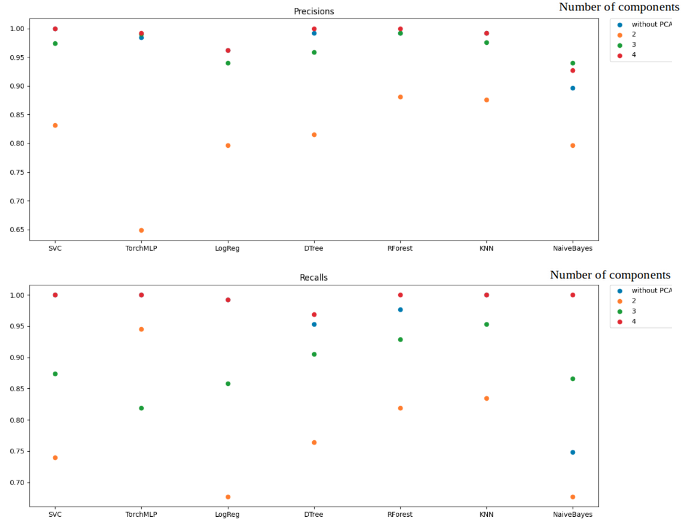
To measure models performances, we will use these two metrics:

* precision: is the ratio of true positives to the total of the true positives and false positives.
* recall: is the ratio of true positives to the total of the true positives and false negatives.

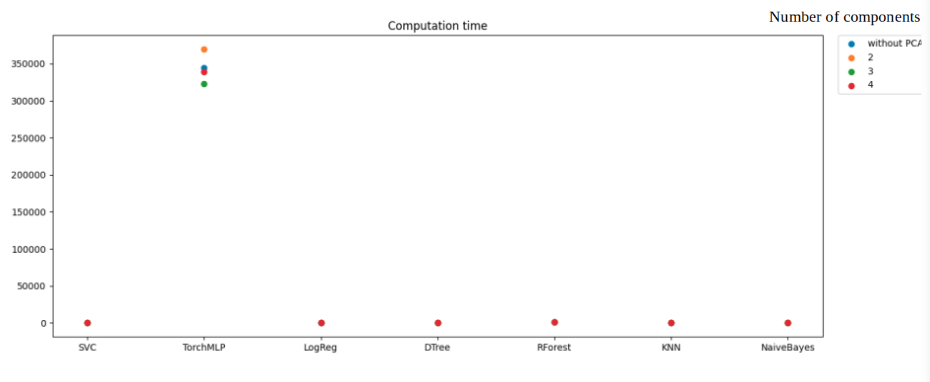
We will compare the different models, while applying PCA to the data.

We will also take a look at the computational time.

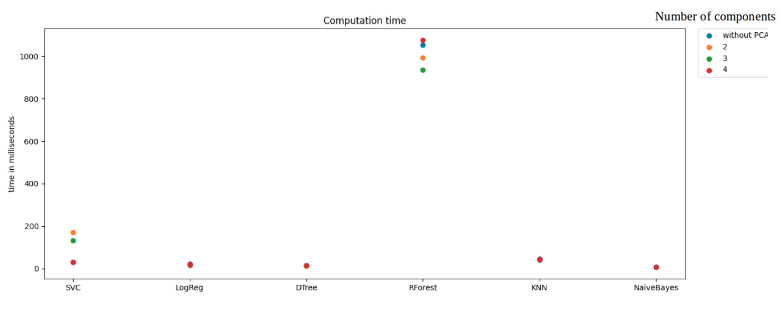
**Dataset Banknote**



One important thing to notice is that we have the worst performance when we apply PCA and keep only two components. Applying PCA is very useful when we have high dimensional data, in order to reduce dimension, prevent overfitting, or denoise data. Here, we can see that applying PCA on few dimensional data is not really helping; it may increase or (more likely) decrease the model performance.

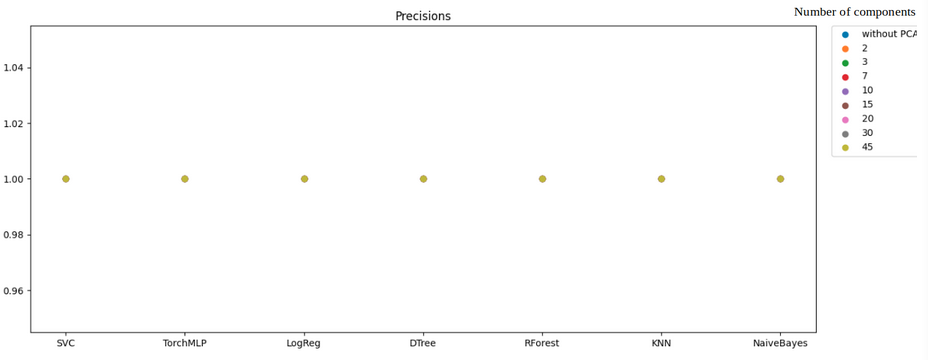


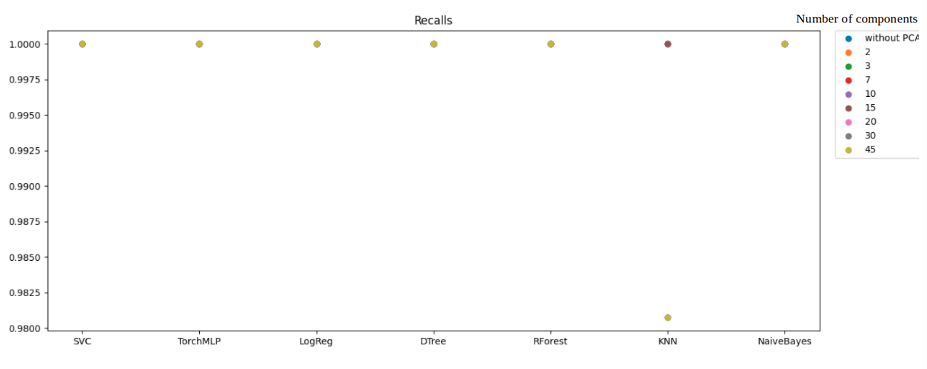
We can notice that the neural network takes a lot more time than the other models. So, let’s take a closer look at the other models (without TorchMLP).

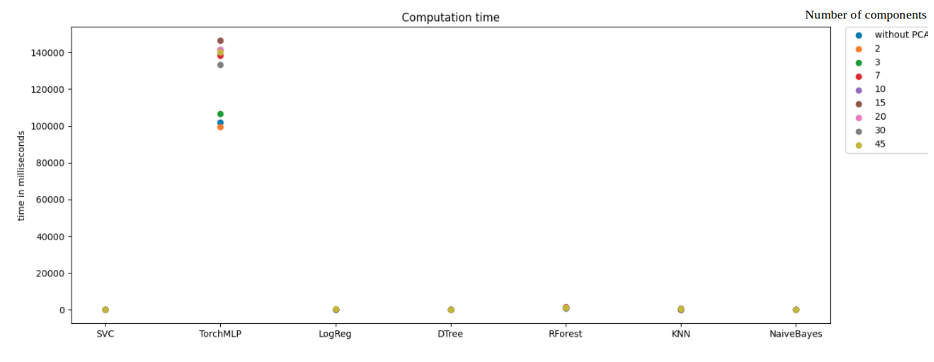


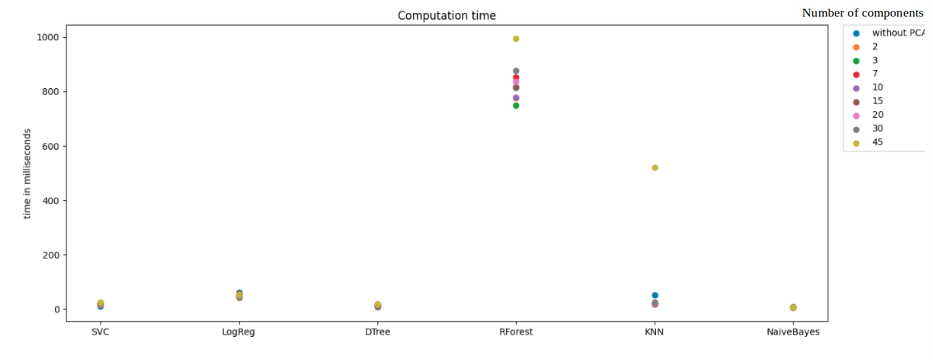
We can notice that the Random Forest model is the one who takes the most time. We used the default scikit-learn Random Forest model which takes as default parameter 100 trees; so it is pretty understandable that it took more time than the other models.

**Dataset kidney**









It is pretty difficult to draw conclusions from these results, because all results are very similar, regardless of the model or the number of components.