Smart Contract Development on Blockchain using Time Series Forecasting

Name : Rajarshi Saha **Roll No. : B2330049**

Supervisor : Champak Dutta

Agenda

Introduction

Scope of the work

Literature Survey

Data Collection

Models used

Experimental Result

Smart Contract

Scope of the work

Elements of the Project

Smart Contract

A smart contract is like a computer program that works like a digital deal. It runs on a blockchain, which is a special, secure system. Once the smart contract is made, it automatically does what it was created to do, and no one can change it or stop it.

Time Series Forecasting

Time series forecasting helps us predict future events by analyzing past data. For example, in the stock market, we look at a stock's prices over the last few months to find patterns. By studying these patterns, we can make educated guesses about what the stock price might be next week or next month.

Activity Done

Experimental analysis on different model behaviours :

1: Transformermodel 2. Informer 3. itransformer

Activity remaining

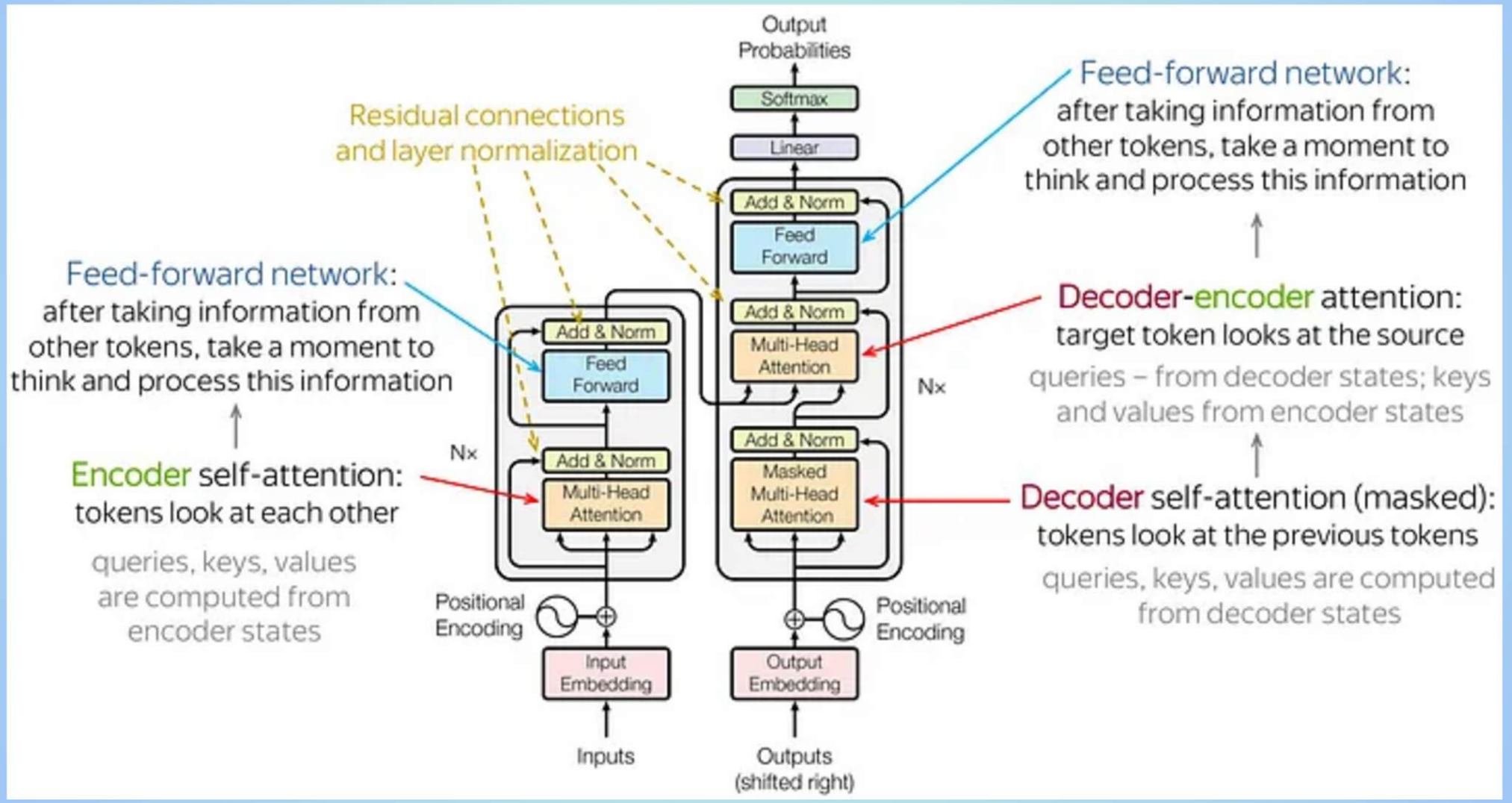
Analysis on the models : 1. Autoformer , 2. TSMixer , 3. Time-LLM

Smart Contract Development

Literature Survey

Paper	dataset	link
Transformers in Time Series: A Survey	dataset 1	https://www.ijcai.org/proceedings/2023/0759.pdf
Informer:Beyond Efficient Transformer for Long Sequence Time-Series Forecasting	dataset 1, dataset 2, dataset 3	https://arxiv.org/pdf/2012.07436
ITRANSFORMER: INVERTED TRANSFORMERS ARE EFFECTIVE FOR TIME SERIES FORECASTING	dataset 1 , dataset 2, dataset 3, dataset 4	https://arxiv.org/pdf/2310.06625
Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting	dataset 1	https://arxiv.org/pdf/2106.13008
TSMixer: An All-MLP Architecture for Time Series Fore□casting	dataset 1 , dataset 2,dataset 3	https://arxiv.org/pdf/2303.06053

Transformer Architechture



source : https://sanchman21.medium.com/evolution-of-transformers-part-1-faac3f19d780

Complexity Comparisons

Table 1: Complexity comparisons of popular time series Transformers with different attention modules.

Methods	Training		Testing
	Time	Memory	Steps
Transformer [Vaswani *et al.*, 2017]	$\mathcal{O}\left(N^2\right)$	$\mathcal{O}\left(N^2\right)$	N
LogTrans [Li *et al.*, 2019]	$\mathcal{O}(N \log N)$	$\mathcal{O}\left(N \log N\right)$	1
Informer [Zhou *et al.*, 2021]	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Autoformer [Wu *et al.*, 2021]	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Pyraformer [Liu *et al.*, 2022a]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	1
Quatformer [Chen *et al.*, 2022]	$\mathcal{O}(2cN)$	$\mathcal{O}(2cN)$	1
FEDformer [Zhou *et al.*, 2022]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	1
Crossformer [Zhang and Yan, 2023]	$\mathcal{O}(\frac{D}{L_{seg}^2} N^2)$	$\mathcal{O}(N)$	1

source : https://www.ijcai.org/proceedings/2023/0759.pdf

Informer Architecture

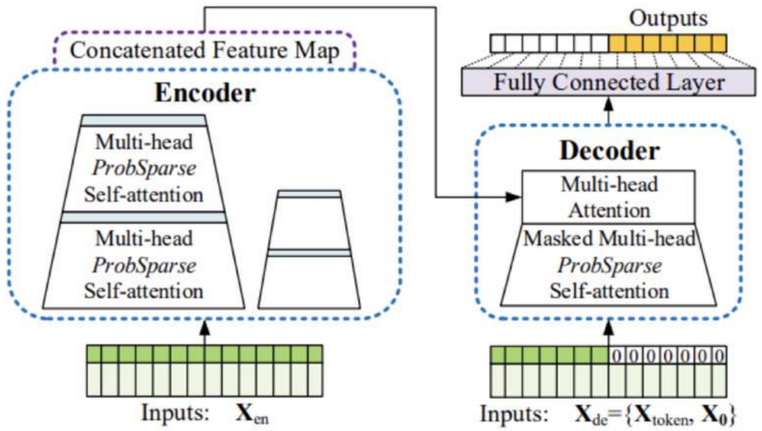


Figure 2: Informer model overview. Left: The encoder receives massive long sequence inputs (green series). We replace canonical self-attention with the proposed *ProbSparse* self-attention. The blue trapezoid is the self-attention distilling operation to extract dominating attention, reducing the network size sharply. The layer stacking replicas increase robustness. Right: The decoder receives long sequence inputs, pads the target elements into zero, measures the weighted attention composition of the feature map, and instantly predicts output elements (orange series) in a generative style.

source : https://arxiv.org/pdf/2012.07436

Self-Attention Distillation in Informer

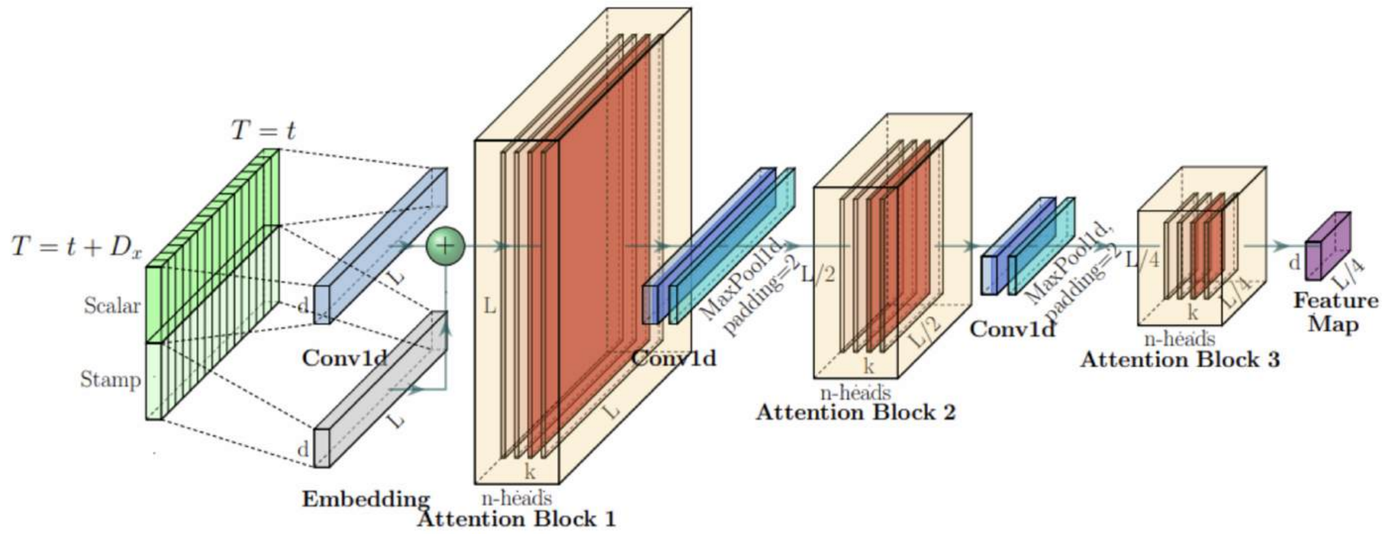


Figure 3: The single stack in Informer's encoder. (1) The horizontal stack stands for an individual one of the encoder replicas in Fig.(2). (2) The presented one is the main stack receiving the whole input sequence. Then the second stack takes half slices of the input, and the subsequent stacks repeat. (3) The red layers are dot-product matrixes, and they get cascade decrease by applying self-attention distilling on each layer. (4) Concatenate all stacks' feature maps as the encoder's output.

source : https://arxiv.org/pdf/2012.07436

itransformer Architecture

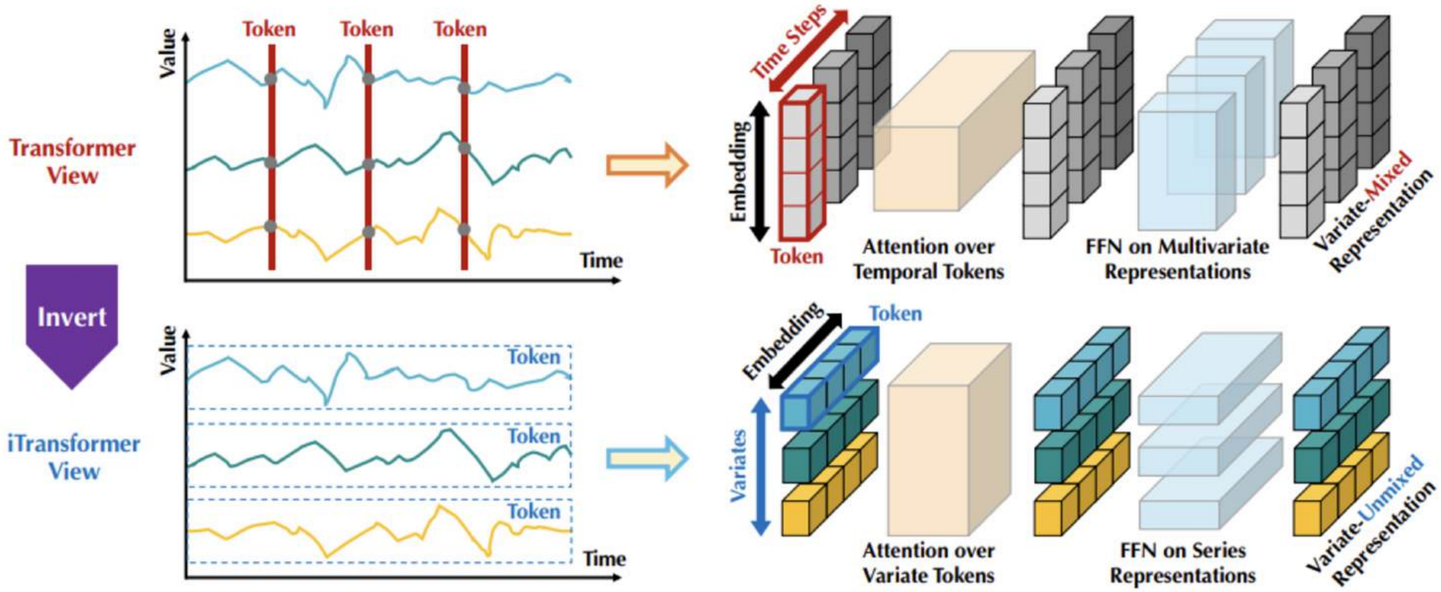


Figure 2: Comparison between the vanilla Transformer (top) and the proposed iTransformer (bottom). Transformer embeds the temporal token, which contains the multivariate representation of each time step. iTransformer embeds each series independently to the variate token, such that the attention module depicts the multivariate correlations and the feed-forward network encodes series representations.

source : https://arxiv.org/pdf/2310.06625

itransformer Architecture

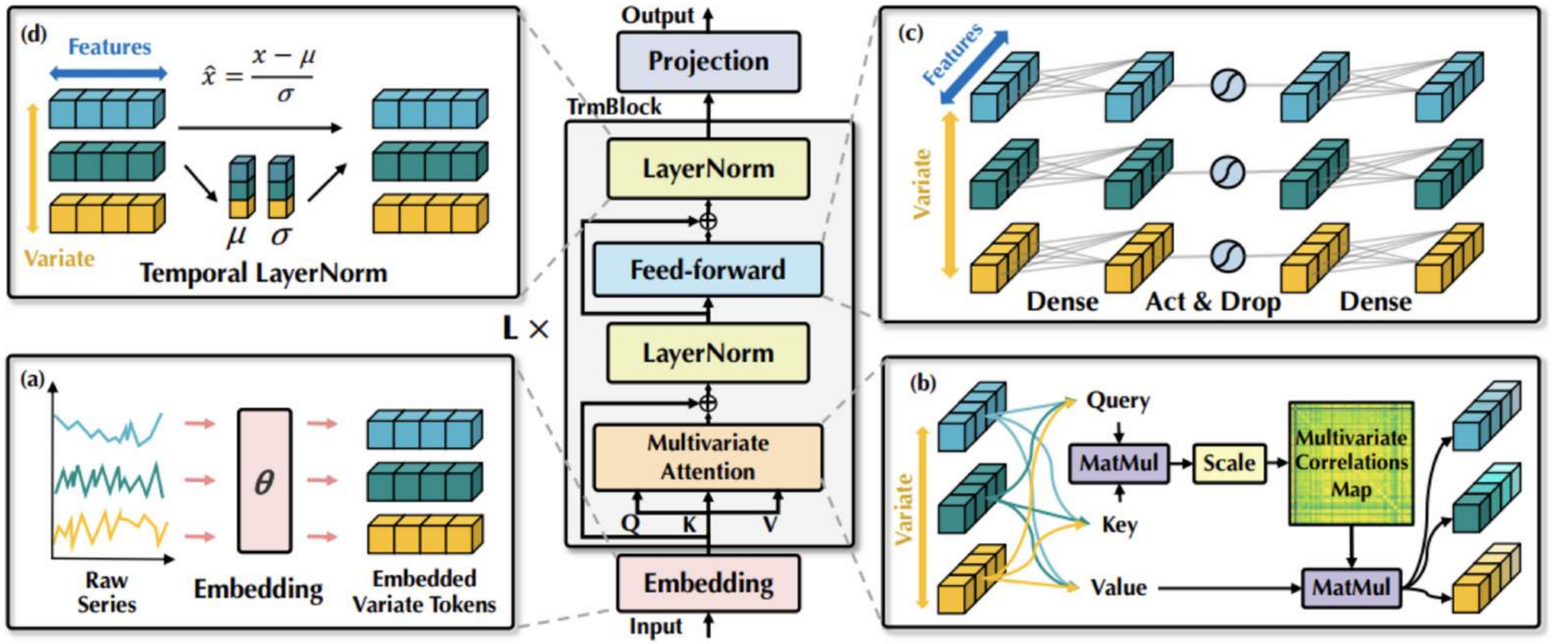


Figure 4: Overall structure of iTransformer, which shares the same modular arrangement with the encoder of Transformer. (a) Raw series of different variates are independently embedded as tokens. (b) Self-attention is applied to embedded variate tokens with enhanced interpretability revealing multivariate correlations. (c) Series representations of each token are extracted by the shared feed-forward network. (d) Layer normalization is adopted to reduce the discrepancies among variates.

source : https://arxiv.org/pdf/2310.06625

itransformer Performance

Table 7: Full performance comparison between the vanilla Transformer and the proposed iTransformer. The results are averaged from all four prediction lengths.

Datasets	ETT		ECL		PEMS		Solar-Energy		Traffic		Weather	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Transformer	2.750	1.375	0.277	0.372	0.157	0.263	0.256	0.276	0.665	0.363	0.657	0.572
iTransformer	**0.383**	**0.407**	**0.178**	**0.270**	**0.113**	**0.221**	**0.233**	**0.262**	**0.428**	**0.282**	**0.258**	**0.279**
Promotion	86.1%	70.4%	35.6%	27.4%	28.0%	16.0%	9.0%	5.1%	35.6%	22.3%	60.2%	50.8%

source : https://arxiv.org/pdf/2310.06625

Autoformer Architecture

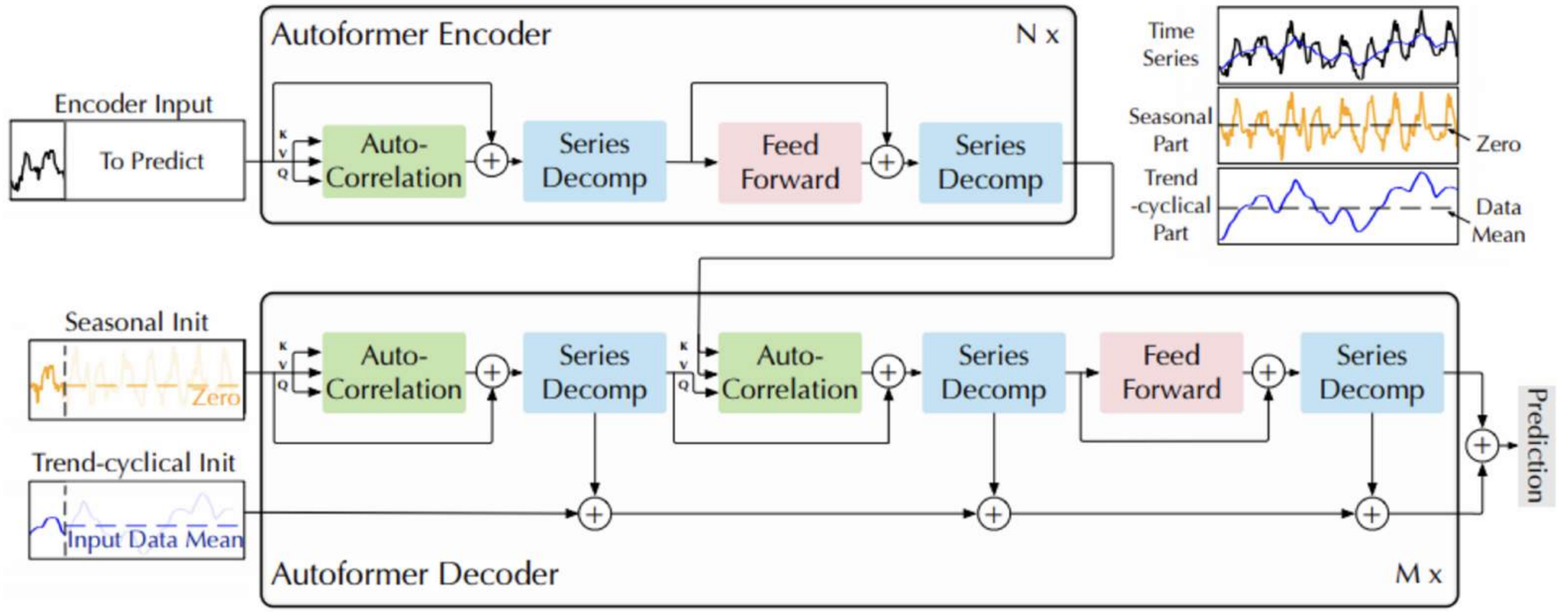


Figure 1: Autoformer architecture. The encoder eliminates the long-term trend-cyclical part by series decomposition blocks (blue blocks) and focuses on seasonal patterns modeling. The decoder accumulates the trend part extracted from hidden variables progressively. The past seasonal information from encoder is utilized by the encoder-decoder Auto-Correlation (center green block in decoder).

source : https://arxiv.org/pdf/2106.13008

Autoformer Architecture

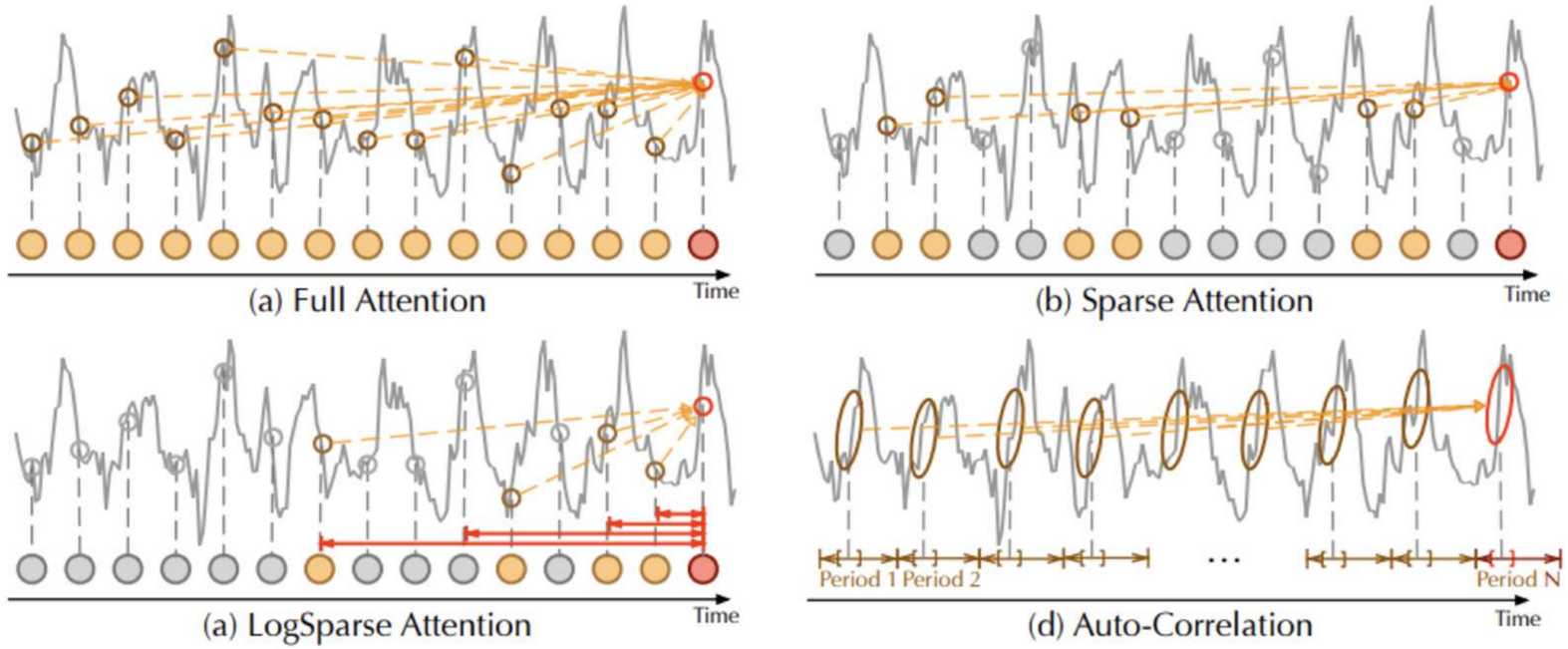


Figure 3: Auto-Correlation vs. self-attention family. Full Attention [41] (a) adapts the fully connection among all time points. Sparse Attention [23, 48] (b) selects points based on the proposed similarity metrics. LogSparse Attention [26] (c) chooses points following the exponentially increasing intervals. Auto-Correlation (d) focuses on the connections of sub-series among underlying periods.

source : https://arxiv.org/pdf/2106.13008

Autoformer Architecture

Table 4: Comparison of Auto-Correlation and self-attention in the multivariate ETT. We **replace** the Auto-Correlation in Autoformer with different self-attentions. The "-" indicates the out-of-memory.

Input Length I		96			192			336		
Prediction Length O		336	720	1440	336	720	1440	336	720	1440
Auto-Correlation	MSE	**0.339**	**0.422**	**0.555**	**0.355**	**0.429**	**0.503**	**0.361**	**0.425**	**0.574**
	MAE	**0.372**	**0.419**	**0.496**	**0.392**	**0.430**	**0.484**	**0.406**	**0.440**	**0.534**
Full Attention[41]	MSE	0.375	0.537	0.667	0.450	0.554	-	0.501	0.647	-
	MAE	0.425	0.502	0.589	0.470	0.533	-	0.485	0.491	-
LogSparse Attention[26]	MSE	0.362	0.539	0.582	0.420	0.552	0.958	0.474	0.601	-
	MAE	0.413	0.522	0.529	0.450	0.513	0.736	0.474	0.524	-
LSH Attention[23]	MSE	0.366	0.502	0.663	0.407	0.636	1.069	0.442	0.615	-
	MAE	0.404	0.475	0.567	0.421	0.571	0.756	0.476	0.532	-
ProbSparse Attention[48]	MSE	0.481	0.822	0.715	0.404	1.148	0.732	0.417	0.631	1.133
	MAE	0.472	0.559	0.586	0.425	0.654	0.602	0.434	0.528	0.691

source : https://arxiv.org/pdf/2106.13008

TSMixer Architecture

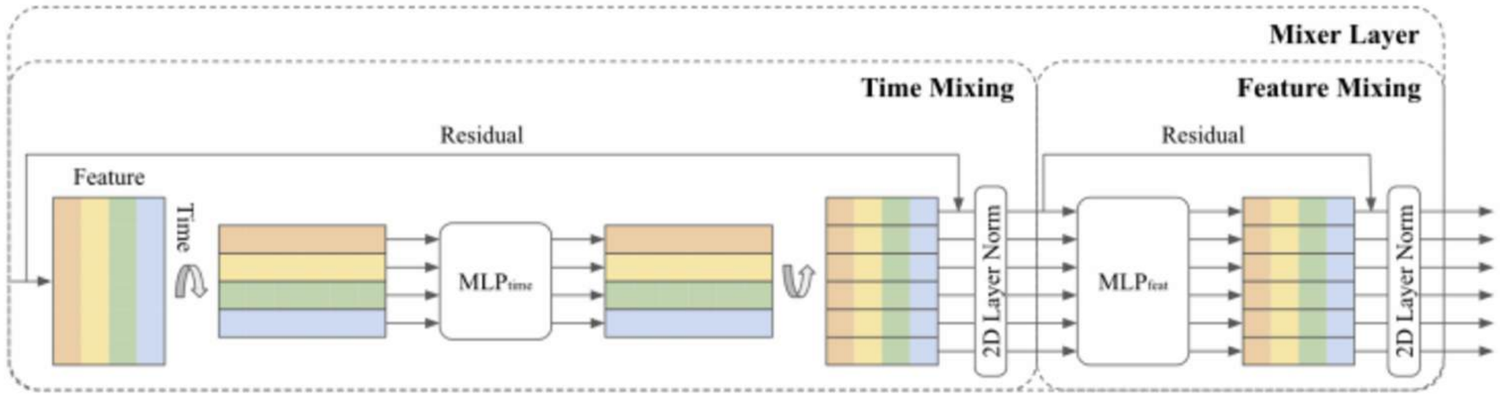


Figure 6: Mixing layers in TSMixer-Ext.

source : https://arxiv.org/pdf/2303.06053

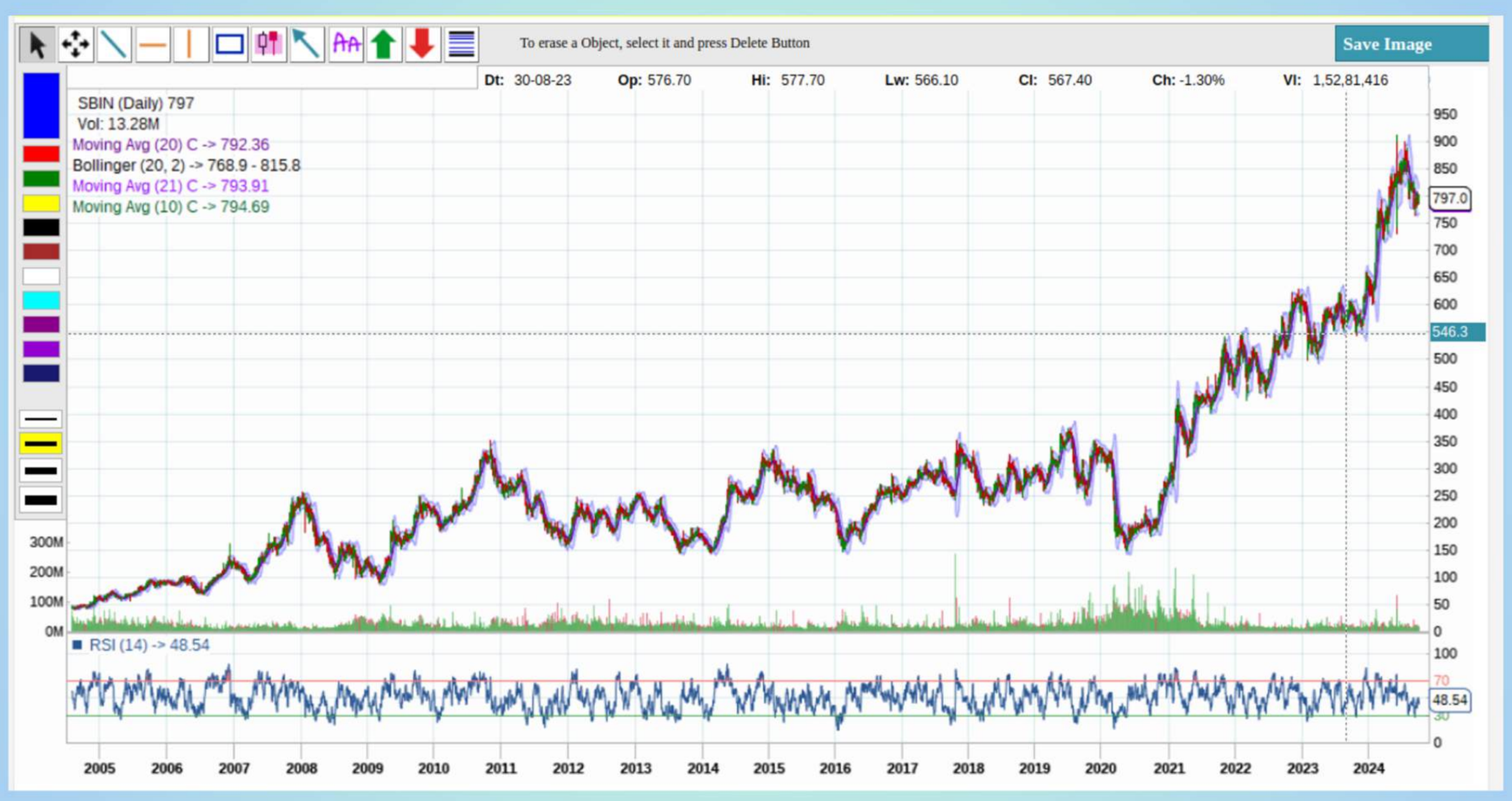
Data Collection

Data Source

Stock market data is used here because it is **freely accessible** and easily obtained using tools like the **yfinance library**. This allows for seamless retrieval of historical stock prices, essential for time series forecasting models. The availability of open financial data makes it **convenient for experimentation** and model training without the need for expensive or proprietary datasets.

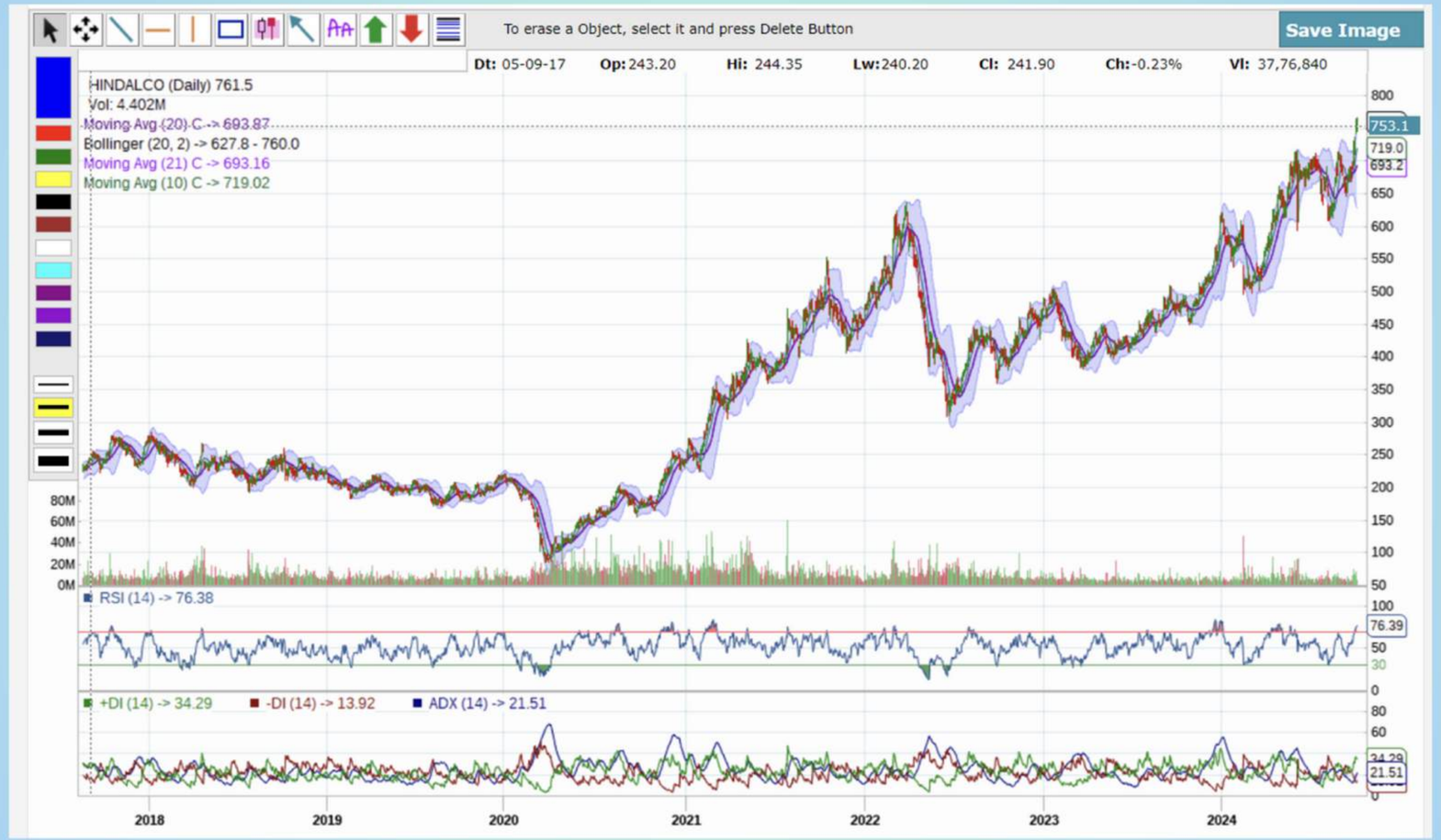
Approx 28 years of data from 01-01-1996 to till date is available here.

State Bank Of India (8 years Data)



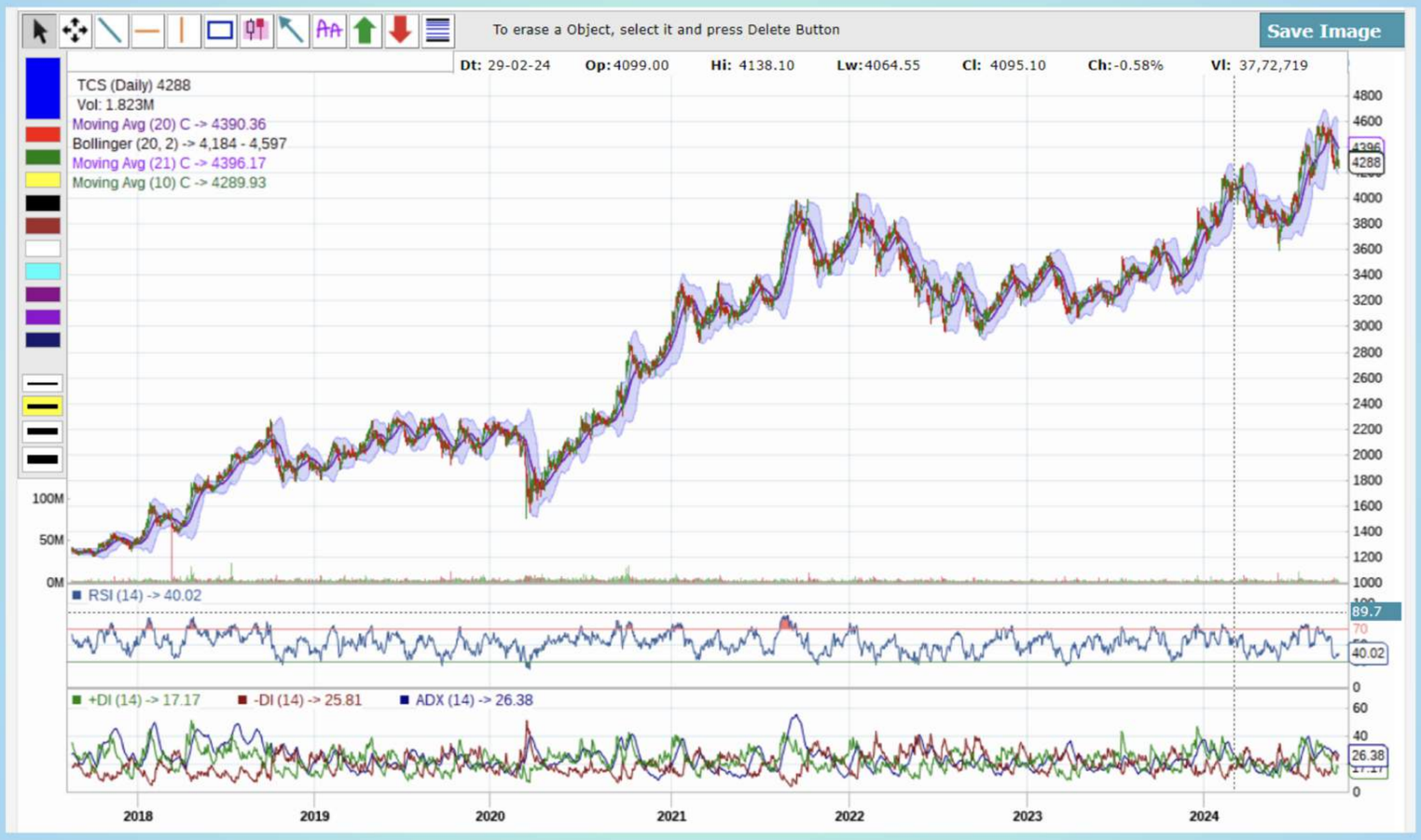
source : https://chartink.com/

Hindalco (8 years Data)



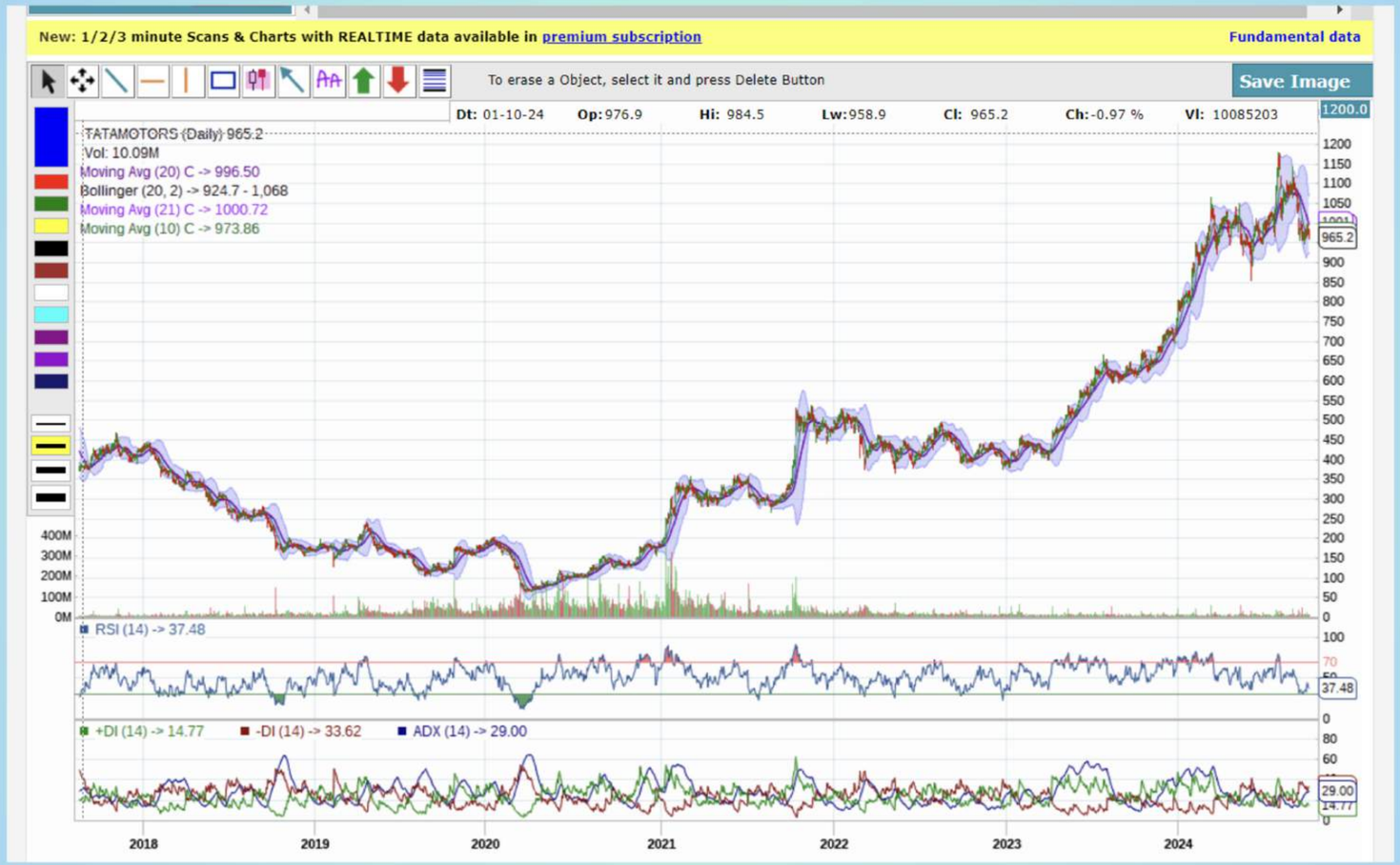
source : https://chartink.com/

TCS (8 years Data)



source : https://chartink.com/

Tata Motors (8 years Data)



source : https://chartink.com/

Reliance (8 years Data)

source : https://chartink.com/

Data Retrieval:

- Used yfinance to retrieve stock price data (e.g., Date, Close, Open, High, Low, Volume) from **01/01/1996 to 31/12/2023.**
- Applied TA-Lib (Technical Analysis Library) to calculate technical indicators .

Parameters used:

- Price Data:
  - Date, Close, Open, High, Low, Volume
- Technical Indicators:
  - RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), MACD Signal
  - ADX (Average Directional Index)
  - Bollinger Bands: Upper Bollinger Band, Lower Bollinger Band
  - Stochastic Indicators: Stochastic SlowK, Stochastic SlowD
- Derived Data:
  - Weekly: wk_close, wk_rsi, wk_macd
  - Monthly: mon_close, mon_rsi, mon_macd

Technical Indicators:

- **RSI (Relative Strength Index):**
  - A momentum indicator that **measures the speed and change of price movements**. It ranges from 0 to 100, where values above 70 suggest the stock is overbought, and below 30 indicates it's oversold.
- **MACD (Moving Average Convergence Divergence):**
  - A **trend-following indicator** that shows the relationship between two moving averages of a stock's price. The MACD line is calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA.
- **MACD Signal:**
  - A 9-period EMA of the MACD line, often used to generate buy or sell signals. When the MACD crosses above the signal line, it's a buy signal, and when it crosses below, it's a sell signal.
- **Upper Bollinger Band:**
  - Part of the Bollinger Bands, which are **volatility indicators**. The upper band is set two standard deviations above the 20-period moving average, showing a potential upper limit of price movement.
- **Lower Bollinger Band:**
  - The lower band is two standard deviations below the 20-period moving average, showing a potential lower limit of price movement.
- **ADX (Average Directional Index):**
  - An indicator that shows the **strength of a trend**, without indicating its direction. Values above 25 suggest a strong trend, while below 20 indicates a weak trend.
- **Stochastic SlowK:**
  - The Stochastic oscillator compares a stock's closing price to its price range over a specific period. SlowK is the slower, more smoothed version of this value and shows the **general momentum** of the stock.
- **Stochastic SlowD:**
  - A smoothed moving average of SlowK, often used to confirm buy or sell signals.

Most of the works done on State Bank of India Data till now

Dataset Description : Shape (7036, 21)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Date	Close	Open	High	Low	Volume	RSI	MACD	MACD Sigr	Upper Boll	Lower Boll	ADX	Stochastic	Stochastic	wk_close	wk_rsi	wk_macd	mon_close	mon_rsi	mon_macd
2	count	7036	7036	7036	7036	7036	7.04E+03	7022	7003	7003	7017	7017	7009	7017	7017	7031	6961	6866	7014	6711	6296
3	mean	46:33.6	161.3654	178.7994	181.1733	176.1297	3.13E+07	51.98166	0.584311	0.575461	172.1209	149.754	26.30813	-87.7922	-87.8251	161.2452	53.93402	2.698634	160.7979	57.18297	10.5408
4	min	01-01-1996 00:00	9.374208	13.4782	13.9358	13.21401	0.00E+00	8.751336	-30.236	-27.0205	10.2251	7.446549	8.291767	-1001.77	-992.94	9.374208	20.56984	-37.4206	9.374208	33.18415	-26.2356
5	25%	26-09-2002 18:00	19.49948	28.38229	28.76795	27.94591	1.30E+07	42.27788	-0.87921	-0.76596	20.47298	18.36326	18.89834	-132.967	-133.881	19.52692	44.64685	-1.02501	19.19543	49.49824	-0.03051
6	50%	24-09-2009 12:00	149.1705	172.4	175.595	169.175	2.07E+07	51.93869	0.190687	0.18894	166.0652	135.7956	24.37954	-63.1475	-62.7804	149.5561	54.15582	0.849356	152.1113	56.23634	6.547599
7	75%	18-11-2016 18:00	240.9911	264.1375	268	260.6863	3.66E+07	61.29498	2.324053	2.231696	257.6564	225.3673	31.52329	-15.6682	-15.4299	241.131	62.93017	6.929783	240.9911	64.42264	16.55085
8	max	29-12-2023 00:00	644.5905	658.7	660.4	646.5	4.47E+08	90.72388	30.99813	27.4948	661.4083	577.9237	69.3671	84.86292	83.73399	637.5584	86.73232	42.65003	609.971	86.71454	64.91261
9	std	NaN	147.6784	152.2748	153.7901	150.49	3.47E+07	12.99392	4.972251	4.666503	154.861	138.6803	10.18892	110.5112	109.2715	147.2743	12.44772	9.886157	146.4173	10.87222	17.04631
10																					

Some Data

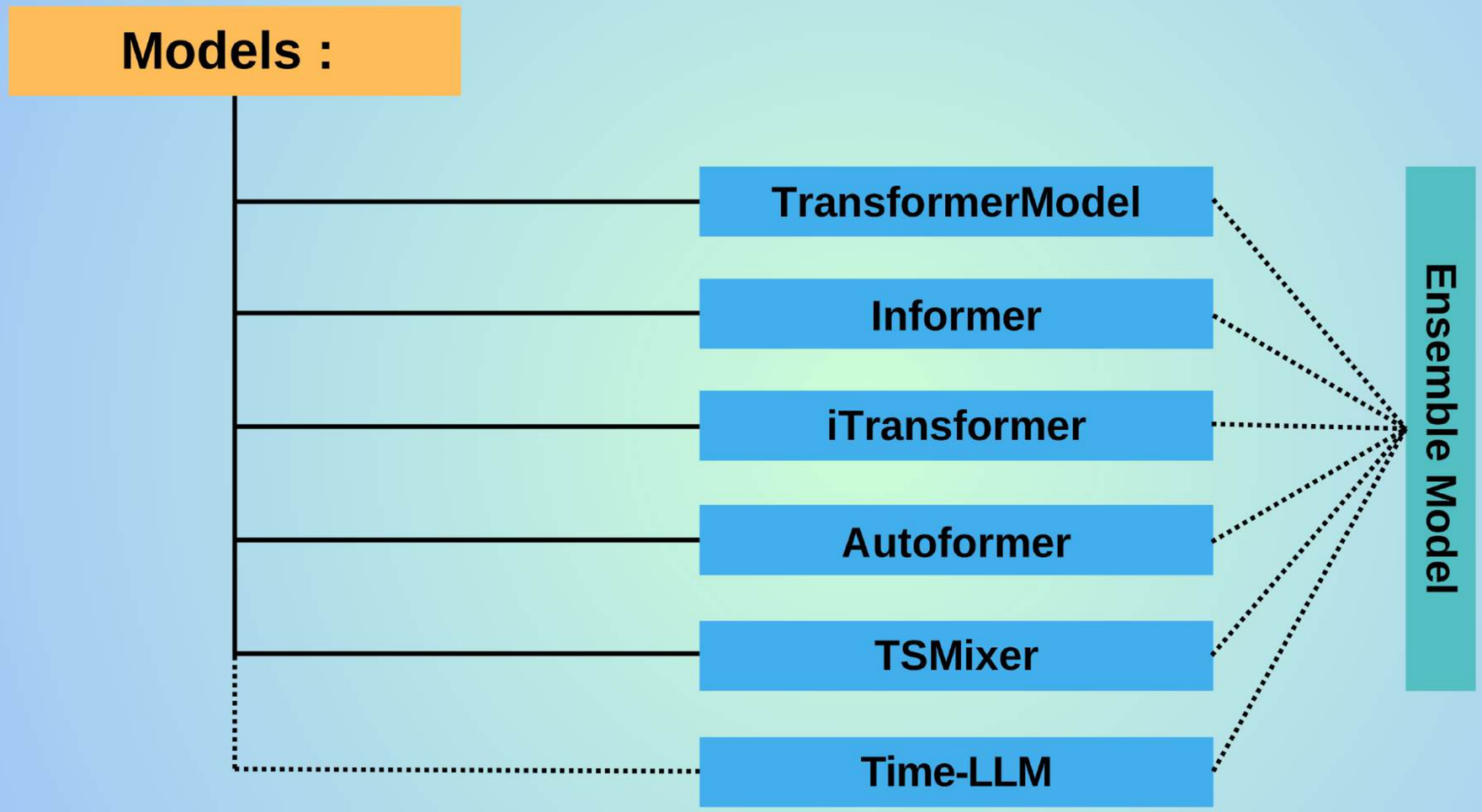
Date	Close	Open	High	Low	Volume	RSI	MACD	MACD Sigr	Upper Boll	Lower Boll	ADX	Stochastic	Stochastic	wk_close	wk_rsi	wk_macd	mon_close	mon_rsi	mon_macd
26-12-2022	576.057739	574	601.7	570.7	13201587	48.24109	-0.88771	3.188401	606.1427	564.062	24.74351	-0.6585	-24.3891	553.7718	55.92561	25.85338	581.2192	72.83305	61.7712
27-12-2022	580.688599	600.4	603.1	593.3	9638618	50.69481	-0.66259	2.418203	605.8856	563.6775	24.00241	15.6959	-5.8898	553.7718	55.92561	25.85338	581.2192	72.83305	61.7712
28-12-2022	579.86853	600.9	607	598.55	7988631	50.24058	-0.54408	1.825745	605.8718	563.5562	22.87358	19.05865	11.36535	553.7718	55.92561	25.85338	581.2192	72.83305	61.7712
29-12-2022	590.239685	600	618	597.15	20130392	55.65232	0.382291	1.537055	606.1732	563.6119	21.74214	32.24682	22.33379	553.7718	55.92561	25.85338	581.2192	72.83305	61.7712
30-12-2022	592.072754	615.25	620.55	611.35	13052895	56.55176	1.249955	1.479635	606.6953	563.683	20.91697	38.1189	29.80813	553.7718	55.92561	25.85338	581.2192	72.83305	61.7712

How to choose these to predict?

Key Principles for Combining Technical Indicators

- **Diversity:**
  - Use different types of indicators (trend, momentum, volatility, volume) for a well-rounded view.
  - Example: Combine Moving Average (trend) with RSI (momentum).
- **Non-Correlation:**
  - Avoid redundant indicators. Choose ones that provide unique insights.
  - Example: Pair Bollinger Bands (volatility) with ADX (trend strength).
- **Timeframe Variation:**
  - Mix short-term and long-term indicators.
  - Example: Use 14-day RSI with 200-day Moving Average.
- **Complementarity:**
  - Ensure indicators work together without contradictions for a clearer picture.

Models Used


Models :
TransformerModel
Informer
iTransformer
Autoformer
TSMixer
Time-LLM
Ensemble Model

TransformerModel

Architecture Overview:

- The TransformerModel from the Darts library is based on the original Transformer architecture for sequence-to-sequence tasks. It consists of stacked encoder and decoder layers, utilizing self-attention mechanisms.

Key Features:

- **Self-Attention:** Each input element attends to all others, effectively capturing long-range dependencies.
- **Components**:
  - **Encoder:** Generates a context representation from the input time series.
  - **Decoder:** Produces the output sequence based on the encoder's representation.

Key Strengths:

- **Long-range Dependency Capture:** Efficiently captures long-range dependencies, suitable for forecasting influenced by distant past values.
- **Parallelization:** Enables faster training on large datasets.

TransformerModel Implementation

```
from darts.models import TransformerModel

# model initialization
   model = TransformerModel(
            input_chunk_length=60,
            output_chunk_length=7,
            d_model=128,
            nhead=8,
            num_encoder_layers=4,
            num_decoder_layers=4,
            dim_feedforward=1024,
            dropout=0.2,
            n_epochs=50,
            optimizer_cls=torch.optim.Adam,
            optimizer_kwargs={'lr': 0.0005},
            lr_scheduler_cls=torch.optim.lr_scheduler.ReduceLROnPlateau,
            lr_scheduler_kwargs={'patience': 5, 'factor': 0.2, 'verbose': True, 'monitor': 'train_loss'},
            pl_trainer_kwargs=self.pl_trainer_kwargs
        )
```

Informer

Architecture Overview:

- Informeris designed to tackle two major limitations of traditional transformers:
  - Memory inefficiency when processing long sequences.
  - Computational complexity due to attention mechanisms.
- Informer introduces **ProbSparse** Attention (probabilistic sparse attention), which reduces computational complexity by only focusing on the most important time steps instead of every single one.
- Additionally, it uses **Distilling operations**: This process downsamples (reduces) the time series, allowing the model to focus on high-level patterns without sacrificing too much detail.

Key Strengths:

- Highly efficient: Works well with long sequences by reducing unnecessary computations (important for datasets with long histories).
- Scalability: Handles very long time series efficiently, making it suitable for large datasets or high-dimensional time series.

Informer Implementation

```
from neuralforecast.models import Informer

# Model initialization
        model = Informer(
            h=7,  # Output horizon (prediction length)
            input_size=60,  # Input window size
            hidden_size=128,
            conv_hidden_size=32,
            n_head=4,
            dropout=0.2,
            encoder_layers=2,
            decoder_layers=1,
            factor=3,
            distil=True,
            loss=MAE(),
            learning_rate=5e-4,
            max_steps=250,  # Adjusted as per your code
            **{'callbacks': [self.save_loss_callback]}  # Pass the callback directly here
        )
```

itransformer

Key Features:

- **Improved Attention Visualization:** Offers visualizations of attention mechanisms, helping users identify influential parts of the input sequence.
- **Explainable Components:** Incorporates techniques like feature importance analysis to clarify predictions.
- **Embedding of Time Points:** Time points from individual time series are embedded into variate tokens, which the attention mechanism uses to capture multivariate correlations. A feed-forward network learns nonlinear representations for each token.

Key Strengths:

- **Focus on Explainability:** Valuable in fields like healthcare and finance, where understanding predictions is crucial.
- **Retained Performance:** Maintains the accuracy and effectiveness typical of Transformer models while prioritizing interpretability.

itransformer Implementation

```
from neuralforecast.models import iTransformer
# Model Initialization
        model = iTransformer(
            h=7,  # Output horizon (prediction length)
            input_size=60,  # Input window size
            n_series=1,  # Number of time series (SBIN in this case)
            hidden_size=512,  # Adjusted for iTransformer
            n_heads=8,
            e_layers=2,
            d_layers=1,
            d_ff=2048,
            factor=1,
            dropout=0.1,
            use_norm=True,
            loss=MAE(),
            learning_rate=0.001,
            max_steps=500,  # Adjusted as per your code
            **{'callbacks': [self.save_loss_callback]}  # Pass the callback directly here
        )
```

Autoformer

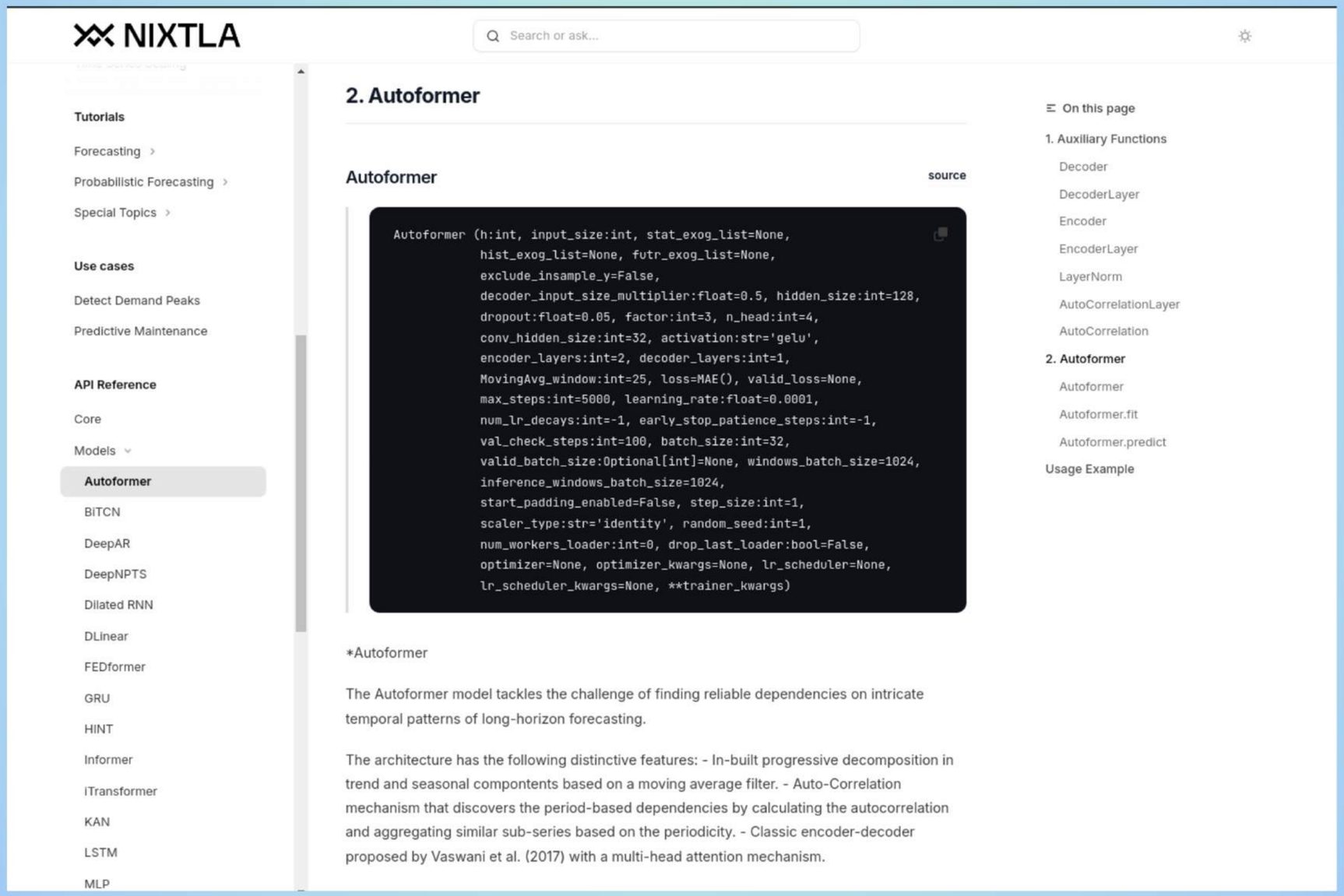
Architecture Overview:

- Utilizes an **auto-correlation** mechanism instead of traditional attention, focusing on repeating patterns in time series data.
- Incorporates a decomposition block to separate data into trend and seasonal components.

Key Strengths:

- **Auto-correlation:** Enhances time series forecasting by emphasizing internal patterns.
- **Decomposition:** Improves accuracy by distinguishing long-term trends from seasonal patterns.
- **Efficiency:** Reduces computational load, making it effective for long time series with recurring patterns.

Autoformer Implementation



TSMixer Model

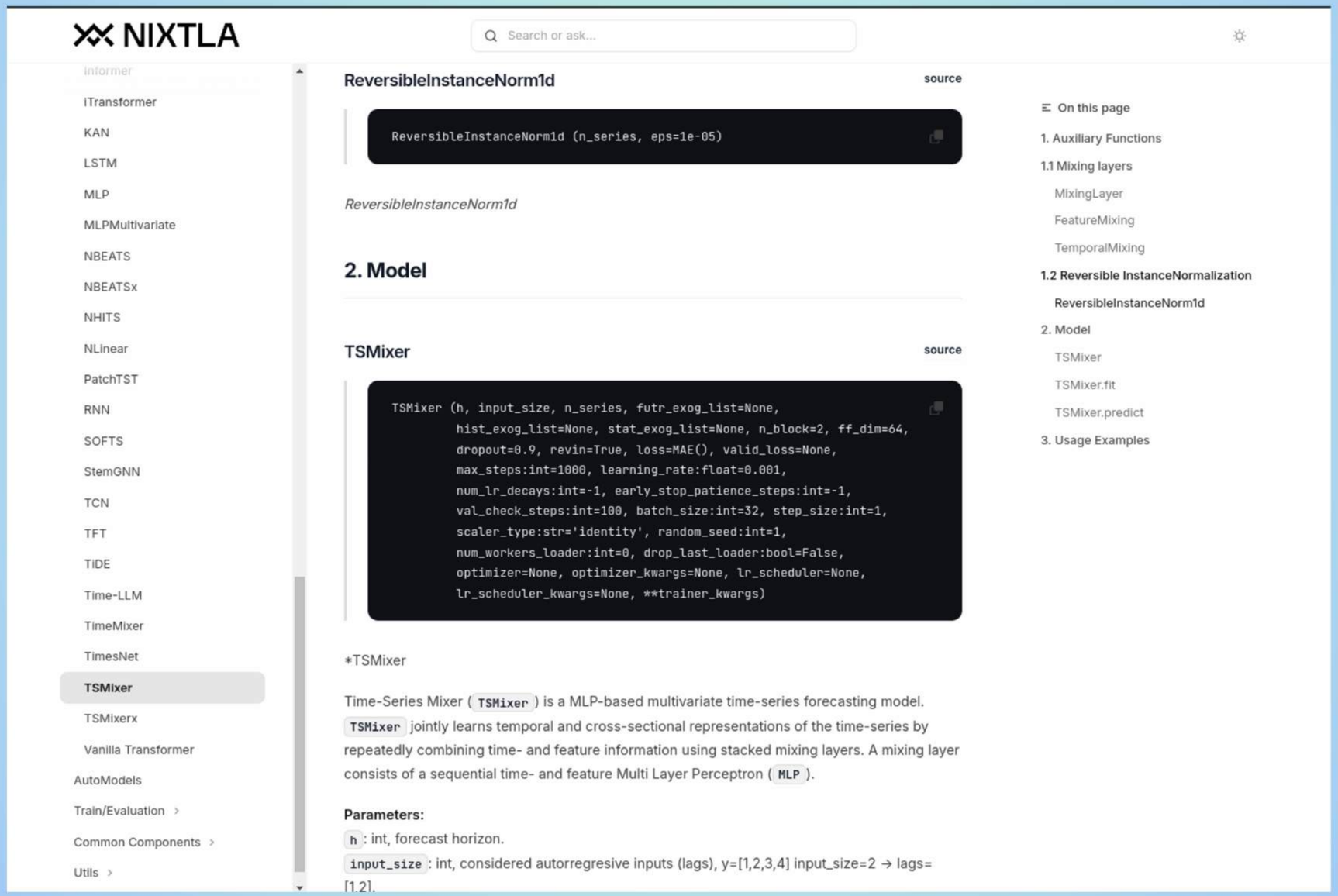
Architecture Overview:

- A simpler, lightweight model that uses token-mixing MLPs instead of self-attention, treating time series data as a series of tokens.

Key Strengths:

- **Simplicity:** Ideal for applications where the complexity of transformers is unnecessary.
- **Efficiency:** Requires less computational power and is faster to train due to its straightforward architecture.

TSMixer Model Implementation



Summary of Key Differences

Model	Key Feature	How It Works	Strengths
TransformerModel (Darts)	Advanced attention mechanism	Utilizes **self-attention** to focus on relevant parts of the input sequence, capturing both local and global dependencies.	Effectively balances local and global dependencies, allowing it to learn complex patterns in time series data.
Informer	ProbSparse Attention + Distilling	Implements a **probabilistic approach** to attention, reducing computation while maintaining key information.	Highly efficient and scalable for processing long sequences, enabling the model to handle large datasets with minimal computational cost.
iTransformer	Focus on interpretability	Integrates **interpretable layers** that clarify the model's decision-making process, enhancing user trust.	Enhances explainability and transparency, making it easier for users to understand model decisions and outcomes, which is crucial in sensitive applications.
Autoformer	Auto-correlation mechanism + Decomposition	Identifies repeating patterns in the data and separates them into **trend** and **seasonal** components for focused analysis.	Automatically identifies and captures long-term trends and seasonal components, improving forecasting accuracy by focusing on internal data patterns.
TSMixer	Token-mixing MLP (no attention mechanism)	Processes time series as a series of tokens using basic **MLP**s, allowing for efficient information mixing between time steps.	A simpler architecture that uses token mixing for efficiency, requiring less computational power and allowing for faster training without the complexity of attention mechanisms.

Experimental Results

The parameters set used for the models

```
# Models initialization
model = TransformerModel(
    input_chunk_length=60,
    output_chunk_length=7,
    d_model=128,  # Increased from 64
    nhead=8,  # Increased from 4
    num_encoder_layers=4,  # Increased from 3
    num_decoder_layers=4,  # Increased from 3
    dim_feedforward=1024,  # Increased from 512
    dropout=0.2,  # Increased regularization
    n_epochs=100,  # More epochs for better convergence
    optimizer_cls=torch.optim.Adam,
    optimizer_kwargs={'lr': 0.0005},
    lr_scheduler_cls=torch.optim.lr_scheduler.ReduceLROnPlateau,
    lr_scheduler_kwargs={
        'patience': 5,
        'factor': 0.2,
        'verbose': True,
        'monitor': 'train_loss'  # Use train_loss if no validation set
    },
    pl_trainer_kwargs=pl_trainer_kwargs,
)
```

Settings 1

input_chunk_length=60,
output_chunk_length=7,

Settings 2

input_chunk_length=60,
output_chunk_length=6,

Settings 3

input_chunk_length=60,
output_chunk_length=8,

Settings 4

input_chunk_length=150,
output_chunk_length=15,

Combinatios of Covariates used

past_cov = TimeSeries.from_dataframe(sbi_data[['RSI', 'Volume', 'Open_Close_Diff','high_minus_low']])

Open - RSI (60X6)

Vol- RSI(60X6)

Open - Vol(60X7)

Open-Vol(60X7)

RSI-Vol-OC(60X7)

Open-Vol-RSI(60X7)

Open - RSI-Vol(60X6)

RSI-Vol-OC-HL(60X6)

MACD-Vol-Bolling- RSI(150X10)

All_paramters_used(60X6)

Informer
OC-All_others(60X7)

itransformer
OC-All_others(60X7)

The parameters set used for the models

setting 1

```
# Model initialization
model = informer(
    h=7,
    input_size=60,
    n_series=1,
    hidden_size=512,
    n_heads=8,
    e_layers=2,
    d_layers=1,
    d_ff=2048,
    factor=1,
    dropout=0.1,
    use_norm=True,
    loss=MAE(),
    learning_rate=0.001,
    max_steps=1000,
    **{'callbacks': [save_loss_callback]}  )
```

setting 2

```
# Model initialization
model = iTransformer(
    h=7,
    input_size=60,
    n_series=1,
    hidden_size=512,
    n_heads=8,
    e_layers=2,
    d_layers=1,
    d_ff=2048,
    factor=1,
    dropout=0.1,
    use_norm=True,
    loss=MAE(),
    learning_rate=0.001,
    max_steps=1000,
    **{'callbacks': [save_loss_callback]} )
```

The parameters set used for the models

setting 3

```
# Model initialization
model = iTransformer(
    h=7,
    input_size=60,
    n_series=1,
    hidden_size=1024,
    n_heads=8,
    e_layers=2,
    d_layers=1,
    d_ff=2048,
    factor=1,
    dropout=0.1,
    use_norm=True,
    loss=MAE(),
    learning_rate=0.0009,
    max_steps=1000,
    **{'callbacks': [save_loss_callback]}  )
```

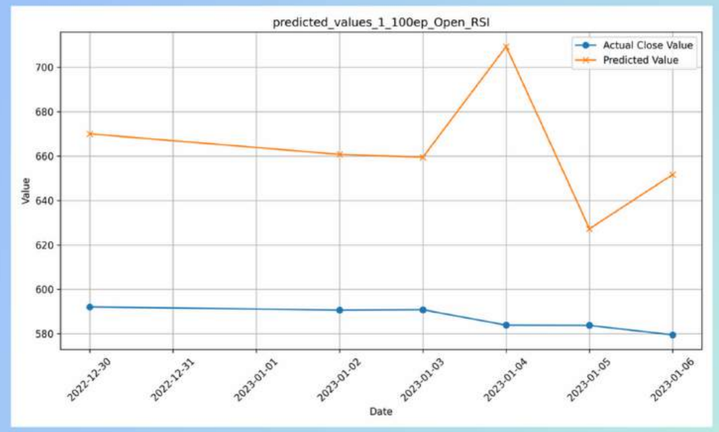
setting 4

```
# Model initialization
model = iTransformer(
 h=8,
 input_size=60,
 n_series=1,
 hidden_size=1024,
 n_heads=8,
 e_layers=2,
 d_layers=1,
 d_ff=2048,
 factor=1,
 dropout=0.1,
 use_norm=True,
 loss=MAE(),
 learning_rate=0.0009,
 max_steps=1000,
 **{'callbacks': [save_loss_callback]} )
```

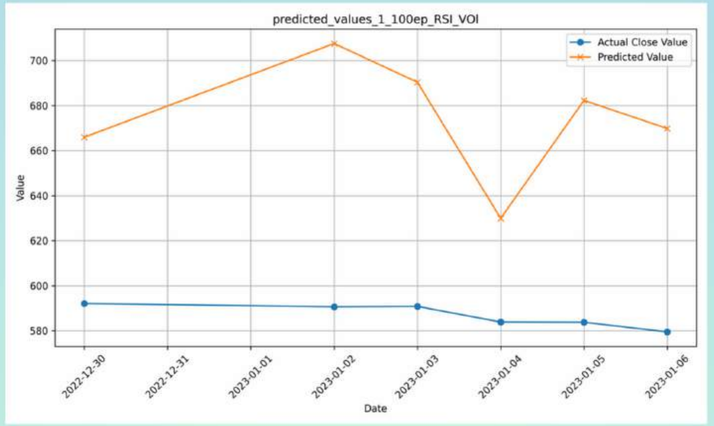
setting 5

```
# Model initialization
model = iTransformer(
 h=15,
 input_size=150,
 n_series=1,
 hidden_size=1024,
 n_heads=8,
 e_layers=2,
 d_layers=1,
 d_ff=2048,
 factor=1,
 dropout=0.1,
 use_norm=True,
 loss=MAE(),
 learning_rate=0.0009,
 max_steps=1000,
 **{'callbacks': [save_loss_callback]} )
```

Predicting on the whole data

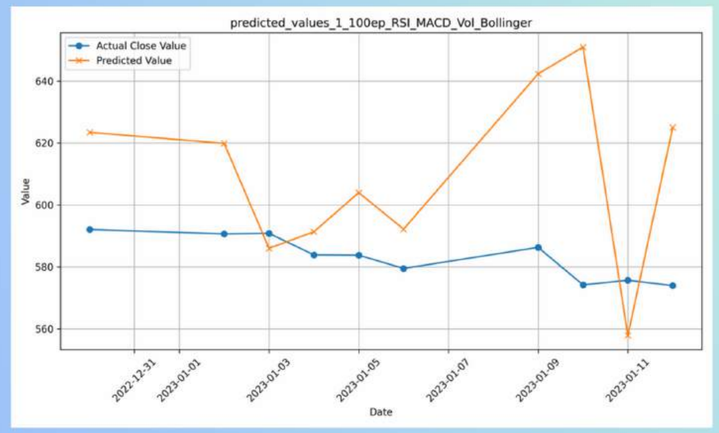


Open - RSI (60X6)

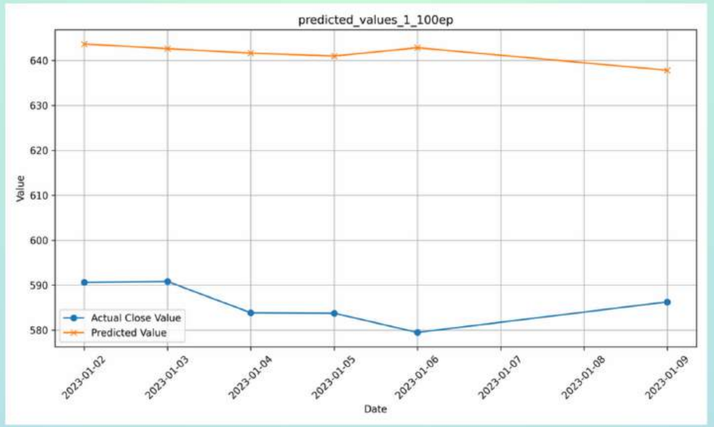


Vol- RSI(60X6)

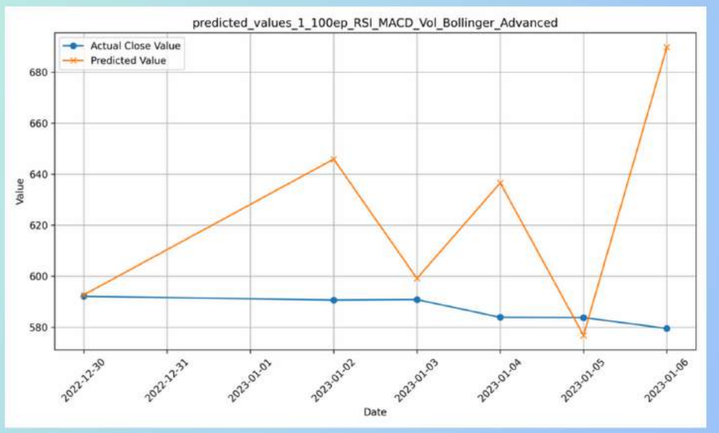
Open - Vol(60X7)



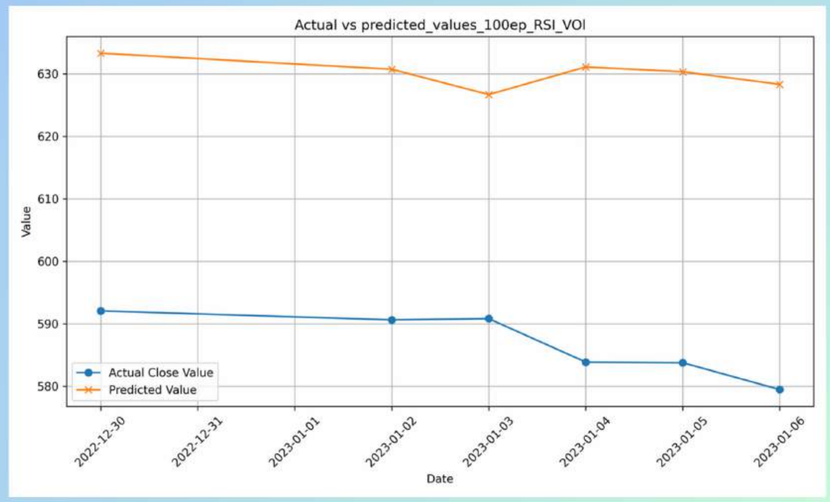
MACD-Vol-Bolling- RSI(150X10)



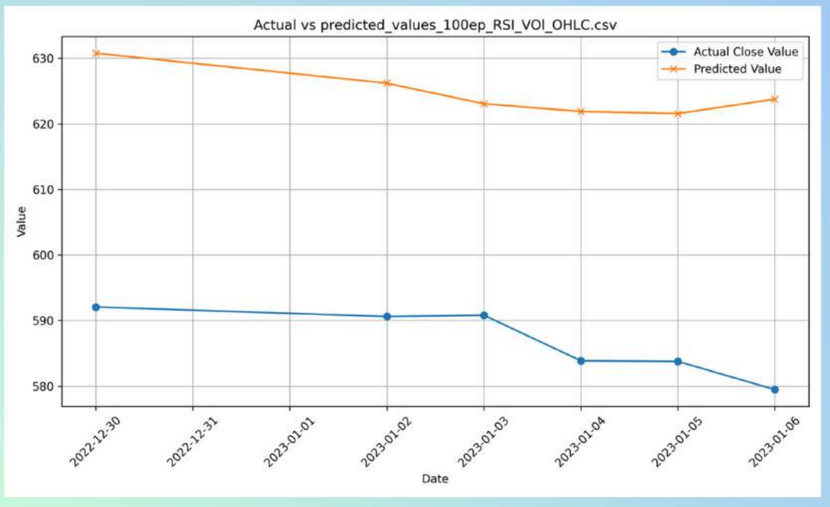
Open - RSI-Vol(60X6)



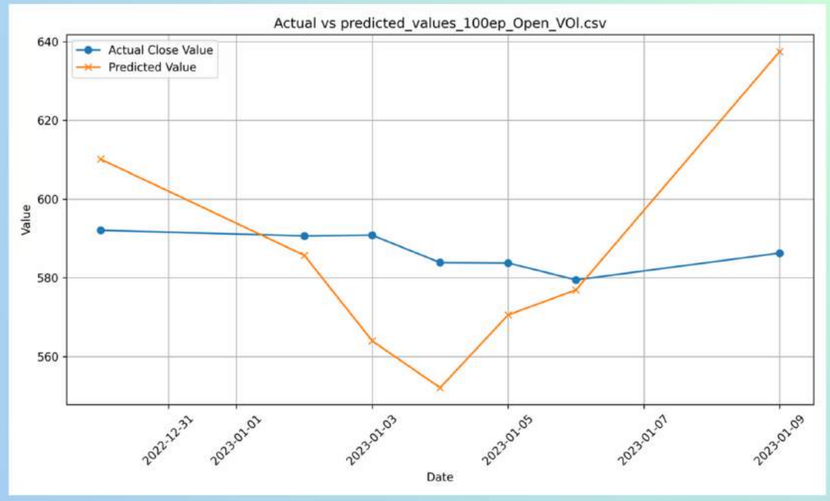
All_paramters_used(60X6)



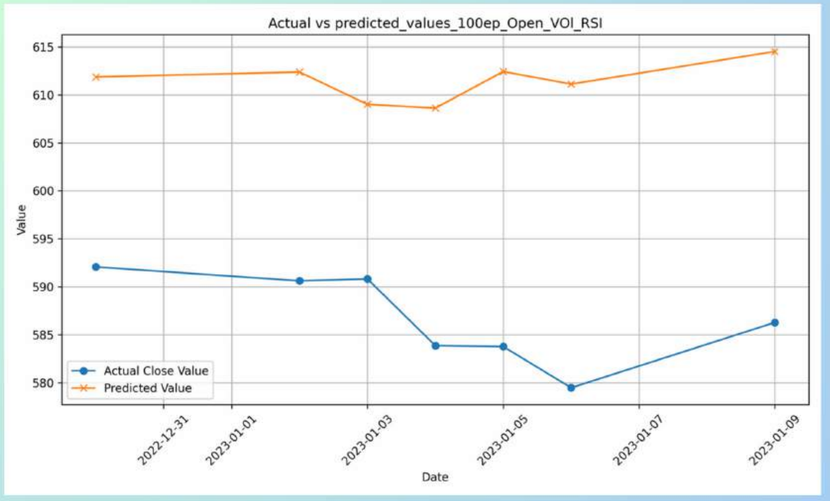
RSI-Vol(60X6)



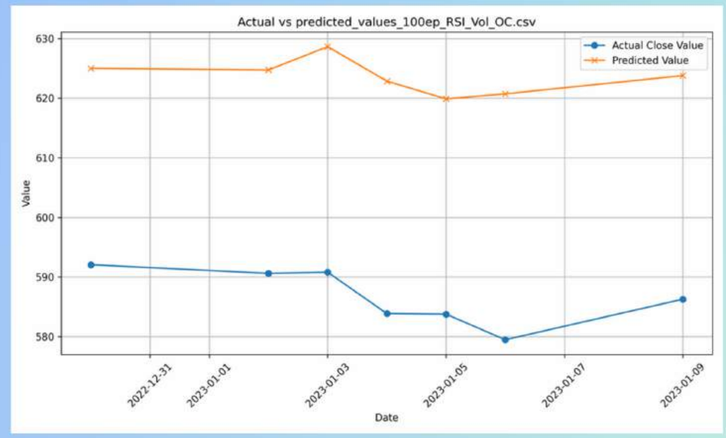
RSI-Vol-OC-HL(60X6)



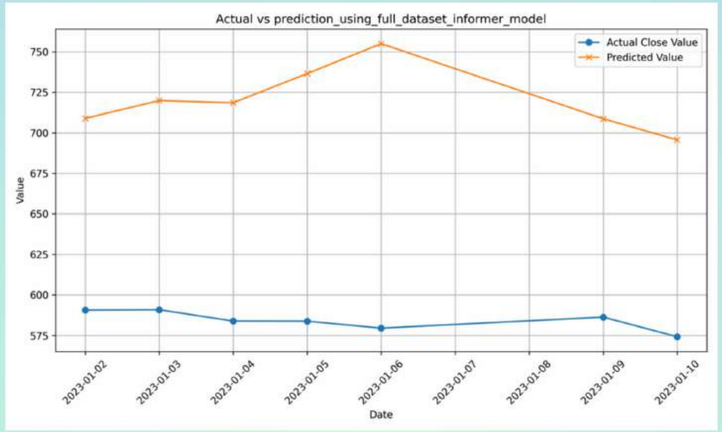
Open-Vol(60X7)



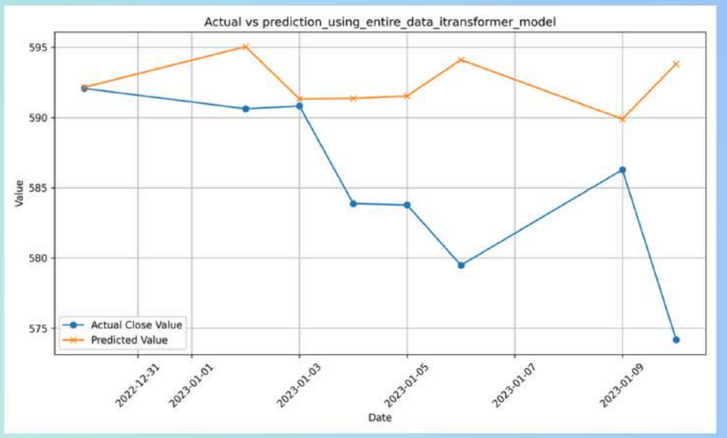
Open-Vol-RSI(60X7)



RSI-Vol-OC(60X7)



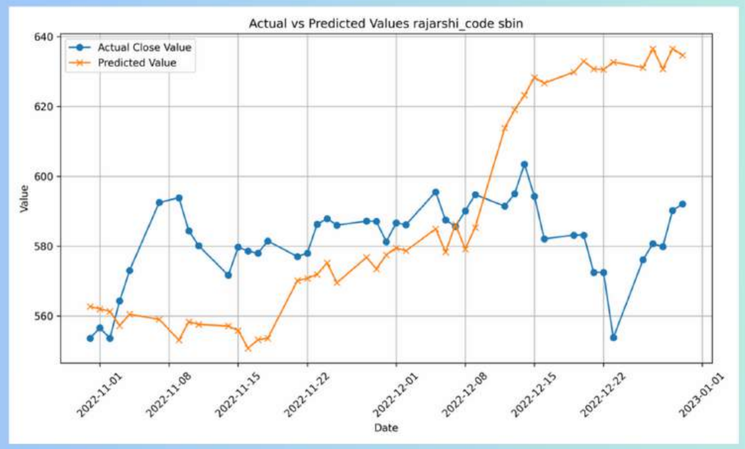
Informer
OC-All_others(60X7)



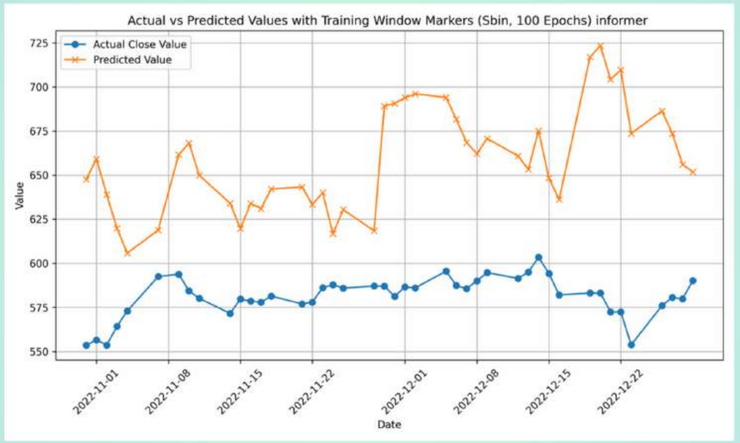
itransformer
OC-All_others(60X7)

OC, derived from "**Open_minus_close**," has significantly improved prediction accuracy. When used alongside other features, it enhanced the **iTransformer** model's performance, leading to more accurate and reliable forecasts.

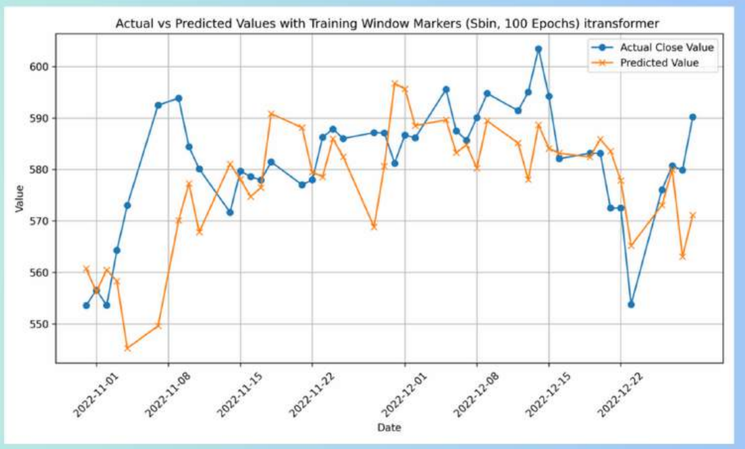
Predicting last 2 months data



Transformermodel

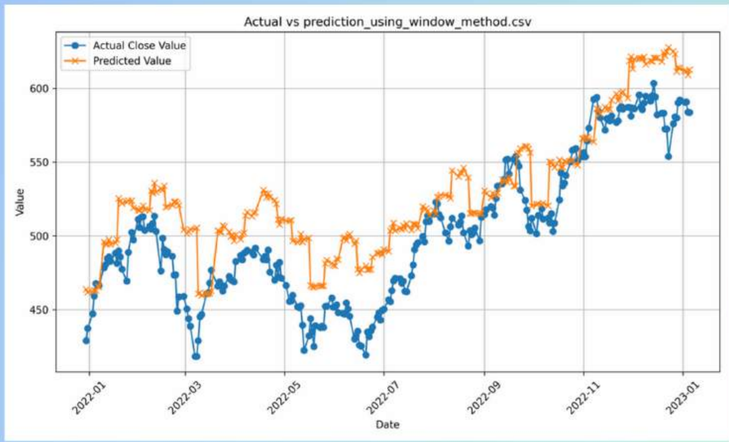


Informer

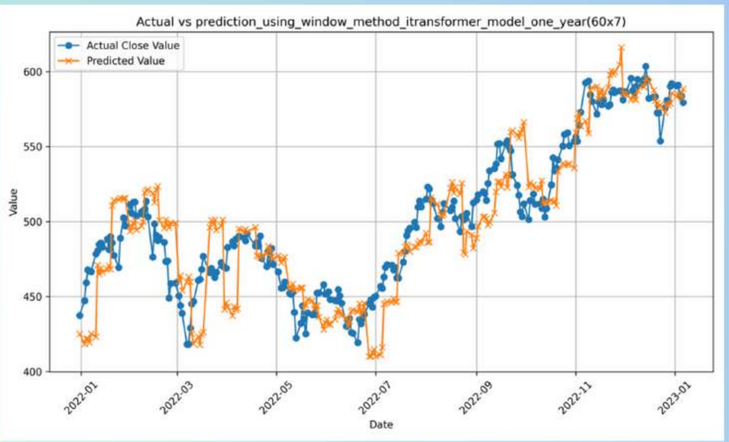


itransformer

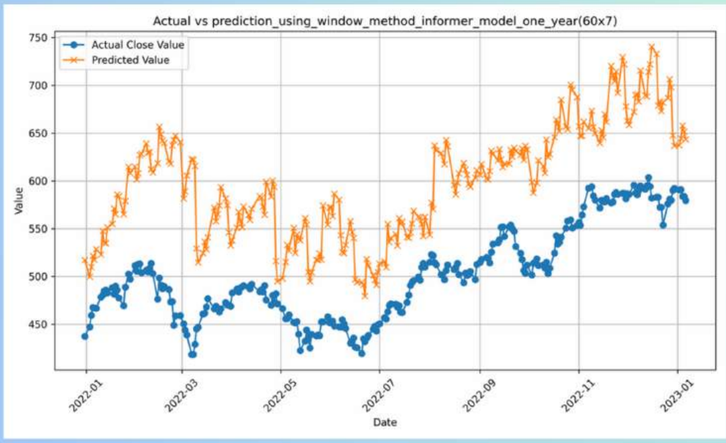
Predicting last 1 year data



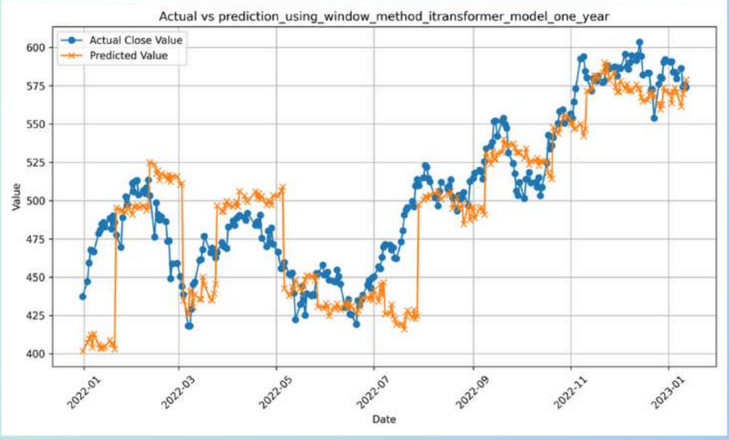
Transformer_model



itransformer_model(60X7)

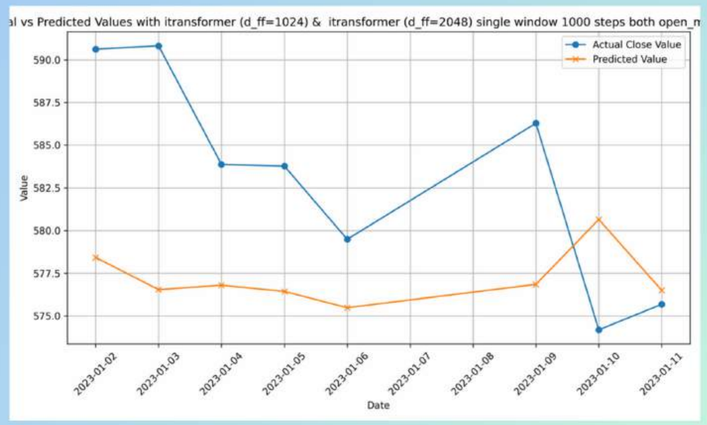


Informer_model

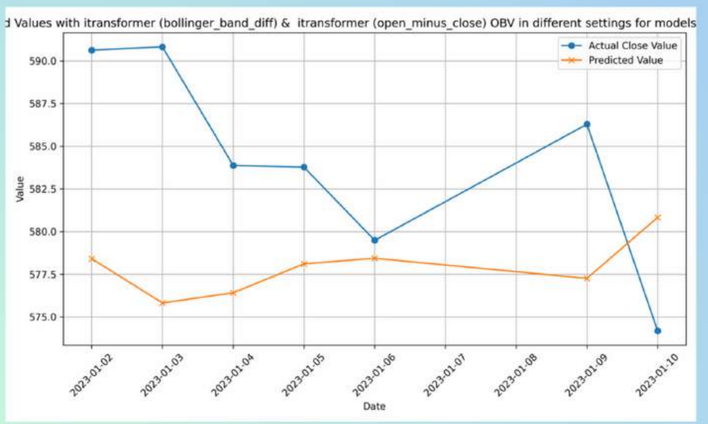


itransformer_model(150X15)

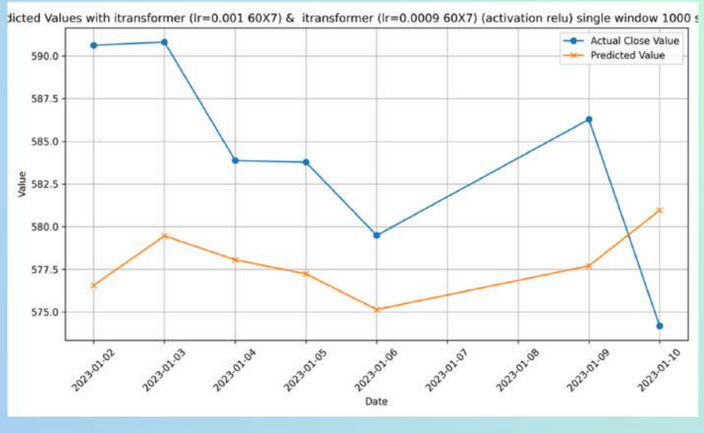
proceeding some sort of ensembling



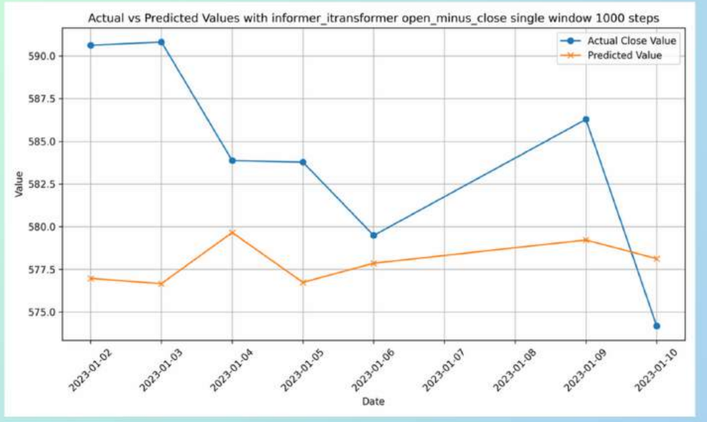
itransformer & itransformer
d_ff=1024 & 2048



itransformer & itransformer
OC & BB & OBV



itransformer & itransformer
lr = 0.001 & 0.0009



informer & itransformer
OC

Performance Comaparison Between Models

	A	B	C	D	E
1	**Data (1996-2022)**	**File_name**	**MAE**	**MSE**	**R^2 error**
2	Whole dataset	predicted_values_100ep_Open_VOl_RSI	24.71899803	632.6520171	-33.05300023
3	Whole dataset	predicted_values_100ep_Open_VOl	21.23457297	698.4516982	-36.59472063
4	Whole dataset	predicted_values_100ep_RSI_Vol_OC	36.9683132	1373.587641	-72.9344521
5	Whole dataset	predicted_values_100ep_RSI_VOl_OHLC	37.78436862	1440.784016	-65.57812434
6	Whole dataset	predicted_values_100ep_RSI_Vol	43.28349086	1894.521839	-86.54519012
7	Whole dataset	prediction_using_entire_data_itransformer_model	7.267330583	93.65121076	-1.80200986
8	Whole dataset	prediction_using_full_dataset_informer_model	136.3311039	18958.52934	-623.7180816
9					
10					
11	Cut 2 months	normal_transformer_with_sbin (epoch 50)	25.26630862	996.5597231	-6.608088482
12	Cut 2 months	prediction_using_window_method_informer_model	76.88227458	6734.175535	-53.50164682
13	Cut 2 months	prediction_using_window_method_itransformer_model	14.18943902	314.1929793	-1.398660042
14					
15					
16	Cut 1 year	prediction_using_window_method (epoch 100)	27.92938585	1099.171213	0.549965019
17	Cut 1 year	prediction_using_window_method_itransformer_model_one_year(150X15)	22.11798577	879.7363805	0.647316317
18	Cut 1 year	prediction_using_window_method_itransformer_model_one_year(60X7)	18.31372278	508.8623802	0.791898842
19	Cut 1 year	prediction_using_window_method_informer_model_one_year(60X7)	94.9070658	10083.28177	-3.123595493
20					
21					
22	Whole dataset	ensemble_prediction_informer_itransformer_model	42.54301035	1841.091616	-59.66732295
23	Whole dataset	ensemble_prediction_itransformer_model	8.215122455	77.4764519	-1.552990242

Smart Contract

What is a Smart Contract?

A smart contract is like a computer program that runs on the blockchain. It automatically enforces rules and actions, like a regular contract, but without needing a middleman (like a lawyer or bank).

How it Works:

If-Then Rules: Smart contracts follow basic "if-then" logic. For example, "If Person A sends money, then Person B will receive the product."

Automatic Execution: Once the conditions are met, the smart contract automatically carries out the agreement. No one needs to approve it; it just happens.

Secure and Transparent: The contract’s terms are written on the blockchain, so everyone can see it, and it can’t be changed once it’s made.

Where it’s Used:

Buying/Selling: Automate payments when goods or services are delivered.

Insurance: Automatically pay claims when certain conditions are met.

Voting: Ensure secure and tamper-proof elections.

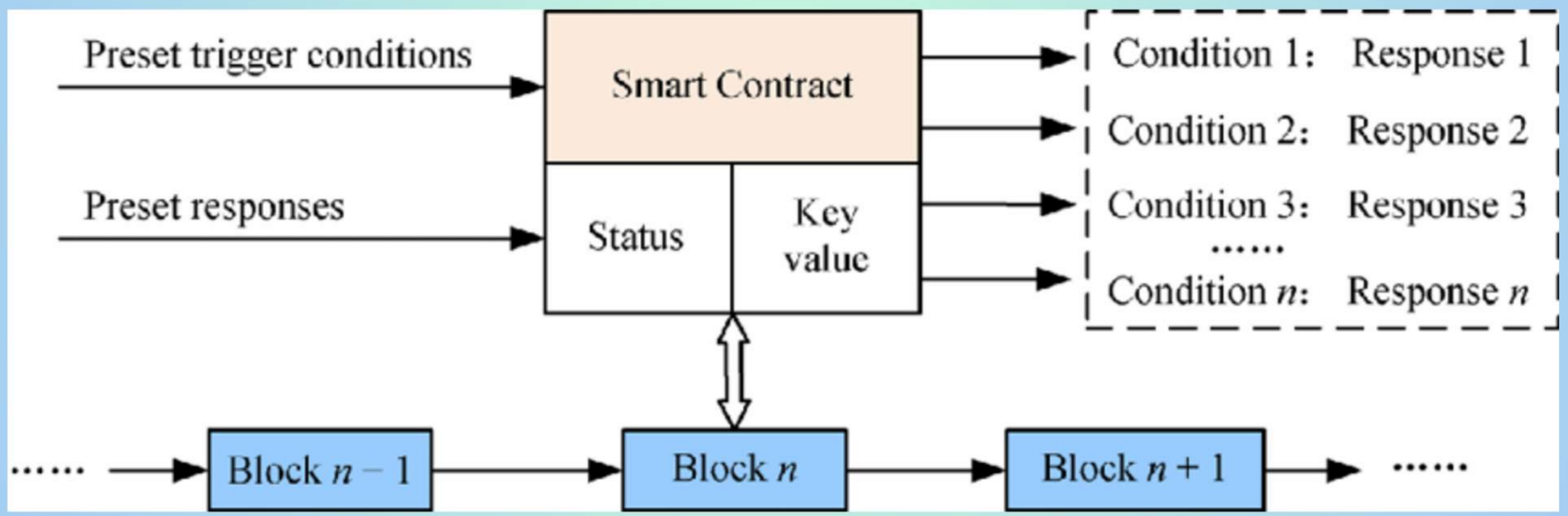
Simple Example:

Imagine a vending machine: You insert money (if you pay).

The machine automatically gives you a snack (then you receive your item). No need for a person to be involved—it's all automatic.

In short, a smart contract is like a self-operating contract that runs automatically when the agreed conditions are met, without needing middlemen.

Smart Contract Structure



Thank You