Product Report

June 2021
NXP Internship

Danial Zarei

Fontys University of Applied Sciences

INTRODUCTION

To avoid disclosing confidential information, this report is separated from my portfolio and will be taken down after my internship's assessment. The R&D-IT department reviews student's theses/dissertations/reports that are resulting from NXP collaborations and where the material concerns any NXP product or technology, via the Publication Approval System. BL (or equivalent organizations) and Law Department approval is required prior to external presentation or publication of any technical paper, article, reports, or presentation.

For this reason, in this report, I'll present a few products in the form of graphs and images to prove my contribution to the project to the university. Keep in mind, graphs and images are partially censored and limited to avoid disclosing confidential information.

Figure 1 on the right is a snapshot of the Exploratory Analysis. Y-axis shows the amount of memory reserved by one single user on a logarithmic scale. X-axis shows the amount of memory they used by the end of the job. As you can observe, jobs are mostly being overestimated by the user. This over-specification results in more jobs being on the pending state since the memory pool is finite which leads to a poor optimization.

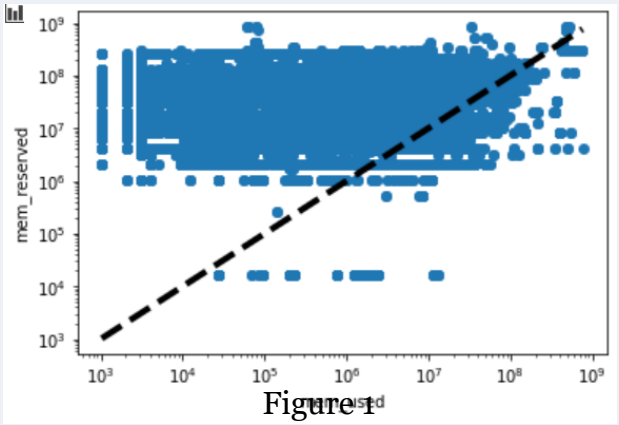


Figure 1

```
errors = np.abs(predictions_mem - Y_test)

np.mean(errors), np.var(errors)

(1659.8360278618504, 8201590.434307204)

plt.scatter(predictions_mem, Y_test)

<matplotlib.collections.PathCollection at 0x1fb2c1d77f0>
```

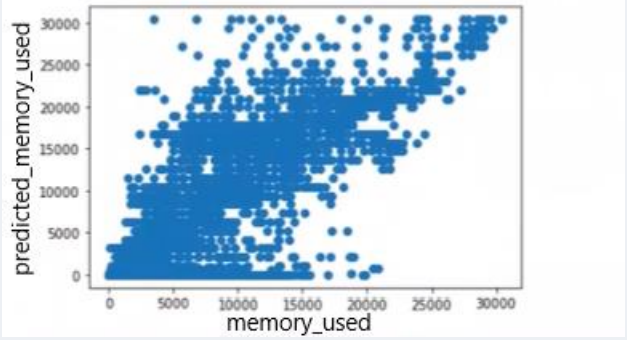


Figure 2

Figure 2 on the left shows how our model (RandomForest) predicts the memory used in the user's level (keep in mind figure 1 is on a smaller dataset with a single user and different projects). Figure 2 is on a bigger dataset with the top 10 users within our department with the highest number of jobs. The Mean Absolute Error as you can see, is 1.6 GB memory used. Y-axis is showing the predicted values and the X-axis is the actual memory used on the test dataset. As you can observe, the model is slightly overestimating the memory.

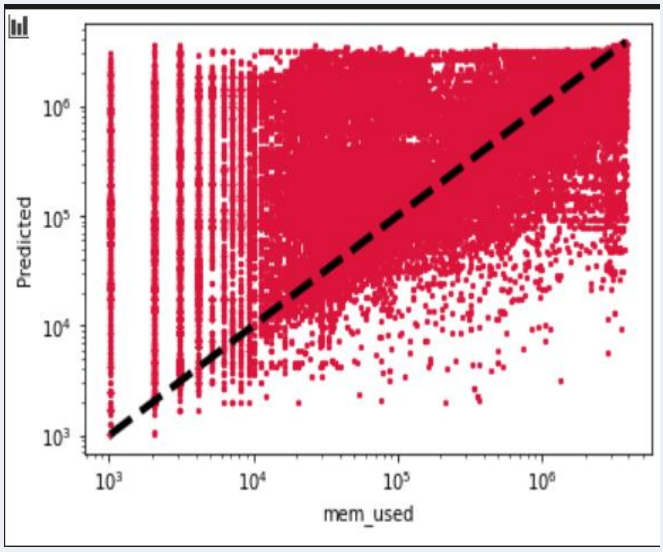


Figure 3 shows RandomForest model 's performance on multiple projects. Y-axis is showing the predicted values and the X-axis is the actual memory used on the test dataset. The Mean Absolute Error is 32GB with 57% mean r2 score. The model performs poorly on memory_used below 10^4. The dataset consists of jobs mostly above 10^4. Since these low memory jobs have many inconsistent features, the models have a hard time in predicting that area. This could be improved by tightening the outlier's borders and remove low memory jobs. But since this was not the company's intention, we decided to look for other solutions.

As discussed in the portfolio, I built a pipeline where data will be prepared, features are automatically selected, and 5 models are trying to predict the memory_used value. This is on a large dataset with top 5 projects with the highest number of submitted jobs and thousands of users. The RandomForest model is not part of this pipeline and it performs better than all of them with 64% mean R2 score. More details have been discussed in the portfolio.

	Name	Mean R2
5	GradientBoostingRegressor	59.82
3	DecisionTreeRegressor	59.54
0	LinearRegression	52.53
1	Lasso	52.19
2	ElasticNet	50.19

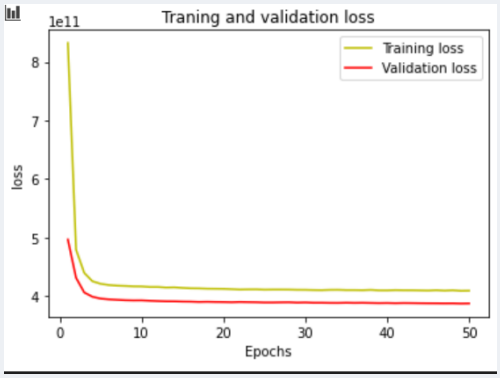
Artificial Neural Network Design

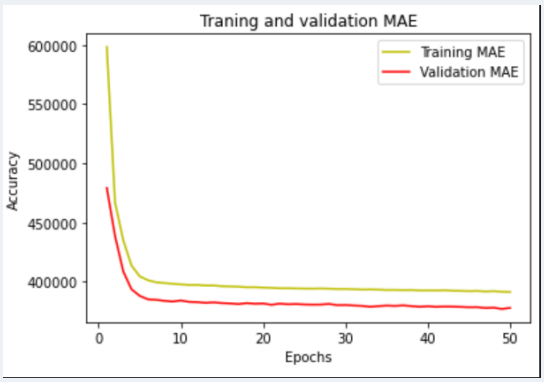
The figure on the right, shows the summary of My ANN model. This model consists of an input layer with 16 nodes, 3 hidden dense layers with each 64 nodes along with a 30% dropout layer to prevent overfitting. Furthermore, I'm using the 'keras.regularizer.l2' to prevent overfitting and better performance on the model.
The last layer is a dense layer with one node which is the memory predicted with a learning rate of 0.0001.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 16)                256
_________________________________________________________________
dense_23 (Dense)             (None, 64)                1088
_________________________________________________________________
dropout_11 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_24 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_12 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_25 (Dense)             (None, 1)                 65
=================================================================
Total params: 5,569
Trainable params: 5,569
Non-trainable params: 0
```

Why: I've experimented with many different architectures by trying 1,2 or 4 hidden layers. I've experimented with different values as the dropout (20% and 10%) also I've tried different learning rates (0.01 and 0.001). The current structure as you can see above is has the least loss and the lowest MAE.

Here you can see how the model has been trained during 50 epochs.





As you can observe, after few epochs, the model doesn't improve as much and hits a plateau. Later, we decided to continue with Random Forest model since it had a slightly better MAE. (37.5 GB MAE vs 32.5 GB MAE)