K-Means Fuzzy Time Series Forecasting

An AI6124 Final Project Presentation by
Reinelle Jan C. Bugnot
G2304329L
bugn0001@e.ntu.edu.sg

Nov. 2023

Chosen Dataset

CapitaLand Ascendas Real Estate Investment Trust (REIT)
15 year range, 1-day resolution

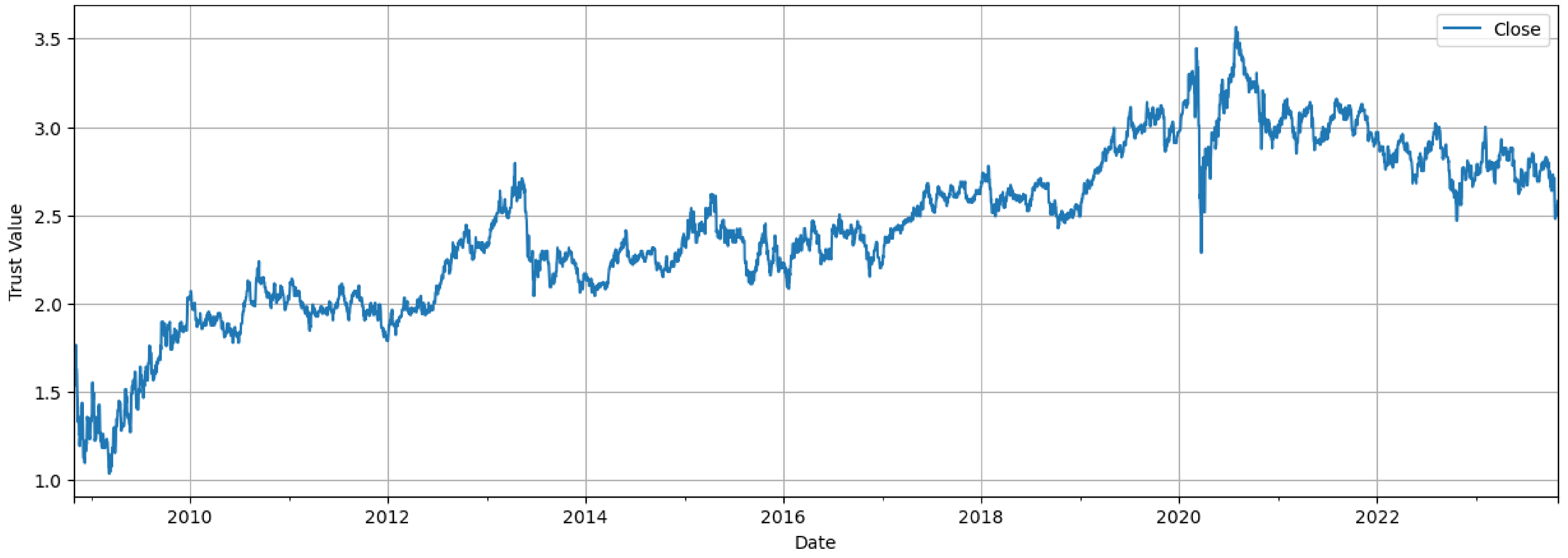


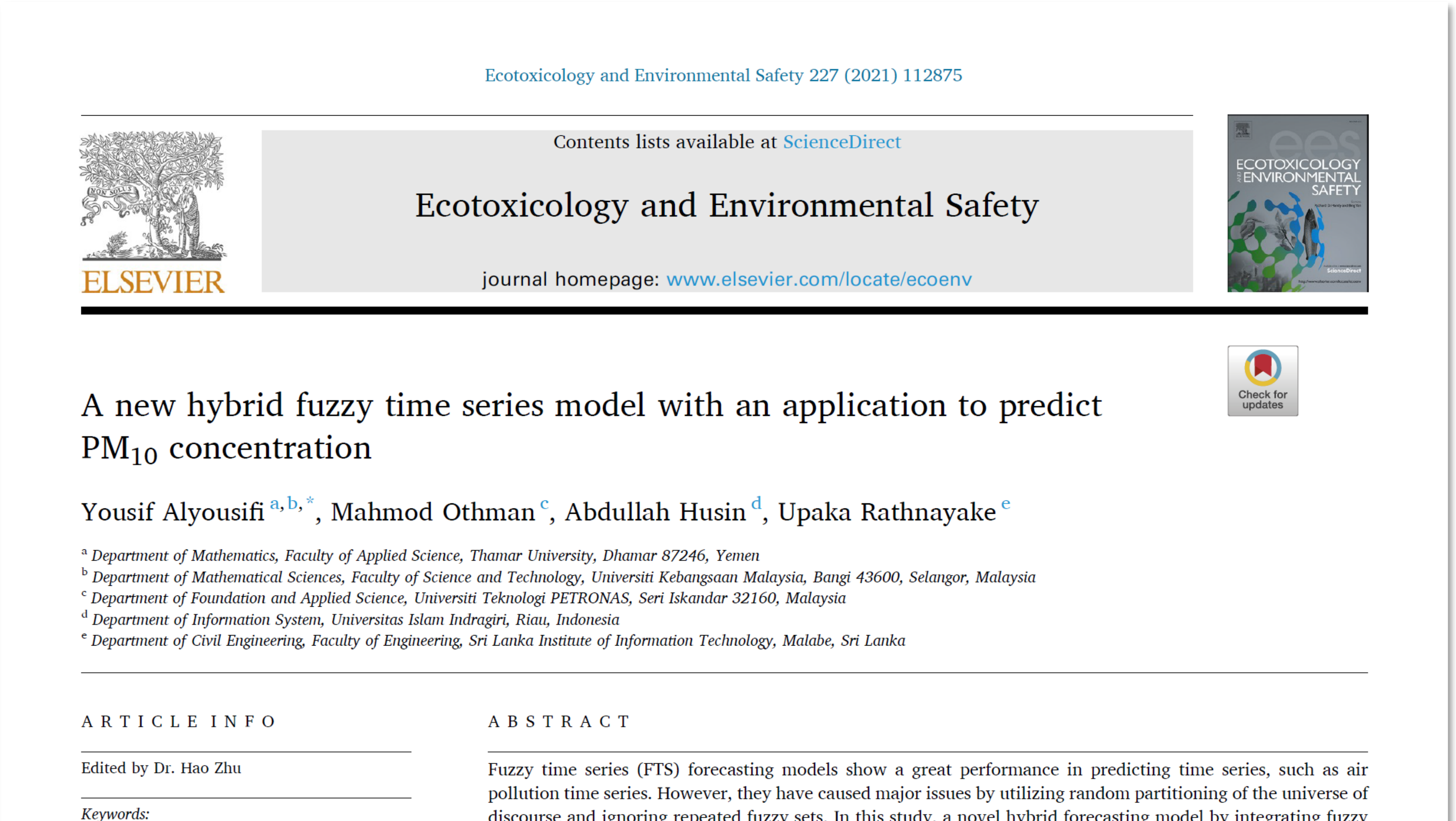
Fig. 1. A17U.SI Time Series

Preprocessing Step: Forward Fill missing data corresponding to Weekends and Holidays

Chosen Hybrid AI Method

K-Means Fuzzy Time Series

Reference Paper by Alyousifi, et. al.



Ecotoxicology and Environmental Safety 227 (2021) 112875

ELSEVIER

Contents lists available at ScienceDirect

Ecotoxicology and Environmental Safety

journal homepage: www.elsevier.com/locate/ecoenv

A new hybrid fuzzy time series model with an application to predict PM_{10} concentration

Yousif Alyousifi[a,b,*], Mahmod Othman[c], Abdullah Husin[d], Upaka Rathnayake[e]

[a] Department of Mathematics, Faculty of Applied Science, Thamar University, Dhamar 87246, Yemen
[b] Department of Mathematical Sciences, Faculty of Science and Technology, Universiti Kebangsaan Malaysia, Bangi 43600, Selangor, Malaysia
[c] Department of Foundation and Applied Science, Universiti Teknologi PETRONAS, Seri Iskandar 32160, Malaysia
[d] Department of Information System, Universitas Islam Indragiri, Riau, Indonesia
[e] Department of Civil Engineering, Faculty of Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

ARTICLE INFO

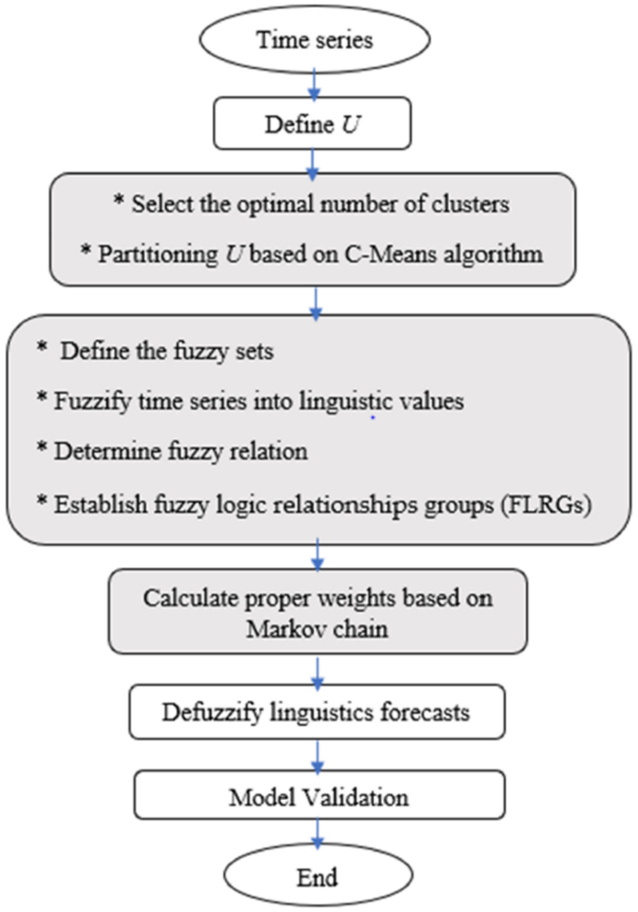
Edited by Dr. Hao Zhu

Keywords:

ABSTRACT

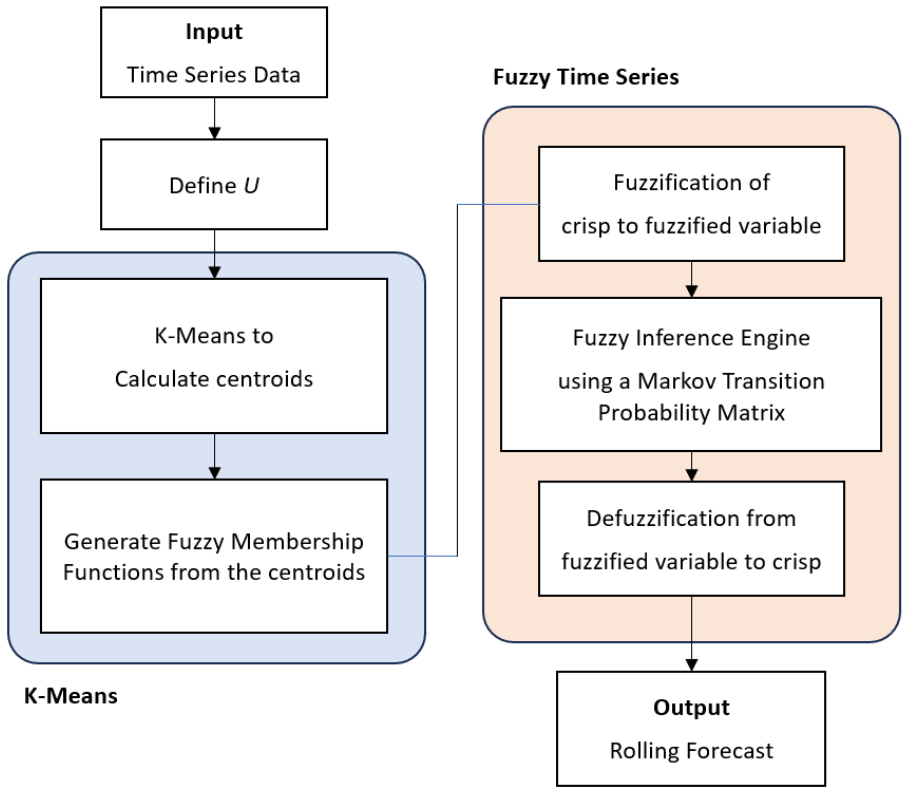
Fuzzy time series (FTS) forecasting models show a great performance in predicting time series, such as air pollution time series. However, they have caused major issues by utilizing random partitioning of the universe of discourse and ignoring repeated fuzzy sets. In this study, a novel hybrid forecasting model by integrating fuzzy

Chosen Hybrid AI Method



Time series
Define U
* Select the optimal number of clusters
* Partitioning U based on C-Means algorithm
* Define the fuzzy sets
* Fuzzify time series into linguistic values
* Determine fuzzy relation
* Establish fuzzy logic relationships groups (FLRGs)
Calculate proper weights based on Markov chain
Defuzzify linguistics forecasts
Model Validation
End

C-Means Fuzzy Time Series Model
(by Alyousifi, et. al.)



K-Means Fuzzy Time Series Model
(my implementation)

K-Means Clustering Algorithm

Generate Centroids. 1-dimensional K-Means clustering that clusters the datapoint on the vertical axis dimension (‘Trust Value’) based on a given number of centroids specified by the parameter *k*.

Fig. 1. A17U.SI Time Series

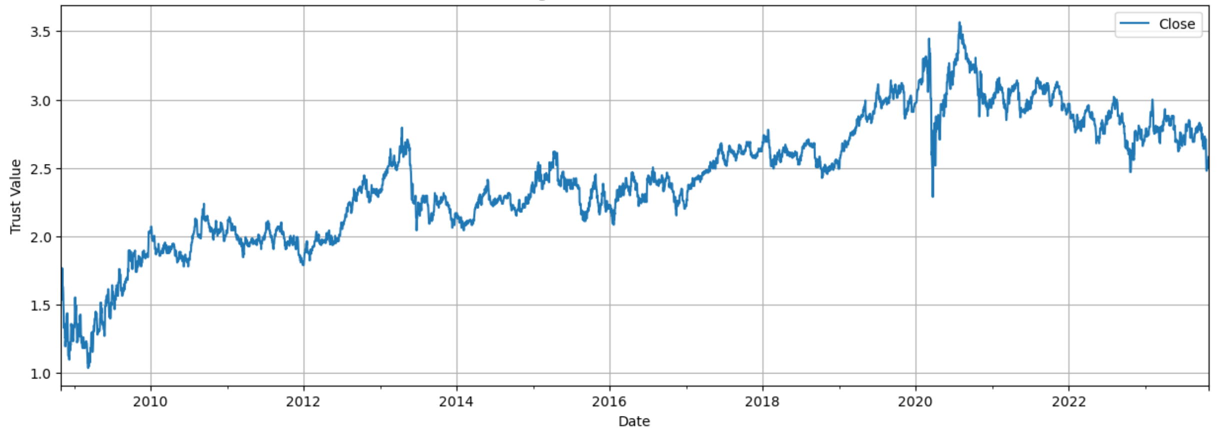
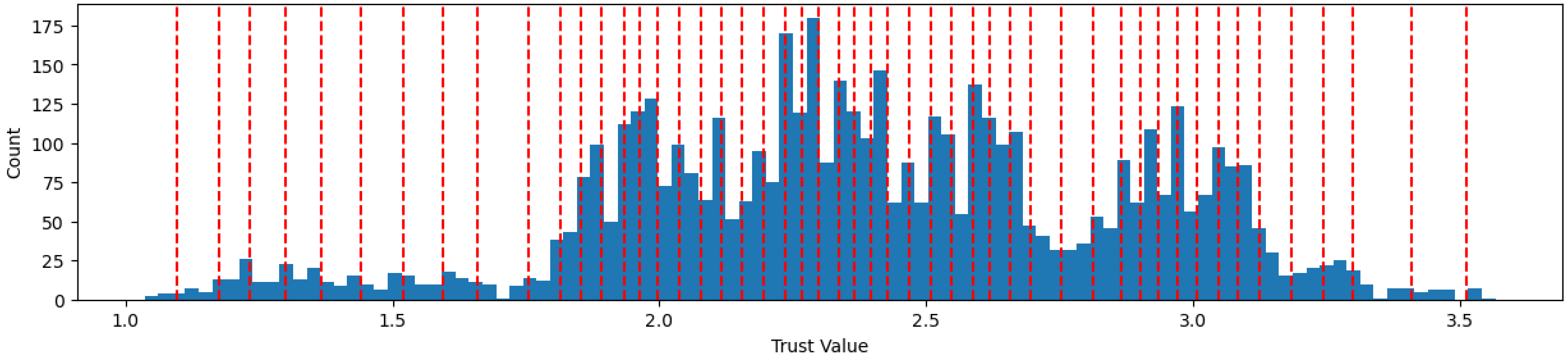


Fig. 4. Datapoint Distribution and Calculated Centroid Positions



K-Means Clustering Algorithm

Generate Membership Functions. From the generated centroids, we can generate triangular membership functions specified by parameters [a, b, c] where a is the centroid of the previous mf, b is the centroid of the current mf, and c is the centroid of the next mf.

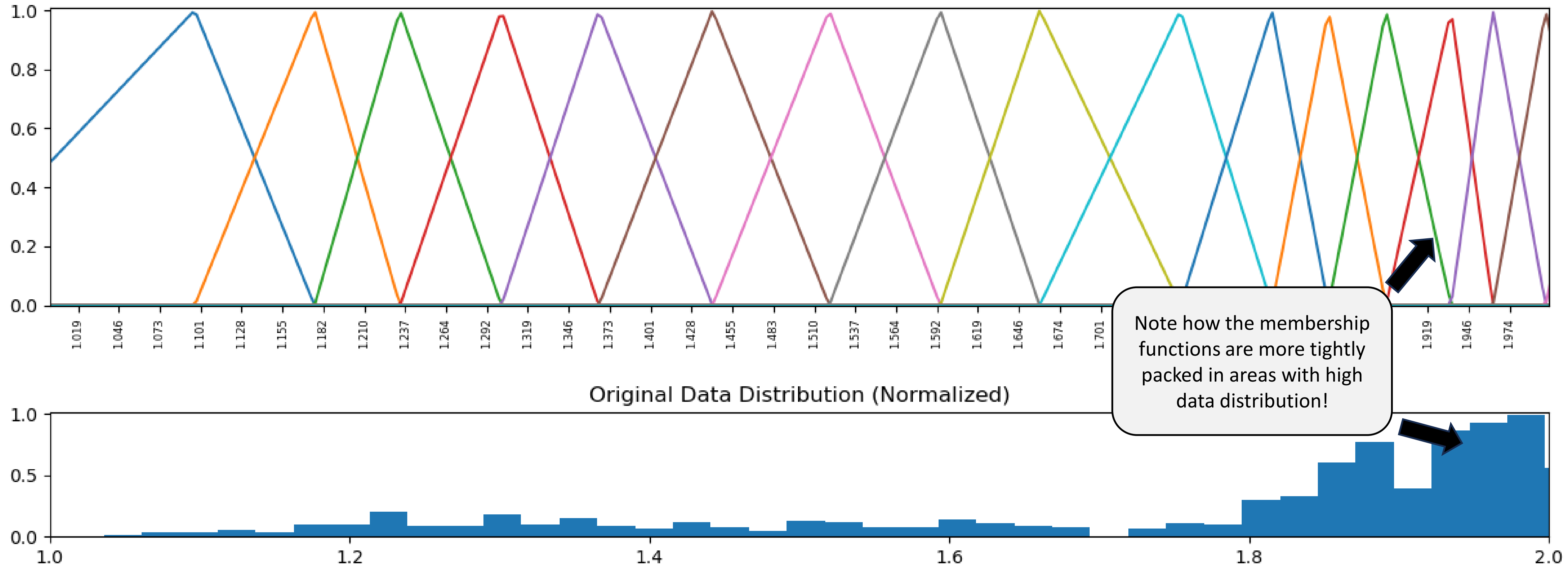


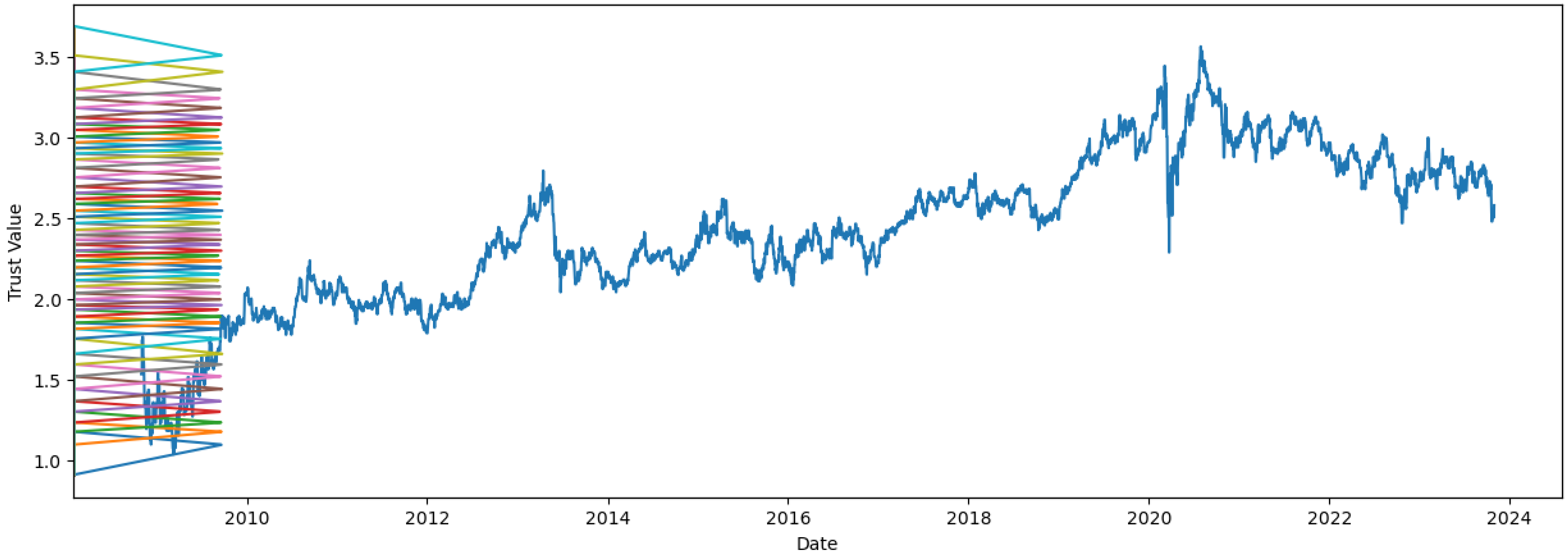
Fig. 5. Triangle Membership Functions from K-Means Centroids
(Zoomed in to the range of 1 to 2)

Note how the membership functions are more tightly packed in areas with high data distribution!

K-Means Clustering Algorithm

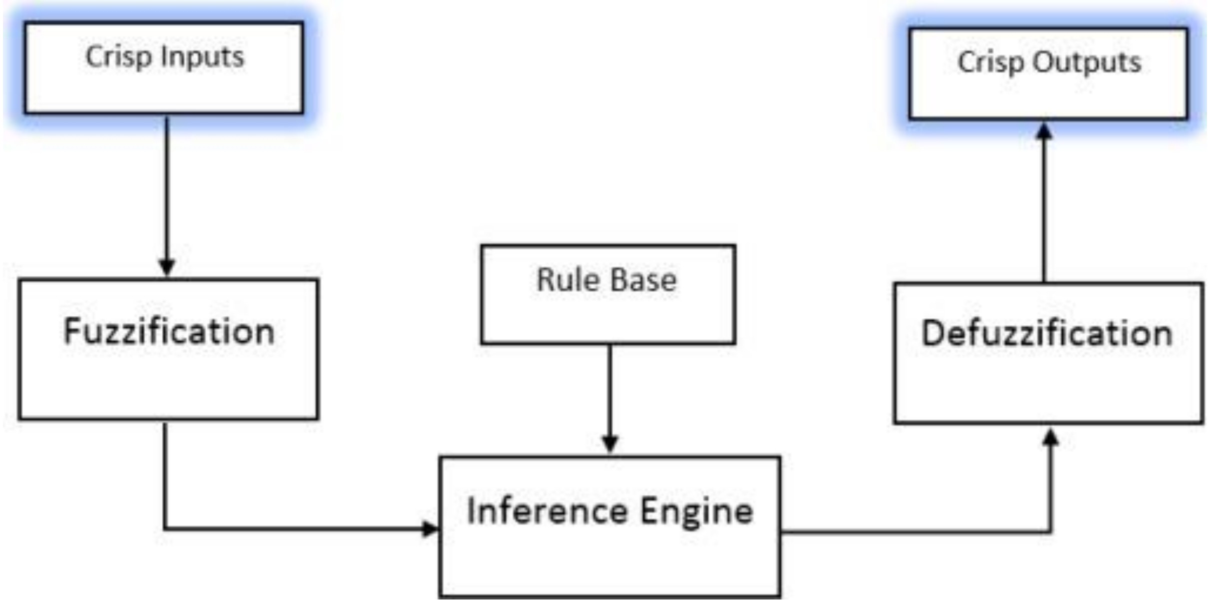
Membership Functions with respect to the input time series.

Fig. 6. A17U.SI Time Series



Fuzzy Time Series Algorithm

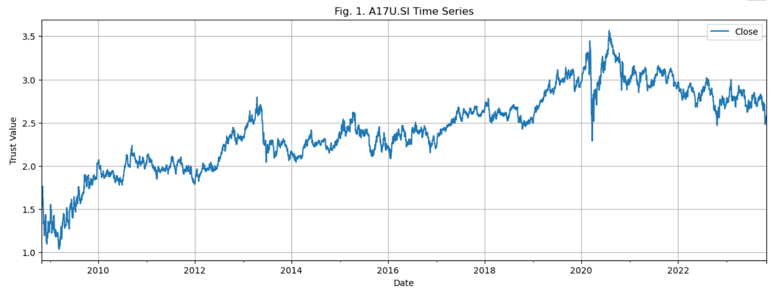
A crisp input goes through a fuzzification process that converts it into its corresponding fuzzified variable, which then goes to a fuzzy inference engine. Based on a set of fuzzy rules, the inference engine outputs a new fuzzified variable that is then converted back to its crisp value via a defuzzification process.

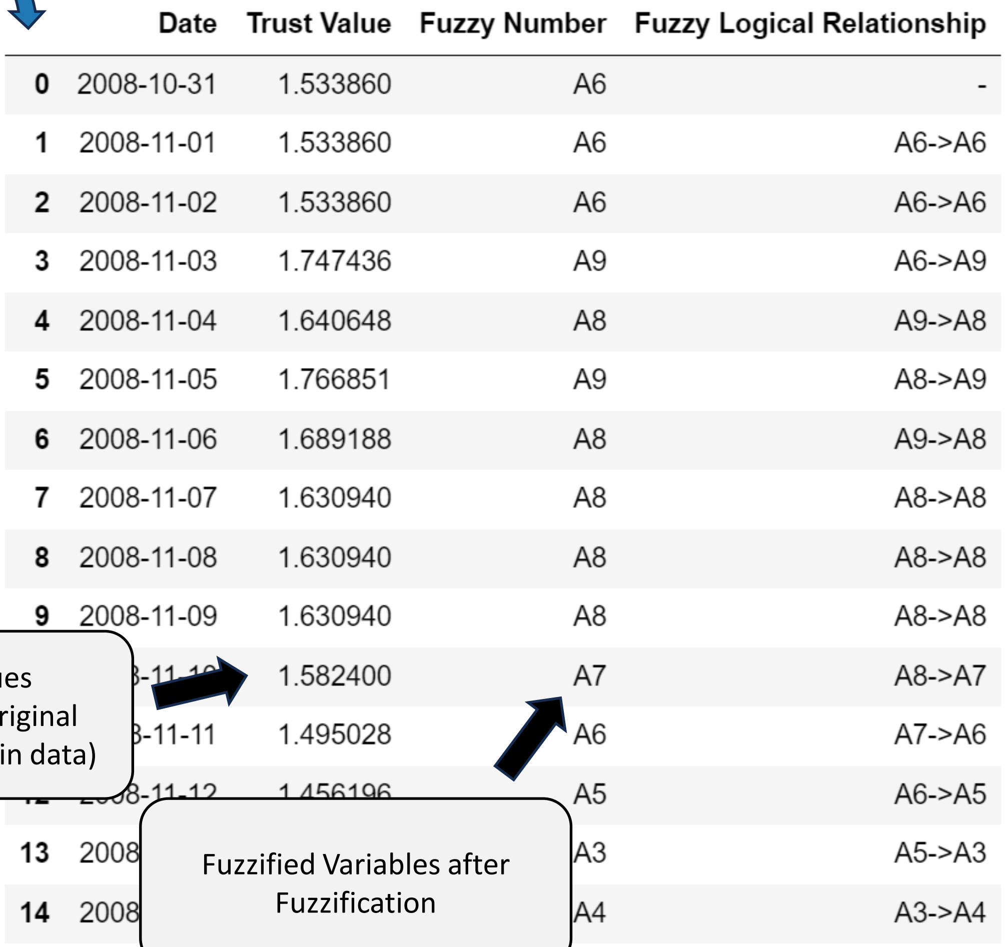


Basic Components of a Fuzzy Inference System.

FTS: Fuzzification

Converting crisp input to fuzzified variables.





	Date	Trust Value	Fuzzy Number	Fuzzy Logical Relationship
0	2008-10-31	1.533860	A6	-
1	2008-11-01	1.533860	A6	A6->A6
2	2008-11-02	1.533860	A6	A6->A6
3	2008-11-03	1.747436	A9	A6->A9
4	2008-11-04	1.640648	A8	A9->A8
5	2008-11-05	1.766851	A9	A8->A9
6	2008-11-06	1.689188	A8	A9->A8
7	2008-11-07	1.630940	A8	A8->A8
8	2008-11-08	1.630940	A8	A8->A8
9	2008-11-09	1.630940	A8	A8->A8
[illegible]	[illegible]	1.582400	A7	A8->A7
[illegible]	3-11-11	1.495028	A6	A7->A6
[illegible]	2008-11-12	1.456196	A5	A6->A5
13	2008	[illegible]	A3	A5->A3
14	2008	[illegible]	A4	A3->A4

Crisp Input values representing our original time series data (train data)

Fuzzified Variables after Fuzzification

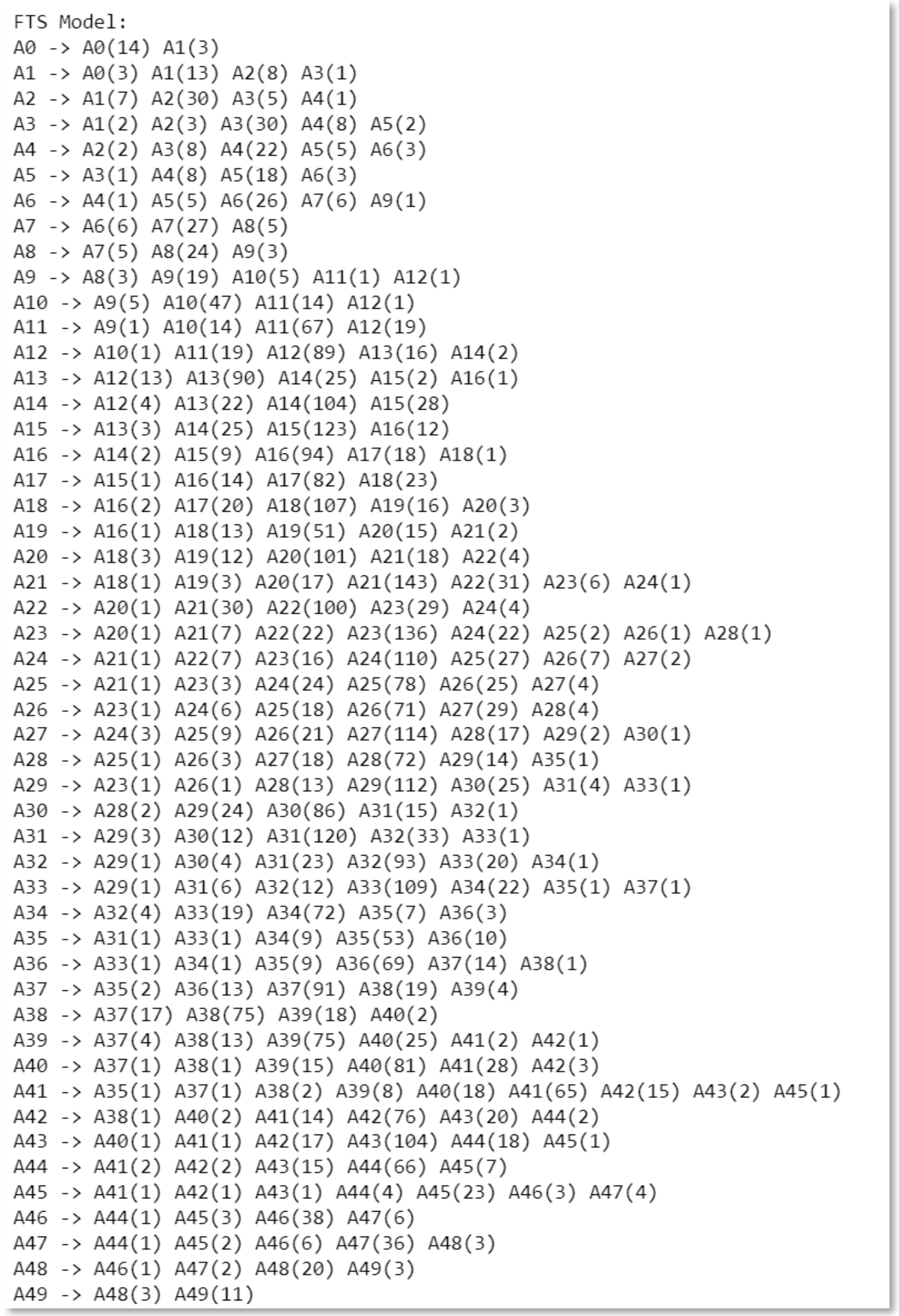
Fuzzy Logical Relationships (FLRs) representing how each fuzzy variable 'flow' from one state to another

TABLE I. Output of the Fuzzification Process

FTS: Inference Engine

The fuzzy rules are derived by grouping the fuzzy logical relationships into groups, which is called the **Fuzzy Logical Relationship Groups** (FLRG), alongside the frequency associated with each fuzzy logical relationship

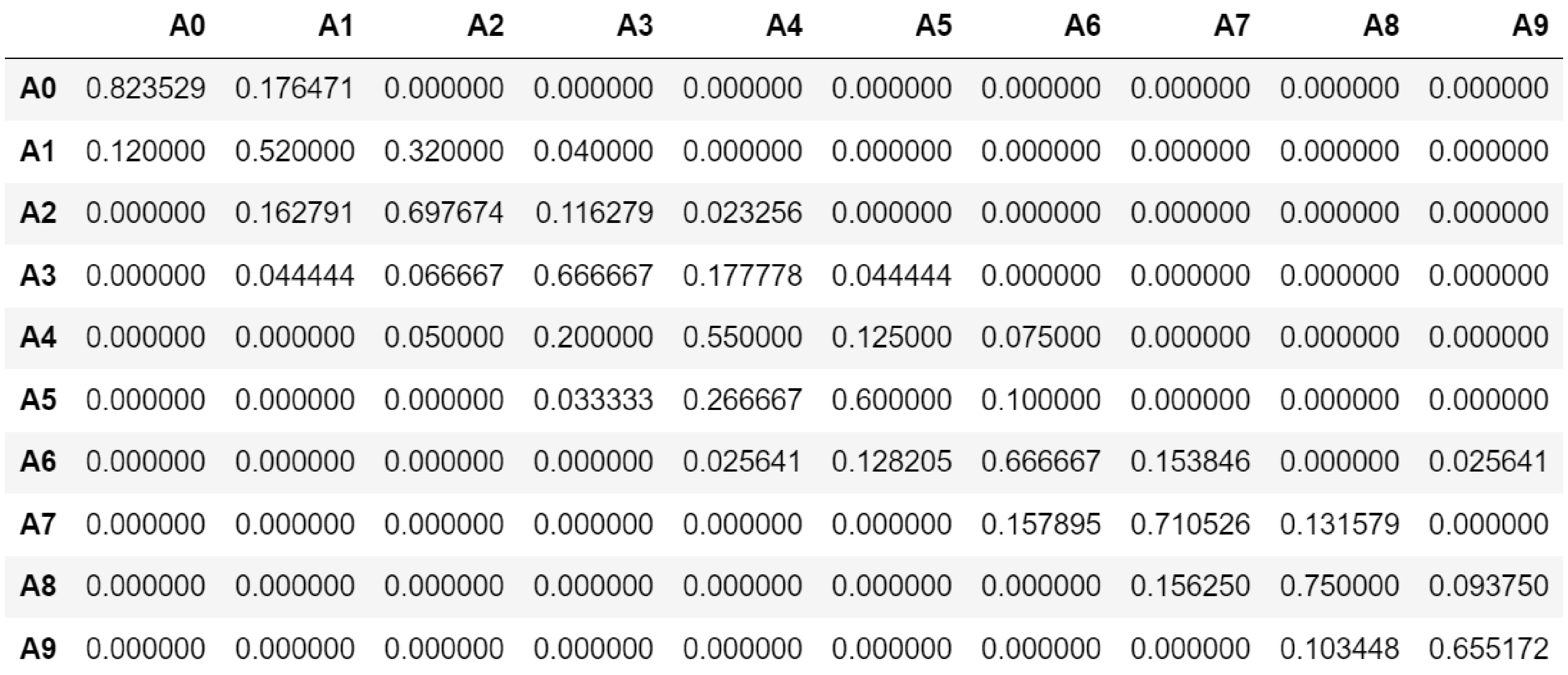
FLRGs tell us all the possible next *states* given the current *state* (input fuzzy variable)



```
FTS Model:
A0 -> A0(14) A1(3)
A1 -> A0(3) A1(13) A2(8) A3(1)
A2 -> A1(7) A2(30) A3(5) A4(1)
A3 -> A1(2) A2(3) A3(30) A4(8) A5(2)
A4 -> A2(2) A3(8) A4(22) A5(5) A6(3)
A5 -> A3(1) A4(8) A5(18) A6(3)
A6 -> A4(1) A5(5) A6(26) A7(6) A9(1)
A7 -> A6(6) A7(27) A8(5)
A8 -> A7(5) A8(24) A9(3)
A9 -> A8(3) A9(19) A10(5) A11(1) A12(1)
A10 -> A9(5) A10(47) A11(14) A12(1)
A11 -> A9(1) A10(14) A11(67) A12(19)
A12 -> A10(1) A11(19) A12(89) A13(16) A14(2)
A13 -> A12(13) A13(90) A14(25) A15(2) A16(1)
A14 -> A12(4) A13(22) A14(104) A15(28)
A15 -> A13(3) A14(25) A15(123) A16(12)
A16 -> A14(2) A15(9) A16(94) A17(18) A18(1)
A17 -> A15(1) A16(14) A17(82) A18(23)
A18 -> A16(2) A17(20) A18(107) A19(16) A20(3)
A19 -> A16(1) A18(13) A19(51) A20(15) A21(2)
A20 -> A18(3) A19(12) A20(101) A21(18) A22(4)
A21 -> A18(1) A19(3) A20(17) A21(143) A22(31) A23(6) A24(1)
A22 -> A20(1) A21(30) A22(100) A23(29) A24(4)
A23 -> A20(1) A21(7) A22(22) A23(136) A24(22) A25(2) A26(1) A28(1)
A24 -> A21(1) A22(7) A23(16) A24(110) A25(27) A26(7) A27(2)
A25 -> A21(1) A23(3) A24(24) A25(78) A26(25) A27(4)
A26 -> A23(1) A24(6) A25(18) A26(71) A27(29) A28(4)
A27 -> A24(3) A25(9) A26(21) A27(114) A28(17) A29(2) A30(1)
A28 -> A25(1) A26(3) A27(18) A28(72) A29(14) A35(1)
A29 -> A23(1) A26(1) A28(13) A29(112) A30(25) A31(4) A33(1)
A30 -> A28(2) A29(24) A30(86) A31(15) A32(1)
A31 -> A29(3) A30(12) A31(120) A32(33) A33(1)
A32 -> A29(1) A30(4) A31(23) A32(93) A33(20) A34(1)
A33 -> A29(1) A31(6) A32(12) A33(109) A34(22) A35(1) A37(1)
A34 -> A32(4) A33(19) A34(72) A35(7) A36(3)
A35 -> A31(1) A33(1) A34(9) A35(53) A36(10)
A36 -> A33(1) A34(1) A35(9) A36(69) A37(14) A38(1)
A37 -> A35(2) A36(13) A37(91) A38(19) A39(4)
A38 -> A37(17) A38(75) A39(18) A40(2)
A39 -> A37(4) A38(13) A39(75) A40(25) A41(2) A42(1)
A40 -> A37(1) A38(1) A39(15) A40(81) A41(28) A42(3)
A41 -> A35(1) A37(1) A38(2) A39(8) A40(18) A41(65) A42(15) A43(2) A45(1)
A42 -> A38(1) A40(2) A41(14) A42(76) A43(20) A44(2)
A43 -> A40(1) A41(1) A42(17) A43(104) A44(18) A45(1)
A44 -> A41(2) A42(2) A43(15) A44(66) A45(7)
A45 -> A41(1) A42(1) A43(1) A44(4) A45(23) A46(3) A47(4)
A46 -> A44(1) A45(3) A46(38) A47(6)
A47 -> A44(1) A45(2) A46(6) A47(36) A48(3)
A48 -> A46(1) A47(2) A48(20) A49(3)
A49 -> A48(3) A49(11)
```

FTS: Inference Engine

We can better represent the FLRGs through a Markov Transition Probability Matrix, showing the probability that the next fuzzy variable is A_b given that the current fuzzy variable is A_a.



	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
A0	0.823529	0.176471	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
A1	0.120000	0.520000	0.320000	0.040000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
A2	0.000000	0.162791	0.697674	0.116279	0.023256	0.000000	0.000000	0.000000	0.000000	0.000000
A3	0.000000	0.044444	0.066667	0.666667	0.177778	0.044444	0.000000	0.000000	0.000000	0.000000
A4	0.000000	0.000000	0.050000	0.200000	0.550000	0.125000	0.075000	0.000000	0.000000	0.000000
A5	0.000000	0.000000	0.000000	0.033333	0.266667	0.600000	0.100000	0.000000	0.000000	0.000000
A6	0.000000	0.000000	0.000000	0.000000	0.025641	0.128205	0.666667	0.153846	0.000000	0.025641
A7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.157895	0.710526	0.131579	0.000000
A8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.156250	0.750000	0.093750
A9	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.103448	0.655172

TABLE II. Markov Transition Probability Matrix (First 10 Fuzzy States)

FTS: Defuzzification

Defuzzification, i.e. to converting the generated fuzzy variables generated by our fuzzy inference engine back to its corresponding crisp value (a.k.a the point forecast for time step t+1), can be performed using the following equation:

Alyousifi's Original Implementation [2]:

$$F(t+1) = \hat{\mathbf{c}} \cdot \mathbf{p_t} + D(F(t))$$

My Implementation:

$$F(t+1) = \mathbf{c} \cdot \mathbf{p_t} + \gamma D(F(t))$$

where

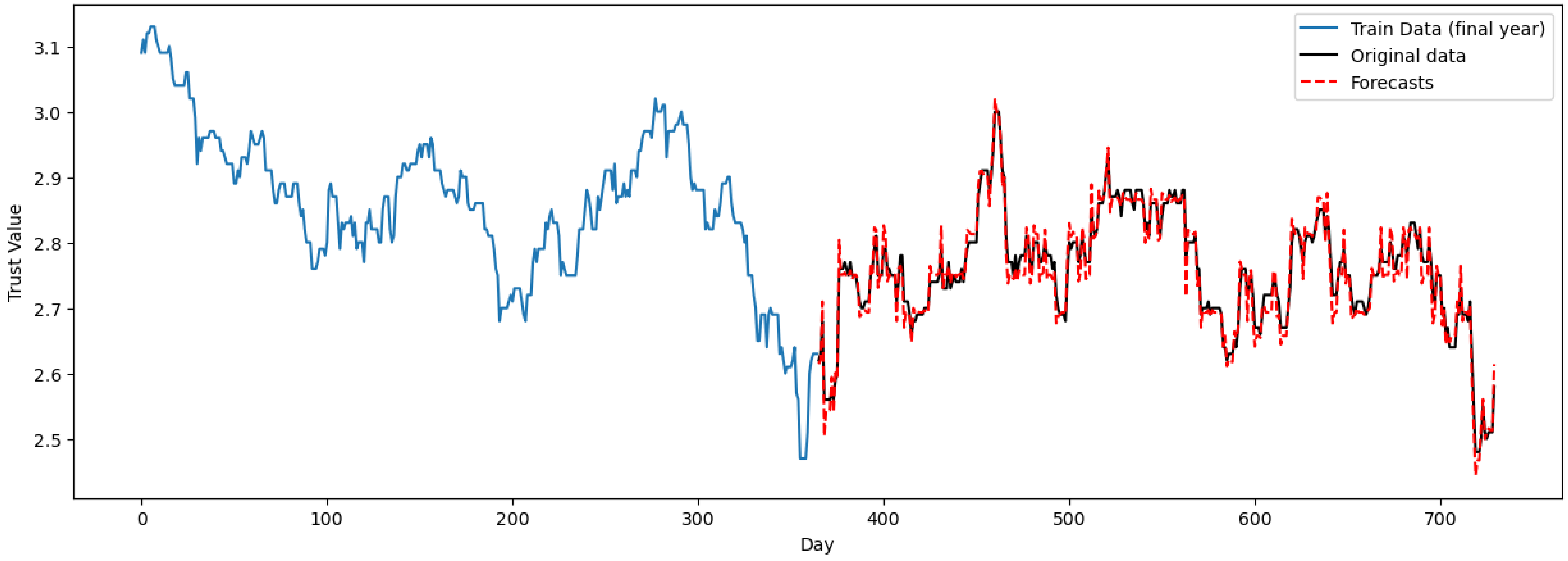
- $\mathbf{c}$ is the vector of centroid values generated from the K-Means clustering process
- $\hat{\mathbf{c}}$ is $\mathbf{c}$ but the centroid value at position *k* corresponding to the fuzzy variable *Ak* is replaced with $F(t)$ or the original crisp value for time step t.
- $\mathbf{p_t}$ is the probability vector corresponding to a row in the markov transition probability matrix P, given the fuzzy variable *Ak* for the current time step
- $D(F(t))$ is the first order differencing of the actual values at time step *t*
- γ is the discount factor that controls the degree on which the differencing influences the final forecast.

The logic behind my modification is to simplify the defuzzification process, as well as remove potential data leakage through the vector c-hat, hopefully making the algorithm much more robust.

FTS: Defuzzification

From the output of the defuzzification process, we can generate rolling forecasts across the entire testing range.

Fig. 8. Forecast Values generated by the KM-FTS model



Benchmarks

To validate the performance of our model, the generated hybrid KM-FTS model is benchmarked across 4 evaluation metrics (Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Thiels' U-statistics, and R-squared) against several models, namely:

BASE-RW: Base random walk model (today's price is tomorrow's price)

GRID-FTS25, **GRID-FTS50**, **GRID-FTS100**: Standard FTS model with 25, 50, and 100 partitions respectively. No K-Means clustering method was used to find the best positions of membership function centroids. Instead, the partitioning is set to be evenly distributed across the universe of discourse *U*.

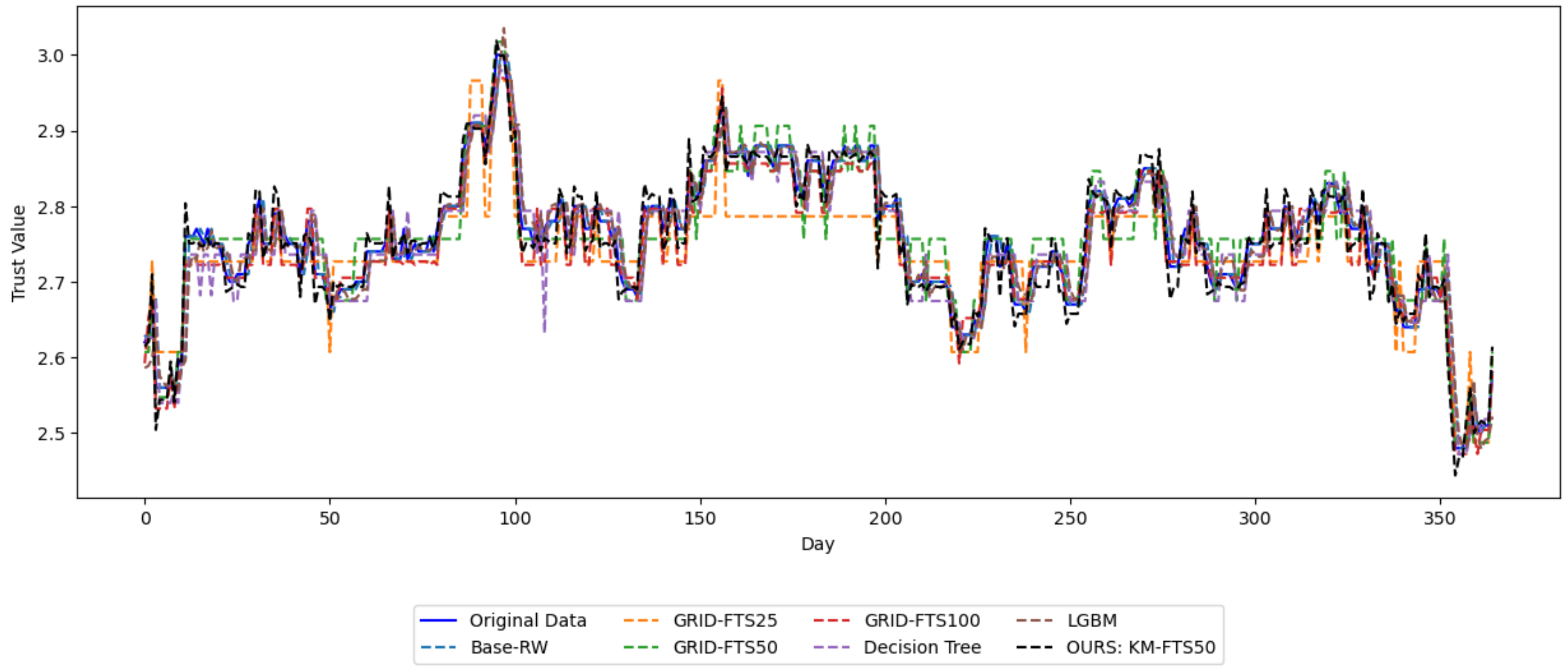
Decision Tree: Decision Tree regressor with 4-fold cross validation.

LGBM: Light Gradient Boosting Model with 4-fold cross validation.

Analysis of Results

The **hybrid KM-FTS model yields the best forecasting performance**, generally surpassing the other models, according to the selected statistical benchmarks. Our KM-FTS model produced the lowest Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), and the highest R-squared value across all other models; only being out-performed by the GRID-FTS100 model in the Theil's U-statistic by a close margin.

Fig. 9. Benchmark Results



	RMSE	MAPE	TU	R2
Base-RW	0.027213	0.621069	1.364670	0.903733
GRID-FTS25	0.042730	1.278573	1.214747	0.762648
GRID-FTS50	0.028577	0.855652	1.044901	0.893838
GRID-FTS100	0.021924	0.633121	0.788783	0.937519
Decision Tree	0.031943	0.871295	1.526671	0.867358
LGBM	0.029023	0.755742	1.333893	0.890498
OURS: KM-FTS50	0.018254	0.535755	0.810689	0.956683

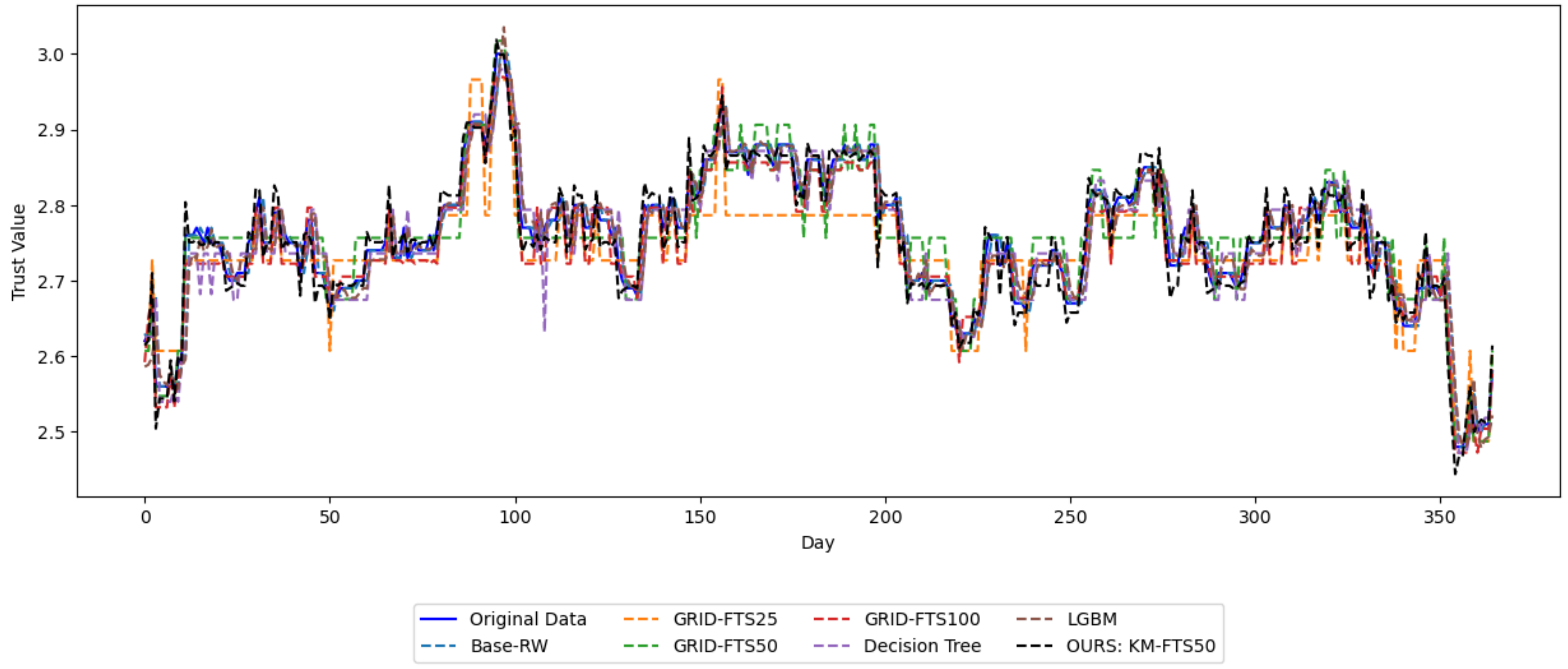
TABLE III. Benchmark Results

Analysis of Results

Key Takeaway KM-FTS outperforms GRID-FTS (implemented using the Chen Model in PyFTS), proving that the addition of K-Means algorithm to select centroid positions drastically improves model performance, as opposed to the uniform partitioning strategy in standard GRID-FTS.

This is because through the K-Means clustering method, we are able to identify areas of high and low concentration of values and are thus able to allocate more membership functions to more finely disaggregate the trend into a series of fuzzy logical relationships. In contrast, a uniform partitioning method implemented in standard FTS models will evenly allocate partitions, even in areas within our universe of discourse where data points are sparse, hence, producing more weakly associated fuzzy logical relationships.

Fig. 9. Benchmark Results



	RMSE	MAPE	TU	R2
Base-RW	0.027213	0.621069	1.364670	0.903733
GRID-FTS25	0.042730	1.278573	1.214747	0.762648
GRID-FTS50	0.028577	0.855652	1.044901	0.893838
GRID-FTS100	0.021924	0.633121	0.788783	0.937519
Decision Tree	0.031943	0.871295	1.526671	0.867358
LGBM	0.029023	0.755742	1.333893	0.890498
OURS: KM-FTS50	0.018254	0.535755	0.810689	0.956683

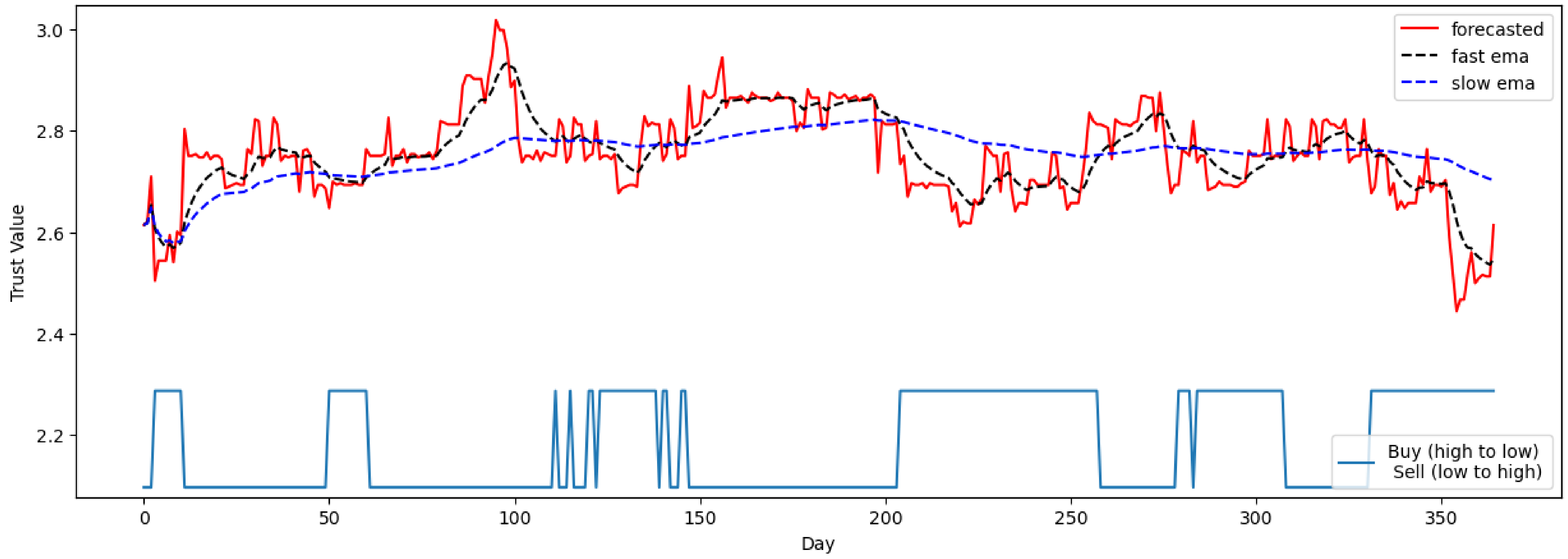
TABLE III. Benchmark Results

Making Money $$$

Generating Buy and Sell signals from the forecast plot requires the use of two moving averages (in this project, I chose exponential moving average or EMA): slow EMA and fast EMA. The difference is the number of periods to which the averaging is performed.

In this project, I found that the model performs best using a **12-day EMA** (fast) and **132-day EMA** (slow). We can define a buy signal whenever the slow EMA crosses below the fast ema, and a sell signal whenever the slow EMA crosses above the fast EMA

Fig. 10. Calculating Buy and Sell Signals



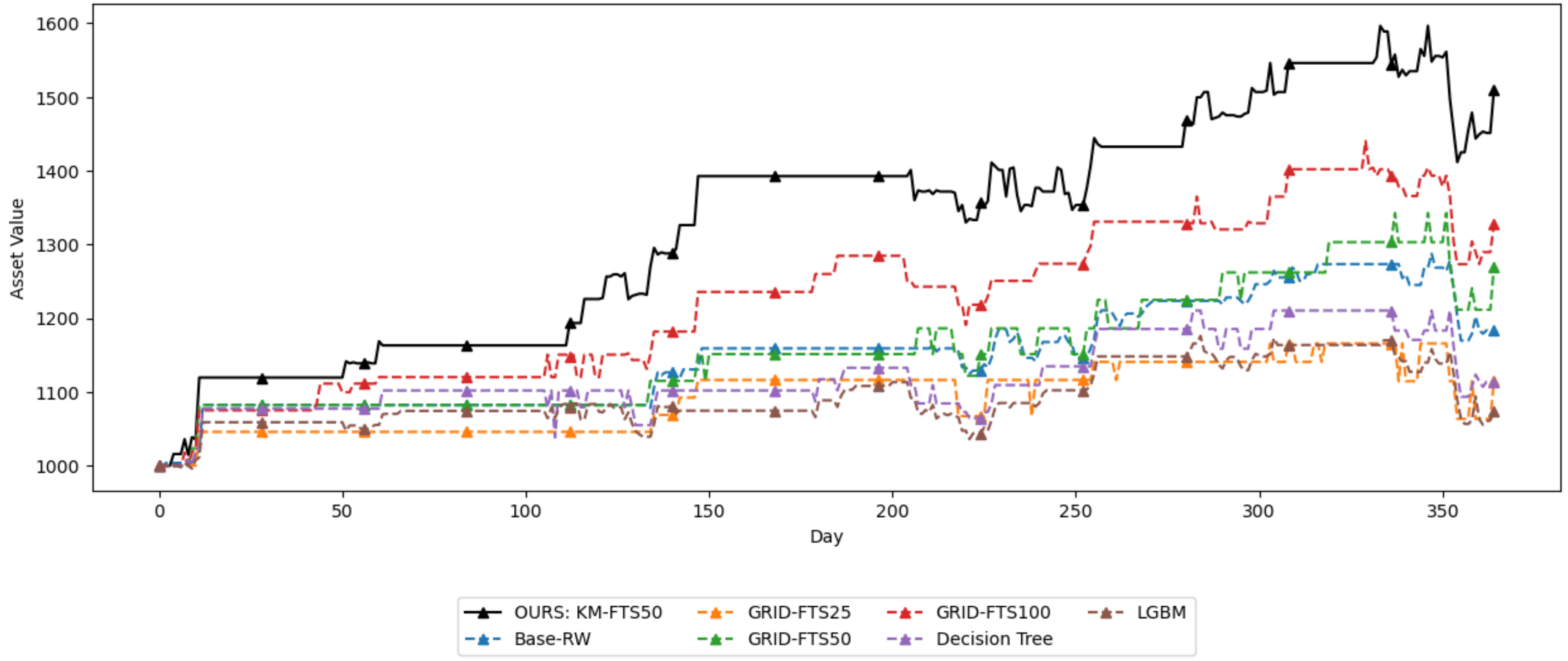
Making Money $$$: Benchmarks

To calculate profit and compare across the benchmark models, I created simple functions that:

1. Finds the best EMA parameters using grid search
2. Calculate trade positions (buy and sell signals) on the model's forecast values using the best EMA parameters
3. Use a greedy trading strategy according to calculated signals (buy all / sell all)

Implementing steps 1-3 to the forecasted values of all the benchmark models, we get the following results:

Fig. 11. Asset Value Curve



	Net Profit	P/L %
OURS: KM-FTS50	509.809640	50.98
Base-RW	183.722203	18.37
GRID-FTS25	114.591405	11.46
GRID-FTS50	269.774842	26.98
GRID-FTS100	327.190604	32.72
Decision Tree	114.148991	11.41
LGBM	73.695670	7.37

TABLE IV. Profit Benchmark

Our portfolio grew by 50.98% over just 1 year!

Of course, it is not all sunshine and rainbows. While our model did net us a profit of $500+ from our initial $1000 investment in the simulation, in actual practice, forecasting models are not expected to perform as well. In fact, machine learning models rarely outperform stock indices over long periods of time. Most likely, the KM-FTS forecasting model we developed is tuned to only perform well on the specific dataset (A17U.SI) and on the specific data range the model was trained on.

Still, this project is a good illustration of how we can leverage machine learning and AI algorithms to analyze data in more ways than one. In the end, it is the trader's responsibility to assess whether or not an opportunity to buy or to sell arises.

Trade responsibly.

Reinelle Jan C. Bugnot
G2304329L
bugn0001@e.ntu.edu.sg
MS AI – Nanyang Technological University