Introduction to Basic Forecasting Methods

Moving Average, Exponential Smoothing, ARIMA, and Rolling Forecasts

Dr. Stavros K. Stavroglou

December 23, 2024

Outline

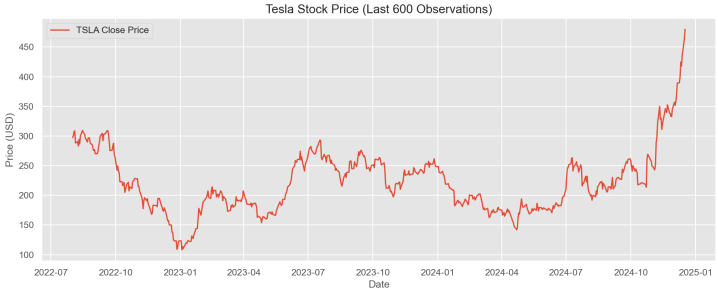
Section 1 Goals

After this section, you should be able to:

- Load a real-world stock price dataset (Tesla)
- Visualize and interpret basic characteristics of the time series
- Explain and implement a **simple Moving Average (MA)** forecast
- Distinguish between **in-sample** and **out-of-sample** forecasting
- Understand how to compute and visualize **confidence intervals**

Tesla Stock Price Dataset

- Data source: `TSLA.csv`, containing date and close price
- We only keep the **last 600 observations** for forecasting demonstration
- **Business frequency** used; missing days (weekends/holidays) are forward-filled



(Tesla close price.)

Moving Average (MA) Forecasting

Idea: Predict future values by averaging a fixed window of recent observations:

$$\hat{y}_{t+1} = \frac{1}{w} \sum_{k=0}^{w-1} y_{t-k},$$

where w is the window size (e.g., 30 days).

Pros:

- Simple, quick to implement, good *benchmark*
- Works best when data has no strong trend or seasonality

Cons:

- Equal weighting of all past points in the window
- *Ignores* more recent observations that might be more informative
- Poor for trending or seasonal data

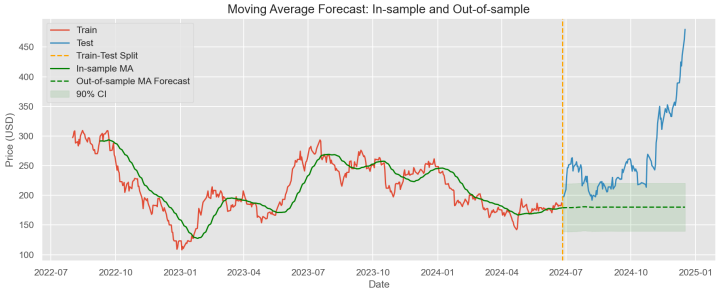
Minimal Code Snippet: Moving Average

Python Example

```
# Generate out-of-sample forecasts iteratively
historical_values = list(train_data.values)
ma_forecasts = []
for _ in test_data:
  if len(historical_values) < window_size:
    forecast_val = np.mean(historical_values)
  else:
    forecast_val = np.mean(historical_values[-window_size:])
  ma_forecasts.append(forecast_val)
  historical_values.append(forecast_val)window_size = 30
ma_train = train_data.rolling(window=window_size).mean()
# Generate out-of-sample forecasts iteratively
historical_values = list(train_data.values)
ma_forecasts = []
for _ in test_data:
  if len(historical_values) < window_size:
    forecast_val = np.mean(historical_values)
  else:
    forecast_val = np.mean(historical_values[-window_size:])
  ma_forecasts.append(forecast_val)
```

Visualizing MA Forecast & Confidence Intervals

- We compare:
  - **Train** set
  - **Test** set
  - **MA in-sample** prediction
  - **MA out-of-sample** forecast
- **Bootstrap** approach for $\sim 90\%$ confidence intervals



(MA forecast with CI.)

Key Idea: Exponential Smoothing

Unlike MA, Exponential Smoothing weights recent observations more heavily:

$$\hat{y}_{t+1} = \alpha\, y_t + (1 - \alpha)\, \hat{y}_t,$$

where $0 < \alpha < 1$ is the smoothing parameter.

Variations:

1. **SES (Simple Exp. Smoothing):** *No explicit trend or seasonality*
2. **Holt's Method:** Adds a **trend** component
3. **Holt-Winters:** Adds **trend** + **seasonal** components

Pros and Cons of Exponential Smoothing

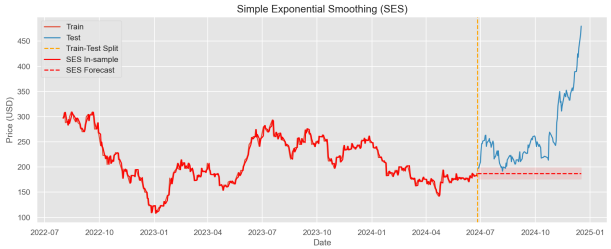
Pros:

- Fast, relatively simple to tune (built-in parameter optimization)
- Good for data with **trend** or **seasonality**

Cons:

- Purely *univariate*, can miss external factors
- May **overfit** if not carefully validated
- Seasonal period must be known or estimated

Sample Forecast: Simple Exponential Smoothing (SES)

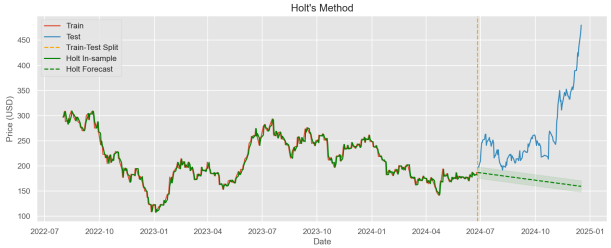


(SES in-sample fit out-of-sample forecast.)

Observation:

- SES might underfit if trend/seasonality is strong
- Good for stable, level series

Sample Forecast: Holt's Method

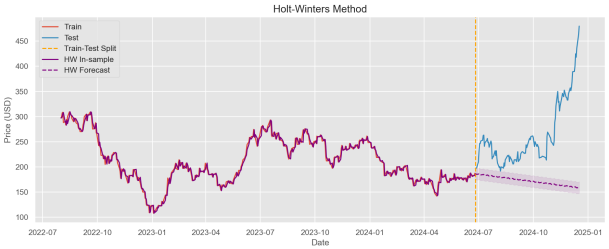


(Holt in-sample fit out-of-sample forecast, accounting for trend.)

Observation:

- Holt adds a trend component
- More flexible than SES alone

Sample Forecast: Holt-Winters Method



(Holt-Winters in-sample fit out-of-sample forecast, accounting for seasonality.)

Observation:

- HW handles recurring seasonal patterns
- Often best for strong seasonality

Why (Auto) ARIMA?

ARIMA stands for:

- **AR (Autoregressive):** Model uses past values
- **I (Integrated):** Differencing to achieve stationarity
- **MA (Moving Average):** Model uses past forecast errors

Auto-ARIMA automatically searches for (p, d, q) to minimize an information criterion (e.g., AIC).

Pros & Cons of Auto-ARIMA

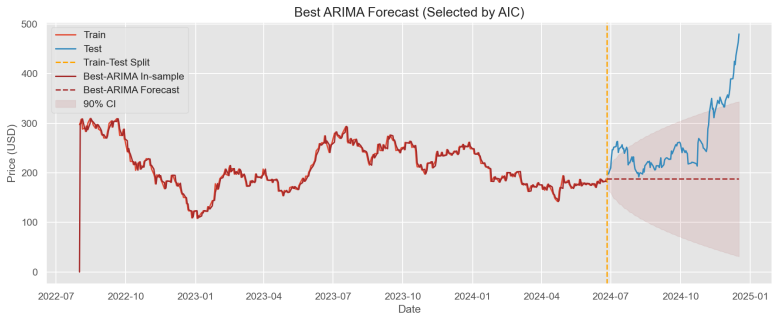
Pros

- Automates a tedious parameter search
- Can **adapt** to diverse patterns (once stationarity is established)
- Often yields better results than naive or simple smoothing methods

Cons

- Can be **computationally expensive** with large data or wide parameter ranges
- Assumes a **linear** ARMA-type structure
- Still **univariate**—no external regressors unless extended

Illustration: Best ARIMA Forecast



(Placeholder: ARIMA in-sample fit and out-of-sample forecast with CIs.) **Key Takeaway:** Auto-ARIMA often outperforms guesswork, but still needs validation.

What is Rolling Forecasting?

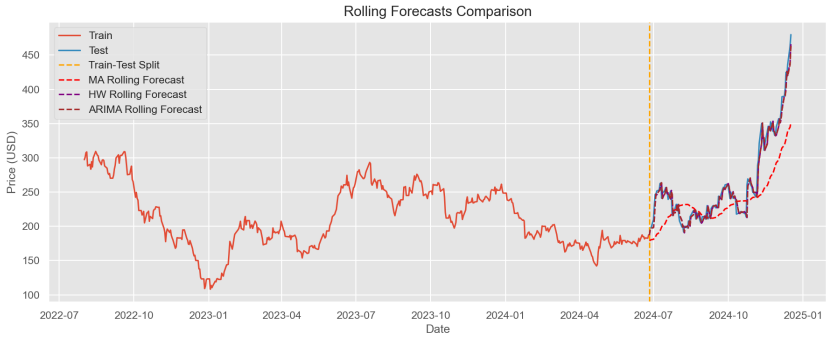
Process:

1. Start with an initial **training window**
2. Fit the model (MA, Holt-Winters, ARIMA, etc.)
3. Forecast **one step** ahead
4. Add the **new actual data** point to your training set
5. **Refit** and repeat until the end of the test set

Why?

- More **realistic** approach for real-time forecasting
- Model is **continuously updated** with new observations

Example: Rolling Forecast Comparison



(Placeholder: Rolling forecasts of MA, HW, ARIMA over the test horizon.) **Observation:**

- **MA** might lag behind quick changes
- **HW** captures short seasonal bursts
- **ARIMA** adapts but needs repeated re-fitting overhead

Common Forecast Accuracy Metrics

Max Error (ME):

$$\max |y_t - \hat{y}_t|.$$

MAE (Mean Absolute Error):

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t|.$$

MSE (Mean Squared Error):

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2.$$

RMSE (Root MSE):

$$\text{RMSE} = \sqrt{\text{MSE}}.$$

MAPE (Mean Absolute Percentage Error):

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right|.$$

MASE (Mean Absolute Scaled Error):

$$\text{MASE} = \frac{\text{MAE of forecasts}}{\text{MAE of naive reference}}.$$

Comparison Tables: Simple vs. Rolling

Example Table: Simple Out-of-Sample Forecasts

Model	ME	MAE	MSE	RMSE	MAPE	MASE
MA	32.5	20.4	723	26.9	9.8	1.15
SES	35.2	19.1	712	26.7	10.3	1.07
Holt	29.8	17.2	625	25.0	9.4	0.96
Holt-Winters	28.3	16.7	590	24.3	9.1	0.93
Auto-ARIMA	27.5	16.3	570	23.9	8.8	0.90

Rolling Forecasts may have a separate table or row of results.
Evaluate both **simple** out-of-sample and **rolling** forecasts to see which approach suits practical needs.

Interpreting These Metrics

- **ME (Max Error):** Worst-case difference between predicted and actual values.
- **MAE (Mean Absolute Error):** Average of absolute errors; measures on-average deviation.
- **MSE (Mean Squared Error):** Penalizes large errors more heavily by squaring them.
- **RMSE (Root MSE):** Square root of MSE, returning to original units.
- **MAPE (Mean Absolute Percentage Error):** Average percentage error; lower means closer forecasts to actual.
- **MASE (Mean Absolute Scaled Error):** Scales errors relative to a naive (benchmark) forecast.

General Rule: *Lower metric values generally imply better performance.*

Lecture 2 Summary

- **Section 1:** Explored a real stock dataset and **Moving Average** forecast
- **Section 2:** Introduced **Exponential Smoothing** variants (SES, Holt, Holt-Winters)
- **Section 3:** Used **Auto-ARIMA** for parameter selection
- **Section 4:** Implemented **Rolling Window Forecasting** for realistic updates
- **Section 5:** Compared model performances using multiple **accuracy metrics**

Key Takeaways

- Start with **simple** methods (e.g., MA) as a baseline
- Use **Exponential Smoothing** when trend/seasonality is evident
- **ARIMA** (and Auto-ARIMA) often outperforms simpler methods but is more complex
- **Rolling forecasts** are more realistic but can be expensive to update
- Always validate with **multiple metrics**, not just one

Next Steps

- **Coding Session:**
  - Run and dissect the accompanying Python script
  - Experiment with **hyperparameters** (e.g., window size, smoothing level)
  - Practice rolling windows in real-time
- **Future Lectures:**
  - Advanced: **SARIMA** (seasonal ARIMA), exogenous variables
  - More robust evaluation: cross-validation, multi-step ahead forecasting
  - Neural network approaches for time series

Thank you!