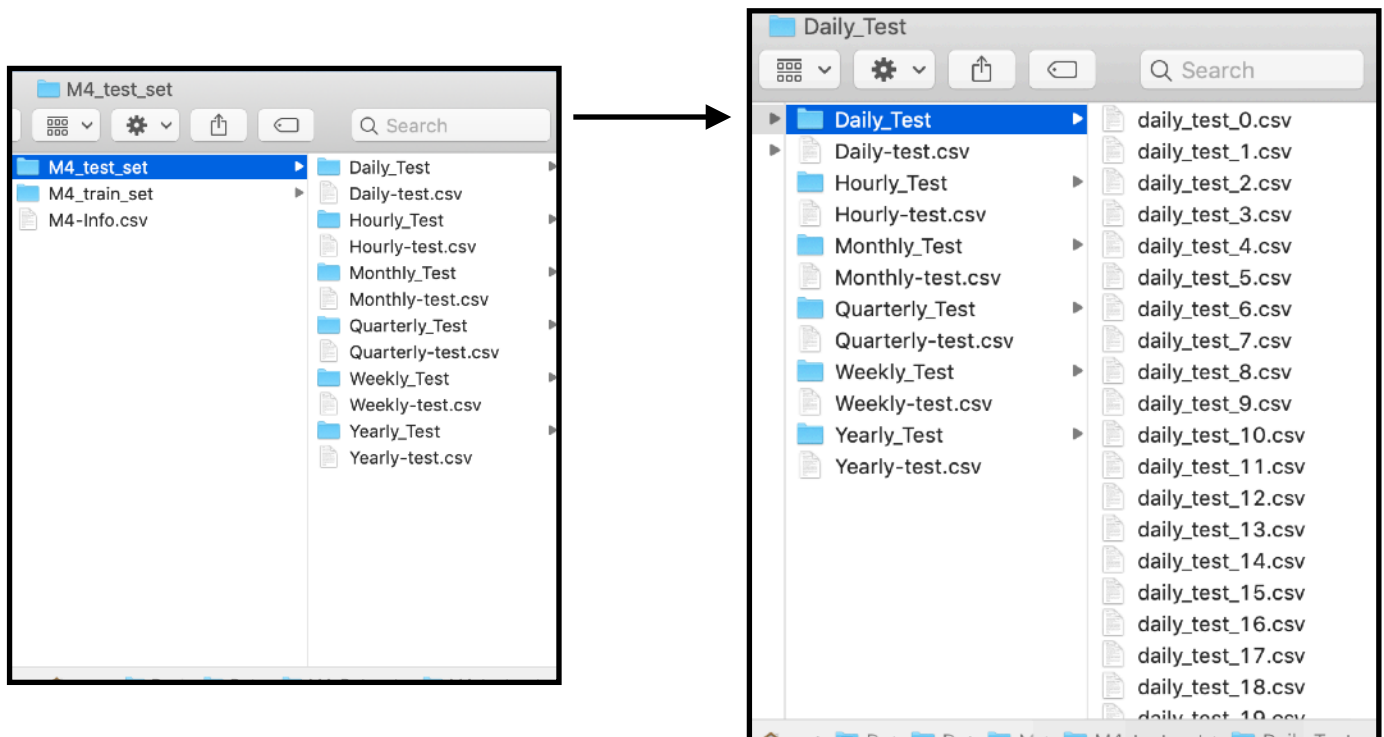


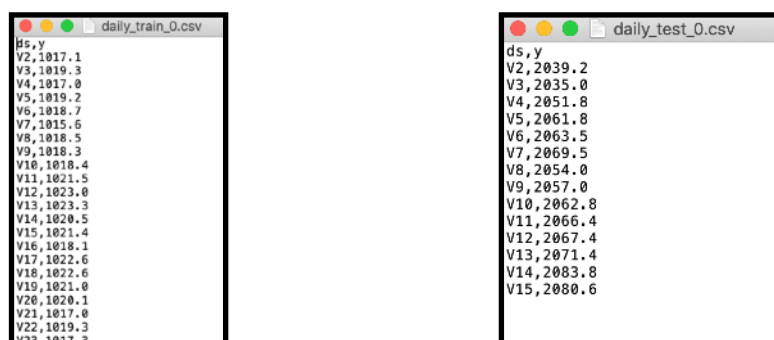
I have downloaded the big M4\_Dataset, there were M4\_train\_set and M4\_test\_set. Under these folders;

- Daily-test.csv - 4227 data files
- Hourly-test.csv - 414 data files
- Monthly-test.csv - 48000 data files
- Quarterly-test.csv - 24000 data files
- Weekly-test.csv - 359 data files
- Yearly-test.csv - 23000 data files

then I have split these big csv files into smaller ones, so they can be tested quickly and (tests-results) variables can be handled with accuracy.



this is how a train and test file looks (ds, y is for facebook.prophet);



In the code, I am traversing over this files with an f stream, for now we are only trying a few files, but when the improvements are made, this file number can be pushed up.

```
for i in range(3): # here size_daily etc. can be used but since it takes very long, i use a smaller number for testing

path_train = f'/Users/mac/Desktop/Datasets/M4_Dataset/M4_train_set/Daily_Train/daily_train_{i}.csv'
path_test = f'/Users/mac/Desktop/Datasets/M4_Dataset/M4_test_set/Daily_Test/daily_test_{i}.csv'
```

Then I am trying to prepare the data for neural networks, adding time steps, splitting to train and validation sets in train file and not splitting in test file. Normalising between 0.001 and 1 (when i normalise to 0 percentage error becomes inf)

I will add more parameters to the parameters list to try many different optimisations.

But there is one thing I did not understand well;

The MAPE scores vary very strongly. This did not happen at my first trial, i get generally about 0.04 - 0.08 and i was surprised since this looked better than MAPE scores on the M4 Competition paper.

But somehow on this attempt, scores seem abnormal to be, sometimes 0.01, sometimes 10.xx, which is a very big variation. I am not sure if I am calculating them false, like a transformation (MinMaxScaler) problem...

```
2020-10-26 09:46:38.721262: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2020-10-26 09:46:38.736715: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x7fce
2020-10-26 09:46:38.736731: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor d
Epoch 00011: early stopping
14/14 [=====] - 0s 1ms/step - loss: 0.1661 - mse: 0.1661

Epoch 00192: early stopping
14/14 [=====] - 0s 1ms/step - loss: 0.0196 - mse: 0.0196

Epoch 00038: early stopping
[-0.09285844] [0.0732434] [{'batch_size': 32, 'epochs': 1000}]
1/1 [=====] - 0s 346us/step - loss: 0.0363 - mse: 0.0363
-0.03633064031600952
MAPE: 0.06711055876107955

Process finished with exit code 0
```

This MAPE seems ok, and the mse also indicates it is close.

But this one for example is huge:

```

2020-10-26 09:50:18.497164: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2020-10-26 09:50:18.509683: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x7
2020-10-26 09:50:18.509700: I tensorflow/compiler/xla/service/service.cc:176] StreamExecuto
Epoch 00059: early stopping
15/15 [=====] - 0s 1ms/step - loss: 0.0312 - mse: 0.0312

Epoch 00050: early stopping
15/15 [=====] - 0s 1ms/step - loss: 0.0119 - mse: 0.0119

Epoch 00047: early stopping
[-0.0215268] [0.00966603] [{'batch_size': 32, 'epochs': 1000}]
predicted:
[0.27941835 0.30684072 0.23722345 0.29379782 0.24033278 0.22628753
 0.16114207 0.11848384 0.19196072 0.2996195 0.4233688 ]
MAPE: 22.349962270914784

Process finished with exit code 0

```

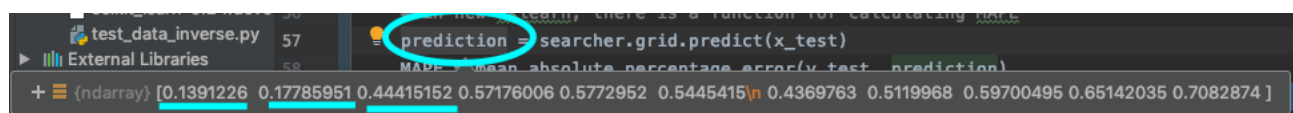
I believe there should be a mistake somewhere, but i could not find it by now. I will keep looking.

## Part 2

I am adding some more explanations, so things can be clear.

While debugging I have compared the code I have written first and then second. In the first code, I am using MAPE with real numbers (e.g. 1850, 1900, 1920 etc.)

But in the second project I am comparing the normalised numbers. It should not make a difference since the error is percentage style. (One caveat, when using normalised numbers, the smallest can be zero, and cause some problems, so it is better to scale between 0.001 and 1)



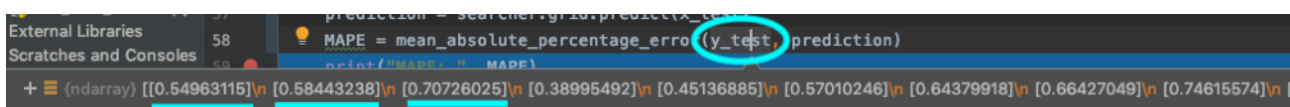
```

test_data_inverse.py 57
External Libraries 58
prediction = searcher.grid.predict(x_test)
MAPE = mean_absolute_percentage_error(y_test, prediction)
+ (ndarray) [0.1391226 0.17785951 0.44415152 0.57176006 0.5772952 0.5445415 0.4369763 0.5119968 0.59700495 0.65142035 0.7082874 ]

```

This is from the normalised code, our prediction list has values of: 0.13, 0.17, 0.44,...

And the actual normalised y numbers are: 0.54, 0.58, 0.70, ...

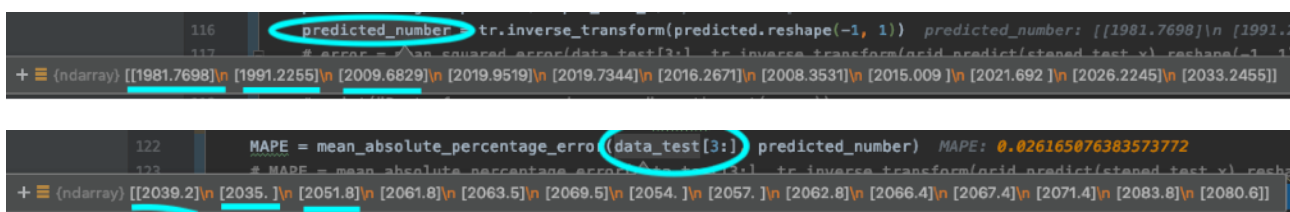


```

External Libraries 58
Scratches and Consoles 59
prediction = searcher.grid.predict(x_test)
MAPE = mean_absolute_percentage_error(y_test, prediction)
print("MAPE: ", MAPE)
+ (ndarray) [[0.54963115] [0.58443238] [0.70726025] [0.38995492] [0.45136885] [0.57010246] [0.64379918] [0.66427049] [0.74615574] ]

```

But in my first code, i had compared actual numbers like this:



```

116
117
predicted_number = tr.inverse_transform(predicted.reshape(-1, 1)) predicted_number: [[1981.7698] [1991.2255] [2009.6829] [2019.9519] [2019.7344] [2016.2671] [2008.3531] [2015.009] [2021.692] [2026.2245] [2033.2455]]
+ (ndarray) [[1981.7698] [1991.2255] [2009.6829] [2019.9519] [2019.7344] [2016.2671] [2008.3531] [2015.009] [2021.692] [2026.2245] [2033.2455]]

122
123
MAPE = mean_absolute_percentage_error(data_test[3:], predicted_number) MAPE: 0.026165076383573772
+ (ndarray) [[2039.2] [2035.] [2051.8] [2061.8] [2063.5] [2069.5] [2054.] [2057.] [2062.8] [2066.4] [2067.4] [2071.4] [2083.8] [2080.6]]

```

And the second MAPE is much, much better. I am not sure if the difference comes from different starting parameters of the weights, but consistently, with same network architecture and same parameters, one code gives better scores.

That, i do not understand why or how...