

Time Series Analysis II: Models & Forecasting

Prepared by Dr. Sumaya Alkubaisi

FNT633
Machine Learning in Finance

Lecture 10



Learning Goals

By the end of this lecture, you will be able to:

- Understand seasonal and multivariate time-series models
- Learn SARIMA, Holt–Winters, Prophet, and VAR models
- Perform seasonal decomposition
- Improve model selection with ACF/PACF
- Compare forecasting models
- Build full advanced forecasts in Python
- Review 2024–2025 global ML advances and Trends

Recap of Lecture 9

- Time Series Basics: You learned what time-series data is and why it's used in finance.
- Trend: The long-term direction of the series over time.
- Seasonality: Patterns that repeat consistently (daily, weekly, monthly, yearly).
- Noise: Random unpredictable changes in the data.
- AR Model: Forecasts the next value using previous values.
- MA Model: Forecasts using previous error terms.
- ARMA Model: Combines AR and MA for stationary series.
- ARIMA Model: Adds differencing to handle non-stationary data.
- Python Forecasting Demo: You used the synthetic stock dataset to fit and generate predictions using ARIMA.

The Need for Seasonal Models

Finance datasets often show periodic patterns:

- Quarterly profits
- Monthly inflation
- Seasonal portfolio flows
- Monthly commodity cycles
- ARIMA alone cannot capture seasonality, hence, we need SARIMA (and other ADVANCED MODELS).

SARIMA = Seasonal ARIMA

$(p, d, q) \rightarrow$ non-seasonal

$(P, D, Q)_s \rightarrow$ seasonal part

s = season length (e.g., 12 months)

Handles both: Trend, Seasonality, and naturally includes noise/error term (always part of the model)

Chapter
10

Advanced Models - Holt–Winters

Holt–Winters is an exponential smoothing forecasting method that breaks a time series into three parts:

- Level: the baseline value of the series (It's what the series would be without trend or seasonal effects.)
- Trend: upward or downward movement
- Seasonality: repeating patterns (daily, monthly, yearly...)

It's widely used in business and finance because it produces smoother and more stable forecasts than ARIMA when data is noisy but has clear seasonal behavior.

Chapter
10

Advanced Models - Holt–Winters

Holt–Winters Types

1. Additive Seasonality

- Seasonal pattern is constant over time (+/- constant).
- it doesn't change as the overall level of the series changes.
- We add the same amount every season.

Example: If our baseline monthly sales are 1,000 units, and December always adds +200 units, then:

November: 1,000 units

December: 1,200 units

- Works best when seasonal fluctuations are roughly the same size each period (Ice cream shop during holidays: During a specific festival, they always sell +100 extra ice creams regardless of whether overall sales trend up or down.)

Chapter
10

Advanced Models - Holt–Winters

Holt–Winters Types

2. Multiplicative Seasonality

- The seasonal effect scales with the series' level ($\times$ factor).
- it grows or shrinks as the baseline changes.
- We multiply the baseline by a seasonal factor.

Example: If December sales are usually 20% higher:

Baseline = 1,000 $\rightarrow$ December = $1,000 \times 1.2 = 1,200$

Baseline grows to 1,500 $\rightarrow$ December = $1,500 \times 1.2 = 1,800$

- Works best when seasonal fluctuations increase/decrease with the trend (E-commerce sales growth: During Black Friday, sales increase by 20%, so if baseline sales double over the years, the seasonal spike doubles too.).

Chapter
10

Advanced Models - Holt–Winters

When to use Holt–Winters:

- The data shows smooth, repeating seasonal patterns
- You want a simple, explainable forecasting model
- ARIMA/SARIMA becomes unstable or too complex due to differencing
- You need fast forecasts with minimal parameter tuning

Advanced Models - Prophet Model (Meta/Facebook)

Prophet is a modern forecasting model designed for business and financial time series. It automatically breaks the data into:

- Trend: long-term increase/decrease
- Multiple seasonalities: daily, weekly, monthly, yearly patterns
- Holiday effects: special events (Eid, Back-to-school season, Black Friday...)
- Missing data handling: it fills gaps smoothly
- Automatic fitting: no complex tuning like ARIMA

It's widely used for:

- Stock index forecasting (daily/weekly trends)
- Volatility cycles
- Seasonal patterns such as monthly sales or daily trading volume

Advanced Models - Prophet Model (Meta/Facebook)

Strengths of Prophet:

- Robust to outliers: sudden spikes/drops don't break the model
- Adapts to long-term trend changes: works well with evolving markets
- Easy to tune: fewer parameters than ARIMA/SARIMA
- Excellent for seasonal financial cycles: monthly/weekly business patterns

Prophet Limitations

- Not suitable for high-frequency (millisecond) trading
Prophet is designed for daily or monthly data, not second-by-second movements.
- Not good for multivariate forecasting
 - Prophet forecasts one variable at a time; can only model one main time series at a time, for example, just stock price, or sales, or traffic.
 - it cannot directly include multiple predictors like ARIMAX or LSTM: It cannot naturally include many external variables (features).
- Can oversmooth sudden market jumps
Large shocks (war, announcements, crashes) may be smoothed too much, reducing accuracy.

Chapter
10

Advanced Models - Vector AutoRegression (VAR)

- VAR is a forecasting model for multiple time series at the same time. Instead of predicting one variable (like Prophet, AR or ARIMA...), VAR predicts several connected variables together.

How Var works?

For variables like: returns(t), volatility(t), trading volume(t)

VAR says that each variable depends on:

1. Its own past values: returns today depend on past returns
2. The past values of the other variables: returns depend on past volatility and past trading volume
 - volatility depends on past returns and volume
 - volume depends on past returns and volatility

VAR captures the interaction between variables.

Chapter
10

Advanced Models - VAR in Finance

1. Predicting portfolio risk

You can forecast risk using variables such as: volatility, returns, volume
Because these influence each other, VAR models them together.

2. Macro-economic forecasting

Central banks and economists use VAR to model variables such as: inflation(t), GDP growth(t), unemployment(t)

These variables are interdependent, and VAR captures the relationships.

3. Multi-asset investment modelling

For modelling the behaviour of several assets such as: stocks, bonds, commodities, currencies
VAR helps forecast how one asset's movement affects the others.

Seasonal ACF & PACF

Purpose: Identify seasonality and seasonal AR terms for models like SARIMA.

ACF (Autocorrelation Function)

- Measures how the series correlates with its past values.
- Seasonal pattern: Spikes appear at multiples of the season length (s).
- Example: Monthly data with yearly seasonality ($s = 12$) → spikes at lag 12, 24, 36...
- Helps identify season length and MA terms.

PACF (Partial Autocorrelation Function)

- Measures correlation with a specific past lag after removing effects of closer lags.
- Seasonal AR pattern: Spike at seasonal lag (e.g., lag 12) → indicates seasonal AR order P .
- Shows the direct influence of a past seasonal point.

Rolling Forecast Evaluation

Purpose: Measure how well a model predicts as time moves forward.

Steps:

- Train the model on past data.
- Predict the next value.
- Slide the training window forward to include the new actual value.
- Repeat this process for the entire series.

Why use it:

- Simulates real-world forecasting in finance or business.
- Captures changing patterns over time.
- Provides a realistic evaluation of model accuracy.

Evaluation Metrics: MAE, RMSE, MAPE

Chapter
10**MODEL SELECTION & ACCURACY****Error Metrics Review****MAE — Mean Absolute Error**

Average size of the errors.

Easy to interpret.

RMSE — Root Mean Squared Error

Penalizes big mistakes more than MAE.

MAPE — Mean Absolute Percentage Error

Shows error as a percentage, easier to compare across scales.

AIC / BIC

- Used for model selection
- Lower AIC/BIC = better balance of fit vs complexity
- Common for ARIMA, SARIMA, VAR models

❖ For MAE, RMSE, MAPE, AIC, and BIC — lower values always mean a better model.

MODEL SELECTION & ACCURACY - Error Comparison Table

Example results comparing models:

Model	MAE	RMSE	MAPE
SARIMA	2.1	3.0	4.5%
Holt–Winters	2.4	3.2	4.9%
Prophet	2.0	2.8	4.0%

- Prophet performs best overall in this example (lowest errors).
- SARIMA is close, good for seasonal structured data.
- Holt–Winters performs well but slightly weaker.

Case Study: a synthetic financial index

This lab teaches you how to build and compare forecasting models on a financial time-series.

What is the goal?

The goal is to learn which model forecasts better and understand how each model behaves with trend + seasonality.

More specifically, we will:

- Train SARIMA, Holt–Winters, and Prophet
- Generate future forecasts
- Compare them using error metrics (MAE, RMSE, MAPE)
- See which model performs best on financial-style data

Case Study: a synthetic financial index

Dataset Description:

The dataset is a synthetic monthly financial index created specifically for the lab. It behaves like a real financial time-series and includes:

Trend: a gradual upward or downward movement over time

Seasonality: repeating yearly patterns (e.g., certain months consistently higher or lower)

Noise: random fluctuations similar to market behavior

This synthetic index is designed to mimic real financial data while keeping it clean and easy to use for learning and model comparison.

Python Lab: ADVANCED TIME SERIES MODELS

#Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from prophet import Prophet
```

- **Matplotlib (plt):** Visualize the index and forecasts
- **seasonal_decompose:** Break series into trend, seasonality, and noise
- **SARIMAX:** Build and forecast using SARIMA models
- **ExponentialSmoothing:** Build and forecast using Holt–Winters models
- **Prophet:** Build and forecast using Prophet (trend + seasonality + holidays)

Python Lab: ADVANCED TIME SERIES MODELS

#Create Monthly Dataset

```
np.random.seed(42)
```

```
months = pd.date_range(start="2023-01-01", periods=36, freq=" ME")
```

```
trend = np.linspace(100, 140, 36)
```

```
season = 10 * np.sin(np.arange(36) * (2 * np.pi / 12))
```

```
noise = np.random.normal(0, 3, 36)
```

```
df = pd.DataFrame({  
    "ds": months,  
    "y": trend + season + noise  
})
```

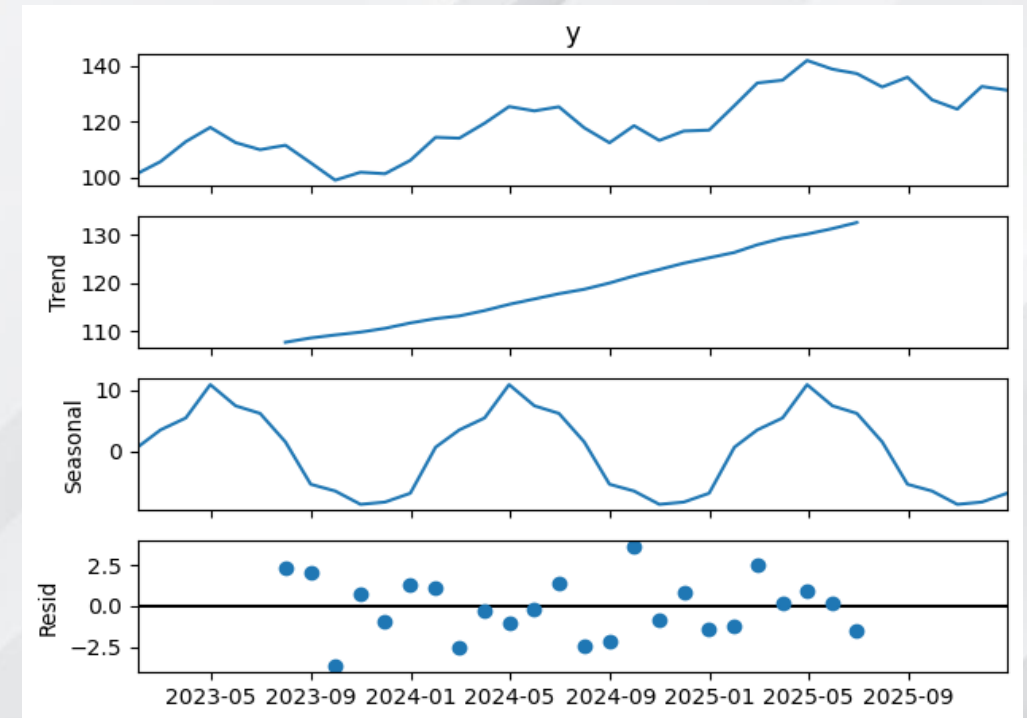
- "ds": dates (time column, used by Prophet)
- "y": synthetic financial index (trend + seasonality + noise)

Python Lab: ADVANCED TIME SERIES MODELS

Decompose Time Series

```
decomp = seasonal_decompose(df.set_index("ds")["y"], model="additive")
decomp.plot()
plt.show()
```

- **df.set_index("ds")["y"]**: selects the index values (y) and sets the date column (ds) as the index.
- **model="additive"**: assumes the series is trend + seasonality + noise (additive model).
- Returns components:
 - Trend: long-term direction
 - Seasonal: repeating pattern
 - Residual: random noise



Python Lab: ADVANCED TIME SERIES MODELS

Fit SARIMA

```
sarima = SARIMAX(df["y"], order=(1,1,1), seasonal_order=(1,1,1,12))  
sarima_fit = sarima.fit()  
df["sarima_pred"] = sarima_fit.predict()
```

- Creates a SARIMA model using the financial index (y) as the data.
- `order=(1,1,1)`: non-seasonal ARIMA parameters (p,d,q)
- `seasonal_order=(1,1,1,12)`: seasonal SARIMA parameters (P,D,Q,s) with 12-month seasonality

Python Lab: ADVANCED TIME SERIES MODELS

Fit Holt–Winters

```
hw = ExponentialSmoothing(df["y"], trend="add", seasonal="add", seasonal_periods=12)
hw_fit = hw.fit()
df["hw_pred"] = hw_fit.fittedvalues
```

- **trend="add"**: model has an additive trend (steady increase/decrease).
- **seasonal="add"**: model has additive seasonality (seasonal effect is constant size).

Python Lab: ADVANCED TIME SERIES MODELS

Fit Prophet

```
model = Prophet(yearly_seasonality=True)
model.fit(df)
future = model.make_future_dataframe(periods=6, freq="ME")
forecast = model.predict(future)
```

❖ **model.make_future_dataframe(periods=6, freq="M"):**

- Creates a future timeline of 6 additional months.
- Prophet will generate predictions for these months.
- freq="ME": monthly frequency.

❖ **forecast = model.predict(future):** Prophet predicts:

- The historical period (36 months)
- Plus 6 future months

- ❖ Prophet by default generates a future dataframe (e.g., 180 days ahead), while SARIMA and Holt-Winters only forecast if you explicitly give them steps to predict.

Python Lab: ADVANCED TIME SERIES MODELS

Plot Predictions

```
plt.figure(figsize=(12,5))  
# Plot actual data  
plt.plot(df['y'], label="Actual Data")  
# Plot SARIMA forecast  
plt.plot(df['sarima_pred'], label="SARIMA Forecast")  
# Plot Holt-Winters forecast  
plt.plot(df['hw_pred'], label="Holt-Winters Forecast")  
# Plot Prophet forecast  
plt.plot(forecast['yhat'], label="Prophet Forecast") # forecast comes from  
Prophet  
  
plt.title("Model Forecast Comparison")  
plt.xlabel("Date")  
plt.ylabel("Index Value")  
plt.legend()  
plt.show()
```

Python Lab: ADVANCED TIME SERIES MODELS

Plot Predictions - Output

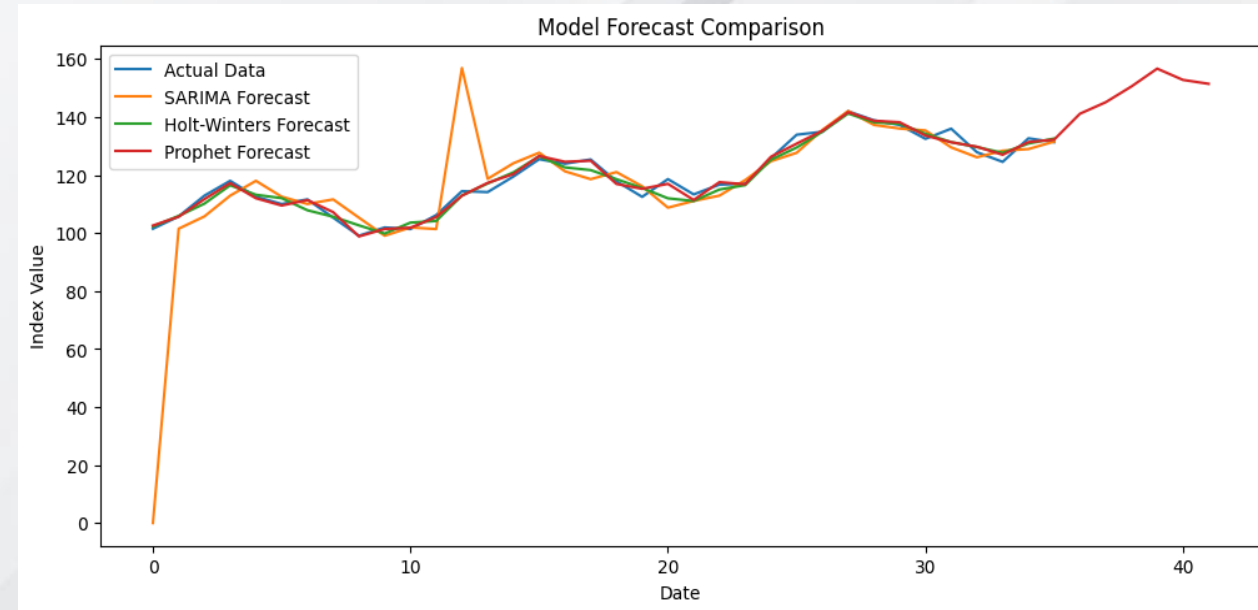
Visual: Line chart comparing actual data and model predictions

Historical Fit

- All models track actual data closely in historical period (Date 0–30)
- SARIMA spike at Date ≈ 12 : likely an outlier
- Holt–Winters & SARIMA: follow historical trend fairly well
- Prophet: closely follows historical trend, slightly overestimates at the end

Future Forecast

- Only Prophet predicts beyond historical data
- Prophet shows upward bias in long-term forecast (Date > 36)
- Holt–Winters & SARIMA → no future predictions in this lab setup



Python Lab: ADVANCED TIME SERIES MODELS

#Compare Errors

```
from sklearn.metrics import mean_squared_error
```

Compute RMSE for in-sample predictions

```
rmse_sarima = np.sqrt(mean_squared_error(df["y"], df["sarima_pred"]))
```

```
rmse_hw = np.sqrt(mean_squared_error(df["y"], df["hw_pred"]))
```

```
rmse_prophet = np.sqrt(mean_squared_error(df["y"], forecast["yhat"][:len(df)])) # Only align historical points
```

```
print("RMSE Scores:")
```

```
print("SARIMA RMSE:", round(rmse_sarima, 3))
```

```
print("Holt-Winters RMSE:", round(rmse_hw, 3))
```

```
print("Prophet RMSE:", round(rmse_prophet, 3))
```

Compare visually

```
models = ["SARIMA", "Holt-Winters", "Prophet"]
```

```
rmse_values = [rmse_sarima, rmse_hw, rmse_prophet]
```

```
plt.figure(figsize=(8,5))
```

```
plt.bar(models, rmse_values, color=["orange", "green", "red"])
```

```
plt.title("RMSE Comparison Across Models")
```

```
plt.ylabel("RMSE")
```

```
plt.show()
```

Python Lab: ADVANCED TIME SERIES MODELS

#Compare Errors - Output

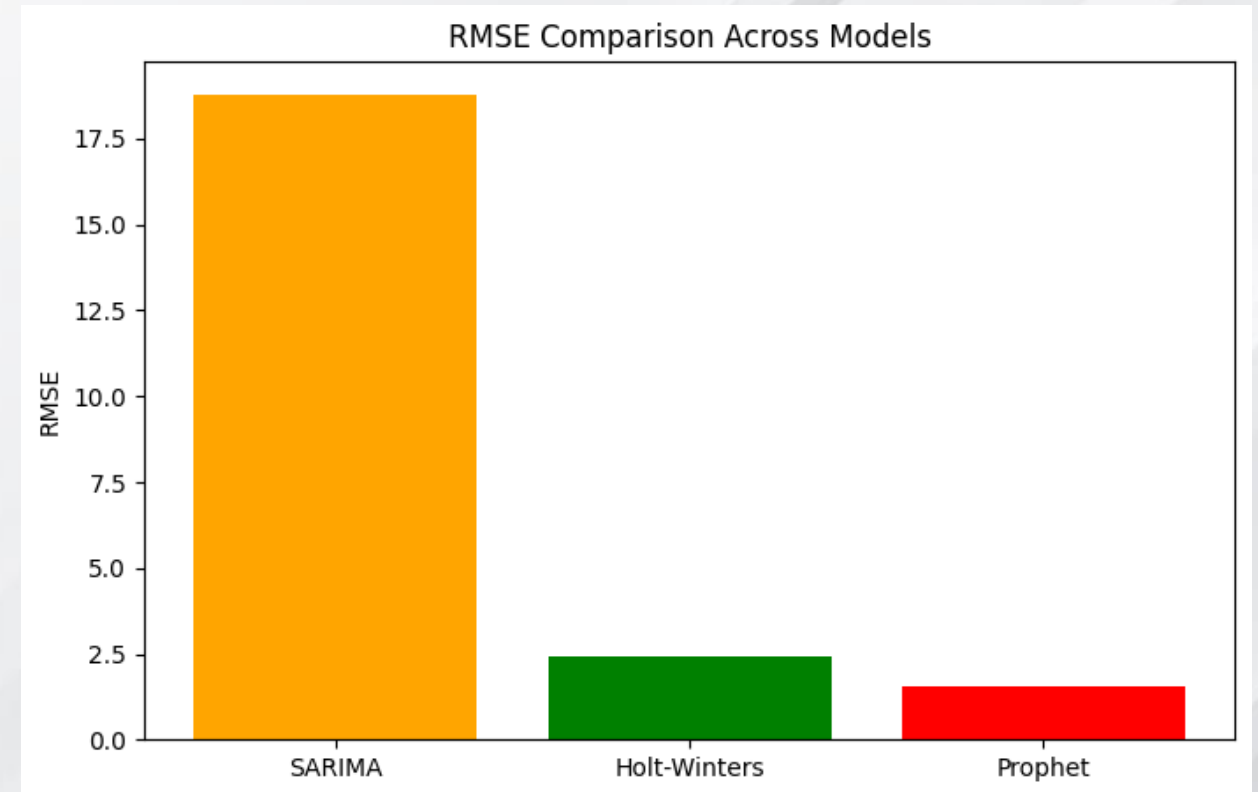
Visual: Bar chart of RMSE across models

Short-Term Accuracy (historical data)

Prophet RMSE = 1.54; most accurate

Holt-Winters RMSE = 2.41; accurate

SARIMA RMSE = 18.79; poor due to spike



Latest Machine Learning Research and Trends

Latest Time Series Research

Long-Horizon Transformers for Finance

Transformers are deep learning models originally for NLP.

- They predict financial data better than ARIMA/SARIMA for long-term horizons.
- Handles long-range dependencies better than ARIMA, LSTM, or Holt–Winters.
- Useful for monthly or quarterly forecasts where trends and seasonality span many periods.

Hybrid ARIMA + LSTM (2025)

- Combines ARIMA (linear trends) + LSTM (nonlinear patterns).
- Reduces forecast errors in noisy financial series.
- Impact on FinTech: Better predictions for prices, volatility, and risk in trading and portfolio management.

Latest Machine Learning Research and Trends

Latest Time Series Research

FFT / Frequency Preprocessing

- Uses Fourier Transform (FFT) or wavelet transforms to analyze the frequency components of financial time series before feeding them into Transformer models.
- Essentially separates periodic/seasonal patterns from noise.
- Helps models detect repeating cycles (daily, weekly, monthly) that are hard to learn from raw data.
- Reduces forecast error for series with strong seasonal or cyclical behavior.
- Impact on FinTech: Improves accuracy in predicting stock prices, trading volumes, or sales cycles.
- Useful for planning, portfolio strategy, and automated trading systems.

Latest Machine Learning Research and Trends

Diffusion & Generative Models (2025)

- Conditional diffusion models generate synthetic financial data, like prices or volatility series, based on historical patterns.
- Can simulate multiple realistic scenarios for the future, including extreme events.
- Captures rare but impactful events (“tail risks”) that are difficult for standard models.
- Useful for testing how strategies perform under extreme market conditions.
- Impact on FinTech: Supports stress testing, risk management, and scenario analysis for banks, hedge funds, and insurers.
- Helps in modeling potential losses and preparing for market shocks.

Latest Machine Learning Research and Trends

Diffusion & Generative Models

- Conditional diffusion models generate synthetic financial data, like prices or volatility series, based on historical patterns.
- Can simulate multiple realistic scenarios for the future, including extreme events.
- Captures rare but impactful events (“tail risks”) that are difficult for standard models.
- Useful for testing how strategies perform under extreme market conditions.
- Impact on FinTech: Supports stress testing, risk management, and scenario analysis for banks, hedge funds, and insurers.
- Helps in modeling potential losses and preparing for market shocks.

Latest Machine Learning Research and Trends

RAG & Retrieval for Finance

- Retrieval-Augmented Generation (RAG) combines an LLM with a specialized financial knowledge base (FinDER dataset) to answer questions accurately.
- The system retrieves relevant documents and uses them to produce context-aware answers.
- Improves factual correctness compared to standard LLMs, which may hallucinate.
- Can handle complex financial questions or queries over large document collections.
- Impact on FinTech: Useful for decision-making, regulatory reporting, audits, and compliance checks.
- Reduces errors in automated financial advice or document summarization.

Lecture

10

Recap

- SARIMA: Handling trend, seasonality, and noise in time series
- Holt–Winters: Exponential smoothing for trend and seasonal patterns
- Prophet: Automatic decomposition with trend, multiple seasonalities, and holiday effects
- VAR (Vector AutoRegression): Multivariate time-series modeling, capturing interdependencies
- Seasonal ACF & PACF: Identifying seasonal correlations and lags
- Full Python Forecasting Workflow: Decomposition, model fitting, prediction, and evaluation
- 2025 ML Innovations: Latest research in time series, regression/classification, trees, and ensemble methods

References

- López de Prado, M. Advances in Financial Machine Learning. Wiley.
- Klaas, J. Machine Learning for Finance: Principles and Practice for Financial Insiders. Packt Publishing.
- Coursera – Machine Learning & Time Series Courses
(Used for structure, pedagogical flow, and conceptual clarity.)
- Hyndman, R.J. & Athanasopoulos, G. Forecasting: Principles and Practice. OTexts.
- Box, G.E.P, Jenkins, G.M., & Reinsel, G.C. Time Series Analysis: Forecasting and Control.
- Brownlee, J. Introduction to Time Series Forecasting. Machine Learning Mastery.

References

- SpringerLink. 2025. “Transformer Variants for Long-Horizon Forecasting: A Systematic Review”.
- Khalid, et al. 2025. “Hybrid ARIMA–LSTM Pipelines for Financial Time Series Forecasting.”
- <https://www.sciencedirect.com/science/article/pii/S0950705125006094>
- MDPI. 2025. “FFT and Wavelet Preprocessing for Transformer-Based Time Series Forecasting.”
- IJCAI. 2025. “Conditional Diffusion Models for Financial Time Series and Tail Risk Modeling.”
- Emergent Mind / ArXiv. 2025. “Retrieval-Augmented Generation for Finance (FinDER Dataset).”