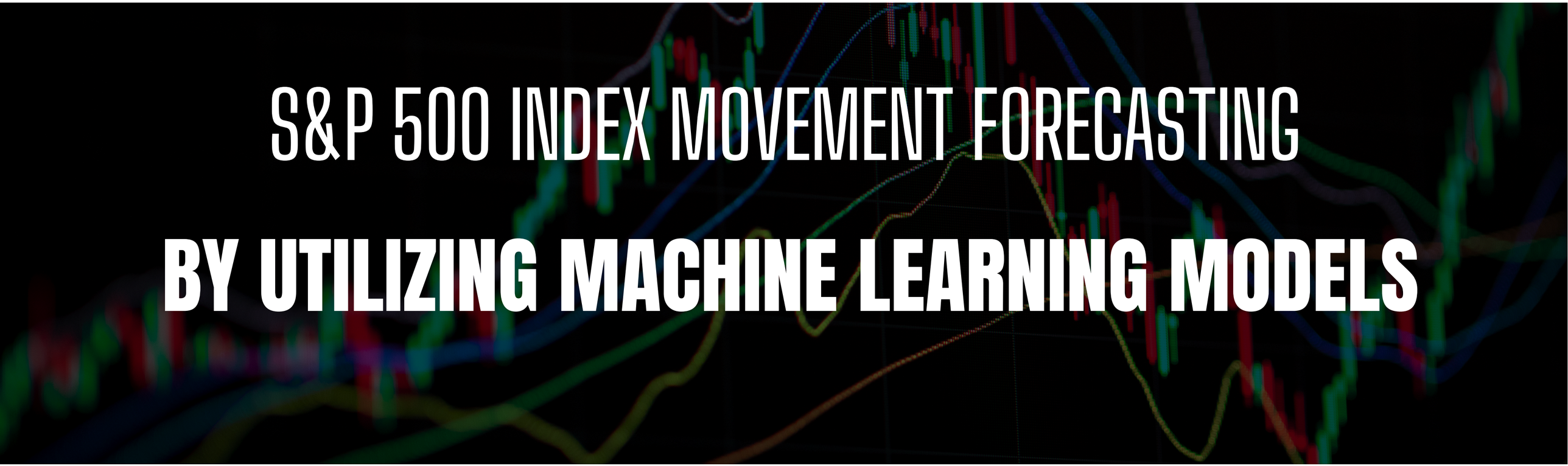
Final Thesis Presentation



# S&P 500 INDEX MOVEMENT FORECASTING

# BY UTILIZING MACHINE LEARNING MODELS

Presented by Le Thi Hong Ha

# AGENDA

# KEY CHARACTERISTICS OF **STOCK DATA**

- Temporal/ time series
- Non-stationary
- Non-linear
- Random walk

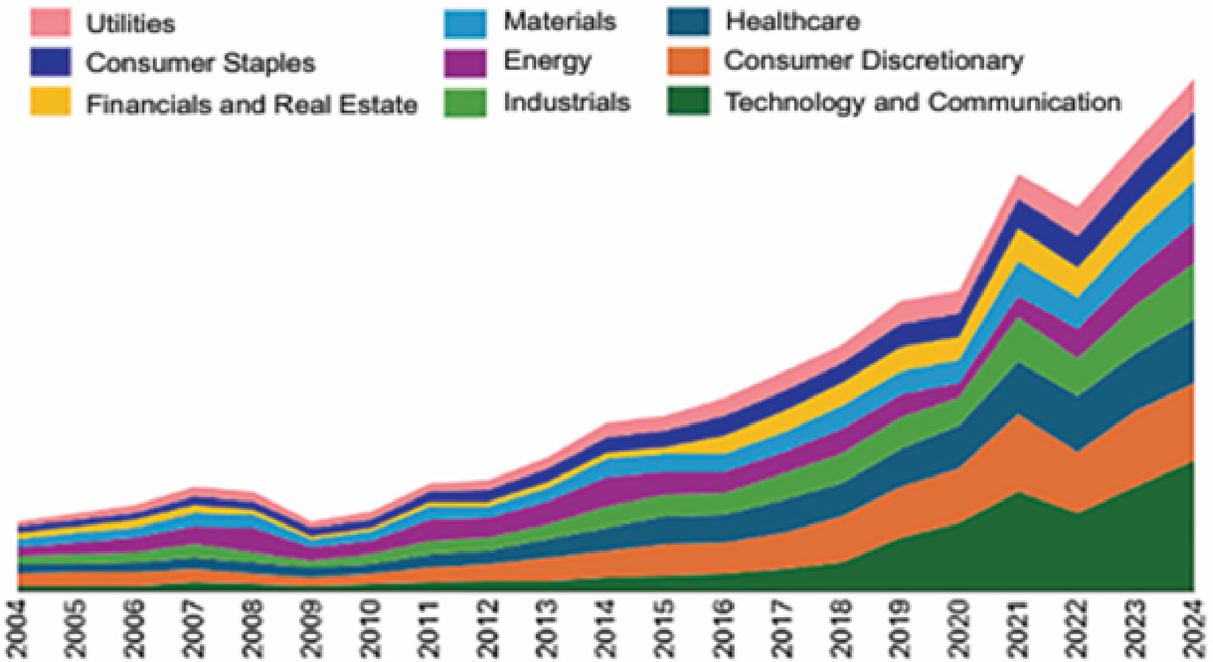
(Hung et al., 2024, Vuong et al., 2024)

No clear pattern of movement

Influenced by external events

# **WHY** THE S&P 500?

- 503 leading companies across various industries
- 80% of the U.S stock market ~ 50% of the worldwide market
- Important to the domestic and global economy
- More stable and impactful than a single stock




**Sectors' distribution in S&P 500 index** (Jha and Jha, 2024)

[1] Hung, M.C., Chen, A.P. and Yu, W.T., (2024) AI-Driven Intraday Trading: Applying Machine Learning and Market Activity for Enhanced Decision Support in Financial Markets. IEEE Access, 12, pp.12953–12962
[2] Vuong, P.H., Phu, L.H., Van Nguyen, T.H., Duy, L.N., Bao, P.T. and Trinh, T.D., (2024) A bibliometric literature review of stock price forecasting: From statistical model to deep learning approach. Science Progress, SAGE Publications Ltd.
[3] Jha, R. and Jha, R., (2024) Identification of Key Impacting Sectors Driving S&P 500 Variation using Statistical Modeling. The American Journal of Student Research. [online]

# Aim and Objectives

## Aim

- Provide a comprehensive examination of S&P 500 index movement forecasting through the application of five tree - based ensemble algorithms and the PyCaret AutoML tool

- Focus on
  - Model performance accuracy improvement
  - Simplicity that both machine learning (ML) experts & non-experts can apply
  - Time & resource consumption efficiency

## Objectives

**1. Data preparation**
To setup PyCaret for preparing the working environment and to pre-process the S&P 500 dataset.

**2. Model training**
To select and train the tree-based ensemble models (Random Forest, XGBoost, AdaBoost, CatBoost, and LightGBM) for the research.

**3. Model evaluation**
To compare the models' performance and choose the two best ensemble learning models based on evaluation error metrics such as MAE and RMSE.

**4. Model improvement**
To improve the overall research model performance by tuning hyperparameters and blending the two top ensemble learning models to leverage their combined strengths


# FORECASTING TECHNIQUES
# IN STOCK MARKET

The **importance of financial forecasting**: plays a crucial role in investment decision-making, as researchers or investors or stakeholders enable to anticipate market movement, make data-driven choices, enhance portfolio management strategies and risk assessment

- Statistical techniques: ARIMA and its variants
- Machine Learning techniques
  - Traditional Machine Learning: Random Forest, LightGBM, XGBoost, Linear Regression, etc,
  - Deep Learning: LSTM, Transformer, CNN, etc,
- Hybrid techniques

(Balasubramanian et al., 2024; Vuong et al., 2024)

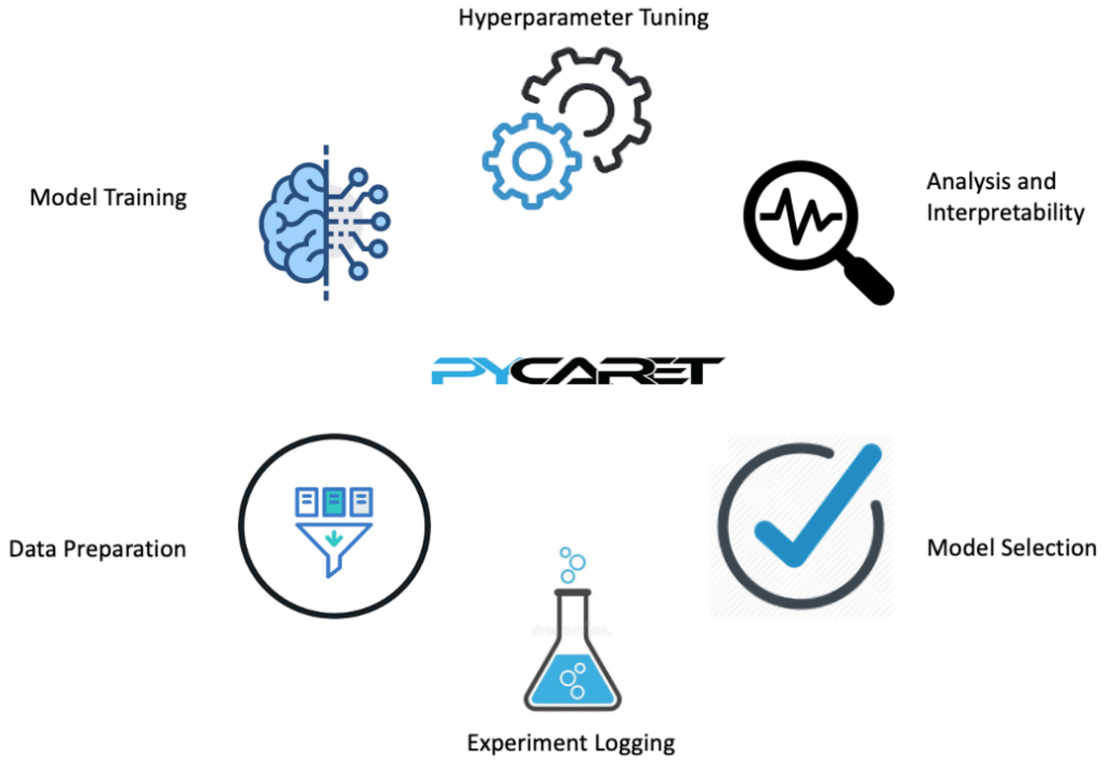
[1] Balasubramanian, P., Chinthan, P., Badarudeen, S. and Sriraman, H., (2024) A systematic literature survey on recent trends in stock market prediction. PeerJ Computer Science, 10.
[2] Vuong, P.H., Phu, L.H., Van Nguyen, T.H., Duy, L.N., Bao, P.T. and Trinh, T.D., (2024) A bibliometric literature review of stock price forecasting: From statistical model to deep learning approach. Science Progress, SAGE Publications Ltd.

# Literature Review - AutoML

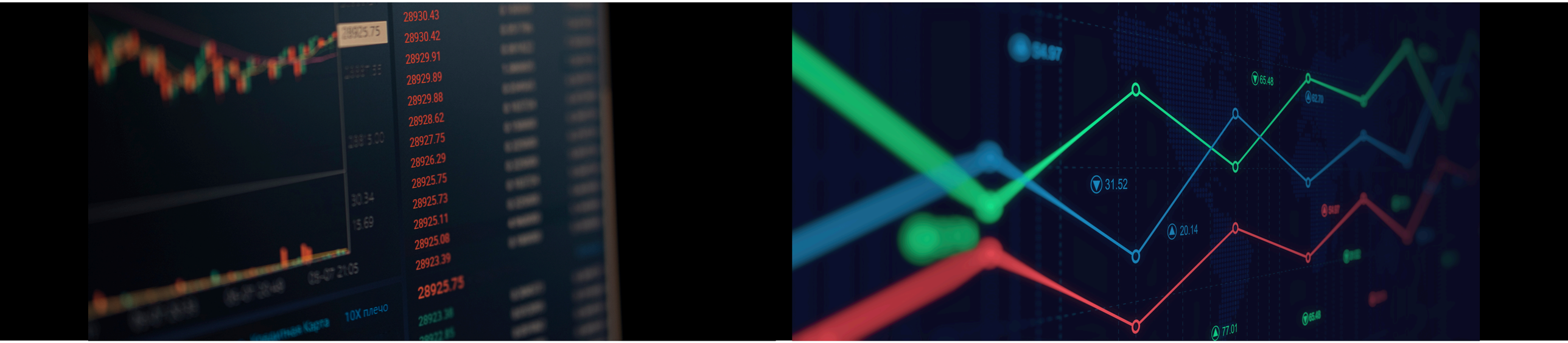
PyCaret is an open-source, low-code machine learning library for Python that streamlines and automates machine learning workflows (Moez, 2020).

PyCaret ‘abstract aways’ coding; users can replace lengthy scripts of code with just a few simple commands. (Great for experimentation and analysis.)

- Accessible tool designed for individuals with minimal expertise in machine learning  (Westergaard et al., 2024).




[1] Moez, A., (2020) PyCaret: An open source, low-code machine learning library in Python. https://www.pycaret.org.
[2] Westergaard, C., Erden, U., Mateo, A., Lampo, M., Akinci, C. and Topsakal, O., (2024) Time Series Forecasting Utilizing Automated Machine Learning (AutoML): A Comparative Analysis Study on Diverse Datasets. Information (Switzerland), 151.




# RESEARCH GAPS

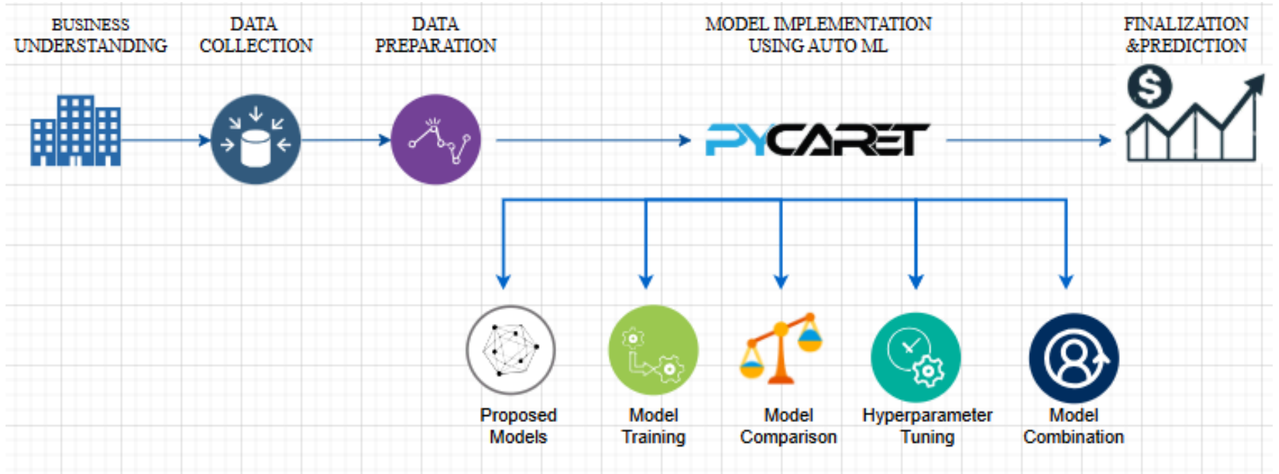
- Complexity with data pre-processing, cross-validation, hyperparameter tuning that require domain expertise
- Computational cost and time constraints
- Limited number of models used for training (two-three models)
- Mainly short-term forecast (one-step ahead)

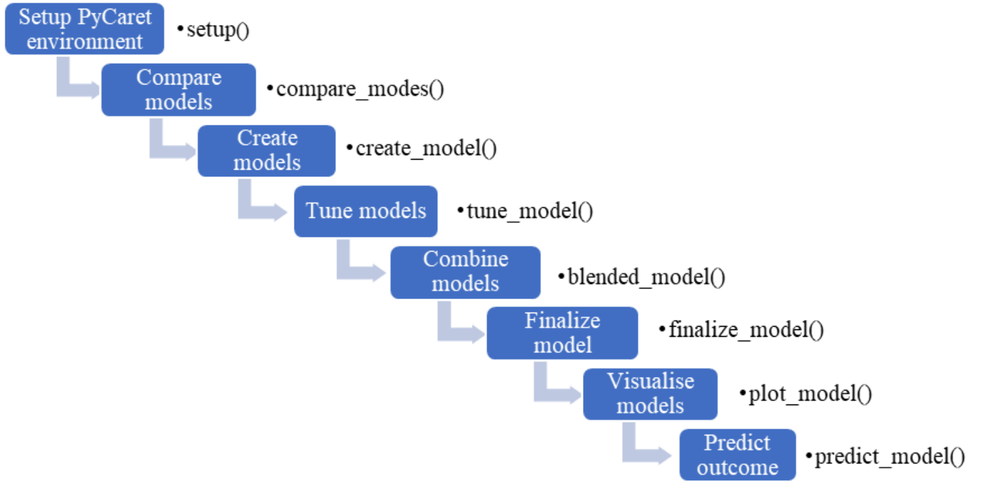
## Problem Statement

This study:

- **aims** to address research gaps related to the challenges faced by both ML experts and non-experts.
- **tackles** the complexity of hyperparameter tuning, the difficulties of forecasting across various time scales, the demand for intensive computational resources in ML models, and how non-experts can benefit from ML learning with minimal domain knowledge.
- **contributes** to the growing field of financial time-series prediction.
- **demonstrates** the potential of automated ML and ensemble algorithms in capturing complex stock market dynamics.

# Research Methodology - Flow Chart




01

## Data Collection

Retrieved from Kaggle & Yahoo Finance
Data description:

- Input features
- Number of observations
- Data types
- Time frame

02

## Data preparation

- Missing values
- Duplicate values
- Outliers
- Transformation
- Feature selection

03

## Model training

- Train & test set split
- Cross-validation

04

## Evaluation metrics

- 7 error metrics: RMSE, MAE, MAPE, R2, RMSSE, MASE, SMAPE.
- Focus on RMSE & MAE

05

## Hyperparameter tuning

- Default setting
- Customized setting

06

## Hybrid model

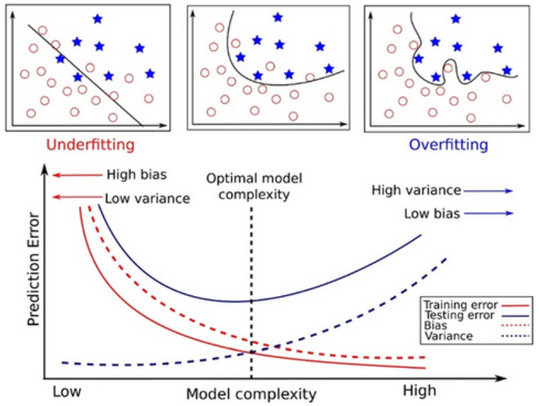
Combine the two top models to investigate its forecasting capability

07

## Finalization

- Finalize the model with the whole dataset
- Predict unseen data

# Research Methodology - Proposed Models




## Bias and variance trade-off

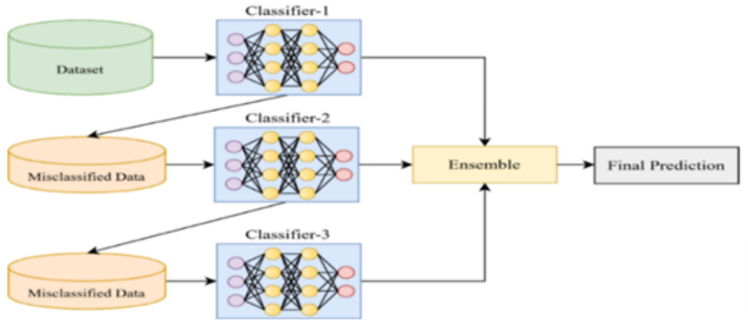
Ensemble methods reduce the risk of overfitting or underfitting and improve the model accuracy (Ranglani, 2024).
They combine multiple models (weak learners) into a more optimal performing model (strong learner) (Sadasivan and Singh, 2024).

### Bagging: Random Forest

multiple decision trees are trained on parallel subsets of the training data and aim to reduce variances

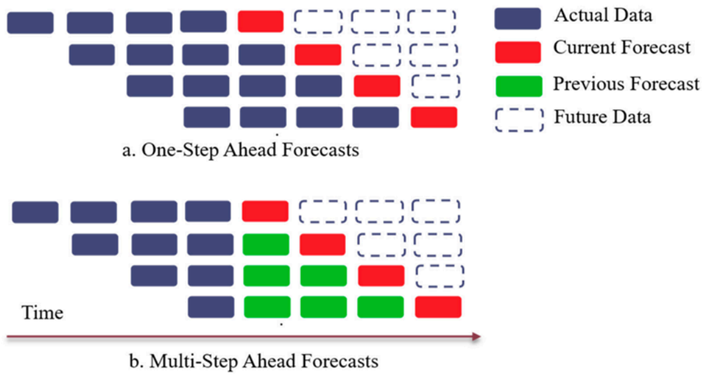

### Boosting: XGBoost, LightGBM, AdaBoost, CatBoost

multiple decision trees are trained sequentially and aim to minimize the errors of their predecessor




[1] Ranglani, H., (2024) Empirical Analysis of the Bias-Variance Tradeoff Across Machine Learning Models. Machine Learning and Applications: An International Journal, [online] 114, pp.01–12.
[2] Sadasivan, P. and Singh, R., (2024) A Review of Prediction Techniques used in the Stock Market. ICST Transactions on Scalable Information Systems, [online] 11. Available at: https://publications.eai.eu/index.php/sis/article/view/7535.

# Research Methodology - Cross-validation




**One-step and Multi-step ahead** (Suradhaniwar et al., 2021)

One-step ahead forecasting = short-term forecasting
Multi-step forecasting = long-term forecasting.

Multi-step forecasting model is when a model predicts a sequence of values instead of one value (Limouni et al., 2023).


**Walk-forward cross-validation**
Sliding window technique

To establish the cross-validation, this study applies:

- One-step ahead forecasting method to predict the next day's index value (t+1)
- Multi-step ahead forecasting techniques to predict the index values for the next multiple days (t+n)

Train set = 80%
Validation set = 10%
Test set = 10%

Cross-validation: walk-forward with a rolling/sliding window technique.

[1] Limouni, T., Yaagoubi, R., Bouziane, K., Guissi, K. and Baali, E.H., (2023) Accurate one step and multistep forecasting of very short-term PV power using LSTM-TCN model. Renewable Energy, 205, pp.1010–1024.
[2] Suradhaniwar, S., Kar, S., Durbha, S.S. and Jagarlapudi, A., (2021) Time series forecasting of univariate agrometeorological data: A comparative performance evaluation via one-step and multi-step ahead forecasting strategies. Sensors, 217.

# Research Methodology - Evaluation Metrics

## 7 metrics in PyCaret:

- Mean Absolute Error (MAE),
- Root Mean Squared Error (RMSE),
- Mean Absolute Percentage Error (MAPE),
- Root Mean Squared Scaled Error (RMSSE),
- Mean Absolute Squared Error (MASE),
- Symmetric Mean Absolute Percentage Error (SMAPE), and
- R-Squared (R2).

(Moez, 2020; Karthika, 2023)

**Proposed evaluation metrics focus more on RMSE and MAE because they cover:**

- Symmetry in over/under forecasting.
- The ability to compare errors for different subsets of time series at different scales.

$$RMSE = \sqrt{[(\frac{1}{n}) * \sum(y_i - \hat{y}_i)^2]} \qquad MAE = (\frac{1}{n}) * \sum|y_i - \hat{y}_i|$$

Where:

n represents the number of data points

$yi$ represents the actual value for the i-th data point

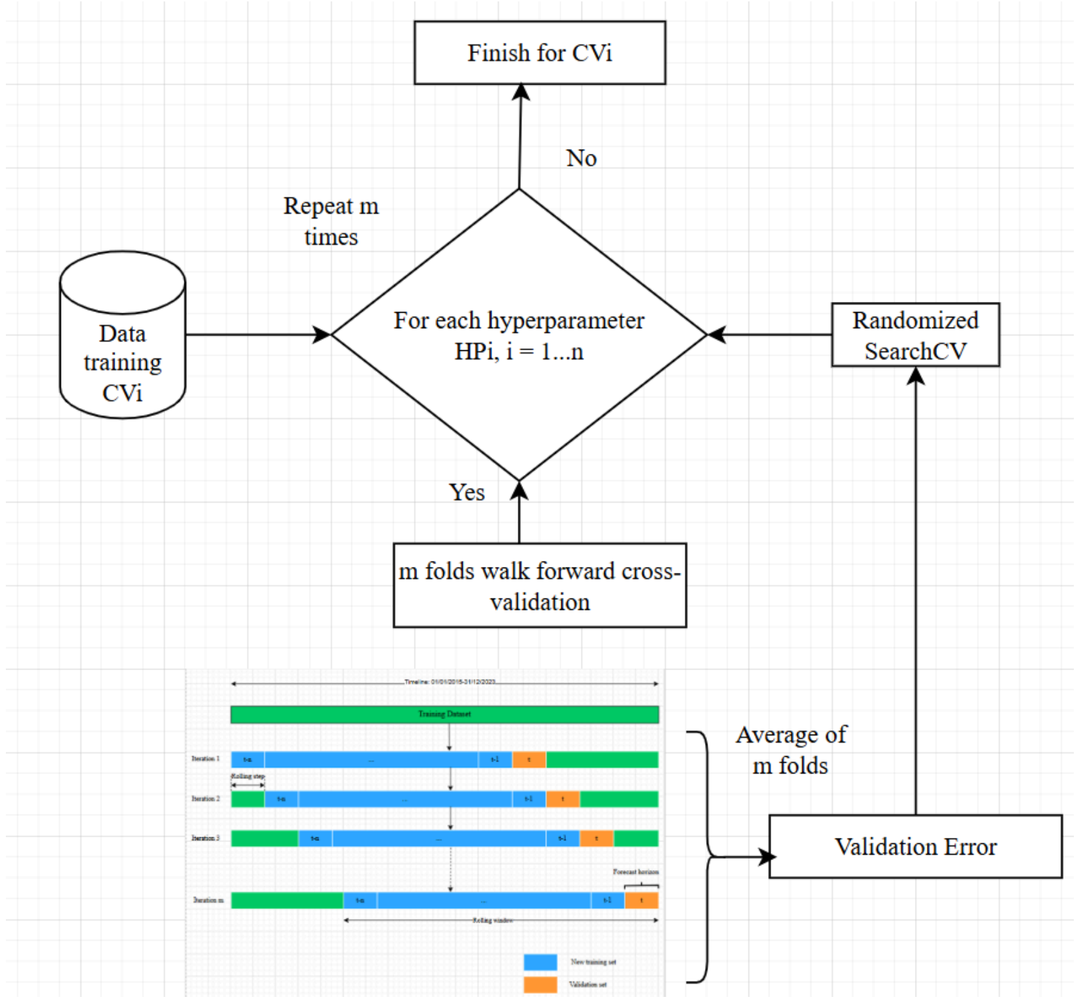
$\hat{y}i$ represents the predicted value for the i-th data point

[1] Moez, A., (2020) PyCaret: An open source, low-code machine learning library in Python. https://www.pycaret.org.
[2] Karthika, G., (2023) Leveraging PyCaret for Time Series Analysis-A Low Code Approach. Journal of Artificial Intelligence & Cloud Computing, [online] pp.1-4

# HYPERPARAMETER **TUNING**

- Tuning libraries: Scikit-learn & Optuna for Bayesian optimization
- Search algorithms: Randomized Search CV and Grid Search
- Fold number default = 3

PyCaret's default configuration and customized settings make it very flexible, and require less coding knowledge to use.



# Implementation & Analysis

## Key Libraries

- Yfinance for fetching the historical S&P 500 data from Yahoo Finance source
- Pandas for data manipulation,
- NumPy for numerical computations,
- Matplotlib for data visualization,
- Seaborn for advanced statistical plotting.
- PyCaret's time series library, Scikit-learn and sktime for model training and evaluation

## Data Collection

S&P 500 index dataset includes

- 5 columns (dimensions): Open/High/Low/Close/Volume
- 2516 rows (observations)
- 3 data types:
  - Datetime (New York time),
  - Floating values,
  - Integer values.

```
DatetimeIndex(['2015-01-02 00:00:00-05:00', '2015-01-05 00:00:00-05:00',
               '2015-01-06 00:00:00-05:00', '2015-01-07 00:00:00-05:00',
               '2015-01-08 00:00:00-05:00', '2015-01-09 00:00:00-05:00',
               '2015-01-12 00:00:00-05:00', '2015-01-13 00:00:00-05:00',
               '2015-01-14 00:00:00-05:00', '2015-01-15 00:00:00-05:00',
               ...
               '2024-12-17 00:00:00-05:00', '2024-12-18 00:00:00-05:00',
               '2024-12-19 00:00:00-05:00', '2024-12-20 00:00:00-05:00',
               '2024-12-23 00:00:00-05:00', '2024-12-24 00:00:00-05:00',
               '2024-12-26 00:00:00-05:00', '2024-12-27 00:00:00-05:00',
               '2024-12-30 00:00:00-05:00', '2024-12-31 00:00:00-05:00'],
              dtype='datetime64[ns, America/New_York]', name='Date', length=2516, freq=None)
```

## Data Clean

- No duplicate values
- Missing values = 92
  - Handle missing values with Front Fill methods

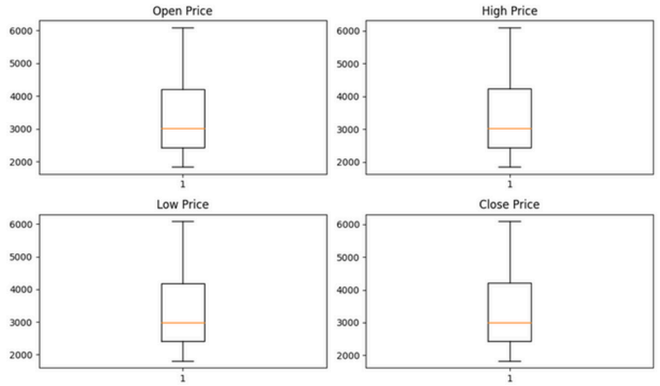
| | |
|---|---|
| No of observations before cleaning | 2516 |
| Missing values | 92 |
| No of observations after cleaning | 2608 |

## Data Transform

- Tranform DatetimeIndex to PeriodIndex
- Select the Close index as primary feature to analysis
  - Other input features are reduced

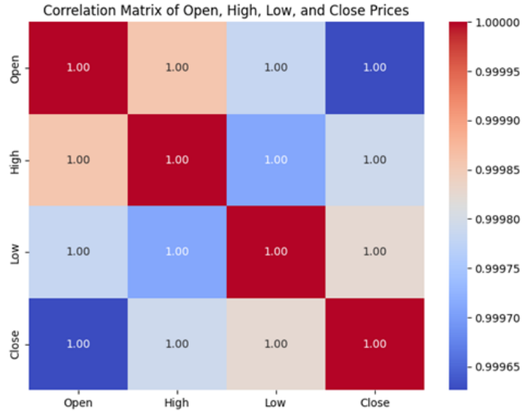
```
PeriodIndex(['2015-01-19', '2015-02-16', '2015-04-03', '2015-05-25',
             '2015-07-03', '2015-09-07', '2015-11-26', '2015-12-25',
             '2016-01-01', '2016-01-18', '2016-02-15', '2016-03-25',
             '2016-05-30', '2016-07-04', '2016-09-05', '2016-11-24',
             '2016-12-26', '2017-01-02', '2017-01-16', '2017-02-20',
             '2017-04-14', '2017-05-29', '2017-07-04', '2017-09-04',
             '2017-11-23', '2017-12-25', '2018-01-01', '2018-01-15',
             '2018-02-19', '2018-03-30', '2018-05-28', '2018-07-04',
             '2018-09-03', '2018-11-22', '2018-12-05', '2018-12-25',
             '2019-01-01', '2019-01-21', '2019-02-18', '2019-04-19',
             '2019-05-27', '2019-07-04', '2019-09-02', '2019-11-28',
             '2019-12-25', '2020-01-01', '2020-01-20', '2020-02-17',
             '2020-04-10', '2020-05-25', '2020-07-03', '2020-09-07',
             '2020-11-26', '2020-12-25', '2021-01-01', '2021-01-18',
             '2021-02-15', '2021-04-02', '2021-05-31', '2021-07-05',
             '2021-09-06', '2021-11-25', '2021-12-24', '2022-01-17',
             '2022-02-21', '2022-04-15', '2022-05-30', '2022-06-20',
             '2022-07-04', '2022-09-05', '2022-11-24', '2022-12-26',
             '2023-01-02', '2023-01-16', '2023-02-20', '2023-04-07',
             '2023-05-29', '2023-06-19', '2023-07-04', '2023-09-04',
             '2023-11-23', '2023-12-25', '2024-01-01', '2024-01-15',
             '2024-02-19', '2024-03-29', '2024-05-27', '2024-06-19',
             '2024-07-04', '2024-09-02', '2024-11-28', '2024-12-25'],
            dtype='period[B]')
```

# Implementation & Analysis




## Outliers

After transforming the data, there is no outliers




## Correlation of Open/High/Low/Close

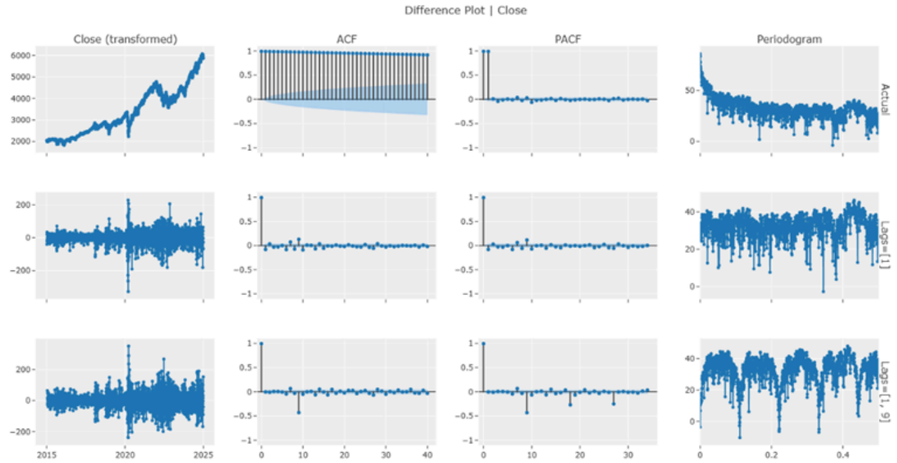
There is a strong correlation amoing these features. For the feature selection, the Close index is chosen as the target variable to forecast

| | |
|---|---|
| Seasonal Period(s) Tested | [9, 7, 28, 56, 14, 2] |
| Significant Seasonal Period(s) | [9, 7, 28, 56, 14, 2] |
| Significant Seasonal Period(s) without Harmonics | [9, 56] |
| Remove Harmonics | False |
| Harmonics Order Method | harmonic_max |
| Num Seasonalities to Use | 1 |
| All Seasonalities to Use | [9] |
| Primary Seasonality | 9 |
| Seasonality Present | True |
| Seasonality Type | mul |
| Target Strictly Positive | True |
| Target White Noise | No |

## Seasonality

the original data after transformed demonstrates the presence of seasonality (9)


Comparison of original and transformed dataset

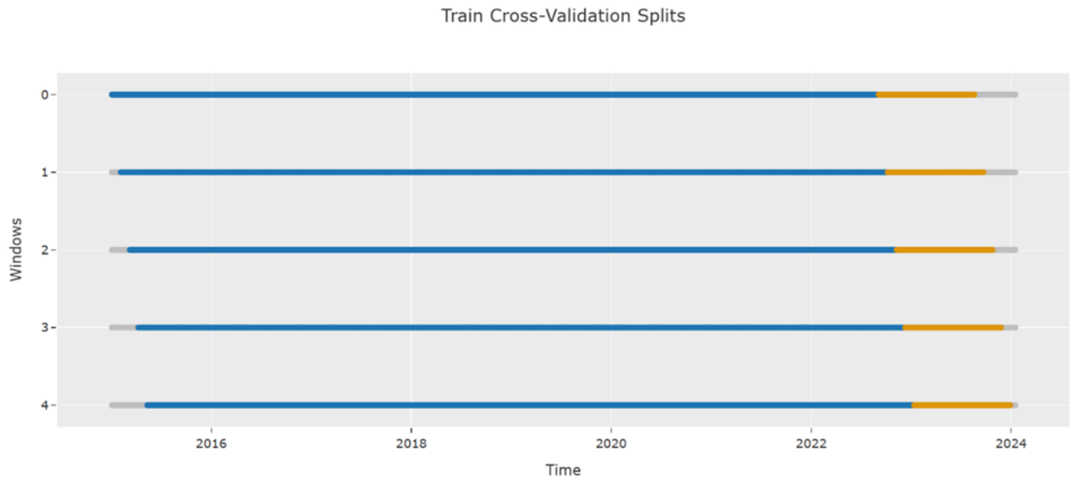



## Differencing

Time series models can remove trends and seasonality (with a lag of 9) to render the data stationary prior to modeling the remaining autoregressive properties.

# Implementation & Analysis


Train-Test set split




Walk-forward cross-validation

Train set = from January 1, 2015, to December 31, 2023, ~ 90% of the total dataset with 2,087 observations.

- 80% is allocated as the new train subset
- 10% is designated as the validation subset.

Test set = from January 1, 2024, to December 31, 2024, ~ 10% of the dataset with 251 observations.

Window_length = the number of consecutive past time steps => lags are t-1, t-2, ...t-2000.
Forecast_horizon = an array of 251 days (a year)
Step_length = 23 trading days move forward
Fold stratergy = slidingwindowsplitter

# Results & Discussion

| | Model | MASE | RMSSE | MAE | RMSE | MAPE | SMAPE | R2 | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| lightgbm_cds_dt | Light Gradient Boosting w/ Cond. Deseasonalize & Detrending | 2.5771 | 1.9882 | 170.5037 | 197.2796 | 0.0411 | 0.0412 | 0.1575 | 109.7220 |
| catboost_cds_dt | CatBoost Regressor w/ Cond. Deseasonalize & Detrending | 2.6481 | 2.1119 | 175.3713 | 209.7909 | 0.0414 | 0.0422 | 0.0905 | 2.6660 |
| xgboost_cds_dt | Extreme Gradient Boosting w/ Cond. Deseasonalize & Detrending | 2.8709 | 2.2548 | 190.0390 | 223.9509 | 0.0449 | 0.0459 | -0.0242 | 1.8600 |
| rf_cds_dt | Random Forest w/ Cond. Deseasonalize & Detrending | 3.2494 | 2.5102 | 215.9076 | 249.8545 | 0.0508 | 0.0522 | -0.3012 | 3.4900 |
| ada_cds_dt | AdaBoost w/ Cond. Deseasonalize & Detrending | 3.2831 | 2.5672 | 218.4317 | 255.7089 | 0.0510 | 0.0528 | -0.3779 | 1.9280 |
| naive | Naive Forecaster | 4.9859 | 3.9325 | 331.7977 | 391.7467 | 0.0770 | 0.0812 | -2.0995 | 0.5360 |

| Model | RMSE | MAE |
|---|---|---|
| Tuned CatBoost | 181.5163 | 152.8886 |
| Tuned LightGBM | 189.1265 | 161.3337 |
| Original LightGBM | 197.2796 | 170.5037 |
| Original CatBoost | 209.7909 | 175.3713 |

=> 1st is LightGBM model
2nd is CatBoost model

Parameters with default tuning

**CatBoost:**

degree=5,
fe_target_rr=[WindowSummarizer(lag_feature={'lag': [9, 8, 7, 6, 5, 4, 3, 2, 1]}, n_jobs=1)],
regressor=<catboost.core.CatBoostRegressor object at 0x7d50a0958ed0>, sp=9, window_length=16

| Model | RMSE | MAE |
|---|---|---|
| CatBoost | 181.5163 | 152.8886 |
| Ensemble Forecaster | 181.9637 | 154.2943 |
| LightGBM | 189.1265 | 161.3337 |
| Naive Forecaster | 279.4357 | 234.0787 |

**LightGBM:**

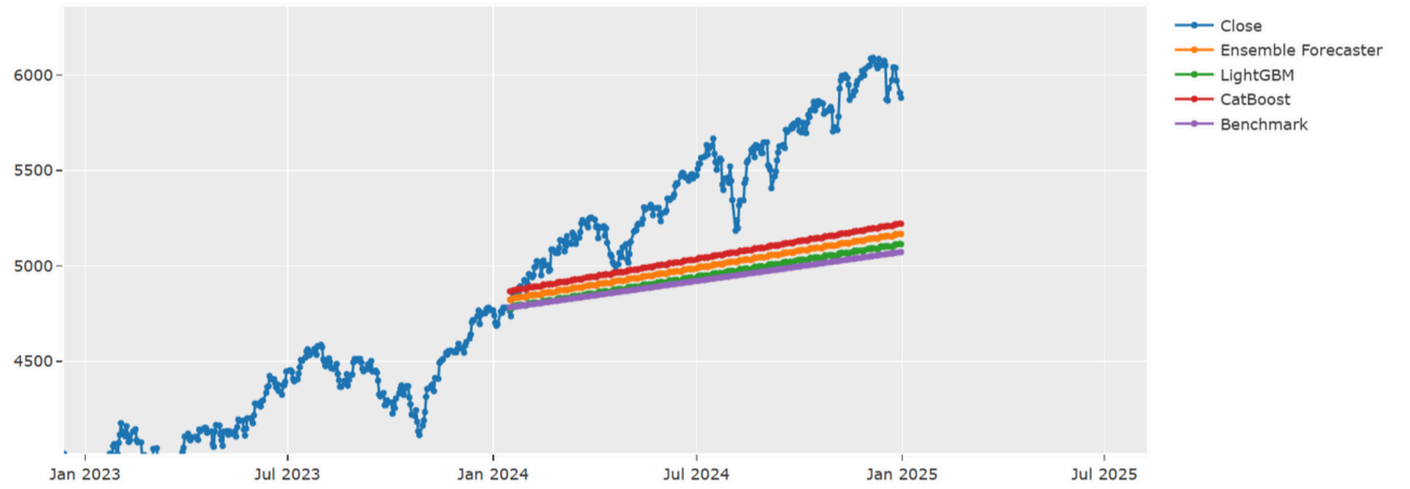
degree=6,
deseasonal_model='multiplicative',
fe_target_rr=[WindowSummarizer(lag_feature={'lag': [9, 8, 7, 6, 5, 4, 3, 2, 1]}, n_jobs=1)],
regressor=LGBMRegressor(bagging_freq=4, colsample_bytree=1, learning_rate=0.1140287744246445,
max_depth=3, min_child_samples=14, n_estimators=49, n_jobs=-1,
num_leaves=189, random_state=499, reg_alpha=1.9861002257175493e-06,
reg_lambda=0.10653911378587111, subsample=0.5)

After hyperparameter tuning,

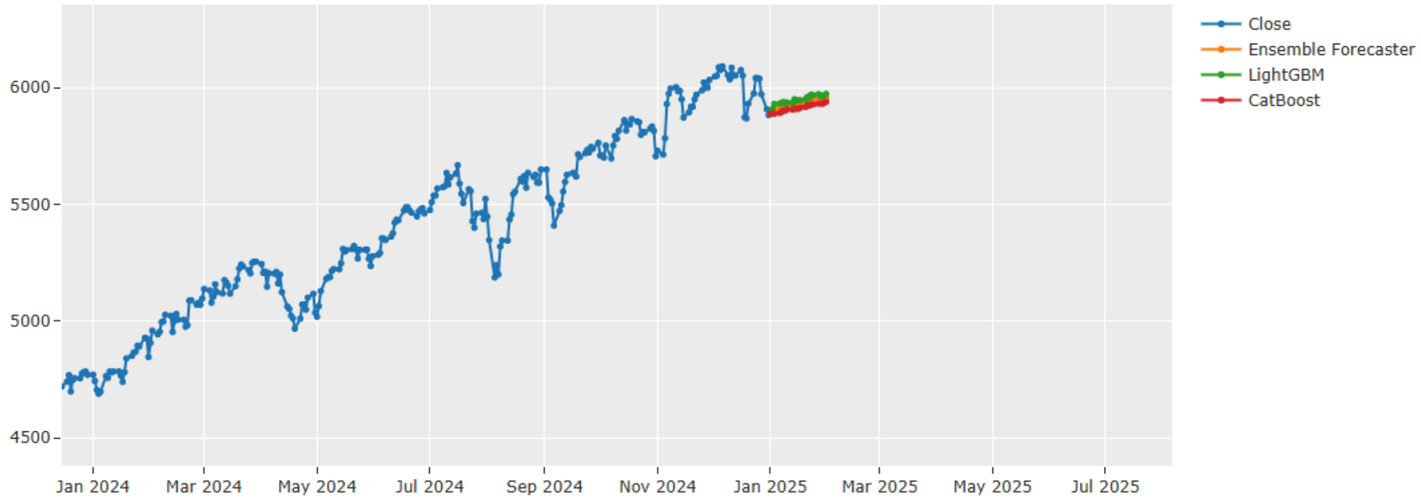
- **CatBoost** model became the 1st with the lowest RMSE and MAE scores, outperforming the hybrid model, the LightGBM model, and the benchmark model

# Results & Discussion

## Prediction on test set




## Prediction on unseen data




## 1st CatBoost

| | y_pred |
|---|---|
| 2025-01-01 | 5887.4177 |
| 2025-01-02 | 5887.2251 |
| 2025-01-03 | 5887.0376 |
| 2025-01-06 | 5891.0721 |
| 2025-01-07 | 5894.8372 |
| 2025-01-08 | 5901.6729 |
| 2025-01-09 | 5900.8861 |
| 2025-01-10 | 5905.7275 |
| 2025-01-13 | 5905.2330 |
| 2025-01-14 | 5908.9466 |
| 2025-01-15 | 5908.7630 |
| 2025-01-16 | 5908.5846 |
| 2025-01-17 | 5912.6282 |
| 2025-01-20 | 5916.4024 |
| 2025-01-21 | 5923.2472 |
| 2025-01-22 | 5922.4695 |
| 2025-01-23 | 5927.3200 |
| 2025-01-24 | 5926.8347 |
| 2025-01-27 | 5930.5574 |
| 2025-01-28 | 5930.3829 |
| 2025-01-29 | 5930.2136 |
| 2025-01-30 | 5934.2664 |
| 2025-01-31 | 5938.0497 |

## 2nd Hybrid model

| | y_pred |
|---|---|
| 2025-01-01 | 5890.4673 |
| 2025-01-02 | 5897.1479 |
| 2025-01-03 | 5908.2537 |
| 2025-01-06 | 5910.8217 |
| 2025-01-07 | 5914.3791 |
| 2025-01-08 | 5919.6184 |
| 2025-01-09 | 5917.6401 |
| 2025-01-10 | 5920.6152 |
| 2025-01-13 | 5919.3967 |
| 2025-01-14 | 5928.9807 |
| 2025-01-15 | 5927.6986 |
| 2025-01-16 | 5926.7592 |
| 2025-01-17 | 5928.7753 |
| 2025-01-20 | 5933.2397 |
| 2025-01-21 | 5941.2514 |
| 2025-01-22 | 5939.4190 |
| 2025-01-23 | 5948.1196 |
| 2025-01-24 | 5946.8983 |
| 2025-01-27 | 5950.7916 |
| 2025-01-28 | 5946.7602 |
| 2025-01-29 | 5945.8246 |
| 2025-01-30 | 5950.6040 |
| 2025-01-31 | 5955.0829 |

## 3rd LightGBM

| | y_pred |
|---|---|
| 2025-01-01 | 5893.5169 |
| 2025-01-02 | 5907.0707 |
| 2025-01-03 | 5929.4699 |
| 2025-01-06 | 5930.5712 |
| 2025-01-07 | 5933.9210 |
| 2025-01-08 | 5937.5639 |
| 2025-01-09 | 5934.3940 |
| 2025-01-10 | 5935.5030 |
| 2025-01-13 | 5933.5605 |
| 2025-01-14 | 5949.0148 |
| 2025-01-15 | 5946.6342 |
| 2025-01-16 | 5944.9338 |
| 2025-01-17 | 5944.9225 |
| 2025-01-20 | 5950.0770 |
| 2025-01-21 | 5959.2557 |
| 2025-01-22 | 5956.3684 |
| 2025-01-23 | 5968.9192 |
| 2025-01-24 | 5966.9619 |
| 2025-01-27 | 5971.0257 |
| 2025-01-28 | 5963.1375 |
| 2025-01-29 | 5961.4356 |
| 2025-01-30 | 5966.9416 |
| 2025-01-31 | 5972.1162 |

# Conclusion & Future Works

## Contribution of the study

- Streamlined model implementation
- Performance evaluation and model selection
- Improvement through hyperparameter optimization
- Accessibility in financial ML: Leveraging PyCaret's AutoML capablities

## Limitation & Challenges

- Univariate time series without exogenous variables
- Currency and global dynamics
- Temporal granularity
- Tool

## Future works

- Incorporate multivariate variables to achieve more accurate and comprehensive forecasting results.
- Optimize cross-validation strategies and hyperparameter tuning by experimenting with diverse custom configurations
- Temporal granularity: hours or minutes
- Explore other AutoML tools that can offer valuable insights into their efficiency and adaptability when applied to ML and DL techniques