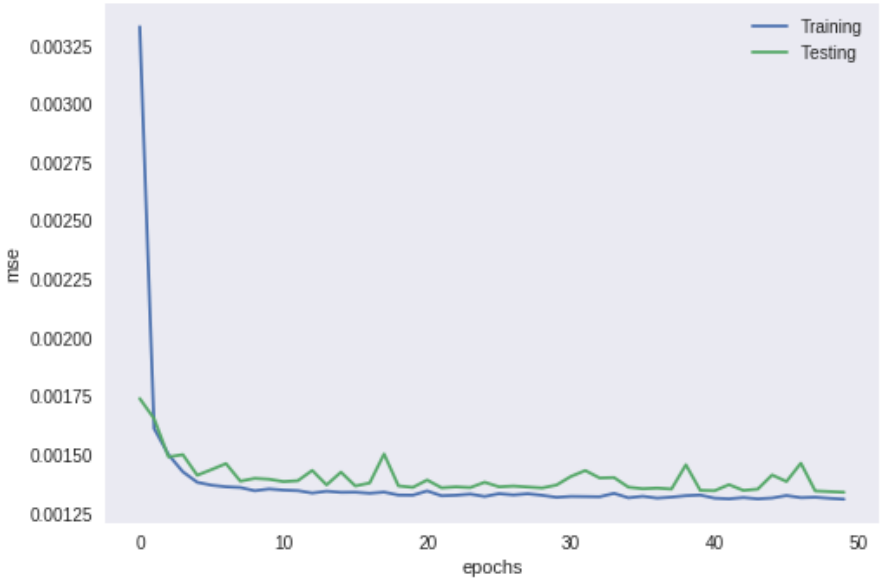
PW14

Authors: Flavia Pittet, Fabio Vitali

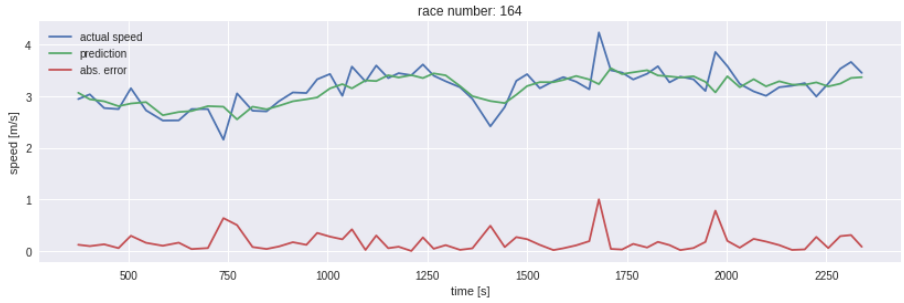
Exercise

1. Change the number of units and epochs of the LSTM network. Show the configuration that performed the best.

- We increased the number of epoch or units, but we couldn't reach an overfitting threshold (our MSE kept decreasing), but the Test correlation coefficient was at its maximum with the following model :
  - Epochs: 50
  - NB_UNITS: 8
  - Parameters: 393
  - MSE Training Set: 0.0013066387226170985
  - MSE Test Set: 0.001344820556779941
  - Training Correlation Coeff: 0.6571079224437987
  - Test Correlation Coeff: 0.642635956687881



```
Max Error:  1.001929939882353
```



2. What is the largest error (speed prediction) you observed? Do you observe that most of those large errors show up for high speeds ? or low speeds? Why?

- **It's not related to either high speed or low speed, the largest errors shows up when the actual speed increases or decreases too quickly.**

3. Compute the correlation between the next speed (model output) and the current speed (model input). Does your LSTM perform better than just using the current speed as a prediction of the next speed ?

```
Training correlation coefficient: 0.6571079224437987
Test correlation coefficient: 0.642635956687881
```

Figure 1: alt text

It is better than just using the current speed.

4. Using the predicted speeds for a given race, compute the expected time for a race and compute the difference between the real race time and the predicted race time in minutes. Provide the code of the cell that computes this prediction error.

- **We should multiply each predicted speed by each intermediate distance travelled between each measure to get the intermediate time and then sum them up.**
  Unfortunately, we failed implementing that, so we only multiply the total distance with the mean value of the predicted speeds to get a result.

```
def predict_time(y_pred, data, race):

  race_df = data.loc[data['race'] == race]
  #filter features
  race_np = race_df[['time','distance']].values
  # We get the last recorded time in the dataset
```

```
  exact_time = race_np[-1][0]
  # We divide the total distance travelled by the mean predicted speed
  estimate_time = race_np[-1][1] / y_pred.mean()

  error = np.abs((exact_time - estimate_time))
  return (estimate_time / 60, error / 60 )


#We use the same prediction for run 164 used in 1)
estimate, error = predict_time(y_pred_o, original_dataset, 164)

print("Error is: ",  error, "minutes")
```

Output: Error is: 0.2977072561254848 minutes

Our estimate only differs from the exact time by ~0.30 minutes (~= 18 seconds)