# Machine Learning Challenge:

By Amaury de Guillebon

## Objectives:

The goal of this project is to build a ML system to verify the veracity of claims. The problem is opened to any solutions and strongly encourages creativity. We will rate the success of our models on the following criteria: accuracy, prediction accuracy on the test set.

## Introduction:

We start this challenge with different resources at our disposal:

- The dataset link: https://huggingface.co/datasets/health_fact
- Huggingface transformers: http://huggingface.co/models

The approach chosen is the following: First, we import the dataset, that we pre-process by removing any non-alphabetical character or any NAN. Then, we adapt the classes to a form that is easier to work with: basically, we characterize all the words present in our strings into a given number using a token embedding. After that, we build and train a keras model that will try to accurately predict the targets of the test dataset.

### 1. Data processing:

First of all, we get rid of the NANs present in the important classes (claims, explanations, label, etc…). This is essential since NANs would cause a problem later on when processing the data. Moreover, it raises the ethical questions of 'when should we consider an argument as valid?'; we made the choice to answer this question by saying that without one of the essential information, the row is not worth anything.

Then we have a look at the data and especially the classes distribution. The uneven distribution led to simplify the problem into a binary one with only 'true' and only 'false' labels. We concluded that any unproven, or misleading claim was by definition not true (as illustrated in the code) and therefore classified as false. We also got rid of all non-alphabetical characters such as the '…', the '"', the '[' etc…

After that, we created a vocabulary of all the tokens associated with an embedding (glove from gensim 4.0.1).

Finally, we changed all the sentences in sequences to be processed by the model.

### 2. The Keras Model:

We created and trained 3 different models: The last state model with 256 hidden units, the mlp model with an output shape of 256 and the bag of vector with an output shape of 300. We merged two vectors corresponding to the sentence embeddings of the claims and the explanation. The embedding layer used, being from Keras. Then we pass the vectors into a Dense layer adaptable to the input shape with only one node to match the shape of the targets.

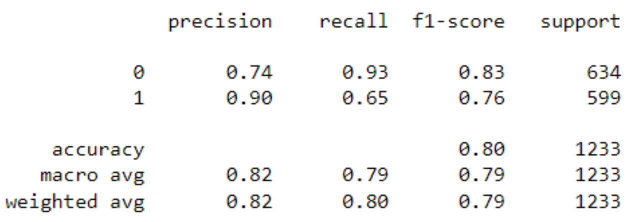
## 3. Results:

Here is a summary of the performances of the models on the **validation set**:

| | Accuracy | Comb. Accuracy | F1 score |
|---|---|---|---|
| Last State Rnn | 0.82% | 0.81% | 0.81% |
| MLP | 0.83% | 0.82% | 0.81% |
| Bag of vectors | 0.80% | 0.79% | 0.79% |

## 4. Best models:

After testing the model with the testing set, we observe very good results for both the MLP and the Laste State models:

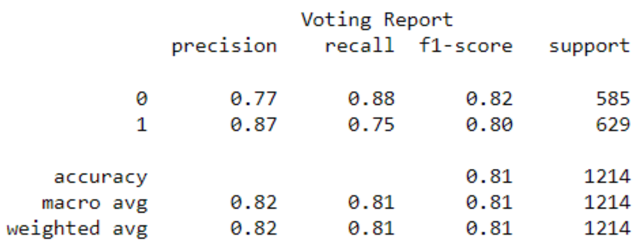


```
              precision    recall  f1-score   support

           0       0.74      0.93      0.83       634
           1       0.90      0.65      0.76       599

    accuracy                           0.80      1233
   macro avg       0.82      0.79      0.79      1233
weighted avg       0.82      0.80      0.79      1233
```

Fig 1. MLP’s evaluation on the test set



```
                   Voting Report
              precision    recall  f1-score   support

           0       0.77      0.88      0.82       585
           1       0.87      0.75      0.80       629

    accuracy                           0.81      1214
   macro avg       0.82      0.81      0.81      1214
weighted avg       0.82      0.81      0.81      1214
```

Fig 2. Last State’s evaluation on the test set

Here are their associated confusion matrix:

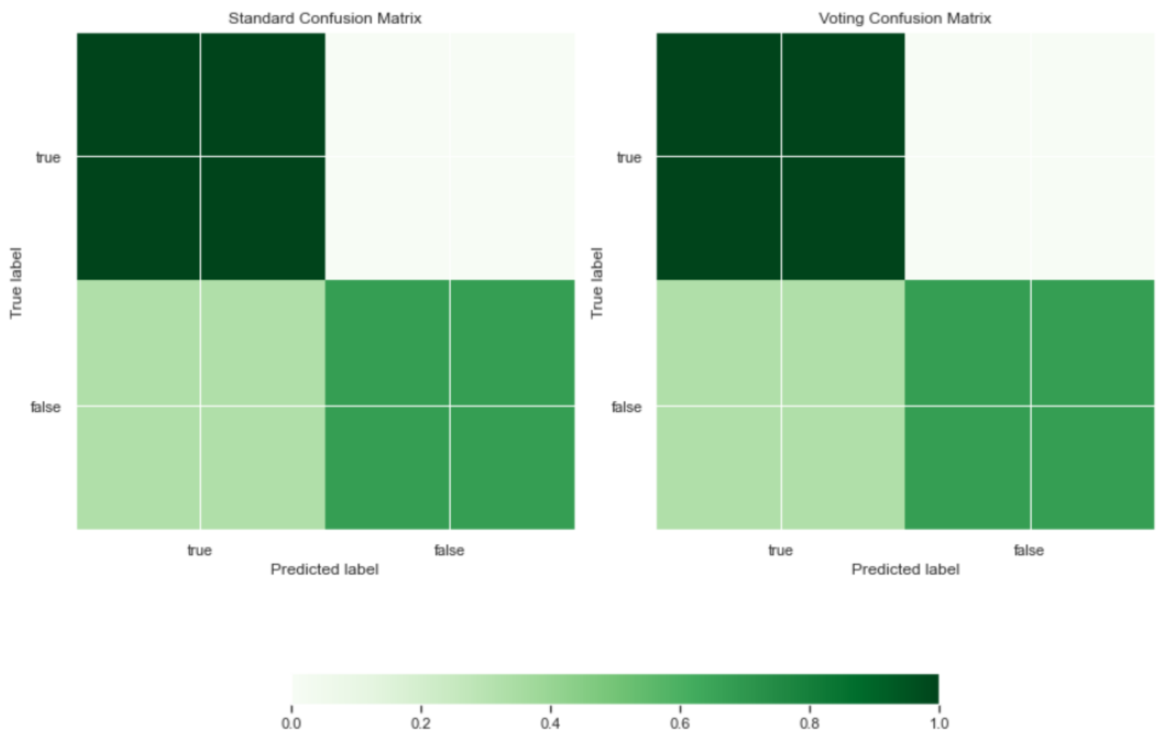


Fig 3. MLP’s confusion matrix

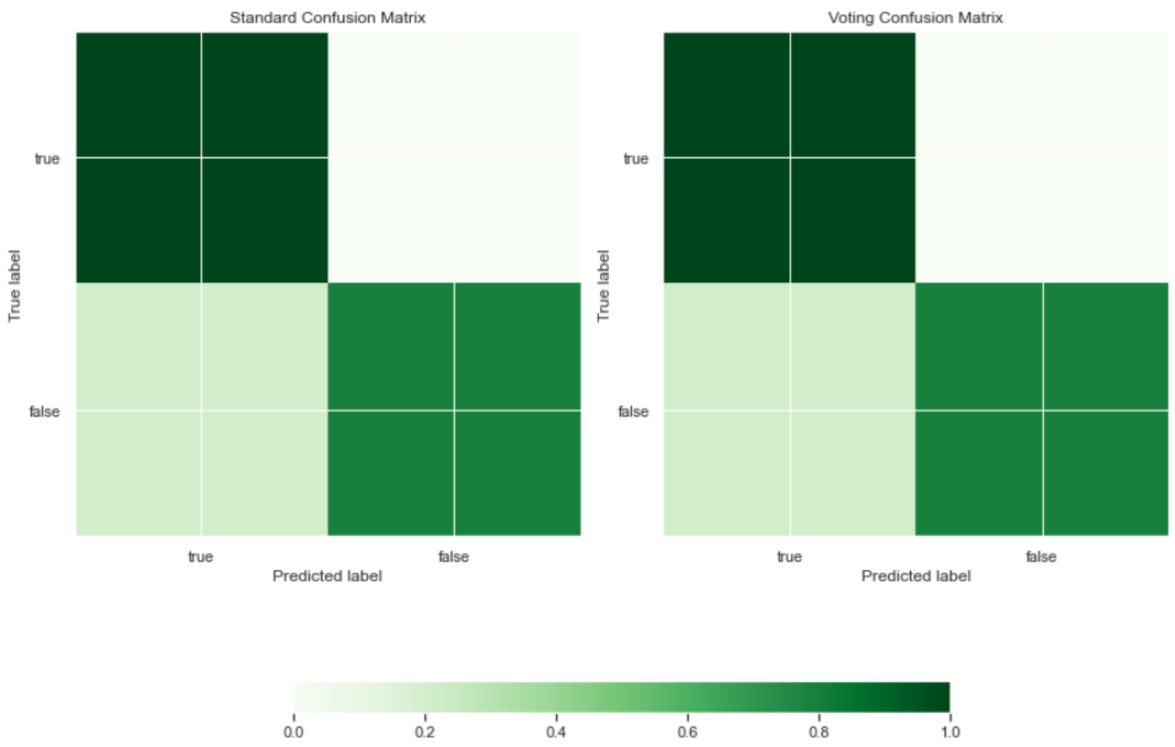


Fig 4. Last State’s confusion matrix

## 5. Observations:

We notice that the last_state_rnn and mlp modules are able to predict quite accurately the target of the test dataset. However, we can notice that some targets are predicted as true even so they are in fact false. This deficiency can be cause by many parameters, but the fact that we simplified all the 'mixture' targets as 'false' might be a reasonable explanation.

## 6. Conclusion:

We managed to generate several models with a good accuracy (more than 80%) and at least two of these models predict the target of a test dataset with a decent precision, reaching a f1 score of about 80%.

**Where it can be improved:**

1) We have been limited by the time needed to train some models (4h 25mins 48s for the last_state_rnn), therefore it was complicated to try new and more complex models. This is the result of a quite demanding model in addition to a limited network availability.
2) We witness some overfitting when training our models, as shown by the val_loss that remains quite high for some of our models.
3) We made the assumption that 'mixture' and 'unproven' would mean 'false' 100% of the time, which can be changed.

**How YOU can improve it:**

1) Have access to a faster network.
2) When running your model, add dropout=0.3 (or 0.2 or 0.4 depending on the results), this layer should lower the overfitting and is already added to the cod in the function get_model.
3) You can keep 'mixture' as 'mixture' and 'unproven' as 'unproven' by changing the following lines indicated in the conclusion of the code. Then you will have to change the optimizer, the loss function since the targets are no longer binary, and adapt your model to the new shape of the targets.