Utilisation of Time Series Data for Forecasting Oil Temperature and Countries Exchange Rates

BENNABI Malek Mohamed
Université Paris Cité

April 22, 2024

Abstract

In this paper, we harness the power of time series data to forecast two critical parameters: oil temperature and exchange rates. Our research is grounded in the analysis of electricity consumption patterns, which serve as a rich source of time series data. We employ machine learning techniques to identify patterns and trends in this data, which are then used to predict oil temperature and exchange rates.

Our approach involves forcasting the Oil temperature within the Electricity Transformer Temperature data. We also conduct a comparative analysis of different forecasting methods on the Exchange rates data, discussing their respective merits and drawbacks.

1 Introduction

Time series forecasting is the process of predicting future behavior using past signal correlations. Over the past decade, it has emerged as a significant area of study for both practitioners in the industry and academic researchers.
Numerous algorithms have been examined using variety of techniques, such as machine learning and deep learning methods (LSTM, RNN). [1] [2]
Nowadays, this learning problem is widespread in real-world applications where different observations are collected sequentially, such as medical data, electricity consumption temperatures or stock prices, hence the need to anticipate future changes, such as a fall in the exchange rate or a sudden rise in temperature, in order to remedy these perturbations and envisage appropriate solutions.

This project is divided into two parts:

- The first part is to provide a solution to the univariate forecasting problem of predicting the next 100 values of the OT variable for the ETTh1 dataset.
- The second part is aimed at finding a solution to the multivariate forecasting problem of predicting the next 100 values of the '6' variable of the exchange rate dataset, while dealing with the missing values.

2 Oil Temperature forecast

The first step is to train a predictor to accurately predict the next 100 values, which is a simple univariate prediction problem. To do this, we'll need to preprocess the data and split the dataset into a subset of subsequences of the form (input, target) for training. The training data consists of a univariate time series with 1 feature. For this method, we will explain the techniques and their results.
We will also give a brief overview of the preprocessing steps we performed.

2.1 Prophet approch

Prophet is a predictive tool developed by Facebook in 2017. It's designed to analyze time series that show patterns on different time scales, such as annually, weekly and daily.
It also has advanced capabilities to handle the holiday effect, which is a very important factor in many practical cases. [2]

Here's a high-level overview of how Prophet works: The Trend Model in Prophet attempts to fit the data into a piecewise linear or logistic growth curve trend. This allows the model to handle non-linear growths by introducing change points at which the rate is allowed to change.
The Seasonality Model in Prophet is modelled using Fourier series. By default, Prophet will model seasonality with a period of one week and one year, but this can be set to other periods.
Holiday Effect: The user may provide Prophet with a list of dates that represent holidays or special events. Prophet will include these dates as additional regressors in its model.
Error Model: The error term is assumed to be normally distributed. The data is divided into a training set (80% of the data) and a testing set (the remaining 20%).
A Prophet model is created and fitted to the training data. The script then adjusts the predicted values by fitting a quadratic function to the data and generating new y values using this function. These new y values are then assigned back to the 'yhat' column in the forecast DataFrame. The script then saves the adjusted predictions and the original predictions to CSV files.

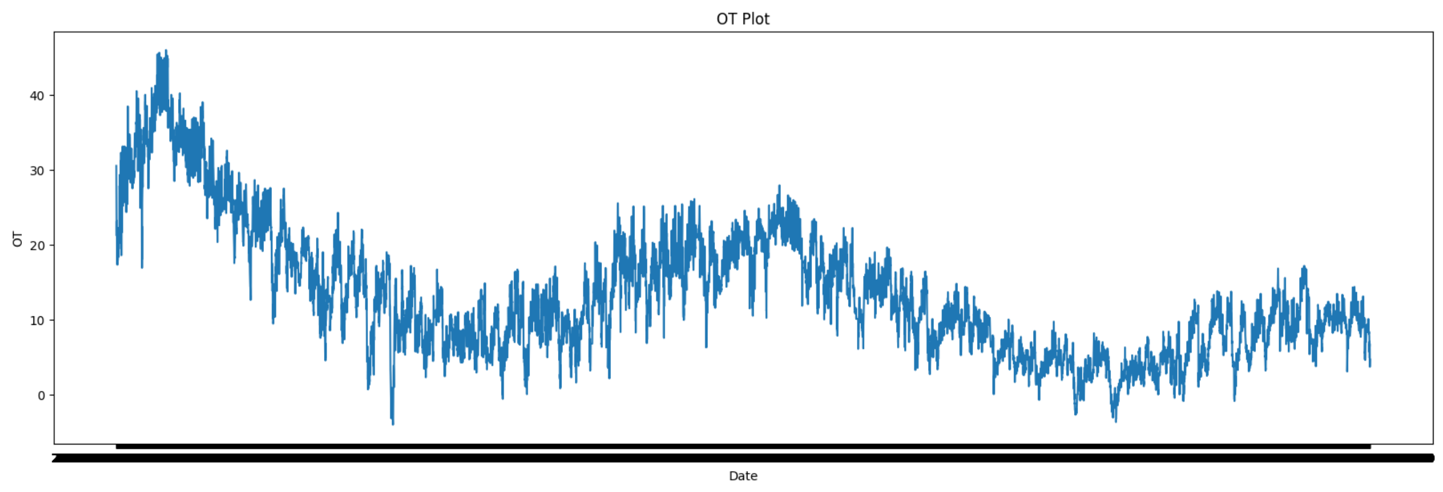


Figure 1: Overview of the OT dataset distribution

2.2 Related works

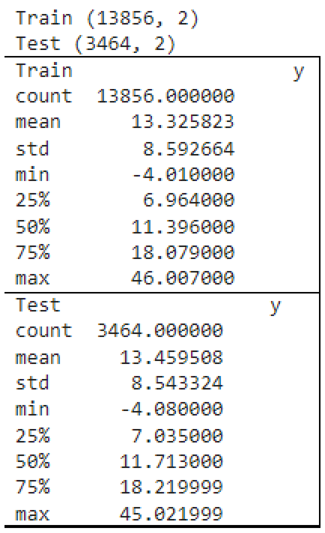
The Prophet model is a statistical model. It employs past data to infer future trends, yet it lacks an understanding of the underlying process that generates the data. Consequently, its predictions are probabilistic, encompassing uncertainty intervals.
The Prophet model is particularly useful for business forecasting tasks due to its ability to generate forecasts in a relatively short timeframe, and its simplicity does not require a high level of familiarity with time-series data.
There are numerous related works that have explored different aspects of this technique. Taylor SJ and Letham B. [3] employed a decomposable time series model comprising three main components: trend, seasonality, and holidays.
Meng, J, Yang, X, Yang, C and Liu, Y [7] conducted a deeply comparative experiment on the LSTM and the Prophet model on pharmaceutical sales prediction.

This is a brief summary of the research conducted on this model and the time series classification. It is an suitable introduction for anyone interested in this field of study.

2.3 Dataset and Pre-processing

The initial dataset consists of three columns (Id, Date and OT) and contains 17,320 times series, spanning from 01/07/2016 00:00 to 22/06/2018 15:00. There are no missing values.
The 'OT' values are distributed between -4.08 and 46.007, with a mean of 13.352. The majority of the dataset is represented by the value of 18.15, which is the 75th percentile. In this case, the dataset was divided into a training set (80% of the data) and a testing set (20% of the data) in order to predict the 100 subsequent values (until 26/06/2018 19:00).

This division yielded 13,856 samples for the training set and 3,464 samples for the testing set, with a diverse range of data to avoid overfitting and to prevent the emergence of discriminatory patterns that favour certain tendencies over others. The 80/20 split was randomly chosen (Figure 2).



```
Train (13856, 2)
Test (3464, 2)
Train                  y
count  13856.000000
mean      13.325823
std        8.592664
min       -4.010000
25%        6.964000
50%       11.396000
75%       18.079000
max       46.007000
Test                 y
count  3464.000000
mean     13.459508
std       8.543324
min      -4.080000
25%       7.035000
50%      11.713000
75%      18.219999
max      45.021999
```

Figure 2: Distribution of the Train/Test sets

2.4 Training

Before initiating the training of the Prophet model, it is necessary to rename the date column to ds (date-stamp) and the target variable (OT in our case) to y. This is because the Prophet model is designed to perform univariate time-series analysis on these two columns.
Then we create an instance of the prophet model and we fit the training data in it and lunch the model's training with the command

$$test\_forecast = model.predict(test)$$

.

This command will perform an analysis of the data and provide insights into the trends observed in the targeted column, with respect to daily, weekly and yearly periods.
We can also obtain plots of the different patterns (Figure3) with the command

$$model.plot\_components(forecast)$$

.

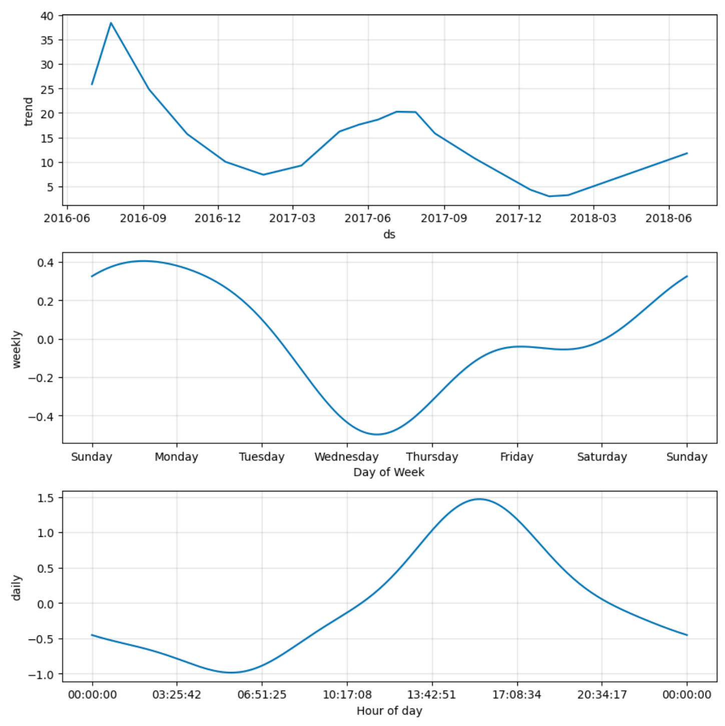


Figure 3: Plot of components

After running the model's training on the train set we can now define the desired time interval and the frequency (Hourly in our example) for predicting the 100 future values (until 28/06/2018 -17420 time series in total).
To do this we use the command:

$$model.make\_future\_dataframe(periods = 100, freq =' H')$$

After extracting the predicted values we can plot the whole result with matplotlib (Figure 4).

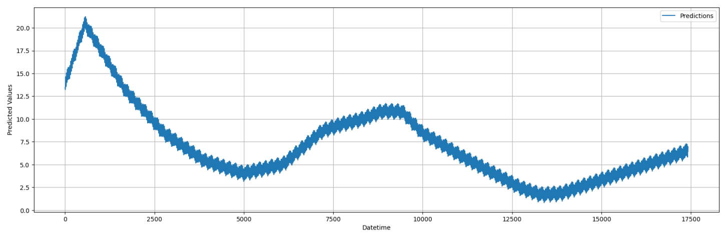
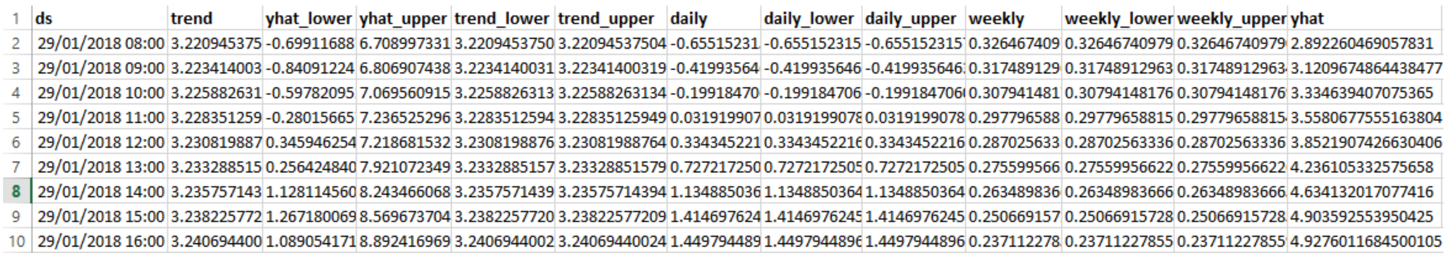


Figure 4: Plot of the future predictions



	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	daily	daily_lower	daily_upper	weekly	weekly_lower	weekly_upper	yhat
2	29/01/2018 08:00	3.220945375	-0.69911688	6.708997331	3.2209453750	3.22094537504	-0.65515231	-0.655152315	-0.655152315	0.326467409	0.32646740979	0.32646740979	2.892260469057831
3	29/01/2018 09:00	3.223414003	-0.84091224	6.806907438	3.2234140031	3.22341400319	-0.41993564	-0.419935646	-0.419935646	0.317489129	0.31748912963	0.31748912963	3.1209674864438477
4	29/01/2018 10:00	3.225882631	-0.59782095	7.069560915	3.2258826313	3.22588263134	-0.19918470	-0.199184706	-0.199184706	0.307941481	0.30794148176	0.30794148176	3.334639407075365
5	29/01/2018 11:00	3.228351259	-0.28015665	7.236525296	3.2283512594	3.22835125949	0.031919907	0.0319199078	0.0319199078	0.297796588	0.29779658815	0.29779658815	3.5580677555163804
6	29/01/2018 12:00	3.230819887	0.345946254	7.218681532	3.2308198876	3.23081988764	0.334345221	0.3343452216	0.3343452216	0.287025633	0.28702563336	0.28702563336	3.8521907426630406
7	29/01/2018 13:00	3.233288515	0.256424840	7.921072349	3.2332885157	3.23328851579	0.727217250	0.7272172505	0.7272172505	0.275599566	0.27559956622	0.27559956622	4.236105332575658
8	29/01/2018 14:00	3.235757143	1.128114560	8.243466068	3.2357571439	3.23575714394	1.134885036	1.1348850364	1.1348850364	0.263489836	0.26348983666	0.26348983666	4.634132017077416
9	29/01/2018 15:00	3.238225772	1.267180069	8.569673704	3.2382257720	3.23822577209	1.414697624	1.4146976245	1.4146976245	0.250669157	0.25066915728	0.25066915728	4.903592553950425
10	29/01/2018 16:00	3.240694400	1.089054171	8.892416969	3.2406944002	3.24069440024	1.449794489	1.4497944896	1.4497944896	0.237112278	0.23711227855	0.23711227855	4.9276011684500105

Figure 5: Example of the Tendencies

2.5 Results

From the preceding experiments, it can be observed that the Prophet model predicted future values 'yhat' based on the combination of hourly, daily and yearly tendencies, with respect to uncertainty intervals ('yhat_lower' and 'yhat_upper'). The Prophet model employs a technique known as Bayesian sampling to estimate these uncertainties [6].
In summary, 'yhat' represents the model's best estimate of the value, while 'yhat_upper' and 'yhat_lower' provide a range of possible values based on the model's estimated uncertainty.

To evaluate this solution's performance we used the MAE (Mean Absolute Error) metric:

$$MAE = \frac{1}{N}\sum_{i=0}^{N}|Y_i - f(X_i)|$$

The scores obtained on the train/test sets and Kaggle submission were mae_tt=0.7122 and mae_f=0.8912, respectively.

2.6 Conclusion

To summarize, in this section, we have discussed the methodology that we have tested for forecasting the next values of the time series. The Prophet model was explained in detail, which is a statistical model based on the concept of "tendencies" of certain patterns or in the data.
Additionally, we provided a brief overview of related work and some of the research focused on prophet-based methods in recent years.

In addition, the dataset and data pre-processing steps were discussed. Finally, the training parameters of the model were presented.
This section provides a comprehensive overview of the methods used for OT values forecast and the steps taken to prepare the data for training the Prophet model.
Although this model does not produce outstanding results, it is easy to deploy and can be implemented quickly.

3 Exchange Rate prediction

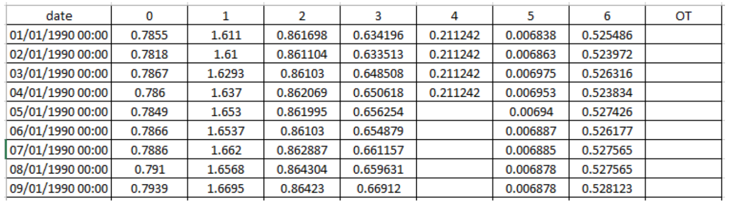
In this second part of the project, we're going to provide a solution to tackle the forecasting problem, to forecast the next 100 values of the '6' variable for the exchange rate dataset while handling the missing values.

3.1 Related works

There are numerous methodologies for performing multivariate forecasting. Two prominent examples are the RNN (Recurrent Neural Network) and the LSTM (Long Short-Term Memory) approach, as illustrated in the A. Sagheer, M. Kotb article [8]. In this study, the authors employed deep LSTM recurrent networks to predict petroleum production. Additionally, the combination of Prophet or Neural Prophet with a regressor enables the prediction of variables based on numerous inputs, as demonstrated in the Oskar.T,1,H.Hewamalagec, Polina P, N.Laptevb C.Bergmeirc study [9], which compared the two previously mentioned models. Furthermore, the article JAGANNATHAN.J, DIVYA, C [10] presents an example of multivariate Prophet being used to predict climate change.

3.2 Dataset and Pre-processing

In order to achieve our objective, we have constructed a dataset comprising 7489 daily currency exchange rate time series, each of which represents a different data point (columns [0-6] and OT). However, 25% of the data is missing (33283 missing values out of the original 59912).
The following table shows the head of the original table.



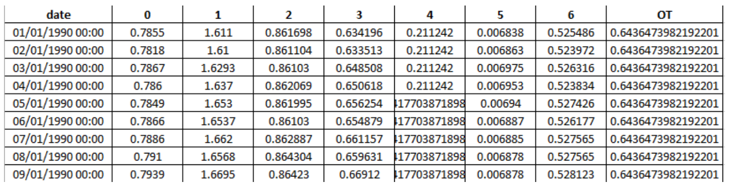
date	0	1	2	3	4	5	6	OT
01/01/1990 00:00	0.7855	1.611	0.861698	0.634196	0.211242	0.006838	0.525486	
02/01/1990 00:00	0.7818	1.61	0.861104	0.633513	0.211242	0.006863	0.523972	
03/01/1990 00:00	0.7867	1.6293	0.86103	0.648508	0.211242	0.006975	0.526316	
04/01/1990 00:00	0.786	1.637	0.862069	0.650618	0.211242	0.006953	0.523834	
05/01/1990 00:00	0.7849	1.653	0.861995	0.656254		0.00694	0.527426	
06/01/1990 00:00	0.7866	1.6537	0.86103	0.654879		0.006887	0.526177	
07/01/1990 00:00	0.7886	1.662	0.862887	0.661157		0.006885	0.527565	
08/01/1990 00:00	0.791	1.6568	0.864304	0.659631		0.006878	0.527565	
09/01/1990 00:00	0.7939	1.6695	0.86423	0.66912		0.006878	0.528123	

Figure 6: Exchange Rate with missing values

In order to overcome the problem of missing values, three methods of completion were tried:

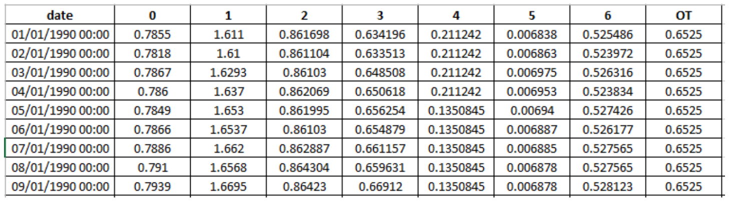
- Mean Completion: Each of the missing values are replaced with the mean of the column df.fillna(df.mean())
- Median Completion: The missing values are replaced with the mean of the column. df.fillna(df.median())
- Interpolation Completion: The missing values are filled based on a second order (quadratic) polynomial interpolation. df.interpolate(method='polynomial', order=2).fillna(method='ffill'), we used the second degree polynom to avoid data overfitting.

We obtained the three following results(Figure 7, 8 and 9)



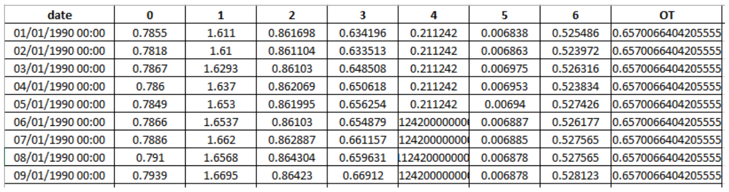
date	0	1	2	3	4	5	6	OT
01/01/1990 00:00	0.7855	1.611	0.861698	0.634196	0.211242	0.006838	0.525486	0.643647398192201
02/01/1990 00:00	0.7818	1.61	0.861104	0.633513	0.211242	0.006863	0.523972	0.643647398192201
03/01/1990 00:00	0.7867	1.6293	0.86103	0.648508	0.211242	0.006975	0.526316	0.643647398192201
04/01/1990 00:00	0.786	1.637	0.862069	0.650618	0.211242	0.006953	0.523834	0.643647398192201
05/01/1990 00:00	0.7849	1.653	0.861995	0.656254	17703871898	0.00694	0.527426	0.643647398192201
06/01/1990 00:00	0.7866	1.6537	0.86103	0.654879	17703871898	0.006887	0.526177	0.643647398192201
07/01/1990 00:00	0.7886	1.662	0.862887	0.661157	17703871898	0.006885	0.527565	0.643647398192201
08/01/1990 00:00	0.791	1.6568	0.864304	0.659631	17703871898	0.006878	0.527565	0.643647398192201
09/01/1990 00:00	0.7939	1.6695	0.86423	0.66912	17703871898	0.006878	0.528123	0.643647398192201

Figure 7: Mean Completion



date	0	1	2	3	4	5	6	OT
01/01/1990 00:00	0.7855	1.611	0.861698	0.634196	0.211242	0.006838	0.525486	0.6525
02/01/1990 00:00	0.7818	1.61	0.861104	0.633513	0.211242	0.006863	0.523972	0.6525
03/01/1990 00:00	0.7867	1.6293	0.86103	0.648508	0.211242	0.006975	0.526316	0.6525
04/01/1990 00:00	0.786	1.637	0.862069	0.650618	0.211242	0.006953	0.523834	0.6525
05/01/1990 00:00	0.7849	1.653	0.861995	0.656254	0.1350845	0.00694	0.527426	0.6525
06/01/1990 00:00	0.7866	1.6537	0.86103	0.654879	0.1350845	0.006887	0.526177	0.6525
07/01/1990 00:00	0.7886	1.662	0.862887	0.661157	0.1350845	0.006885	0.527565	0.6525
08/01/1990 00:00	0.791	1.6568	0.864304	0.659631	0.1350845	0.006878	0.527565	0.6525
09/01/1990 00:00	0.7939	1.6695	0.86423	0.66912	0.1350845	0.006878	0.528123	0.6525

Figure 8: Median Completion



date	0	1	2	3	4	5	6	OT
01/01/1990 00:00	0.7855	1.611	0.861698	0.634196	0.211242	0.006838	0.525486	0.6570066404205555
02/01/1990 00:00	0.7818	1.61	0.861104	0.633513	0.211242	0.006863	0.523972	0.6570066404205555
03/01/1990 00:00	0.7867	1.6293	0.86103	0.648508	0.211242	0.006975	0.526316	0.6570066404205555
04/01/1990 00:00	0.786	1.637	0.862069	0.650618	0.211242	0.006953	0.523834	0.6570066404205555
05/01/1990 00:00	0.7849	1.653	0.861995	0.656254	0.211242	0.00694	0.527426	0.6570066404205555
06/01/1990 00:00	0.7866	1.6537	0.86103	0.654879	124200000000	0.006887	0.526177	0.6570066404205555
07/01/1990 00:00	0.7886	1.662	0.862887	0.661157	124200000000	0.006885	0.527565	0.6570066404205555
08/01/1990 00:00	0.791	1.6568	0.864304	0.659631	112420000000	0.006878	0.527565	0.6570066404205555
09/01/1990 00:00	0.7939	1.6695	0.86423	0.66912	124200000000	0.006878	0.528123	0.6570066404205555

Figure 9: Interpolation Completion

3.3 Experiments

In order to address the problem, three different approaches were tested:

1. Univariate Prophet : We Applied the Step A method on this problem, and we tested it on the three completed version of the dataset(Figure 10, 11, 12)

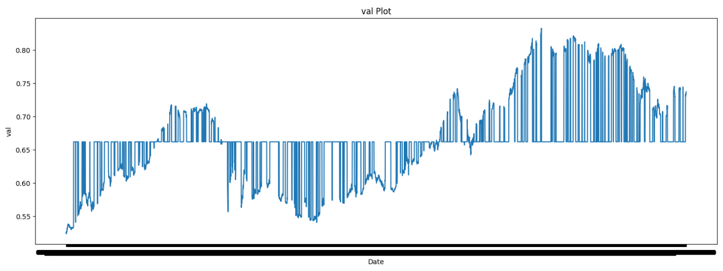


Figure 10: Data distribution mean completion

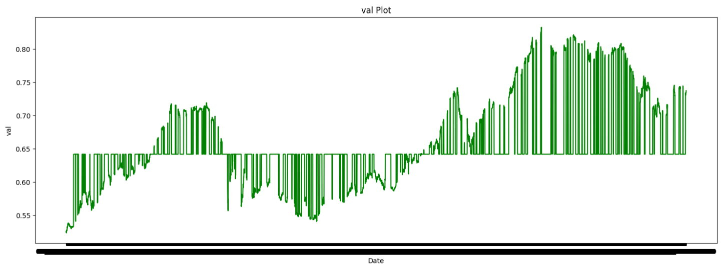


Figure 11: Data distribution median completion

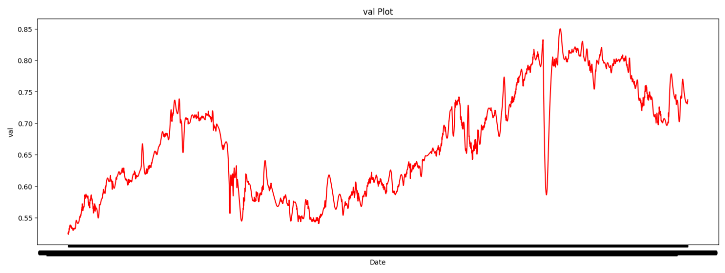
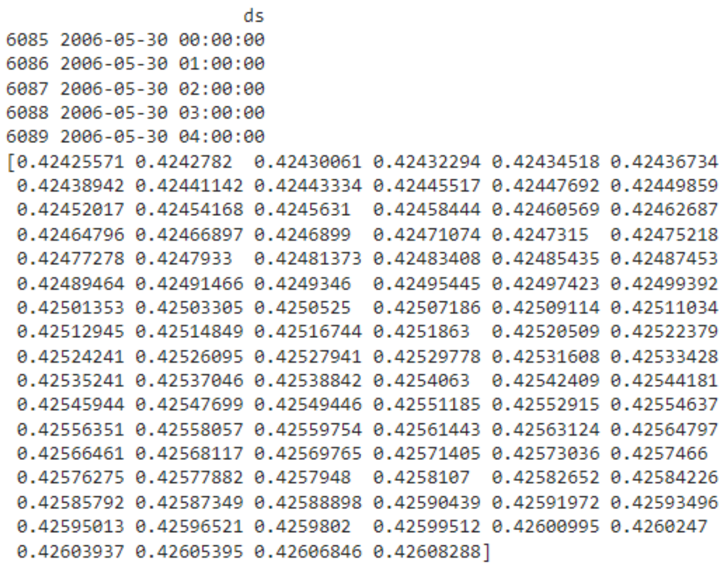


Figure 12: Data distribution interpolation completion

The two columns, previously designated as 'date' and '6', were renamed as 'ds' and 'y', respectively. The dataset was then divided into two distinct sets, 80% of which constituted the training set and 20% the test set. This resulted in a total of 5990 instances in the training set and 1498 instances in the test set.
The training set was then fitted into the Prophet model, resulting in the prediction of the figure 13 and 14 .



```
                         ds
6085 2006-05-30 00:00:00
6086 2006-05-30 01:00:00
6087 2006-05-30 02:00:00
6088 2006-05-30 03:00:00
6089 2006-05-30 04:00:00
[0.42425571 0.4242782  0.42430061 0.42432294 0.42434518 0.42436734
 0.42438942 0.42441142 0.42443334 0.42445517 0.42447692 0.42449859
 0.42452017 0.42454168 0.4245631  0.42458444 0.42460569 0.42462687
 0.42464796 0.42466897 0.4246899  0.42471074 0.4247315  0.42475218
 0.42477278 0.4247933  0.42481373 0.42483408 0.42485435 0.42487453
 0.42489464 0.42491466 0.4249346  0.42495445 0.42497423 0.42499392
 0.42501353 0.42503305 0.4250525  0.42507186 0.42509114 0.42511034
 0.42512945 0.42514849 0.42516744 0.4251863  0.42520509 0.42522379
 0.42524241 0.42526095 0.42527941 0.42529778 0.42531608 0.42533428
 0.42535241 0.42537046 0.42538842 0.4254063  0.42542409 0.42544181
 0.42545944 0.42547699 0.42549446 0.42551185 0.42552915 0.42554637
 0.42556351 0.42558057 0.42559754 0.42561443 0.42563124 0.42564797
 0.42566461 0.42568117 0.42569765 0.42571405 0.42573036 0.4257466
 0.42576275 0.42577882 0.4257948  0.4258107  0.42582652 0.42584226
 0.42585792 0.42587349 0.42588898 0.42590439 0.42591972 0.42593496
 0.42595013 0.42596521 0.4259802  0.42599512 0.42600995 0.4260247
 0.42603937 0.42605395 0.42606846 0.42608288]
```

Figure 13: Predictions using univariate Prophet model

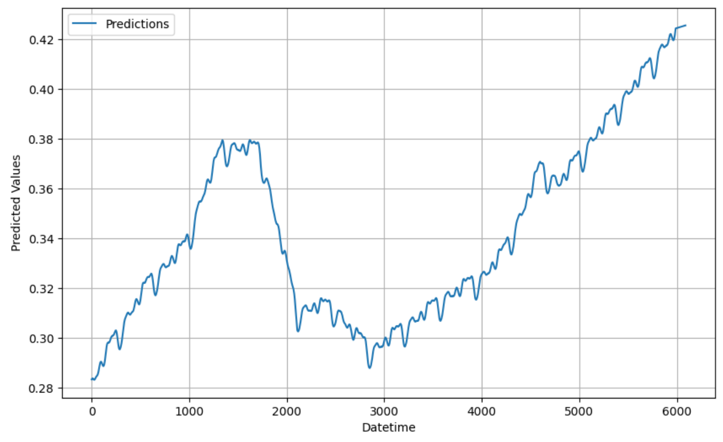
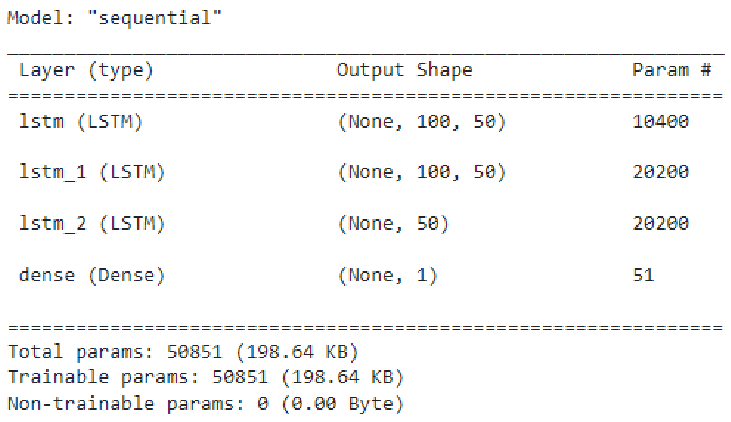


Figure 14: Predictions using interpolation completion and Prophet

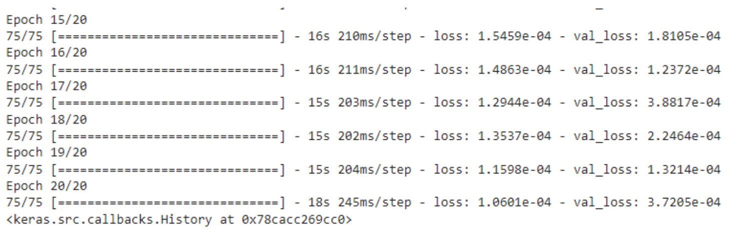
2. LSTM (Long Short-Term Memory): Once the dataset had been completed by the preceding techniques, we proceeded to analyse the '6' column.
In order to ensure that the values were within the appropriate range for LSTM models, we applied a MinMax scaler to manipulate them between 0 and 1. In order to avoid having imbalanced sets, we then split the data into 65% Train and 35% Test. Subsequently, the input was reshaped to a format of [samples, time steps, features], which is the requisite format for LSTM.
The Sequential model was then created using Keras, which is a type of Recurrent Neural Network (RNN) that employs Long Short-Term Memory (LSTM) units, which are a specialised form of RNN unit that is effective for dealing with long-term dependencies in sequence data.
We executed the command *model.summary*() to obtain our model's architecture (Figure 15).



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 lstm_1 (LSTM)               (None, 100, 50)           20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50851 (198.64 KB)
Trainable params: 50851 (198.64 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 15: The Model's architecture

The data was then incorporated into the Keras model and training was initiated for 20 epochs with a batch size of 64.
The model's performance (Loss function [11]) was evaluated after each epoch.



```
Epoch 15/20
75/75 [==============================] - 16s 210ms/step - loss: 1.5459e-04 - val_loss: 1.8105e-04
Epoch 16/20
75/75 [==============================] - 16s 211ms/step - loss: 1.4863e-04 - val_loss: 1.2372e-04
Epoch 17/20
75/75 [==============================] - 15s 203ms/step - loss: 1.2944e-04 - val_loss: 3.8817e-04
Epoch 18/20
75/75 [==============================] - 15s 202ms/step - loss: 1.3537e-04 - val_loss: 2.2464e-04
Epoch 19/20
75/75 [==============================] - 15s 204ms/step - loss: 1.1598e-04 - val_loss: 1.3214e-04
Epoch 20/20
75/75 [==============================] - 18s 245ms/step - loss: 1.0601e-04 - val_loss: 3.7205e-04
<keras.src.callbacks.History at 0x78cacc269cc0>
```

Figure 16: Keras model's performance after every epoch

Subsequently, the model's prediction (Blue) was displayed on the train (Orange) and test (Green) sets of the 100 most recent values (Figure 17).

Finally, an inverse transform was applied in order to eliminate the data scaling, resulting in the plot of the predicted values (Figure 18).

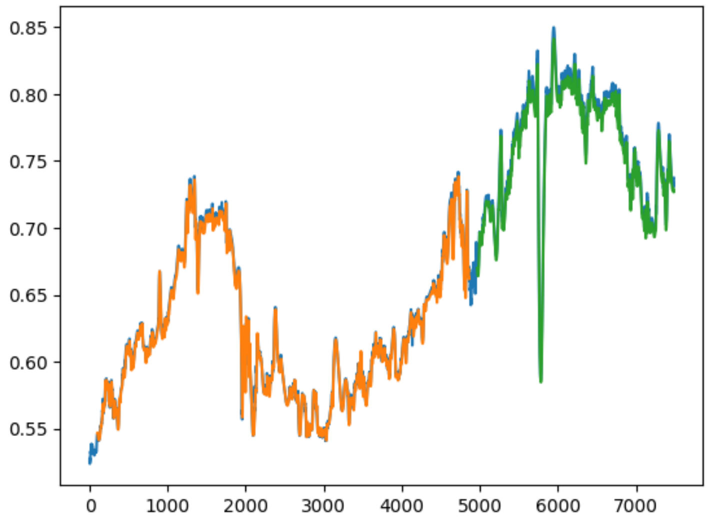


Figure 17: Model's prediction on train and test sets

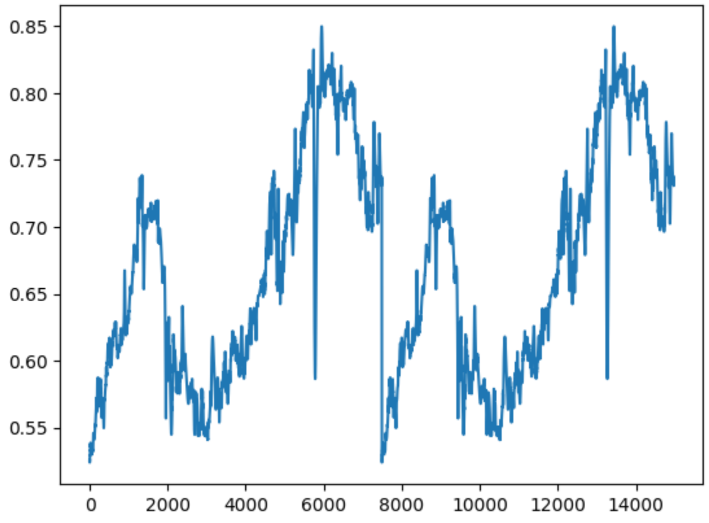
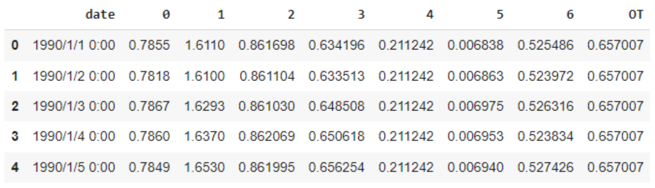


Figure 18: LSTM model's prediction

The same methodology was employed on the other completed datasets (mean and median), the results of which will be discussed later.

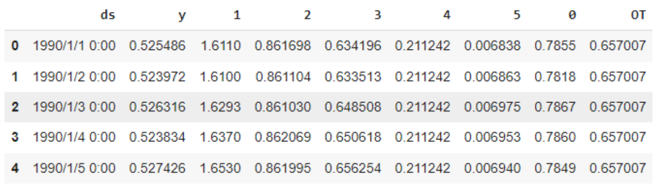
3. Multivariate Prophet : Now, we employed the previously cited prophet model in a multivariate approach. The dataset, which had been completed by the interpolation method, was imported and the start and end dates of the training and testing sets were defined. (Figure 19).

   We plotted the different data distribution on the same figure to observe the relation between the variables visually (Figure 21).
   Then we rename the date and the '6' column to 'ds' and 'y' and we select the relevant columns (Figure 20),



	date	0	1	2	3	4	5	6	OT
0	1990/1/1 0:00	0.7855	1.6110	0.861698	0.634196	0.211242	0.006838	0.525486	0.657007
1	1990/1/2 0:00	0.7818	1.6100	0.861104	0.633513	0.211242	0.006863	0.523972	0.657007
2	1990/1/3 0:00	0.7867	1.6293	0.861030	0.648508	0.211242	0.006975	0.526316	0.657007
3	1990/1/4 0:00	0.7860	1.6370	0.862069	0.650618	0.211242	0.006953	0.523834	0.657007
4	1990/1/5 0:00	0.7849	1.6530	0.861995	0.656254	0.211242	0.006940	0.527426	0.657007

Figure 19: Head of completed dataset



	ds	y	1	2	3	4	5	0	OT
0	1990/1/1 0:00	0.525486	1.6110	0.861698	0.634196	0.211242	0.006838	0.7855	0.657007
1	1990/1/2 0:00	0.523972	1.6100	0.861104	0.633513	0.211242	0.006863	0.7818	0.657007
2	1990/1/3 0:00	0.526316	1.6293	0.861030	0.648508	0.211242	0.006975	0.7867	0.657007
3	1990/1/4 0:00	0.523834	1.6370	0.862069	0.650618	0.211242	0.006953	0.7860	0.657007
4	1990/1/5 0:00	0.527426	1.6530	0.861995	0.656254	0.211242	0.006940	0.7849	0.657007

Figure 20: Relevant columns

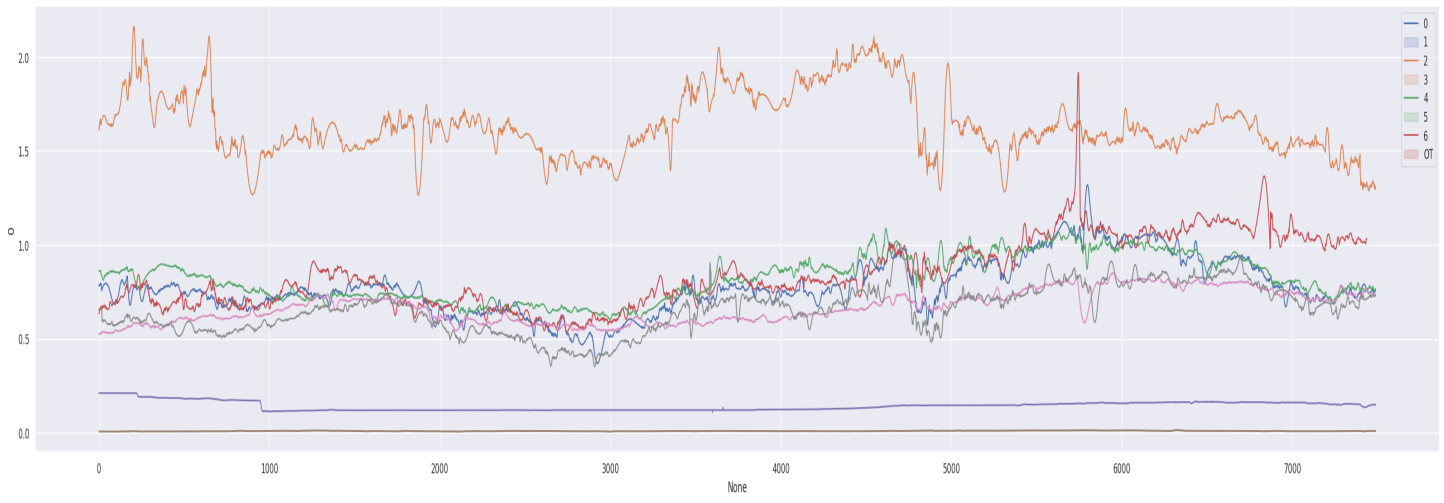
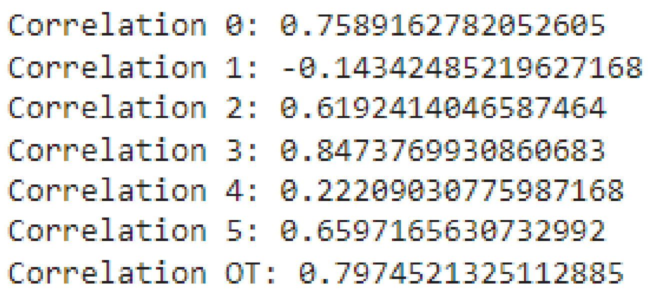


Figure 21: Dataset distribution

And to understand the relation between the different variables we calculated the correlation between each variable and the '6' variable (Figure 22)



```
Correlation 0: 0.7589162782052605
Correlation 1: -0.14342485219627168
Correlation 2: 0.6192414046587464
Correlation 3: 0.8473769930860683
Correlation 4: 0.22209030775987168
Correlation 5: 0.6597165630732992
Correlation OT: 0.7974521325112885
```

Figure 22: Correlation between variables

We notice that the best matching variables are the '3', 'OT' and '0'.
These values are close to 1, indicating that there is a high correlation between the three columns and the '6' column.

We fitted the train/test sets into the Prophet model and we initiated the training process, then we defined a forecast time range of 100 and we plotted the forecast and the components chart.(Figure 23 and 24).
In the forecast graphic, the light blue chart represents the uncertainity interval and the black dots represents the real values.

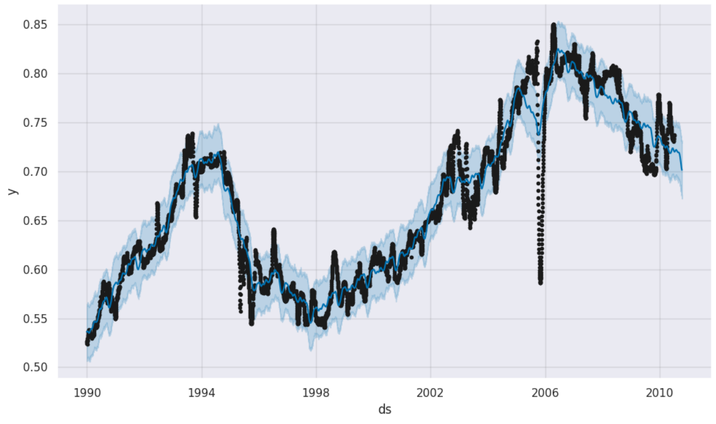


Figure 23: Forecast plot visualisation

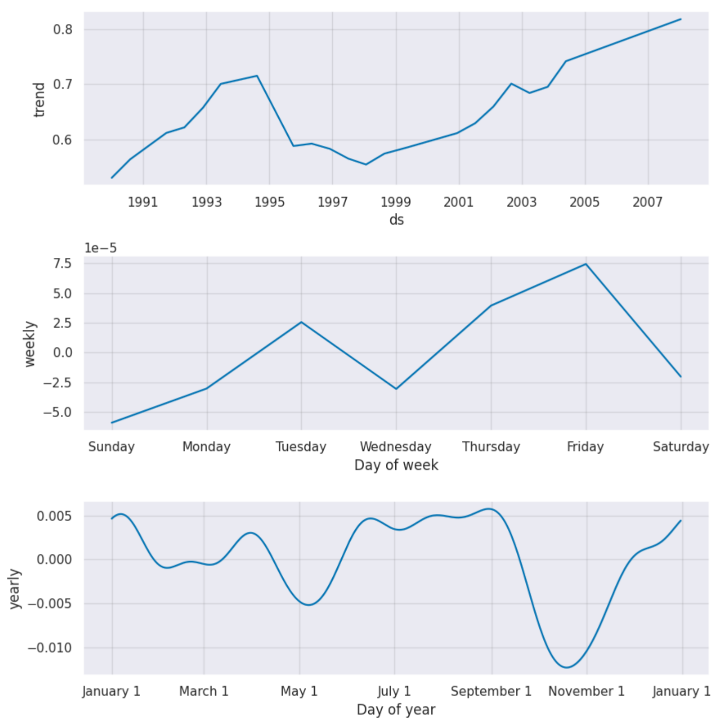


Figure 24: Forecasting components

By adding the weekly and the yearly seasonality to our initial baseline model, we displayed the 100 future predicted values and we plotted a second time the components (Figure 25).

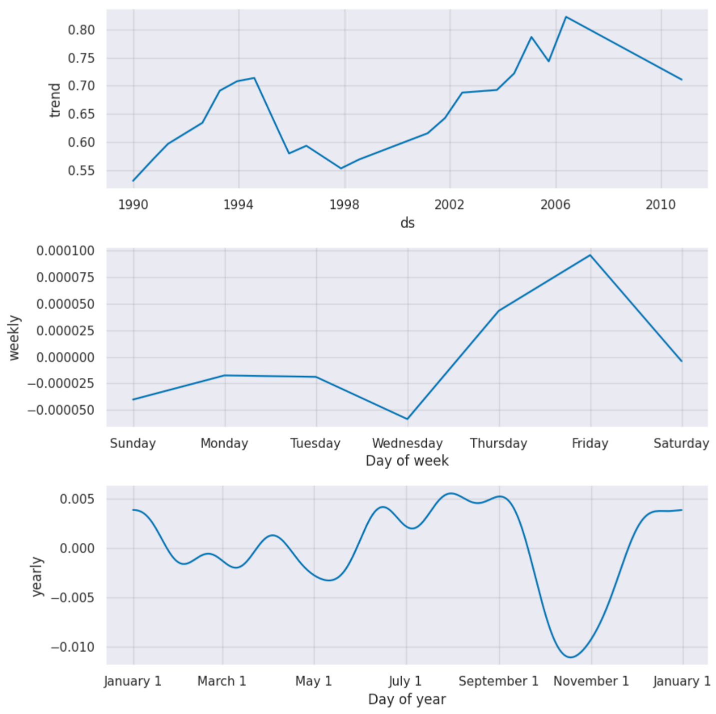


Figure 25: Future forecasting components

We observe that the yearly and the weekly tendencies didn't change very much from our baseline model components. However, the trend exhibited a notable change when the future prediction period was extended.

Finally, in order to enhance the predictive capabilities of our model, we incorporated an additional regressor comprising the columns '0', '3' and 'OT'. We standardised the input values within the regressor and tested the model's predictions for the subsequent 100 values on each case.

3.4 Results

To summarize the previous experiments, we have collected all the results in the table (Table 1).
It is worth noting that the use of other correlated variables enhances the accuracy of our prediction, but we should also be careful to not use too many variables to avoid every column's estimation error (More columns=more error per column).
In our case, the optimal combination was the second-degree interpolation completion of the initial dataset and the use of the values of the column '3' to estimate the future values of the column '6'.
This approach yielded an mean absolute error of 0.0134.

3.5 Conclusion

In this section, we have presented the methodology that we have tested for forecasting the next values of the time series. In detail, we have explained the three models that were used. Additionally, we have provided a brief overview of related work and some of the research focused on LSTM and multivariate prophet methods in recent years. Furthermore, we have discussed the dataset and data pre-processing steps. Finally, we have provided details of the training parameters of the model. Furthermore, it should be noted that there are other methods of completing missing values that have not been discussed in this section(for example, the completion of missing data using association rules). The primary objective in this report was to identify a model that produces acceptable results and is both fast and easy to implement.

4 Summary

In conclusion, the two sections discuss different methods for forecasting future values from past time series data. The first section presents the univariate prophet method, which is a statistical technique that uses tendency and seasonality to predict certain patterns or motifs in the data. Although this method is fast and simple to implement, it can be less precise to determine the appropriate values if the dataset's size were voluminous.
Conversely, the second section presents three distinct models for performing multivariate prediction from exchange rates data. The third model outperforms the first and second models in precision and in execution time.
The results of the two sections are encouraging the establishment of a data monitoring system, whether to track the temperature of the oil or to monitor the exchange rate, with the objective of anticipating potential disruptions and implementing proactive repair measures.

Method	Approach	Completion	Mae	Columns used
Univariate	Prophet	Mean	0.3419	/
		Median	0.3483	/
		Interpolation	**0.3292**	/
	LSTM	Mean	0.0440	/
		Median	0.0633	/
		Interpolation	**0.0357**	/
Multivariate	Prophet	Median	0.1010	/
		Interpolation	**0.0134**	'3'
			0.0226	'0' and 'OT'
			0.0228	'OT'
			0.0223	'0' and '3'
			0.0197	'3' and 'OT'
			0.0240	'3' and '0' and 'OT'

Table 1: Summary of the previous experiments result

References

[1] Makridakis, Spyros. Time series prediction: Forecasting the future and understanding the past. International Journal of Forecasting. 1994.

[2] TOWARDS DATA SCIENCE. Time Series Analysis in Python: An Introduction. 2018. [consulted on 01/04/2024]. https://towardsdatascience.com/time-series-analysis-in-python-an-introduction-70d5a5b1d52a

[3] MEDIUM. Prophet. 2020. Available at: https://medium.com/@vinay1996/prophet-f729877e7e5a

[4] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2

[5] P.Anand, M.Sharma, A.Sarolia, "Time Series Analysis for Predicting Anomaly using Prophet Models", AIJR Abstracts, pp. 70–70, Feb. 2024

[6] Facebook, Inc. (2024). 'Uncertainty Intervals — Prophet 0.7 documentation'. Available at: https://facebook.github.io/prophet/docs/uncertainty_intervals.html (Accessed: 20 April 2024).

[7] Meng, J., Yang, X., Yang, C., Liu, Y. (2021). Comparative Analysis of Prophet and LSTM Model. Journal of Physics: Conference Series, 1910(1): 12-59.

[8] A.Sagheer, M.Kotb, Time series forecasting of petroleum production using deep LSTM recurrent networks, Neurocomputing, Volume 323, 2019, https://doi.org/10.1016/j.neucom.2018.09.082.

[9] TRIEBE.O, HEWAMALAGE.H, PILYUGINA.P, et al. Neuralprophet: Explainable forecasting at scale. 2021. https://doi.org/10.48550/arXiv.2111.15397

[10] JAGANNATHAN.J, DIVYA, C. Time Series Analyzation and Prediction of Climate using Enhanced Multivariate Prophet. International Journal of Engineering Trends and Technology, 2021. https://ijettjournal.org/archive/ijett-v69i10p212.

[11] Ketkar, N. . Introduction to Keras. In: Deep Learning with Python. Apress, Berkeley, CA. 2017. https://doi.org/10.1007/978-1-4842-2766-4_7

[12] Chih-Hung Wu, Chian-Huei Wun and Hung-Ju Chou, "Using association rules for completing missing data," Fourth International Conference on Hybrid Intelligent Systems, Japan, 2004, pp. 236-241, doi: 10.1109/ICHIS.2004.91.