**Models:**

1. **PyCaret (We used this for prediction)**
2. **LSTMs**
3. **AutoKeras**
4. **MLBox**
5. **TPOT**

Most of them, were all codes I used from my previous Data Science personal projects but I had to update them since some had deprecated.

**Big Picture**

Since time is a large constraint, we want to optimize the process. We will go from a Bottom-up approach; meaning we will use AutoML models to find the “best model” and tune it. We will not be going the top down approach by fine-tuning models such as SVM or Random Forest as it is an inefficient way of deriving “optimal” models for the following reason:

1. Much like Deep Learning, it is a highly iterative process
2. Tuning Bias-Variance Trade off for different models in 7 days is ridiculous

We are going to follow the bottom guide to try to minimize error in the lowest possible timeframe.

1. basic Neural Network as a baseline (LSTM)
2. Build AutoML models   
   **AutoML is used for 3 reasons.**
3. It has the lowest opportunity cost (error with respect to time consumption)
4. The best AutoML result will serve as a baseline across all AutoML models
5. Some AutoML models returns features with highest variance (Principle Components)

**This is important as we want to maximize the effect of Adj. R2.**

**Reason:**

**R2** does not penalize the addition of useless features. Meaning as we add more features, the value of R2 will never decrease. This is however not desirable, we can keep adding useless features.

In the case of **Adj R2**,

a. If you **add useless variables** to a model, the value will decrease.

b. If you **add useful variables**, adj. R2 will increase.

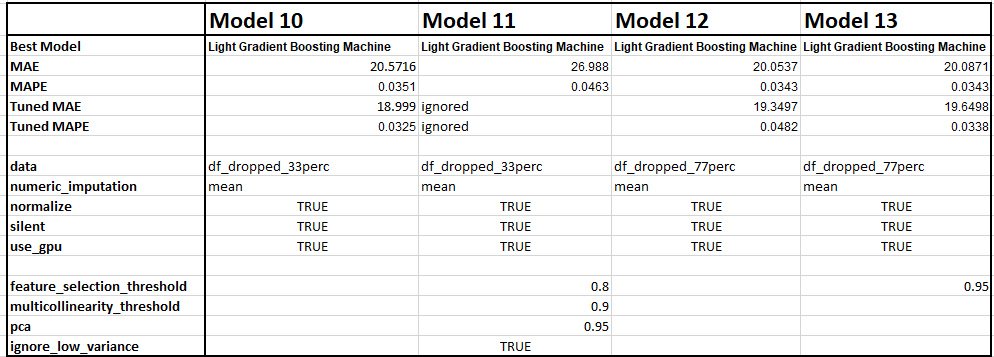
With that in mind, we want to **maximize the effect of Adj. R2**, therefore theoretically **reducing** computational cost and **error rate** after model tuning.

**We attempt to use the following AutoML models:**

* 1. pycaret
  2. Autokeras
  3. tpot
  4. mlbox

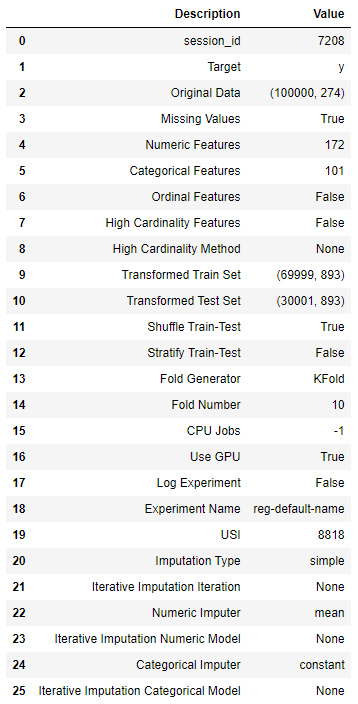
1. After running each AutoML Models, we will
   1. Analyze the results and gather informational data via output of the metrics provided by the models.
   2. choose the most important features, via results of AutoML,
2. We will then rerun all the models after choosing the most important features.
3. If results are not good enough, we might need to spend time to finetune, keeping in mind performance of autoML. Thus we keep track of end target whilst being mindful of opportunity cost (cos I am currently working)

**PyCaret Results**



Quick Comparison reveals Model 10 has the best Tuned MAE of 18.999, where MAPE is 3.25%. The rest are inferior and are not described thoroughly to keep the report short.

**df\_dropped\_33perc**: so long as there are more than 33% missing values, the columns Is dropped  
**df\_dropped\_77perc**: so long as there are more than 67% missing values, the columns Is dropped



Transformed data is 893 columns, as compared to 274 columns, thus feature has been engineered for us. This may be the reason why Model 10 performed due to dropping columns with high null-values.

**df\_dropped\_77perc:** performed worse as columns with <67% missing values are kept. Meaning even columns with 50% missing values are used and are filled with mean values. Resulting in decrease in performance. This is on some level, evidence of presence of unimportant features.

Most importantly, MAE is 19.2441. This means the average absolute error is ~19.2. Which is fairly good, given we used almost all columns (even the unimportant ones). Thus we might be confident feature selection might produce a better performing model.

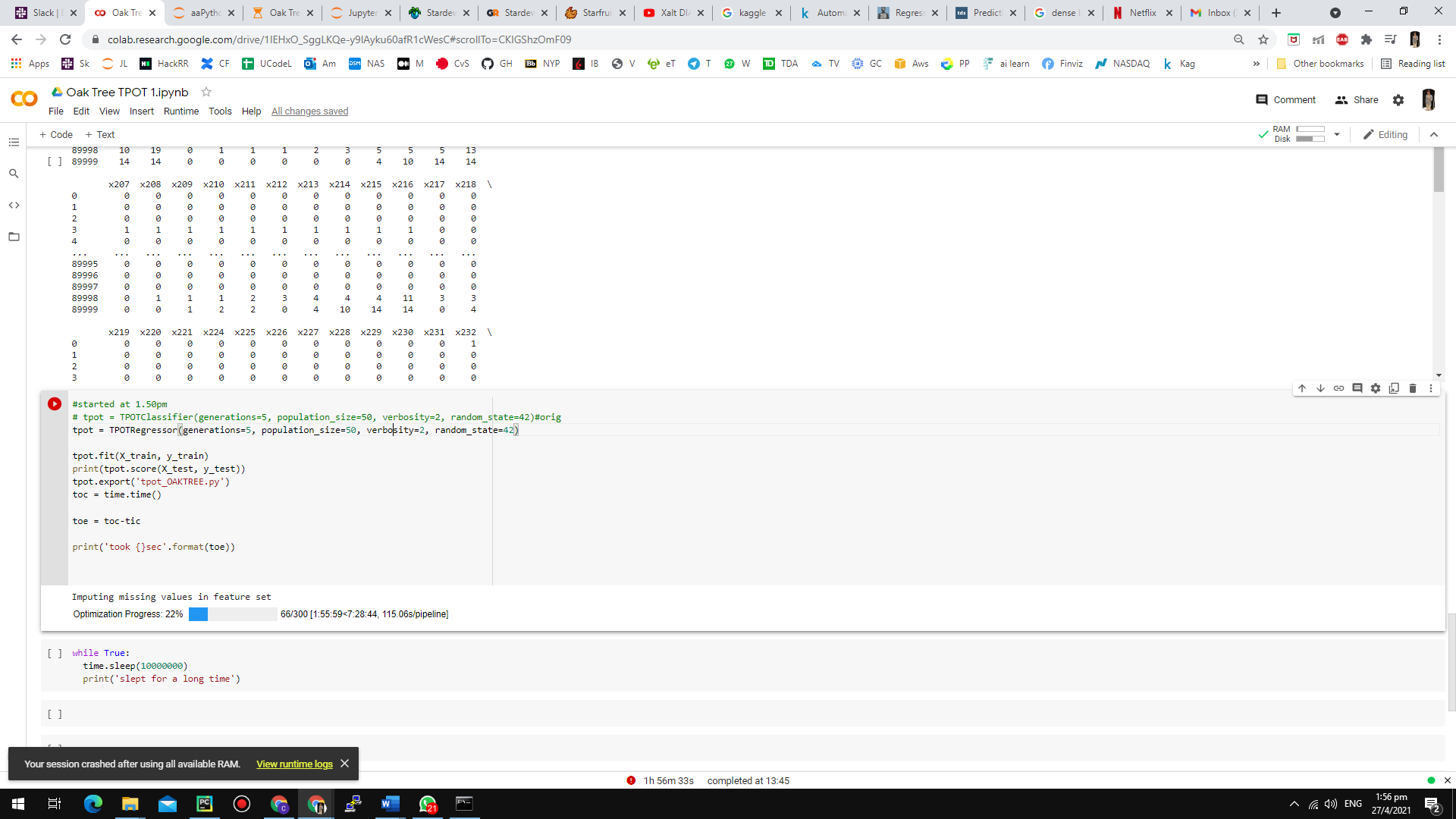
**AUTOKERAS**

After running AutoKeras, we will not be using AutoKeras for the following reason:

1. It was unable to converge
2. MAE: 1000

**TPOT**

I was running TPOT on colab as I was unable to run locally as I was training various models. After running TPOT on colab it was unable to optimize fully



**MLBOX**

MLBox was unable to converge as well as most, if not all of the accuracy metrics were -inf.