Forecasting Models

Forecasting aims to create models to predict the value that a variable of interest will take soon.

- Given a set of **ordered records**, we want to predict the value of a variable of interest for the next record;
- Contrary to classification, the values assumed by the target variable are **numeric and potentially infinite - continuous** - in classification, the values are **discrete**;
- Usually, the data to be forecasted is **time-dependent**, and are called **time series**;
- Against classification, the **target** is a **continuous variable**, and not a class - the result information is called **predictor**;
- After training the predictor, we can apply it to predict the value of the target in **future time steps**;
- Considering a function f that maps the **input** to the **output**, and a function $\hat{f}$ that maps the **input** to the **predicted output**;
  - The best estimation of $\hat{f}$ is the one **closest** to f - **minimizes the square error**;
  - $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$;
  - The **MAE (Mean Absolute Error)** is the **absolute value** of the **error** - $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$;
  - The **RMSE (Root Mean Squared Error)** is the **square root** of the **MSE** - $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$;
  - The **MAPE (Mean Absolute Percentage Error)** is the **absolute value** of the **error** divided by the **actual value** - $MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y_i}|}{y_i}$;
  - The **R^2 (Coefficient of Determination)** is the **proportion of the variance** in the **dependent variable** that is **predictable** from the **independent variable(s)** - $R^2 = 1 - \sum_{i=1}^{n}\frac{(y_i - \hat{y_i})^2}{(y_i - \bar{y_i})^2}$, where $\bar{y_i}$ is the **mean** of the **actual values**;

* We want R^2 to be **close to 1**;

There are several **forecasting models**:

- **Simple models**:
  - **Simple average** - predicts the mean of the time series;
    * $x_t = \frac{1}{n}\sum_{i=1}^{n} x_i$;
  - **Persistence model** - predicts the last value of the time series;
    * $x_t = x_{t-1}$;
  - **Rolling mean** - predicts the mean of the last w values of the time series;
    * $x_t = \frac{1}{w}\sum_{i=1}^{w} x_{t-i}$;
- **Regression models**:
  - **Linear regression** - predicts the value of the time series based on a linear combination of the previous values;
    * $x_t = \beta_0 + \beta_1 x_{t-1} + \beta_2 x_{t-2} + ... + \beta_p x_{t-p}$;
  - **ARMA**;
  - **ARIMA**;
  - **SARIMA**;
- **Neural networks - LSTMs**.

ARMA (Autoregressive Moving Average) Models

- ARMA models, aka **Box-Jenkins**, combine **2 models**:
  - **Autoregressive Model (AR)** - the value of the variable at time t is a **linear combination** of the values of the variable at **previous time steps**;
    * $AR(p)$ assumes that our model depends on the last p values of the time series;
    * If $p = 2$, the forecast X_t has the form $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \epsilon_t$;
  - **Moving Average Model (MA)** - the value of the variable at time t is a **linear combination** of the **errors** at **previous time steps**;
    * $MA(q)$ assumes that our model depends on the last q errors of the time series;
    * If $q = 2$, the forecast X_t has the form $X_t = \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \epsilon_t$;

So, the **ARMA** model is a **linear combination** of the **previous values** and the **previous errors**. If $p = q = 2$, the forecast X_t has the form $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \epsilon_t$.

3 Steps to Build an ARMA Model

1. **Identify** the **order** of the model - p and q;
2. **Estimate** the **parameters** of the model - $\phi_1, \phi_2, \theta_1, \theta_2$;
3. **Validate** the **model**.

Things to Keep in Mind

- The time series must be **stationary - constant mean** and **constant variance**;
- Have ate least **100 observations**;
- It is important to confirm whether the time series contains a seasonal component.

ARIMA (Autoregressive Integrated Moving Average) & SARIMA (Seasonal ARIMA) Models

> **Integrated series** are **stationary** series that are **differenced** to become **stationary** - subtracting the value of the previous time step from the current one.

- **ARIMA** models are **ARMA models** applied to **integrated series**, composed of 3 components:
  - AR Model - p;
  - **Integrated Component** - d - the number of times the series was **differenced** to become **stationary**;
  - MA Model - q;
- **SARIMA** models are **ARIMA models** applied to **seasonal series**;
  - Used to remove the **seasonal component** of the series;
  - $SARIMA(p, d, q)(P, D, Q)_s$ - s is the **seasonal period**;
  - P, D, Q are the **order** of the **seasonal ARIMA model**.

> Both require the time series to be **stationary**. If it is not, remove trend, seasonality and apply differencing.

Matrix Profile

> **Matrix Profile** is a **time series data structure** that allows us to **find similar subsequences** in a time series.

The MP has many highly desirable properties:

- **Exact** - the MP methods provide no false positives nor false negatives;
- **Simple and Parameter-Free** - the MP methods have no parameters to tune;
- **Incrementally Maintainable** - the MP methods can be updated in real-time;
- **Free of the Curse of Dimensionality** - the MP methods are not affected by the number of dimensions;
- **Allows Anytime Algorithms** - the MP methods can be stopped at any time and still provide a valid result;
- **It can Leverage Hardware** - the MP methods can be parallelized and distributed.
- **It can be constructed in Deterministic Time** - the MP methods can be constructed in linear time;
- **It can handle Missing Data** - the MP methods can handle missing data.

LSTM (Long Short-Term Memory) Models

> **LSTM** is a **recurrent neural network** that can **learn long-term dependencies**.

- **Recurrent Neural Networks (RNN)** are a class of **neural networks** that allow previous outputs to be used as inputs while having hidden states;
- **LSTM** is a type of **RNN** that can **learn long-term dependencies**;
- ...