.2010250091552734 seconds.
- execute time of Method 2 is : 0.13640499114990234 seconds
- execute time of Method 3 is : 0.25228285789489746 seconds

As a result, method 1 is not suitable for data with huge number of features.
While method 2 seems to have the quickest run time, it in fact is not a reliable way, as we change the feature sets in our train data, the correlation coefficients changing in spite that it can list the top features.

Method 3 can be regarded as the best method in all. It has the short run time and higher reliability. Based on my test, the feature set with the lowest NMAE of 0.104280 comes from when Alpha is 0.372759 and feature number is 9. It can provides with smaller number of features as well as reasonable range of NMAE.

## References


[1] A. Ng, "Machine learning - standford university [full course]." https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN, Feb. 2017.

[2] Scikit-learn.org, "Feature selection module documentation." http://scikit-learn.org/stable/modules/feature_selection.html, June 2018.