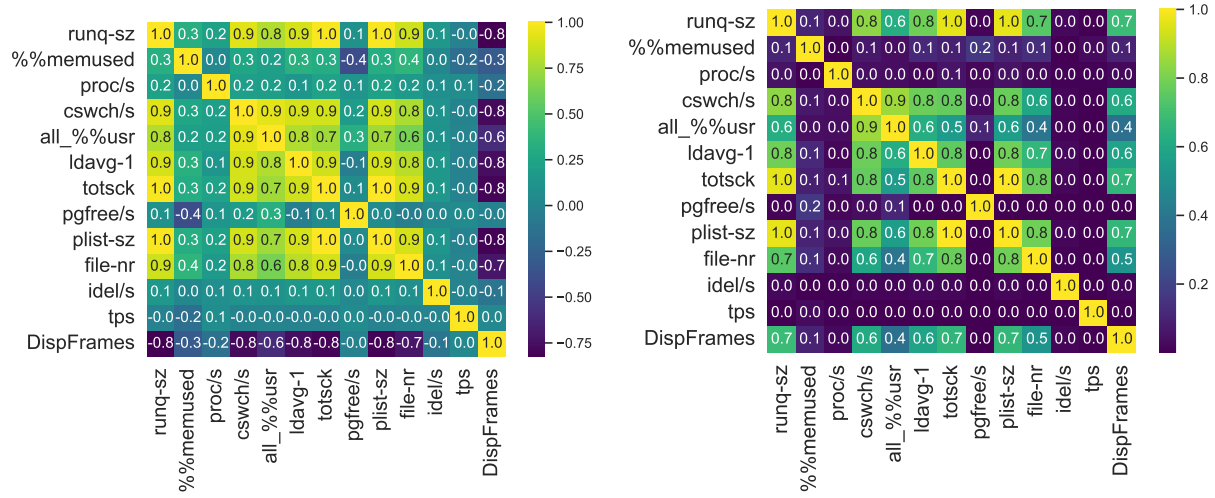
EP2420 *Project 1 - Week 3*

Federico Giarrè

November 16, 2023

3.2 Heuristic Method



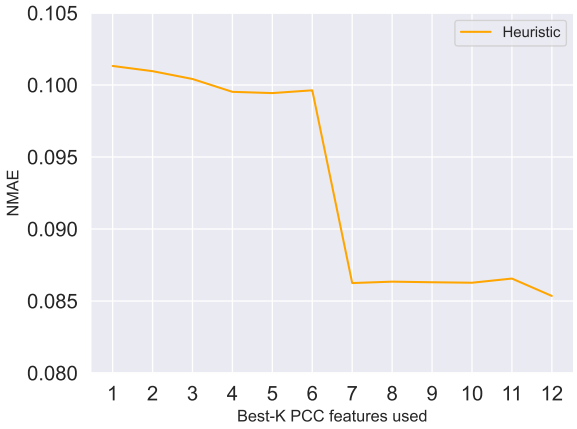
(a) Heatmap for the PCC of each feature

(b) Heatmap for the squared PCC of each feature

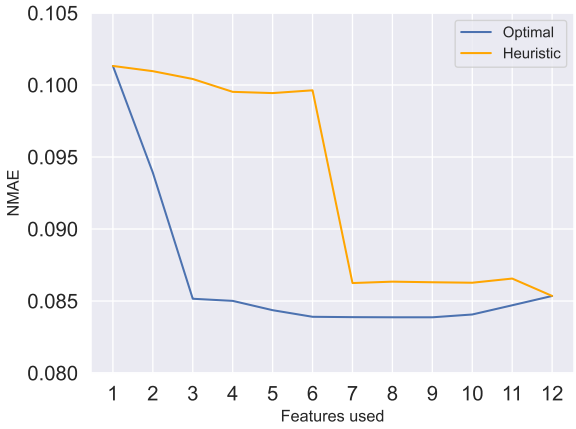
Figure 1: PCC 1(a) and squared PCC 1(b) for each feature of the dataset

After computing the Pearson Correlation Coefficient (PCC) for each combination of features, we can obtain the results shown in the heatmap in Fig.1(a). Some features can be considered correlated (i.e. the number of context switches per second and the CPU usage) and most of the single features are to be considered highly correlated to the FrameRate (DispFrames), given that six out of twelve values reach a PCC of -0.8 [1]. By squaring the values of the correlation, we obtain the heatmap in Fig. 1(b), and according to that we can find an ordering of the features based on their coefficient of determination. The ordering (from the highest to lowest) is the following: **('runq-sz', 'plist-sz', 'totsck', 'cswch/s', 'ldavg-1', 'file-nr', 'all_%%usr', '%%memused', 'proc/s', 'idel/s', 'tps', 'pgfree/s')**.

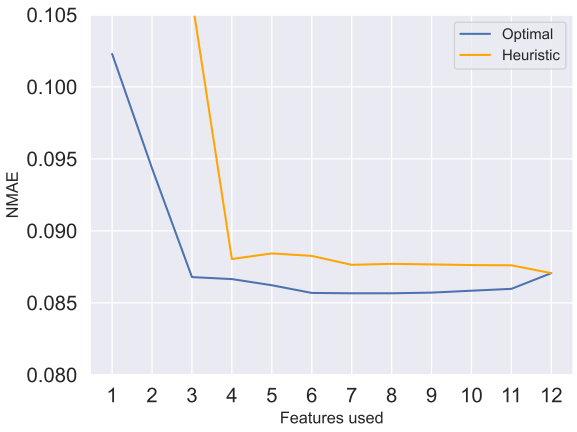
[1]PCC returns for each couple of features a value in the range [-1,+1]. The closer the value is to the extremes, the higher the correlation is. A value near 0 indicates the absence of correlation between the two features.



(a) NMAE with respect to an increasing number of features used (Heuristic method)



(b) Comparison between the Heuristic result and the minimum of Optimal method result



(c) Comparison between the Heuristic result and the minimum of Optimal method result

Figure 2: NMAE results of using the Heurist method to select features 2(c) and the same result compared to the minimum error achieved by the optimal method 2(b). A comparison with different order of features selected is shown in 2(c)

.

Fig. 2 shows the NMAE results (on the y axis) with respect to an increasing number of features used in the dataset. When selecting the features for the Heuristic method, the top-k features in terms of squared PCC were picked, thus following the ordering given in the previous paragraph. The results were then compared to the best results achieved by the Optimal method in Fig. 2(b). When analyzing this comparison, it is possible to notice how better the Optimal method can perform when selecting fewer features (two to six in this case), but the difference is error is sensibly narrowed from seven features and on. When looking at the behavior of the Heuristic method, we can notice a drop in the NMAE when adding the 7th top feature: the CPU usage. In order to assess the importance of the feature, the test in Fig. 2(c) was conducted, in which the CPU was added as the first feature and then the experiment progressed as the previous. What is notable, is that the drop in NMAE is presented very early with respect to the ones in figures 2(a) and 2(b), but then converges in the exact same way. Moreover, no matter the feature ordering, the time taken to compute the twelve models for the Heuristic method is of $\sim 0.14s$

This comparison leads to some conclusions that can be applied to this dataset: *i)* For a smaller number of features, picking the top-k features is not always the optimal strategy, since some important features may

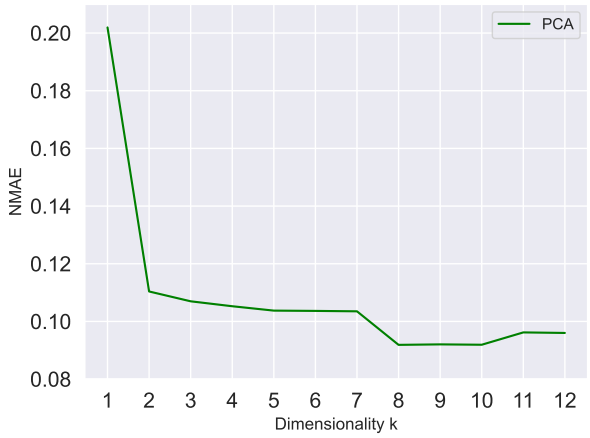
not be used from the start; *ii)* Due to the higher number of models trained with the optimal method and given the low error difference, the Heuristic method could be preferred after a certain number of features are added; *iii)* Time-wise, the Heuristic method is more convenient, completing the computation in $\sim 0.14s$ instead of the $\sim 40s$ taken by the optimal method.

3.3 Quiz

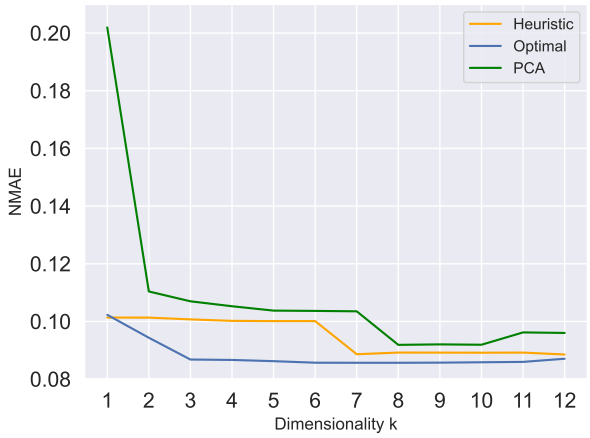
When using Heuristic and Optimal methods, the aim is to discard enough information to make computations faster without impacting the error too much. At the same time, it is possible to discard features that are not important or that, not having any relation with the label, may have a negative impact on the model. On the other hand, PCA projects the feature dimension into a smaller one, making computation faster while also reducing the noise in the data. Contrarily to the other two methods, using PCA information that is not essential will still be part of the dataset, so the risk of dropping important information is reduced.

Task 4 Dimensionality reduction using Principal Component Analysis (PCA)

4.1 PCA results



(a) Error with respect to the dimensionality of the dataset



(b) Comparison of PCA error with respect to Heuristic and Optimal method

Figure 3: NMAE result for PCA with respect to the number of dimension 3(a) and comparison with other methods 3(b)

PCA tests consisted of using PCA in order to reduce the dimensionality of the dataset to k dimension (with $k \in 1..12$) and train k models, one for each dimensionality. As it is shown in 3(a), the error is very high when $k = 1$, but quickly decreases and follows the same trend as the Heuristic method, with a step of further decreasing at $k = 8$ (instead of 7 of the Heuristic, as shown in Fig. 3(b)). In general, PCA shows a higher error than the other two methods in this dataset, and at the same time has a mean training time of $\sim 0.25s$, which is near to two times the one of the Heuristic method ($\sim 0.14s$). Even if this difference is almost negligible in this case, with a bigger dataset this can become some attention-worthy complexity. Given n the amount of samples, p the amount of starting dimensions, the time complexity of PCA is $O(n * p * min(n, p) + p^3)$, and since $n >> p$ the cost of doing PCA and training a regression model can be seen as $O(2(n * p^2 + p^3))$, so the final cost to run PCA and train a model 12 times becomes $O(24(n * p^2 + p^3))$, close to double the time needed to train 12 models with the Heuristic method ($O(12 + 12(n * p^2 + p^3))$ operations), which is also

reflected in the time recorded: $\sim 0.14s$ for the Heuristics and $\sim 0.25s$ for the PCA. operations[2].

4.2 Quiz

There are various ways to perform the hyperparameter-search that will be needed in Task 5. One of the ways in which we could perform hyperparameter-search would be using KerasTuner [3], which would search an hyperparamenter set on the defined space of possible hyperparameters and get back the hyperparameter that obtained the best results. However, since the dataset analyzed is quite small and Task 5 requires searching only for optimizer, regularizer, and number of hidden layers and nodes, we could simply pick some meaningful values and loop through them. For instance:

- **Optimizer**: Keras implements 10 optimizers, but some of them are useful only for big or sparse or time-variant samples (such as AdamW, Adamax, Adafactor and Ftrl)
- **Regularizer**: L1 and L2 regularizers have to be tested, for λ some recommended values can be tested (such as 0.3, 0.4, and 0.6).
- **Hidden layers and nodes**: Plausible number for the number of layers and nodes can be obtained with the following rules of thum[4]:
  - The number of hidden neurons should be between the size of the input layer and the size of the output layer.
  - The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
  - The number of hidden neurons should be less than twice the size of the input layer.
  - The number of hidden layer should be one (or higher) if the data is not linearly separable, but one layer can already approximate any function that contains a continuous mapping.

[2]https://alekhyo.medium.com/computational-complexity-of-pca-4cb61143b7e5
[3]https://keras.io/keras$_tuner$/
[4]Heaton, J. (2008). Introduction to neural networks with Java. Heaton Research, Inc..