Multi-Model Financial Time Series Forecasting

Fundamental Concepts for Time Series Analysis

→ A **time series** is a sequence of ordered data through time → $y_1, y_2, y_3, \ldots, y_n$

→ The **trend** is a smooth, sustained upward or downward movement over time

$y_t = \text{trend}(t) + \text{noise}$

Note: If there is a trend, AR and ARMA fail, that's why we need ARIMA (the "I" in Integrated) to remove the trend.

→ The **seasonality** is a pattern that repeats at certain fixed periods. For example every 7 days, every 12 months, every 2 years, every presidential year,...)

$y_t = y_t - m$ m : cycle lenght Note: ARIMA does not manage it well.

→ The **periodicity** is similar to the seasonality, but more general → any pattern that repeats in cycles. Is the time it takes for the pattern to repeat.

→ The **noise** is the unpredictable or random part in the series. This is what remains when the trend, the seasonality, the periodicity and any pattern are removed.

$y_t = \text{unpredictable patterns} + \varepsilon_t$ ε_t is white noise : $\varepsilon_t \sim N(0, \sigma^2)$ → Normal distribution → Mean = 0 and Std. Dev = σ so Variance = σ^2

→ **Mean**: Central value of a set of data

$$\mu = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Variance: It measure how dispersed the data is with respect to the mean. Cuadratic dispersion

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (X_i - \mu)^2$$

→ low variance: data points close together
→ high variance: data points very far apart

Standard Deviation: Square root of variance. Dispersion

$\sigma = \sqrt{\sigma^2}$ → it has the same measure units of the data

Covariance: It measure how 2 variables move together.

$$Cov(X, Y) = \frac{1}{n}\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})$$

→ $Cov > 0$ → When X goes up, Y tends to go up
→ $Cov < 0$ → When X goes up, Y tends to go down
→ $Cov = 0$ → There is no linear relationship between X and Y

Autocorrelation: It measures how a variable relates to itself at different points in time.

$$\rho_K = \frac{Cov(X_t, X_{t-K})}{\sigma^2}$$

Compares X_t with X_{t-K}, where K is the lag

→ $\rho > 0$: high values follow high values (trend)
→ $\rho < 0$: high values follow low values (alternating)
→ $\rho = 0$: no temporal dependency

Differencing: This tool removes trends and seasonality.

→ Normal Difference: $\Delta y_t = y_t - y_{t-1}$

→ Seasonal Difference: $\Delta_m y_t = y_t - y_{t-m}$

Seasonal Process: The series depends on what happened in previous steps.

$$y_t = \varphi \cdot y_{t-m} + Noise$$

Regression: Relation between a target and the features:

$$y_t = \beta_0 + \beta_1 \cdot X_{1,t} + \beta_2 \cdot X_{2,t} + \dots + \varepsilon_t$$

In time series, regression with lags:

$$y_t = \beta_1 \cdot y_{t-1} + \beta_2 \cdot y_{t-2} + \dots$$ → AR Model

Errors (Residuals): Difference between the predicted value and the real value.

$$\varepsilon_t = \hat{y}_t - y_t$$

MA models use errors from the past as main feature.

Stationarity: A series is stationary:

→ the mean doesn't change over time.
→ the variance doesn't change.
→ the autocorrelation doesn't change over time.

AR Process: Relation between past values:

$$y_t = \varphi \cdot y_{t-1} + \varepsilon_t$$

MA Process: Relation between past errors:

$$y_t = \theta \cdot \varepsilon_{t-1} + \varepsilon_t$$

Granger Causality: A series X "causes Granger to" Y if X contains information that helps predict Y.

Lag: A temporary delay

Ex: Lag 1 → yesterday
Lag 7 → a week ago

Additive Model: $y_t = trend + seasonality + noise$

Multiplicative Model: $y_t = trend \cdot seasonality \cdot noise$

Structural Model (Decomposition): A series can be divided: $y_t = T_t + S_t + N_t$ (Trend, Seasonality, Noise)

• AR Model

• An Autoregressive model predict using its past. Depends linearly of its previous values plus an error term.

order p model → AR(p): $X_t = \varphi_1 \cdot X_{t-1} + \varphi_2 \cdot X_{t-2} + \dots + \varphi_p \cdot X_{t-p} + \varepsilon_t$

X_t : value of the series at time t — p : take the last p values of the series.

$\varphi_1, \varphi_2, \dots, \varphi_p$: model parameters (weights) → the model has to "learn" those parameters.

$\varepsilon_t \sim N(0, \sigma^2)$: Noise

- The AR doesn't manage well trends, neither sessonalities or external variables. Slow decay of autocorrelation

• MA Model

- A moving average model describe a time series as a linear combination of current and previous random errors. This method models previous errors not previous values

order q model → MA(q): $X_t = \mu + \varepsilon_t + \theta_1 \cdot \varepsilon_{t-1} + \theta_2 \cdot \varepsilon_{t-2} + \dots + \theta_q \cdot \varepsilon_{t-q}$

X_t : value of the series at time t — q : take the last p values of the series

$\theta_1, \theta_2, \dots, \theta_q$ model parameters (weights)

$\varepsilon_t \sim N(0, \sigma^2)$: Noise — μ : process mean

- $E[X_t] = \mu$. Also the autocorrelation is cut off after q lags.

• ARMA Model

- ARMA (p, q) is a combination of: an AR(p) model → depends on past values; an MA(q) model → depends on past errors. } Usually to predict on sessonal exercises.

ARMA(p, q): $X_t = \varphi_1 \cdot X_{t-1} + \varphi_2 \cdot X_{t-2} + \dots + \varphi_p \cdot X_{t-p} + \varepsilon_t + \theta_1 \cdot \varepsilon_{t-1} + \theta_2 \cdot \varepsilon_{t-2} + \dots + \theta_q \cdot \varepsilon_{t-q}$

X_t : value of the series at time t
$\varphi_1, \varphi_2, \dots, \varphi_p$: autoregressive coefficients

$\varepsilon_t \sim N(0, \sigma^2)$: White Noise
$\theta_1, \theta_2, \dots, \theta_q$: moving average coefficients

- The current value depends of the previous p values and q previous errors.
- ARMA requires an stationary series (constant μ and σ^2 over time)

· ARIMA Model

- Autoregressive, Integrated (differentiated), Moving Average Model. Basically, ARIMA is ARMA but applied after differentiating the series to make it stationary, so originally the series has trend and is not stationary.
- ARIMA(p, d, q) combines:
  - AR(p) : dependencies on past values
  - d differentiations : remove trends and convert the series to stationary.
  - MA(q) : dependencies on past errors

ARIMA(p, d, q) :

→ Differentiate d times : $y_t = \Delta^d X_t = \Delta(\Delta^{d-1} X_t)$

$d=1$: $\Delta X_t = X_t - X_{t-1}$
$d=2$: $\Delta^2 X_t = X_t - 2X_{t-1} + X_{t-2}$
$d=3$: $\Delta^3 X_t = X_t - 3X_{t-1} + 3X_{t-2} - X_{t-3}$

Normalmente $d=1$ máximo $d=2$

→ Find y_t with ARMA(p, q) :

$$y_t = \varphi_1 \cdot y_{t-1} + \varphi_2 \cdot y_{t-2} + \dots + \varphi_p \cdot y_{t-p} + \varepsilon_t + \theta_1 \cdot \varepsilon_{t-1} + \theta_2 \cdot \varepsilon_{t-2} + \dots + \theta_q \cdot \varepsilon_{t-q}$$

· SARIMA Model

- Means S (seasonal) + ARIMA. A SARIMA model combines:
  - → Non-seasonal ARIMA components
  - → Seasonal ARIMA components
  - → A seasonal period s (ex. $s=12$ → monthly data)

- Basically SARIMA is ARIMA + Seasonal ARIMA applied to lags that occur every S periods.

$$SARIMA(p, d, q)(P, D, Q)_s$$

Non-Seasonal Components:

- p: autoregressive order
- d: number of simple differentiations
- q: moving average order

Seasonal Components:

- P: seasonal AR order
- D: number of seasonal differentiations
- Q: seasonal MA order
- s: seasonality period (monthly → 12, quarterly → 4, weekly → 7)

- ARIMA + seasonal components (repeats every season)

$$\Phi \cdot (B^s) \cdot \varphi \cdot (B)(1-B)^d(1-B^s)^D \cdot X_t = \emptyset(B^s) \cdot \theta(B) \cdot \epsilon_t$$

Where: $\varphi, \theta \rightarrow$ non-seasonal part, $\Phi, \emptyset \rightarrow$ seasonal part, B is the lag operator → $BX_t = X_{t-1}$

• SARIMAX Model

- SARIMA + External Variables. SARIMAX model predicts using the past of the series, the seasonality and external or exogenous variables.

$$SARIMAX(p, d, q)(P, D, Q)_s$$

↳ variables that does not belong to the series but help to explain its behaviour.

Non-Seasonal ARIMA — Seasonal component — Exogenous variables Z_t

$$\Phi \cdot (B^s) \cdot \varphi \cdot (B)(1-B)^d(1-B^s)^D \cdot X_t = \lambda \cdot Z_t + \emptyset(B^s) \cdot \theta(B) \cdot \epsilon_t$$

Where: $\varphi, \theta \rightarrow$ non-seasonal part, $\Phi, \Theta \rightarrow$ seasonal part, B is the lag operator $\rightarrow B X_t = X_{t-1}$, $s \rightarrow$ seasonal period

$d, D \rightarrow$ simple & seasonal differentiation $\quad \lambda \rightarrow$ External variables coefficient $\quad Z_t \rightarrow$ Vector of external variables

• VAR Model

. The Vector Autoregressive model is the multivariate extension of AR Model because it can handle multiple time series simultaneously. Each variable depends of his own lag and the lags of all other variables in the system.

. Definition : Suppose K series $Y_{1,t}, Y_{2,t}, \ldots, Y_{K,t}$ with p lags :

$$Y_t = c + A_1 \cdot Y_{t-1} + A_2 \cdot Y_{t-2} + \ldots + A_p \cdot Y_{t-p} + \varepsilon_t$$

X_t : vector of [Kx1] of variables over time t (dimension)

c : vector of constants (biases)

$A_1, A_2, \ldots, A_p$: coefficient matrices of [K x K] for each lag

ε_t : vector of random errors with covariance $\sum_\varepsilon$ → Noise

. VAR is the generalization of AR, so each variable depends of its own past values and the past values of the other variables.

. The K series have to be stationary

• VARMAX Model

. Vector Autoregressive Moving Average with Exogenous variables.

- Vector Autoregressive → Several series that affect each other
- Moving Average → Depend of past errors
- Exogenous variables → External variables affect the series behaviours.

- Suppose K series Y_t and m exogenous variables Z_t

$$\text{VARMAX}(p,q,r): \quad Y_t = \lambda + \sum_{i=1}^{p} A_i \cdot Y_{t-i} + \sum_{j=1}^{q} B_j \cdot \varepsilon_{t-j} + \sum_{h=1}^{r} C_h \cdot Z_{t-h} + \varepsilon_t$$

- Y_t: vector of K endogenous variables (series to model) $[K \times 1]$
- λ: vector of constants (biases) $[K \times 1]$
- $\sum_{i=1}^{p} A_i \cdot Y_{t-i}$: VAR component: Autoregressive coefficient matrices $[K \times K]$
  - Each matrix describes how past variables across all series affect the current value
  - Internal dynamic of endogenous variables
- $\sum_{j=1}^{q} B_j \cdot \varepsilon_{t-j}$: MA component: Moving averages coefficient matrices $[K \times K]$
  - They represent how past errors influence current variable errors
- $\sum_{h=1}^{r} C_h \cdot Z_{t-h}$: Exogenous variables component:
  - $\rightarrow Z_t$: vector of m variables
  - $\rightarrow C_h$: coefficient matrices that relate the exogenous with the endogenous
- ε_t

• ARCH Model

- Autoregressive Conditional HeteroskEdasticity. Designed to capture changing volatility over time.
- The variance of the series is not constant, but depends on past error → Recent large errors imply high future volatility.

• Bear in Mind the following concepts for ARCH, GARCH,...

- We can describe 2 types of variances in Finance:

Well-known variance → 1- Variance (standard): It is a statistical measure that indicates how much the values are dispersed with respect to their mean. Usually used for daily or monthly returns, series of assets/portfolio
→ higher variances, higher dispersion of returns, higher risk

2- Temporal Variance: Usually describes 2 situations:

→ **Time-varying variance:** The variance that changes over time. This refers to the fact that volatility is not constant, but rather moves in regimes or clusters.

. Here the temporal variance is the conditional variance at each point of time: σ_t^2 rather than a unique global variance.

→ **Variance decomposition over time:** How variance accumulates or decomposes depending on the time horizon

. Daily variance
. Monthly variance
. Annual variance ≈ Daily variance · 252 (trading days)

. In finance, the temporal variance can be interpreted as Volatility (normally volatility = std = $\sqrt{\sigma^2}$)

. Here, we assume that volatility is not constant → Heteroskedasticity → Non-constant Variance

Conditional Heteroskedasticity → The volatility depends on the past, it's not fixed.

It depends on past variables or available information — Non-constant variance

. AR, MA, ARMA models assume the errors as a Normal Distribution, that means a constant variance

$\varepsilon_t \sim N(0, \sigma^2)$ → Not in this case (ARCH)

. Unconditional Variance vs Conditional Variance → ARCH is based on Conditional Variance

Unconditional Variance: Long-term average value

Conditional Variance: Variance given what happened before

$Var(\varepsilon_t \mid \text{past information}) = h_t$

. ARCH models are applied to returns, not prices → because the returns tend to be approximately stationary.

$r_t = Ln\left(\frac{P_t}{P_{t-1}}\right)$ (log returns) — P_t: price period t; P_{t-1}: price previous period t

$Z_t \sim N(0,1)$

- ARCH(q):

Mean Ecuation: $r_t = \mu + \varepsilon_t$ → $\sigma_t \cdot Z_t$

Conditional Variance Ecuation:

$$h_t = \alpha_0 + \alpha_1 \cdot \varepsilon^2_{t-1} + \alpha_2 \cdot \varepsilon^2_{t-2} + \dots + \alpha_q \cdot \varepsilon^2_{t-q}$$

h_t: Conditional Variance at t

α_0: Constant

Squared Past Errors → past errors influence the future volatility

It squared because the volatility measured dispersion, and dispersion depends on magnitude not direction

$\alpha_i \geq 0$: ARCH coefficients

q: lags

• GARCH Model

- Generalized Autoregressive Conditional HeteroskEdasticity → the future volatility depends on past volatility and past errors.

- GARCH(p, q):

$$h_t = \sigma^2_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \cdot \varepsilon^2_{t-1} + \sum_{j=1}^{p} \beta_j \cdot \sigma^2_{t-j}$$

$\alpha_i, \beta_j \geq 0$

$h_t = \sigma^2_t$: Conditional Volatility at time t → changes over time → expected uncertainty respect the return at time t

α_0: Constant → base volatility

ARCH part:
- → α_i: ARCH coefficients
- → ε^2_{t-i}: Squared past errors (magnitude)

GARCH part: How past variance affects present variance
- → σ^2_{t-j}: Past variances (past volatility)
  - ↳ if volatility was high yesterday, it tends to remain high today
- → β_j: GARCH coefficients

• DCC-GARCH Model

- This is a multivariate model based on GARCH, allows that correlation between many series change over time. DCC means Dynamic Conditional Correlation.

- Each series has its own volatility (GARCH) and the correlation between series is dynamic, not fixed.

- Suppose, we have 2 series of returns $r_{1,t}$ and $r_{2,t}$:
  - → Each return has an univariate GARCH:

$$\sigma^2_{1,t} = \alpha_0 + \alpha_1 \cdot \varepsilon^2_{1,t-1} + \beta_1 \cdot \sigma^2_{1,t-1} \qquad \sigma^2_{2,t} = \gamma_0 + \gamma_1 \cdot \varepsilon^2_{2,t-1} + \delta_1 \cdot \sigma^2_{2,t-1}$$

  - → The correlation between $\varepsilon_{1,t}$ and $\varepsilon_{2,t}$ changes over time:

$$R_t = (1-a-b) \cdot \bar{R} + a(u_{t-1} \cdot u^T_{t-1}) + b \cdot R_{t-1}$$

Conditional correlation matrix at time t

Average historical correlation

Standardized error ε_t / σ_t

a, b: parameters that control the correlation dynamics.

- Captures conditional volatility of each series and captures dynamic correlations between series → This allows capturing co-movements in volatility. During a crisis, stocks become more correlated. In normal times the correlation may be low.

Timeline

AR	VAR	ARCH
MA	VARMAX	GARCH
ARMA		DCC-GARCH
ARIMA		BEKK-GARCH
SARIMA		
SARIMAX		

Model	Type	Captures	Handles Trend?	Handles Seasonality ?	Handles Multiple Series?	Exogenous Vars?	Volatility Modeling?	Typical Use Cases
AR(p)	Univariate	Autoregressive dependence	✓ via AR roots (weak)	✗	✗	✗	✗	Simple stationary series
MA(q)	Univariate	Shock/innovation dependence	✗	✗	✗	✗	✗	Noise/signal smoothing
ARMA(p,q)	Univariate	AR + MA short-memory dynamics	✗	✗	✗	✗	✗	Stationary economic/industrial data
ARIMA(p,d,q)	Univariate	Trend + ARMA	✓ via differencing	✗	✗	✗	✗	Non-stationary series with trend
SARIMA(p,d,q)(P,D,Q)_s	Univariate	Trend + seasonality + ARMA	✓	✓	✗	✗	✗	Monthly/quarterly business cycles
SARIMAX(p,d,q)(P,D,Q)_s	Univariate	Trend + seasonality + exogenous inputs	✓	✓	✗	✓	✗	Forecasting with external drivers (weather, marketing, etc.)

Model	Type	Captures	Handles Trend?	Handles Seasonality ?	Handles Multiple Series?	Exogenous Vars?	Volatility Modeling?	Typical Use Cases
VAR(p)	Multivariate	Dynamic interactions across series	✓ via differencing	✗	✓	✓ (as VARX)	✗	Macro-finance systems, multiple indicators
VARMAX(p,q)	Multivariate	VAR + MA + exogenous	✓ via differencing	✗	✓	✓	✗	Complex systems with external forces
ARCH(q)	Univariate	Time-varying variance from past shocks	✗	✗	✗	✗	✓ (variance only)	Financial returns, volatility clustering
GARCH(p,q)	Univariate	Persistent volatility (ARCH + lagged variance)	✗	✗	✗	✗	✓	Risk forecasting, asset volatility
DCC-GARCH	Multivariate	Dynamic correlations + individual GARCH	✗	✗	✓	✗	✓ (vol. + correlations)	Portfolio risk, co-movement in markets
BEKK-GARCH	Multivariate	Full variance–covariance dynamics (positive-definite)	✗	✗	✓	✗	✓	Complex multivariate volatility, asset interactions

Fundamental Concepts for understanding Stochastic Pricing Models

→ An **stochastic process** is a variable that evolves over time and its value has randomness.

- The price of an asset tomorrow is not deterministic → So the price can't be predicted, we can only describe its distribution
- At smaller time-frames, the price changes a bit at each step → Each change has a predictable part (trend) and a random part → It is formalized using SDEs - Stochastic Differential Equations

→ White Noise (Brownian Motion)

. It is the random component, it is the "error" or "shock" that can't be predicted. Also can be described in the following properties:

* Has a 0 mean ($\bar{X} = 0$)
* Has a constant variance → Homoscedasticity
* Has a null correlation with the past errors.
* Independent increments

. The Brownian Motion does not follow a Normal Distribution. The increments (independents) follow a Normal Distribution

$W_{t+h} - W_t \sim N(0, k)$ → The increment of the process between time t and time t + h has a N (normal distribution) with mean 0 and variance K. t + h does not depend of t

or

$dW_t \sim N(0, dt)$

$E[W_{t+h} - W_t] = 0$ → There is no uptrend or downtrend

$Var[W_{t+h} - W_t] = K$ → Variance proportional to time

↓

The larger the time interval, the more dispersed the jump can be

→ **Drift vs Diffusion**: Any movement of an asset is broken down into 2 forces:

1. **Drift (μ - The Trend)**: This is the deterministic part. It's the force that constantly pushes the price up or down. If a stock yields an average of 10% annually, that's its drift. To conclude this is the predictic, expected or the trend in the change of the assets.

2. **Diffusion (σ - The Volatility)**: This is the noisy and random part. It's the amplitude with which the price oscillates around its trend due to several events (news, fear, euphoria, ...). It's multiplied by the random factor (dW_t or $W_{t+h} - W_t$). This part says how much volatility has an assets in its random behaviour.

→ The logarithm of the price is often modeled because there can't exist negative prices. So, the model assumes that the prices follow a log-normal distribution. We are talking about log-returns.

→ The Markovian Property: "The market has no memory" → The future price depends only of the current price.
↳ Memoryless This is the opposite of the previous models, where the past matters a lot

→ Stochastic Volatility: Random and changing volatility. Not only is the price uncertain, but so is the price volatility.
↳ Volatility Clusters: volatility does not change uniformly. It tends to appear in waves or clusters → periods of high volatility followed by periods of high volatility and periods of low volatility followed by periods of low volatility. Those patterns are clusters.

→ For this project, we will implement the models in below:

1. Geometric Brownian Motion
2. Merton - Jump Diffusion
3. Heston
4. Kou DEJD
5. NIG

→ Wishart Stochastic Volatility (pending) → Too complex!! → Multidimensional Heston Model
→ Rough Bergomi (rough volatility)

· Geometric Brownian Motion (GBM Model)

. This is the base model for modern finance. The GBM is a stochastic process (incorporates randomness). It says that the price of an asset grows exponentially over time but it is constantly hit by random shocks proportional to its current price.

. Unlike Simple Brownian Motion, GBM Model:

→ Never produces negative values
→ Has average growth
→ Its volatility scales with prices (variability increases when the price is high)

$$dS_t = \mu \cdot S_t \cdot dt + \sigma \cdot S_t \cdot dW_t$$

- dS_t: The little change in price
- dt: Infinitesimal increment of time
- μ: Average growth rate
- S_t: Asset price at time t
- σ: Asset volatility (percentage)
- dW_t: Increment of a standard Brownian Motion (Random Noise)

. Drift ($\mu \cdot S_t \cdot dt$) → Represents the deterministic average price growth. It is the predictable component of change or expected growth.

. Diffusion ($\sigma \cdot S_t \cdot dW_t$) → Represents the random part, the uncertainty of the price. It depends of the current price (S_t) and the Brownian Motion

→ dW_t (Brownian Motion): It is a continuous stochastic process, with independent and normally distributed increments:

$$dW_t \sim N(0, dt)$$

. As you can see, the main limitation of this model is that assumes the volatility as constant. That's why this is the base model.

. The solution of the SDE (Itô's Formula):

$$S_t = S_0 \cdot e^{\left(\mu - \frac{\sigma^2}{2}\right)\cdot t + \sigma \cdot W_t}$$

S_0 → Initial price

$\left(\mu - \frac{\sigma^2}{2}\right)$ → Itô's Correction (due to volatility)

. For simulation:

$$S_{t+dt} = S_t \cdot e^{\left(\mu - \frac{\sigma^2}{2}\right)\cdot \Delta t + \sigma \cdot \sqrt{\Delta t} \cdot Z_t}$$

$Z_t \sim N(0,1)$

· Merton Jump Diffusion Model

. The GBM model assumes the price as a continuous "path" (without gaps). But the truth is that in reality, the price can "jump" or moves with gaps. Prices may experience large jumps due to unexpected events.

. The Merton model introduces a Poisson process to capture those random jumps.

. Can be summarized as a GBM with occasional random jumps.

. This equation has the advantage of clearly split the continuous parte (GBM), the adjusted trend for drifts and the discrete jumps.

$K = E[Y - 1]$ → Average size of net multiplicative leap

$Y = 1 + r$ r → percentage change of the jump

$$dS_t = (\mu - \lambda \cdot K) \cdot S_t \cdot dt + \sigma \cdot S_t \cdot dW_t + S_t \cdot dJ_t$$

- dS_t: The little change in price
- dt: Infinitesimal Increment of time
- μ: Average growth rate
- λ: Intensity of the Poisson process → Number of expected jumps per time unit
- S_t: Asset price at time t
- σ: Asset volatility
- dW_t: Brownian Motion (Continuous random normal noise)
- dJ_t: Jump increment → Discrete jumps

. The magnitude of the jump is continuous, the quantity of jumps have to be discrete or integer.

- Drift adjusted by jumps $((\mu - \lambda \cdot K) \cdot S_t \cdot dt)$ → This is the average growth rate but adjusted to compensate the expected effect of the jumps.

  → On average, multiplicative jumps cause the price to change by K with each jump. As they (jumps) occur with intensity λ, the expected effect per time unit is $\lambda \cdot K$.

- Continuous Component - GBM $(\sigma \cdot S_t \cdot dW_t)$ → The same GBM's component. It represents the little continuous random fluctuations of the price.

  → Assumes $dW_t \sim N(0, dt)$. Also the volatility scales with the price S_t.

- Jumps Components $(S_t \cdot dJ_t)$ → This component models discrete jumps in the price.

  → $dJ_t = 0$ if a jump does not occur. Generally:

  $$dJ_t = (Y-1) \cdot dN_t \quad \text{where:} \quad \begin{cases} Y \sim \text{Lognormal}(\nu, \delta^2) \rightarrow \text{jump size} \\ N_t \sim \text{Poisson}(\lambda) \rightarrow \text{number of jumps} \end{cases} \} \text{If the jump occurs, } S_t \text{ multiplies by } Y$$

- The discrete formula for the solution of the previous model:

$$S_t = S_0 \cdot e^{((\mu - \lambda \cdot K - \sigma^2/2) \cdot t + \sigma \cdot W_t)} \cdot \prod_{i=1}^{N_t} Y_i$$

Initial price

Adjusted GBM by the average of the jumps

Product of all jumps that occur until time t

- For simulation:

$$S_{t+dt} = S_t \cdot e^{((\mu - \lambda \cdot K - \sigma^2/2) \cdot \Delta t + \sigma \cdot \sqrt{\Delta t} \cdot Z_t)} \cdot Y^{d \cdot N_t}$$

• Heston Model

. In this model, the volatility is random too (stochastic process). It is more powerful because:
  → The volatility is not constant
  → Has mean-reversion (tends to return to a certain level)
  → The volatility rises in crisis scenarios but falls in stable periods.

. The asset price S_t and its variance (squared volatility/standard deviation) follow coupled stochastic processes. The prices fluctuate as a GBM, but the volatility is not fixed. The variance follows a CIR (Cox-Ingersoll-Ross) process that guarantee that always will be positive.

. The equations of the Heston model are:

Price Equation:

$$dS_t = \mu \cdot S_t \cdot dt + \sqrt{V_t} \cdot S_t \cdot dW_t^{(1)}$$

- dS_t: Change in price
- μ: Asset drift (Average growth rate)
- S_t: Asset Price
- $\sqrt{V_t}$: Random Instant Volatility
- $dW_t^{(1)}$: Brownian Motion # 1 (random noise)

. The volatility tends to fluctuates in random shocks but tends to return to an average value.

Variance Equation:

$$dV_t = \kappa \cdot (\theta - V_t) \cdot dt + \sigma_v \cdot \sqrt{V_t} \cdot dW_t^{(2)}$$

- dV_t: Change in variance
- V_t: Instant Variance
- $dW_t^{(2)}$: Brownian Motion # 2 (noise)
- $\kappa > 0$ (speed of reversion to the mean): How fast the market calms down after a volatility spike.
- $\theta > 0$ (long term variance level): volatility always wants to return to this "normal" level
- $\sigma_v > 0$ (volatility of variance) → vol of vol: How quickly and how strong can the asset's volatility change → How much uncertainty/variability does the volatility process have

. The volatility changes over time. As the variance follows a CIR process (always positive) → Feller Condition:

$$2 \cdot \kappa \cdot \theta = \sigma_v^2$$

→ the variance goes up if $V_t < \theta$
→ the variance goes down if $V_t > \theta$
} Mean Reversion

. Correlation between Price and Volatility:

$$dW_t^{(1)} \cdot dW_t^{(2)} = \rho \cdot dt$$

ρ: correlation between both noises

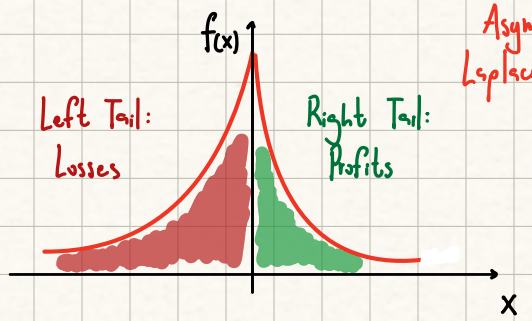
→ Usually if $\rho < 0$:
  * If the price goes down, the volatility goes up
  * Reproduce "volatility clustering"

. The solution does not exist in a simple and defined form like GBM. It has various approaches.

• Kou Model (Double Exponential Jump Diffusion)

. The problem of the original Merton model is that only assumes positives jumps (this isn't very realistic because in the market we see both euphoric periods where the assets rise and crashes that trigger falls. The Kou model simulates more frequent crashes and more violent rallies.

. This model replaces the Normal Distribution in Merton Model by the Asymmetrical Double Exponential (sKewness) or Laplace Distribution.



Asymmetrical Laplace Distribution

. The probability of a jump of size X is:

$$f(x) = p \cdot \eta_1 \cdot e^{-\eta_1 \cdot x} 1_{\{x>0\}} + q \cdot \eta_2 \cdot e^{-\eta_2 \cdot x} 1_{\{x>0\}}$$

Profits decay with velocity η_1 and losses with η_2

Usually $\eta_2 < \eta_1$, which means that the left tail is fatter → extreme crashes are more likely that extreme rises.

. Look at the ecuation (almost identical respect Merton's original ecuation):

$$dS_t = \mu \cdot S_t \cdot dt + \sigma \cdot S_t \cdot dW_t + S_t \cdot d\left[\sum_{i=1}^{N_t} (V_i - 1)\right]$$

- dS_t: The little change in price
- μ: Average growth rate
- S_t: Asset price
- σ: Asset volatility
- dW_t: Brownian Motion (Continuous random normal noise)
- J_t: $\sum_{i=1}^{N_t} (V_i - 1)$
- $Y = \text{Ln}(V_i)$ it distributes according the double exponential

. Drift or Deterministic Growth ($\mu \cdot S_t \cdot dt$) → It represents the expected trend of the price, where main parameter is the average growth rate.

. Continuous Noise via Brownian Motion ($\sigma \cdot S_t \cdot dW_t$) → The same GBM's component. It represents the little continuous random fluctuations of the price.

→ Assumes $dW_t \sim N(0, dt)$. Also the volatility scales with the price S_t.

. Jump Process ($S_t \cdot d\left[\sum_{i=1}^{N_t} (V_i - 1)\right]$) → If there is a jump, the price changes as $S_t \to S_t \cdot V_i$. So, $V_i - 1$ is the proportional change of the jump. It is different of 0 when the jump happens.

→ N_t: Poisson Process → Where N_t is the quantity of jumps until time t, it follows a Poisson Process with λ intensity. Of course the jumps occur with randomness, with an expected number of $\lambda \cdot t$

→ V_i: Factors of Jump Sizes → Each jump multiplies the price by a V_i factor. In the Kou Model, the V_i factors are distributed by the double exponential

$$\ln(V_i) \sim \begin{cases} \text{Positive exponential with } \eta_1 \text{ parameter} \to \text{bullish jump} \nearrow & \text{(less frequent)} \\ \text{Negative exponential with } \eta_2 \text{ parameter} \to \text{bearish jump} \searrow & \text{(more frequent)} \end{cases}$$

• Normal Inverse Gaussian (NIG Model)

. The NIG model belongs to a family called Lévy Processes of Infinite Activity*. The model is supported by a Normal Inverse Gaussian distribution, this distribution also belongs to the family of Generalited Hyperbolic.

* It is a stochastic process whose increments can include infinitely many small jumps in any interval, unlike Poisson Process, which has only a finite number of visible jumps, and GBM which has no jumps and is completely continuous. These Lévy Processes combine drift (trend), a possible Brownian component, and a jump structure so dense that infinitely many microscopic oscillations occur between 2 instants, although only a finite number of large jumps. This is achieved by a Lévy measure that has infinite mass around the small jumps.

. This distribution (NIG) allow to represent fat tails (higher probability of extreme events) and having higher peaks than the Normal distribution → Present skewness or asymmetry.

→ What's the problem with the Normal Distribution in Finance?

- Says that extreme events (like collapses, black swans,...) are almost impossible to happen, which is not an accurate representation of markets.
- As the top of the bell is soft, it has a lot of values (returns) near zero (0).

. The NIG distribution is the mix of the Normal Distribution and the Inverse-Gaussian Distribution. We can think the NIG distribution like a sort of Normal distribution with stochastic variance (not fixed one).

. If the market volatility were constant → We would have a Normal Distribution.
. If the market volatility changes randomly following an Inverse Gaussian → We would have a NIG Distribution.

. The NIG model doesn't define as a SDE, but a random variable. Because of the mix of distributions and the SDEs only model continuous process:

→ Creation of a NIG(X) random variable by mixing a Normal variable (Z) and an Inverse Gaussian variable (Y):

$$X = \mu + \beta \cdot Y + \sqrt{Y} \cdot Z \quad , \quad \text{where: } Z \sim N(0,1) \rightarrow \text{standard and normal noise}$$

In "boring" days, little variance
In "crisis" days, big variance
$\leftrightarrow$ $Y \sim IG(\delta, \gamma) \rightarrow$ stochastic variance

→ NIG Probability Density Function:

$$f_{NIG}(x) = \frac{\alpha \cdot \delta}{\pi} \cdot \frac{K_1\left(\alpha \cdot \sqrt{\delta^2 + (x-\mu)^2}\right)}{\sqrt{\delta^2 + (x-\mu)^2}} \cdot e^{\delta \cdot \sqrt{\alpha^2 - \beta^2} + \beta(x-\mu)}$$

- K_1: Bessel Function to manage the exponential decays.

. The Normal has only 2 parameters (μ, σ), while the NIG distribution has 4 parameters $(\alpha, \beta, \mu, \delta)$

1. Alpha (α) - Tail Heaviness: Controls how quickly the probability of extreme events decreases.
   - high α → light tails, looks like a Normal Distribution
   - low α → fat tails, means high extreme risk.

2. Beta (β) - SKewness: Controls which way the distribution leans.
   - $\beta = 0$ → Symmetrical - Normal Distribution
   - $\beta < 0$ → The left tail is longer → stock falls are more violent than rises

3. Delta (δ) - Scale: It is analogous to volatility (σ). It determines how wide the central distribution is. Measures how big the dispersion is.

4. Mu (μ) - Location: It is the central displacement → The expected average return. Moves the distribution without deform it.

NIG model doesn't have a simple SDE, but is defined by its increases.

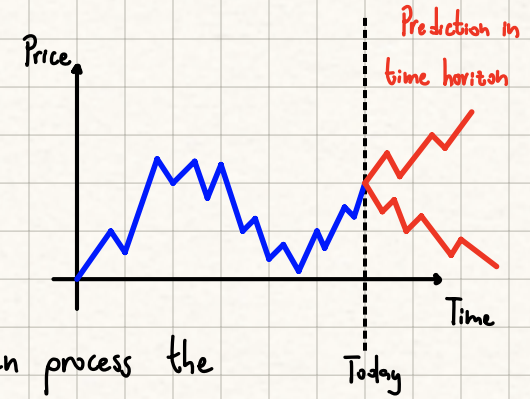
→ The dynamic price process of NIG is defined:

$$S_t = S_0 \cdot e^{L_t} \qquad \text{where} \quad L_t \sim NIG(\alpha \cdot t, \beta \cdot t, \mu \cdot t, \delta \cdot t)$$

- L_t is a Lévy Process (independent and stationary increases), has a NIG distribution in the increases and represents the log-returns of the asset.

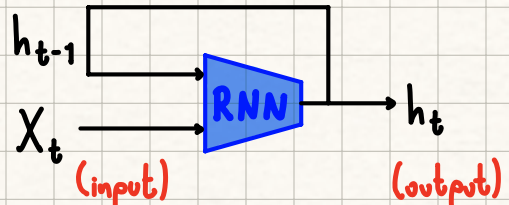
- NIG (Infinite Activity) → It has no continuous component. The path is formed purely by an infinite number of minuscule jumps.
  - Most of the jumps are so small that they appear as a continuous line.
  - Occasionally, there are large jumps
  - This better captures the market's microstructure (tick to tick)

RNN



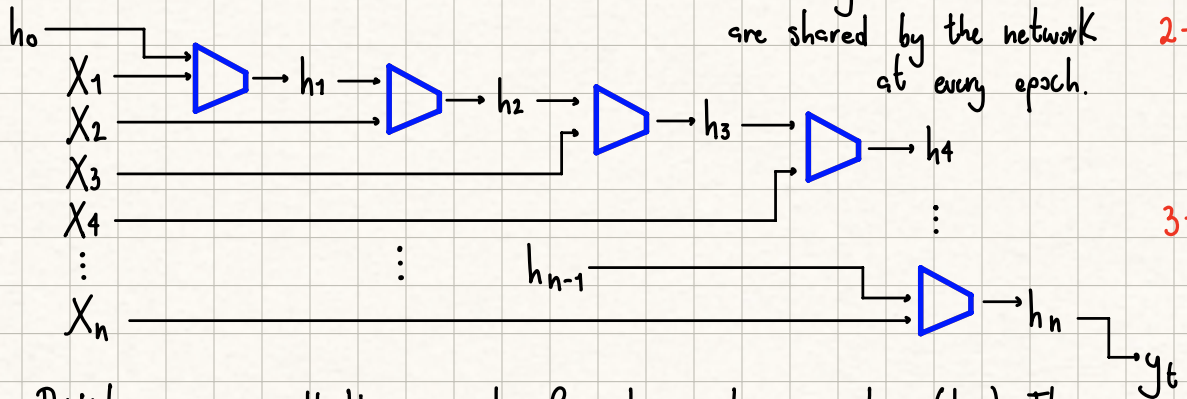
- Designed with the goal of processing sequences → Data with an order
- In order to predict the next value, the n previous values are used
- In a nutshell, the network process a value and calculate other as output. Then process the combination between the next value and the first output and so on so on...

- Here is a quick representation of the Recurrent Neuronal Networks.



- There are 3 main components
  - Input in each step: Value in the time series.
  - Hidden state: Intern memory that accumulates past data
  - Output: What the network produces in the step (prediction)

- Suppose the series:



The weights and bisses are shared by the network at every epoch.

- RNNs compress all the passed information into a vector (h_t). This vector is updated at each step, retaining only what is considered relevant.

The Vanishing / Exploding Gradient Problem: To optimize the NNs, we use the Backpropagation (gradients of each parameter, those gradients are put in the Gradient Descent to minimize the loss function and update the previous weights). As the gradient at some point will explode (tends to ∞) or vanish (tends to 0), it will affect the search of the optimal weights and bisses

Steps:

f could be ReLu or tanh
g could be Softmax, Sigmoid or Linear

1- Initialized the first hidden state (h_0)

2- The network receives the first input (X_1):

$$h_1 = f(W \cdot X_1 + U \cdot h_0 + b)$$

3- Calculate h_2 and X_2 get in:

$$h_2 = f(W \cdot X_2 + U \cdot h_1 + b)$$

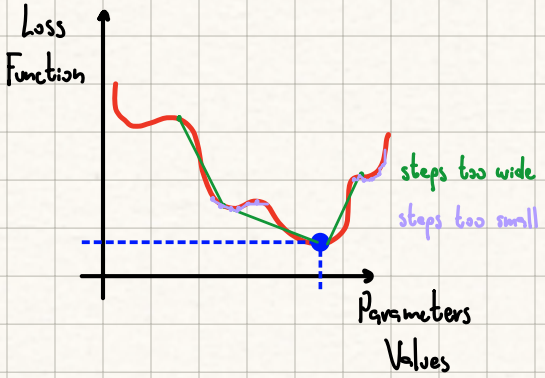
↳ Has information of X_1 and X_2

4- The previous 2 steps are repeated themselves.

$$h_n = f(W \cdot X_n + U \cdot h_{n-1} + b)$$

5- At each step or at the end, the following is done:

$$y_t = g(V \cdot h_n + C)$$



Derivative of the hidden state respect the previous one: $\frac{\partial h_t}{\partial h_{t-1}} = W^T \cdot (1 - h_t^2)$ → For tanh

↳ Multiplication of the recurrent weights by the activation function derivative

Gradient ← $\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\left(\frac{\partial h_{t-1}}{\partial h_{t-2}}\right)\left(\frac{\partial h_{t-2}}{\partial h_{t-3}}\right)$ ...

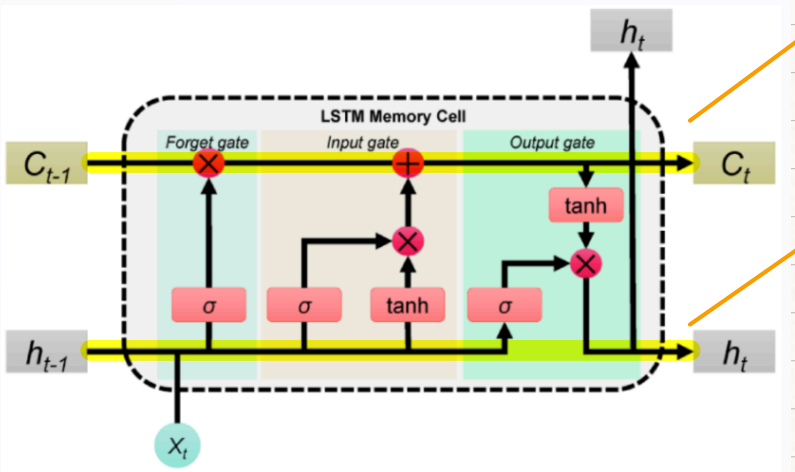
If the numbers: { > 1: numbers too big → The gradient explodes; < 1: numbers too small → The gradient vanishes ↳ There is no learning } The problem of RNNs, that why LSTM or GRU exist

LSTM

. To avoid the Gradient problems, the Long-Short Term Memory uses 2 separate paths to make predictions (next value), one path is for long-term memories and the other one is for short-term memories.

. Intuitively, these networks add 3 key actions:
1) Decide what to forget 2) Decide what new information to keep 3) Decide how much of that updated memory to show

. Basically, there are 2 types of memories { C_t: cell state → long term memory; h_t: hidden state → short term memory



Long Term memory → There are no weights and biases, therefore the Gradient problems disappear.

Short Term memory → The input and following elements in the flow are connected to biases and weights, so they could be modified.

. LSTMs only use 2 activation functions:

Sigmoid → turns any input into a number between (0, 1)

$\sigma(x) = \frac{1}{1+e^{-x}}$ $\sigma'(x) = \sigma(x)(1-\sigma(x))$

Tanh → turns any input into a number between (-1, 1)

$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ $\tanh'(x) = 1 - \tanh(x)^2$

- Suppose the series: $X_1, X_2, X_3, \ldots, X_t$ (t steps)

Steps: At each step the LSTM execute 4 main operations:

1. **Forget Gate** → decide how much old information should be kept

$$f_t = \sigma(W_f \cdot [h_{t-1}, X_t] + b_f)$$ → 0: completely forget / 1: remember everything

2. **Input Gate** → decide what new information will be added to the memory:

a. What information is relevant:

$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i)$$

b. What new content is being stored:

$$\hat{C}_t = \tanh(W_c \cdot [h_{t-1}, X_t] + b_c)$$

Later, those 2 components will be mixed to write in the memory.

3. **Memory Update (Cell State)** → the cell is update by combining what was forgotten and what will be added:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$$ → The Key of LSTM → "Forget the unnecessary and write what's new relevant"

4. **Output Gate** → it says which part of the memory should influence the exit from the current step.

$$O_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o)$$

- The hidden state is calculated: $h_t = O_t \cdot \tanh(C_t)$
- The prediction: $y_t = g(W_y \cdot h_t + b_y)$

g is:
- Softmax → Multi-class classification
- Sigmoid → Binary classification
- Linear → Regression

Temporal Fusion Transformer

. Combines the best of the RNNs/LSTMs to capture the short-term dynamics and the Attention of Transformers to catch long-term dependencies.

The predicted value for the i-th time series at time t for quantile q and forecast horizon τ

$$\hat{y}_i(q,t,\tau) = f_q(\tau, \hat{y}_{i,t-K:t}, z_{i,t-K:t}, x_{i,t:t+\tau}, s_i)$$

Static covariates from the i-th time series

The function used to compute the compute forecast

The forecast horizon

Historical target values for the i-th time series from time t-K to time t

Unknown inputs (past covariates) for the i-th time series from time t-K to time t

Known inputs (future covariates) for the i-th time series from time t-K to time t+τ

↳ To predict the future value $\hat{y}$ of the series i, at time t, for horizon τ, and for quantile q, the TFT uses a function f_q that receives several types of inputs: past covariates, future covariates and static covariates.

. Let's go step by step:

1. $\hat{y}_i(q,t,\tau)$ → final prediction to get
   - i: indicates which series it is (ex. 1, 2, 3, ...)
   - t: the current time point (today)
   - τ: how many steps into the future do we want to predict
   - q: quantile

Quantile (q) → Is a way to measure uncertainty. Instead of predicting a single value, the TFT predicts several possible "levels"
↳ Builds a Confidence Interval
↳ Examples:
- quantile 0,1: Pessimistic prediction → 10% probability that the value will be lower
- quantile 0,5: Median prediction → the most typical prediction
- quantile 0,8: Optimistic prediction → 90% probability that the value will be lower
  - $q(0,1)$: probably the stock will be at least at 80 USD
  - $q(0,5)$: the central prediction is 100 USD stock price for tomorrow
  - $q(0,9)$: probably the stock won't exceed 130 USD tomorrow

2. $f_q(\dots)$ → internal function that produces the prediction for the quantile q
   ↳ Contains the whole TFT architecture
   - variable selector, multi-head attention,
   - static encoder, several gates,
   - LSTM encoder-decoder

3. $\hat{y}_{i,t-k:t}$ → These are the target historical values (what we want to predict). From the t-K time (K steps) to time t (now). } Past Memory

Ex: predict the tomorrow's Close Price with the last 30 days

4. $z_{i,t-k:t}$ → Past covariates. Are input variables known only up to the present moment, only accompany the target variable. These are not the target but they influence it.

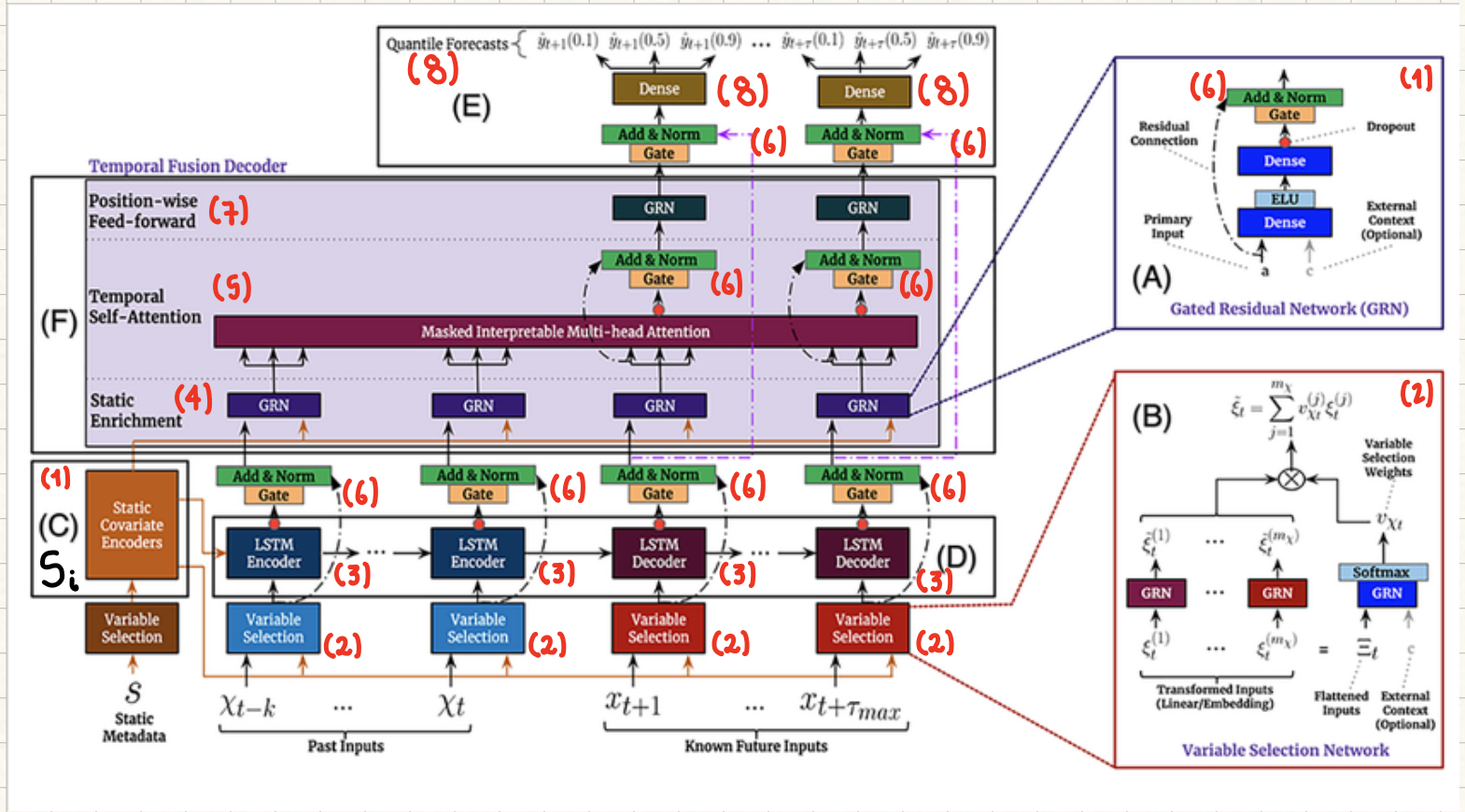
→ In the common ML terminology, these are the features or independent variables.

5. $x_{i,t:t+\tau}$ → Future covariates or known inputs. It includes information that we know in advance about the future → Real information available about the future.

→ For example: forecasting of other sources, calendar days, end-period indicators, holidays

is_holiday, is_month_end, is_trading_day, week_of_the_year, is_quarter_end, ...

6. s_i → Static covariates → Features that doesn't change in the series through time.



. The TFT architecture can be divided in 3 big modules:

1. Input processing and variable selection

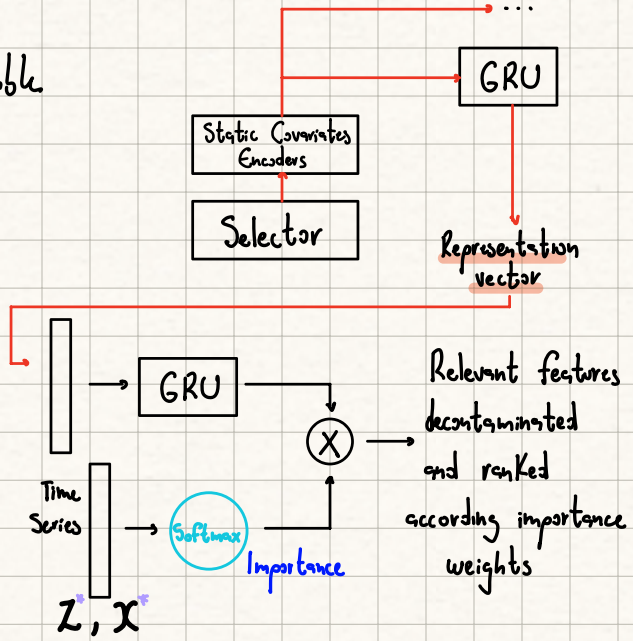
2. Temporal processing with LSTM plus "static enrichment".

3. Fusion via transformer plus prediction (in quantiles).

(1) Static Covariate Encoders (S_i) → The static covariates are passed by the Variable Selector Network. Then they are passed by a **GRN** (Gated Residual Network) that "learns" the useful representation → It produces static context vectors

(2) Corresponds to: $\hat{y}_{i,t-k:t}$, $z^*_{i,t-k:t}$, $x^*_{i,t:t+\tau}$ → The network decides which variables are important in each time step.

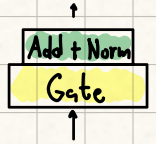
Targets — Past Inputs — Known future inputs



(3) LSTM Encoder & Decoder → Captures the short-term dynamic. The Encoder process the past input (Historical Data) and generates a kind of "summary" internally. In the other hand, the Decoder process the Known future inputs, preparing the representations to be aligned with the future horizon.

(4) The Static Enrichment (enriches the whole sequence using S_i) → the static vectors are combined with the LSTM outputs through GRN, a gate and residual connectors.

(5) Temporal Self-Attention → The transformer is in charge of finding long-term dependencies in the series (past and future covariates). Basically decided when to "pay attention" in order to predict. This process uses masks, allowing to identify seasonal patterns, cycles, loops, atypical events and trends.

(6) 

Each module (GRN, Attention-Transformer, Feed-Forward) is accompanied of 3 blocks (Residual Add + Normalization Layer + Gate) with the aim of stabilizing training, controlling how much of each module is passed and preventing the model from overfitting or becoming too complex when unnecessary.

(7) The Position-wise Feed-Forward is a block (GRN) where same independent transformation are applied in order to prepare the final vector to predict each future horizon.

(8) Dense Outputs (Quantile Predictions) → For each future step ($t+\tau$) quantile predictions are generated: $\hat{y}_{t+\tau}(0,1)$, $\hat{y}_{t+\tau}(0,5)$, $\hat{y}_{t+\tau}(0,9)$ → These are obtained using a fully-connected (dense) layer for each quantile.

. Here is a quick summary: