Time Series Data Analysis

1. Time Series Data Analysis in Finance

Time Series data analysis in finance involves the study of historical financial data to understand patterns, trends, and relationships, make forecasts and inform investment decisions. Key aspects of time series data analysis in finance include:

1) Stock Price Analysis
2) Volatility Analysis
3) Risk Management
4) Financial Forecasting
5) Econometric Analysis
6) Portfolio Optimization
7) Trading Strategy Development
8) Seasonality Analysis
…

2. Time Series Data Analysis in Financial Crime Risk

In the field of financial crime risk, time series data analysis helps financial institutions and regulatory agencies identify, monitor, and respond to suspicious activities and potential financial crimes.

2.1. Data

Transaction Data

It includes records of financial transactions, such as wire transfers, credit card transactions, cash withdrawals, and fund transfers. They are time dependent.

Customer Data

It includes customer profile data, such as name, date of birth, gender, SSN, address, phone number, etc.; financial information data, such as annual income, employment status, credit score, credit balance, investment holdings, transaction history, etc.; transaction history data, such as account activity, transaction dates and times.

Customer Behavior Data

It includes login times, account access patterns, app usage, interactions with websites.

Market Data

It includes stock prices, exchange rates, interest rates, and commodity prices. Those can be used for detecting insider trading, market manipulation and financial crimes that involve market movements. Time series analysis can uncover abnormal trading patterns.

Trade Data

It includes records of international trade transactions, such as invoices, bills of lading, transaction channel, and customs declarations. Monitoring trade data over time can reveal discrepancies and anomalies.

KYC Data

It includes customer onboarding, including identity verification and know your customer checks, can be analyzed over time to detect discrepancies or suspicious patterns.
…

2.2. Purpose

Transaction Monitoring

By analyzing transaction data in a time series format, it becomes easier to identify unusual patterns, such as sudden spikes in transaction volumes or unexpected changes in transaction behavior.

Anomaly Detection

Statistical process control charts and machine learning algorithms can be applied to identify anomalies or outliers in financial data that may indicate fraudulent or suspicious activities.

Fraud Detection
Using machine learning algorithms can further determine whether the anomalies are fraud or not.

Money Laundering Detection
Patterns of cash flows, international fund transfers, and unusual currency exchanges can be analyzed over time to identify potential money laundering schemes.

Behavioral Analysis
Historical time series data can be established as baseline behavior for customers, accounts, or entities. Deviations from the baselines can raise red flags and prompt further investigation.
…

2.3. Model

Basic Statistical Models

1) Moving Averages
- This smooths out the data and identifies trends or patterns that might be less apparent when looking at the raw, noisy price data.

2) Exponential Smoothing
- Give more weight to recent observations while gradually decreasing the influence of older data.
- This makes it responsive to changes in the data while providing a smoothed forecast that reduces noise and short-term fluctuations.

3) Z-score
- Calculate z-scores can identify transactions or patterns that fall outside a specified threshold.
…

Machine Learning Models

1) Logistic Model
- A binary classification model to identify fraudulent transactions.

2) Decision Tree
- It identifies decision rules that may indicate the fraudulent or suspicious activities, classify customers into different risk categories.

3) Random Forest/XGBoost
- A enhanced tree model which improves predictive accuracy and reduce overfitting.

4) Support Vector Machine
- It separates high-dimensional data into distinct classes. It can detect fraudulent transactions, identifies deviations from established behavior patterns, abnormal market movements, etc.

5) Neural Network
- It is a deep learning model that learn intricate patterns and capture non-linear relationship in time series data.

6) Natural Language Processing
- It analyzes textual trade data for discrepancies and inconsistencies. It can also analyze news and social media data for sentiment and news related to insider trading.

7) Cluster Analysis
- It can cluster customers based on their behavior patterns over time.
…

Time Series Models

1) Autoregressive Model (“AR”)
- It is suitable for modeling time series data with autocorrelation, where the current value of a variable depends on its past values.

2) Moving Average Model (“MA”)
- It is suitable for modeling time series data with moving averages or trends.

3) Autoregressive Moving Average Model (“ARMA”)
- It is a combination of AR and MA, which is suitable for modeling time series data with both autocorrelation and trends.

4) Autoregressive Integrated Moving Average Model ("ARIMA")
- It introduces the integration (differencing) component, which is suitable for non-stationary time series data by differencing it to achieve stationarity. It performs well when data exhibit seasonality or non-stationarity. The choice between ARMA and ARIMA depends on the characteristics of the time series data being analyzed, and whether or not differencing is required to make the data suitable for modeling.

5) Seasonal Decomposition of Time Series
- It decomposes a time series into seasonal, trend, and residual components. It is effective when data exhibit seasonality and complex patterns.
…

3. Time Series Data Analysis in General

3.1. Time Series Data Overview

-We cannot shuffle data, since the chronological order is important in time series data
-The reason why we try to forecast time series data is because we think that the patterns observed in the past are expected to persist in the future.

3.2. Data Frame

- Date: use as index
- QQ plot: quantile quantile, standard deviation/whether normally distributed

3.3. Data Preparation

1. Transform string inputs into data time values
2. Use data as an index
3. Set the frequency (daily, monthly, quarterly, etc.)
4. Fill the missing value
5. Add/remove columns in a data frame
6. Update the data (yfinance)

3.4. Work with Time Series Data

-**White Noise**: does not follow a pattern, (past → future nope), constant mean, constant variance, no autocorrelation in any period
-**Random Walk**: move from one point to another in a series of random steps. No trend/pattern.
RW has a cumulative effect that results in a trend or direction over time, albeit with short-term randomness.
-**Stationarity**:
Dickey-Fuller test
-**Seasonality**:
Decomposition
-Autocorrelation:
ACF (accumulated effect)
PACF (direct effect)

3.5. Data Modeling

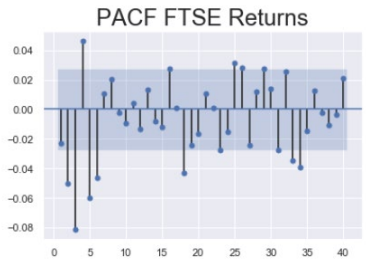
0) How to measure the performances of models?
- ACF to determine the q component (MA component), PACF to determine the p component (AR component)
- LLR test to determine if one model or hypothesis provides a significantly better fit to the data compared to another model. We want higher LLR.
- We want lower information criteria.
- Lower AIC and BIC values indicate better models.
- Look at p values.

1) Auto Regression model (AR model)

$x_t = c + fi_t\, x_{t-1} + e_t$

$|fi_t| < 1$

AR usually performs poorly of predicting the non-stationary data. We may want to transform the prices to returns for stationarity purpose.



-Negatively/positively correlated
-As time goes by, less correlations

-Normalization does not affect stationarity (S&P prices, NIKKI prices), using normalized values won't affect the AR model selection

2) Moving Average model (MA model)
$x_t = c + \text{theta}\ e_{t-1} + e_t$ e is residuals
$|\text{theta}_t| < 1$

-MA model performs poorly in non-stationary data.
-If ACF for prices is all close to 1, infinite MA model.

$x_t = c + \text{theta}\ e_{t-1} + e_t$
$x_{t-1} = c + \text{theta}\ e_{t-2} + e_{t-1}$
assume theta ~ 1 (because the lag is approximately =1 to the current)
$x_t = x_{t-1} - e_{t-2} + e_t$

-Thus, AR component will be introduced, combination of AR and MR are better.

3) Auto Regressive Moving Average model (ARMA model)
$r_t = c + fi_1\ r_{t-1} + theta_1\ e_{t-1} + e_t$

4) Auto Regressive Integrated Moving Average model (ARIMA model)
$\text{delta}\ P_t = c + fi_1\ \text{delta}\ P_{t-1} + theta_1\ e_{t-1} + e_t$

5) Auto Regressive Integrated Moving Average with eXogenous variables model (ARIMAX model)
$\text{delta Pt} = c + \text{beta X} + fi_1\ \text{delta}\ P_{t-1} + theta_1\ e_{t-1} + e_t$
* X can be any variable we are interested in, including a time-varying measurement, a categorical variable, a Boolean value, a combination of external factors, etc.

6) Seasonality Auto Regressive Integrated Moving Average with eXogenous variables model (SARIMAX model)

4. Times Series Modeling using Python

4.1. Python General

-Save the file in the same directory as your python code. So you can do df=pd.read.csv("Index2018.csv")

6.2 Python library

Sklearn, Matplotlib, Seaborn, Arch

6.2 Python Syntax

#Know the count, mean, std, min, IQR, max
df.comp.describe()
df.comp.date.describe()

#False, false, etc.
df_comp.isna()
df_comp.spx.isna().sum()

```
df.comp.spx.plot()
plot.show()

#Frequency
df_comp = df_comp.asfreq('d') #daily

#Missing values
df_comp.isna().sum()
df_comp.spx = df_comp.spx.fillna(method='ffill') #front filling, back filling, note usually filling data with a mean is a bad approach because there is time variant pattern underlying the data

#Add/remove columns
df_comp ['market_value'] = df_comp.spx
del df_comp ['spx']

#Split data, in time series, cannot shuffle, chronological order
size = int (len(df_comp)*0.8)
df = df_comp.iloc[:size]
df_test=df_comp.iloc[size:]
df.tail()

#Auto Regression Model AR(1)
model_ret_ar_1 = ARMA (df.returns, order = (1,0))
results_ret_ar_1 = model_ret_ar_1.fit()
results_ret_ar_1.summary()

#Moving Average Model MA(1)
model_ret_ma_1 = ARMA (df.returns[1:], order = (0,1))
results_ret_ma_1 = model_ret_ma_1.fit()
results_ret_ma_1.summary()

#Auto Regression Moving Average Model ARMA(1)  #note: (1,1) means AR component = 1, MA component = 1
model_ret_ar_1_ma_1 = ARMA (df.returns[1:], order = (1,1))
results_ret_ar_1_ma_1 = model_ret_ar_1_ma_1.fit()
results_ret_ar_1_ma_1.summary()

#Auto Regression Integrated Moving Average Model ARIMA(1)  #note: (1,1,1) means AR component = 1, d=1, MA component = 1
model_ar_1_i_1_ma_1 = ARIMA (df.market_value, order = (1,1,1))
results_ar_1_i_1_ma_1 = model_ar_1_i_1_ma_1.fit()
results_ar_1_i_1_ma_1.summary()

#Auto Regression Integrated Moving Average eXogenous Model ARIMAX(1)
model_ar_1_i_1_ma_1_Xspx = ARIMA (df.market_value, exog = df.spx, order = (1,1,1))
results_ar_1_i_1_ma_1_Xspx = model_ar_1_i_1_ma_1_Xspx.fit()
results_ar_1_i_1_ma_1_Xspx.summary()

#Seaonal Auto Regression Integrated Moving Average eXogenous Model SARIMAX(1)
model_sarimax = SARIMA (df.market_value, exog = df.spx, order = (1,0,1), seasonal_order = (2,0,1,5))
results_sarimax = model_sarimax.fit()
results_sarimax.summary()
```