Assessment

The purpose of the project is to assess your scientific process, IT skills, attention to detail, creativity

and communication skills. We are not necessarily expecting a trading system, although it certainly

should be your goal.

Scientific Method

We want to see you followed a well-defined arc of thought from start to end, regardless of the

outcome. This means clearly defining why you followed a certain path, what your hypothesis was,

your method, results and conclusions.

This should be like a presentation you would do for any scientific conference and

demonstrate your professionalism and effectiveness in communicating your work.

Data

Attached you will find a .csv file with two financial time series spanning 1-Jan-2008 to 1-Jan-2013.

The columns of the .csv file are of the form;

time, ts1, ts2

Some notes on this dataset;

● It’s not important what these time series represent although they are financial markets

● time is expressed using Matlab’s date format

● ts1 refers to time series 1

● ts2 refers to time series 2

● NaN indicates missing data

Remember, you can make money in markets that go up or down by going long or short respectively.

To get you started

Clean Data

The most important aspect of studying any data is making sure it is “clean”. We would like evidence

of how you have done this.

Log Return Space

Some statistical properties of financial markets are better analysed in log return space (r). This is

done by applying the logartihmic transform to the price (p) time series;

Some suggested lines of thinking

● Can ts1 predict ts1?

● Can ts2 predict ts2?

● Can ts1 predict ts2?

● Can ts2 predict ts1?

1. Data Preparation

a. Read the provided .csv file and load the data into a suitable data structure (e.g., pandas DataFrame)

b. Convert time from Matlab's date format to a standard date format (e.g., datetime)

c. Handle missing data (NaN values) by:

i. Identifying the extent of missing data in each time series

ii. Deciding on an appropriate imputation method (e.g., forward fill, interpolation, or dropping missing data)

1. Data Exploration

a. Visualize the time series data (ts1 and ts2) to understand their behavior over time

b. Calculate summary statistics (mean, median, standard deviation, etc.) for each time series c. Investigate the correlation between ts1 and ts2

1. Data Transformation a. Calculate log returns (r) for each time series using the formula: r = log(p(t)/p(t-1)) b. Visualize the log returns for ts1 and ts2
2. Hypothesis Formulation

a. Formulate hypotheses based on the suggested lines of thinking:

i. H1: ts1 can predict ts1

ii. H2: ts2 can predict ts2

iii. H3: ts1 can predict ts2

iv. H4: ts2 can predict ts1

1. Model Selection and Evaluation a. Identify suitable quantitative models for time series analysis (e.g., ARIMA, GARCH, VAR,
2. Machine Learning models like LSTM, etc.)

b. Split the data into training and testing sets (e.g., 80% training, 20% testing)

c. For each hypothesis, apply the selected models and evaluate their performance using relevant metrics (e.g., Mean Squared Error, Mean Absolute Error, etc.)

* 1. Evaluate models for H1: ts1 predicting ts1
  2. Evaluate models for H2: ts2 predicting ts2
  3. Evaluate models for H3: ts1 predicting ts2
  4. Evaluate models for H4: ts2 predicting ts1

d. Perform cross-validation and model selection to choose the best model for each hypothesis

e. Investigate the residuals of the best models for each hypothesis to ensure their validity

1. Model Interpretation and Conclusion
   * 1. Analyze the results of the best models for each hypothesis
     2. Draw conclusions based on the model performance and the significance of their predictive power
2. **Data preparation and visualization:**

* Handling missing values and outliers appropriately
* Plotting the data and observing the patterns and trends (e.g. seasonality, cyclicity, trend)
* Transforming the data into a stationary series if needed (e.g. differencing, log transformation)
* Plotting: raw stock prices, returns, histograms of returns, and boxplots

1. **Statistical analysis:**

* Calculating descriptive statistics and summary measures (e.g. mean, median, standard deviation, skewness, kurtosis)
* Performing correlation analysis and testing for autocorrelation and partial autocorrelation (e.g. ACF, PACF plots)
* Calculating and plotting autocorrelation and partial autocorrelation functions to measure the linear dependence of a time series with its own lagged values (e.g. acf and pacf functions in Python)
* Testing for stationarity using statistical tests (e.g. Augmented Dickey-Fuller test) or visual methods (e.g. rolling mean and standard deviation)
* Stationarity: testing for stationarity using statistical tests such as the Augmented Dickey-Fuller test

1. **Decomposition and periodicity analysis:**

* Decomposing the time series into trend, seasonal and residual components using additive or multiplicative models (e.g. seasonal\_decompose function in Python)
* Seasonal subseries plots to examine the patterns and trends within each season
* Spectral analysis to decompose the time series into its frequency components using techniques such as Fourier analysis or wavelet analysis to identify any periodicity or cycles
* Trend analysis: linear regression analysis, moving average analysis, Hodrick-Prescott filter
* Seasonality analysis: autocorrelation analysis, seasonal decomposition, spectral analysis
* Cyclicity analysis: autocorrelation analysis, spectral analysis

1. **Residual analysis:**

* Serial correlation analysis: autocorrelation function (ACF), partial autocorrelation function (PACF)
* White noise analysis: Ljung-Box test, Breusch-Godfrey test

**Modelling**

1. Feature Engineering:
   1. Select the relevant features for the model, such as past prices, trading volumes, technical indicators, etc.
   2. Generate lagged features, such as the stock price from the previous day or week, to capture time dependencies.
2. Model Selection and Hyperparameter Tuning:
   1. Select a set of candidate models that are appropriate for time series forecasting, such as ARIMA, LSTM, or Prophet.
   2. For each candidate model, train the model using the training set and perform hyperparameter tuning (e.g., using grid search, random search, or Bayesian optimization) to find the optimal set of parameters.
   3. a. Seasonal Autoregressive Integrated Moving Average (SARIMA): A model that captures both linear and seasonal dependencies in a time series
      1. i. Use a grid search over a range of possible parameter values and choose the model with the lowest AIC or BIC value
   4. b. Exponential Smoothing: A model that assigns weights to past observations that decay exponentially over time
      1. i. Test variants such as Holt's linear exponential smoothing and Holt-Winters' exponential smoothing
   5. c. Long Short-Term Memory (LSTM): A type of recurrent neural network that can learn long-term dependencies in a time series
      1. i. Specify the appropriate architecture and hyperparameters, and use techniques such as dropout or early stopping to prevent overfitting
3. Model Evaluation on the Test Set:
   1. Forecast the test set using the trained models with their optimal hyperparameters.
   2. Calculate evaluation metrics for each model, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
   3. Compare the performance of each model using the evaluation metrics.
4. Cross-Validation (Optional):
   1. Perform time series cross-validation (e.g., using a rolling window or expanding window approach) to assess the model's performance more robustly.
   2. Calculate the evaluation metrics for each fold and calculate their mean and standard deviation for each model.
5. Residual Analysis and Diagnostics:
   1. Analyze the residuals of the models (e.g., check for autocorrelation, normality, and homoscedasticity).
   2. Perform diagnostic tests, such as the Ljung-Box test for autocorrelation in residuals, and assess the results.
6. Fine-tuning and retraining the best model: a) Choose the best model based on the evaluation metrics and diagnostic tests. b) Fine-tune the hyperparameters of the best model further if needed.
7. Communicate the results and insights gained from the chosen model, including its strengths, limitations, and potential improvements for future analysis.

Steps:

Decomposition and periodicity analysis aim to identify and evaluate periodic patterns or cycles in time series data. Here's an outline for conducting decomposition and periodicity analysis and evaluating the results:

1. Time series decomposition: a. Apply an additive or multiplicative decomposition method (e.g., seasonal\_decompose function in Python) to the time series data. b. Examine the resulting trend, seasonal, and residual components. c. Visualize the components and assess the presence of periodic patterns or cycles.
2. Autocorrelation Function (ACF) analysis: a. Calculate the ACF for the stationary transformed data. b. Plot the ACF. c. Look for significant peaks at regular intervals, which may indicate periodic patterns.
3. Spectral analysis: a. Apply Fourier Transform or Wavelet Transform to the time series data. b. Analyze the resulting power spectral density or scalogram to identify dominant frequencies. c. Identify significant frequency components corresponding to periodic patterns or cycles.
4. Periodogram analysis: a. Calculate the periodogram of the stationary transformed data. b. Analyze the periodogram to identify dominant frequencies or cycles. c. Look for significant peaks corresponding to periodic patterns.
5. Seasonal subseries plots: a. Create subseries plots for each season or cycle in the data. b. Examine the patterns and trends within each season or cycle. c. Assess the consistency and strength of the periodic patterns.
6. Evaluation: a. Compare the results from time series decomposition, ACF, spectral analysis, periodogram analysis, and seasonal subseries plots. b. Identify any consistent periodic patterns or cycles across the different methods. c. Assess the strength and significance of the identified periodic patterns.

By combining the results of these techniques, you can identify and evaluate periodic patterns or cycles in your time series data. Note that the presence of strong periodic patterns may suggest the use of models that can capture seasonality or cyclical components, such as SARIMA, exponential smoothing state space models, or Fourier-transform-based regression models. If significant non-seasonal cycles are detected, you might also consider using models that can capture cyclical components like Bayesian structural time series models or incorporating cyclical features into machine learning models like LSTM or LightGBM.

Trend

1. Visual inspection of the time series data: a. Plot the raw time series data. b. Look for any noticeable trends, such as increasing, decreasing, or more complex patterns.
2. Moving average analysis: a. Calculate the simple moving average (SMA) or exponential moving average (EMA) of the time series data with various window sizes. b. Plot the moving averages alongside the raw time series data to visually identify any trends.
3. Linear regression analysis: a. Perform a linear regression on the time series data with time as the independent variable. b. Assess the slope and intercept of the fitted line, as well as the statistical significance of the regression coefficients. c. Plot the fitted line over the raw time series data to visualize the linear trend.
4. Non-linear regression analysis (optional): a. If a non-linear trend is suspected, fit a non-linear regression model (e.g., quadratic, exponential) to the time series data. b. Assess the coefficients and goodness-of-fit of the non-linear regression model. c. Plot the fitted curve over the raw time series data to visualize the non-linear trend.
5. Hodrick-Prescott filter (optional): a. Apply the Hodrick-Prescott filter to the time series data to separate the trend component from the cyclical component. b. Plot the trend component alongside the raw time series data to visualize the trend.
6. Evaluation of trend analysis: a. Compare the visualizations and results of the different trend analysis methods (e.g., moving average, linear regression, non-linear regression, Hodrick-Prescott

seasonality analysis and evaluating the results to identify seasonality in time series data:

1. Visual inspection: a. Plot the raw time series data and look for any repeating patterns or regular fluctuations. b. Identify possible seasonal patterns by observing the data at different time scales (e.g., daily, monthly, or yearly).
2. Time series decomposition: a. Decompose the time series into its trend, seasonal, and residual components using additive or multiplicative models (e.g., using the seasonal\_decompose function in Python). b. Visualize the seasonal component and assess its regularity and strength.
3. Autocorrelation function (ACF) analysis: a. Calculate and plot the ACF of the time series data. b. Identify significant autocorrelations at seasonal lags (e.g., lags that are multiples of the suspected seasonal period).
4. Seasonal subseries plots: a. Create subseries for each season by dividing the data into segments corresponding to the suspected seasonal period. b. Plot the seasonal subseries to visualize patterns and trends within each season.
5. Box plots: a. Create box plots for each season to visualize the distribution of the data within each seasonal period. b. Compare the box plots to identify any systematic differences between the seasonal periods.
6. Spectral analysis (optional):
   1. Perform a spectral analysis or Fourier analysis to decompose the time series into its frequency components. b. Identify dominant frequencies that correspond to the suspected seasonal period.
7. Evaluation of seasonality analysis:
   1. Assess the strength and regularity of the seasonal component based on the visualizations and statistical analyses (e.g., ACF, seasonal subseries plots, box plots).
   2. Determine if seasonality is present in the data by examining the consistency of patterns across different time scales and the statistical significance of seasonal autocorrelations.
   3. If seasonality is detected, incorporate the seasonal component into your time series model (e.g., SARIMA, Holt-Winters exponential smoothing).

By following this outline, you will be able to perform a comprehensive seasonality analysis on time series data and evaluate the results to identify the presence of seasonality. This information can then be used to inform the choice of an appropriate time series model that accounts for the seasonal component.

Cyclicity analysis aims to identify recurring patterns in time series data that are not necessarily seasonal. Here's an outline for conducting cyclicity analysis and evaluating the results:

1. Visual inspection of the time series plot: a. Plot the original time series data. b. Observe any recurring patterns that are not strictly seasonal.
2. Autocorrelation Function (ACF) analysis: a. Calculate the ACF for the stationary transformed data. b. Plot the ACF. c. Look for significant peaks at non-seasonal lags, which may indicate cyclical patterns.
3. Spectral analysis: a. Apply Fourier Transform or Wavelet Transform to the time series data. b. Analyze the resulting power spectral density or scalogram to identify dominant frequencies. c. Identify any significant frequency components that do not correspond to the seasonal frequency, indicating potential cyclical patterns.
4. Periodogram analysis: a. Calculate the periodogram of the stationary transformed data. b. Analyze the periodogram to identify dominant frequencies or cycles. c. Look for significant peaks corresponding to non-seasonal cycles.
5. Evaluation: a. Compare the results from the ACF, spectral analysis, and periodogram analysis. b. Identify any consistent patterns or cycles across the different methods. c. Assess the strength and significance of the identified cycles.

Discussion:

1. Stationarity:
   * SARIMA, ARIMA, VAR, and GARCH: Stationarity is an important assumption for these models. If a time series is non-stationary, it should be transformed (e.g., differencing) before fitting these models.
   * LSTM, LightGBM: These models can handle non-stationary data better than traditional models, but transforming the data can still improve their performance.
   * Exponential Smoothing: Requires the data to be stationary, but some variants (e.g., Holt-Winters) can handle trends and seasonality.
2. Trend:
   * SARIMA, ARIMA: These models can handle linear trends by differencing or including a drift term.
   * GARCH: Primarily for modeling volatility, not suitable for modeling trends.
   * VAR: Can model linear trends by including a deterministic term.
   * LSTM, LightGBM: Can capture non-linear trends due to their flexible structures.
   * Exponential Smoothing: Holt's linear method can model linear trends, while other variants can handle non-linear trends.
3. Seasonal:
   * SARIMA: Explicitly models seasonal components.
   * ARIMA, GARCH, VAR: Cannot model seasonal components directly, but seasonal differencing or including seasonal dummy variables can help.
   * LSTM, LightGBM: Capable of modeling seasonal patterns due toheir flexible structures.

* Exponential Smoothing: Holt-Winters method models seasonal components.

1. Cyclical:
   * SARIMA, ARIMA, VAR: Can model cyclical components to some extent but may require additional transformations or exogenous variables.
   * GARCH: Not suitable for modeling cyclical components directly.
   * LSTM, LightGBM: Capable of capturing cyclical patterns due to their flexible structures.
   * Exponential Smoothing: Not well-suited for modeling cyclical components directly.
2. Irregular:
   * SARIMA, ARIMA, GARCH, VAR: Assumes a certain structure in the residuals and may not perform well with highly irregular data.
   * LSTM, LightGBM: More robust to irregular data due to their flexible structures.
   * Exponential Smoothing: Assumes a certain structure in the data and may not perform well with highly irregular data.
3. Noise:
   * SARIMA, ARIMA, GARCH, VAR: Assumes a certain structure in the noise and may not perform well with highly noisy data.
   * LSTM, LightGBM: More robust to noise due to their flexible structures.
   * Exponential Smoothing: Can handle noise to some extent, depending on the chosen variant.
4. ACF:
   * SARIMA, ARIMA, VAR: ACF can help identify the autoregressive (AR) and moving average (MA) components of the models.
   * GARCH: ACFcan help identify the autoregressive (AR) and moving average (MA) components for the conditional variance equation.

* LSTM, LightGBM: ACF might not be as informative for these models, but it can still provide insights into the underlying data structure.
* Exponential Smoothing: ACF can help identify the level of autocorrelation in the data, which can inform the choice of the smoothing parameters.

1. PCF:
   * SARIMA, ARIMA, VAR: PCF can help identify the autoregressive (AR) components of the models.
   * GARCH: PCF might not be as informative for this model, but it can still provide insights into the underlying data structure.
   * LSTM, LightGBM: PCF might not be as informative for these models, but it can still provide insights into the underlying data structure.
   * Exponential Smoothing: PCF can help identify the level of partial autocorrelation in the data, which can inform the choice of the smoothing parameters.

Heteroskedasticity

* If you observe volatility clustering or heteroskedasticity in the data, it might be appropriate to consider models that account for these characteristics. For instance, GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models are designed to capture the time-varying volatility observed in many financial time series data, such as stock prices.
* In such cases, you can incorporate GARCH models or other models that account for conditional heteroskedasticity during the modeling phase of your approach. This would involve selecting a GARCH model, determining its order (p, q), and estimating the parameters using maximum likelihood estimation.
* Additionally, you may consider using a combination of models, such as an ARIMA model for the mean process and a GARCH model for the volatility process, to better capture both the linear dependence and the time-varying volatility in the stock price data.
* By accounting for volatility clustering and heteroskedasticity, you can improve the accuracy and reliability of your time series forecasts, as the selected models will be better suited to capture the underlying characteristics of the stock price data.

In summary, the choice of a model depends on the characteristics of the data revealed through exploratory data analysis. Traditional models like SARIMA, ARIMA, GARCH, and VAR require certain assumptions about the data, while more flexible models like LSTM and LightGBM can handle a wider range of data structures. Exponential Smoothing is a simple yet effective model for certain cases, especially when trends or seasonality are present.

**Analysis:**

1. Data preparation: a. Split the time series data into a training set and a testing set, typically using a specific ratio (e.g., 80% training, 20% testing) or a specific time point.
2. Model selection: a. Choose a set of candidate models for evaluation, such as ARIMA, SARIMA, GARCH, VAR, LSTM, LightGBM, and Exponential Smoothing.
3. Model training and hyperparameter tuning:
   1. For each candidate model, train the model using the training set and perform hyperparameter tuning (e.g., using grid search, random search, or Bayesian optimization) to find the optimal set of parameters.
4. Model evaluation on the test set:
   1. Forecast the test set using the trained models with their optimal hyperparameters.
   2. Calculate evaluation metrics for each model, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
   3. Compare the performance of each model using the evaluation metrics.
5. Cross-validation (optional):
   1. Perform time series cross-validation (e.g., using a rolling window or expanding window approach) to assess the model's performance more robustly.
   2. Calculate the evaluation metrics for each fold and calculate their mean and standard deviation for each model.
6. Residual analysis and diagnostics:
   1. Analyze the residuals of the models (e.g., check for autocorrelation, normality, and homoscedasticity).
   2. Perform diagnostic tests, such as the Ljung-Box test for autocorrelation in residuals, and assess the results.
7. Model selection based on performance and diagnostics:
   1. Review the evaluation metrics, cross-validation results (if performed), and diagnostic tests for each candidate model.
   2. Consider trade-offs between model complexity, interpretability, and performance.
   3. Select the best model based on a combination of performance metrics, robustness, and model diagnostics.
8. Model validation and confidence intervals (optional):
   1. Validate the selected model using out-of-sample data or by performing a hold-out validation.
   2. Calculate prediction intervals (e.g., 95% confidence intervals) for the model's forecasts to quantify the uncertainty associated with the predictions.

By following this outline, you can systematically test and evaluate different models to determine the best model for forecasting your time series data. Remember that the choice of the best model depends not only on its performance but also on its robustness, interpretability, and ability to meet the assumptions and requirements of the specific problem at hand.

hyperparameter tuning methods for each of the mentioned models:

1. ARIMA (Autoregressive Integrated Moving Average):
   * p: Order of the autoregressive (AR) component
   * d: Degree of differencing
   * q: Order of the moving average (MA) component
2. SARIMA (Seasonal Autoregressive Integrated Moving Average):
   * p: Order of the autoregressive (AR) component
   * d: Degree of differencing
   * q: Order of the moving average (MA) component
   * P: Order of the seasonal autoregressive component
   * D: Degree of seasonal differencing
   * Q: Order of the seasonal moving average component
   * m: Seasonal period
3. GARCH (Generalized Autoregressive Conditional Heteroskedasticity):
   * p: Order of the GARCH component
   * q: Order of the ARCH component
   * Distribution: Choice of distribution (e.g., Gaussian, Student's t)
4. VAR (Vector Autoregression):
   * p: Number of lags to include in the model
5. LSTM (Long Short-Term Memory):
   * Number of hidden layers
   * Number of hidden units per layer
   * Learning rate
   * Dropout rate
   * Activation function (e.g., ReLU, Tanh)
   * Batch size
   * Number of training epochs
6. LightGBM (Light Gradient Boosting Machine):
   * Number of boosting rounds (n\_estimators)
   * Learning rate
   * Max depth of trees
   * Number of leaves (num\_leaves)
   * Minimum data in leaf (min\_data\_in\_leaf)
   * Feature fraction (feature\_fraction)
   * Bagging fraction (bagging\_fraction)
   * Bagging frequency (bagging\_freq)
   * L1 regularization (lambda\_l1)
   * L2 regularization (lambda\_l2)
   * Objective function (e.g., regression, classification)
7. Exponential Smoothing: For Simple Exponential Smoothing:
   * Alpha: Smoothing factor for the level component (0 < alpha < 1)

For Holt's Linear Exponential Smoothing:

* + Alpha: Smoothing factor for the level component (0 < alpha < 1)
  + Beta: Smoothing factor for the trend component (0 < beta < 1)

For Holt-Winters' Seasonal Exponential Smoothing:

* + Alpha: Smoothing factor for the level component (0 < alpha < 1)
  + Beta: Smoothing factor for the trend component (0 < beta < 1)
  + Gamma: Smoothing factor for the seasonal component (0 < gamma < 1)
  + Seasonal periods (e.g., 12 for monthly data with yearly seasonality)

For each model, hyperparameter tuning can be performed using methods such as grid search, random search, or more advanced optimization techniques like Bayesian optimization. The goal is to find the optimal combination of hyperparameters that yield the best performance on the given time series data, typically measured using metrics such as mean squared error (MSE),

1. split\_train\_test(data, test\_size): This function should split the given data into training and testing sets based on the specified test size, and return both the sets.

Functions for Evaluating the Model: 2. evaluate\_model(model, X\_train, y\_train, X\_test, y\_test, metrics): This function should train the given model using the provided training data and evaluate its performance using the specified evaluation metrics on the testing data. It should return the evaluation results.

Functions for Cross-Validation and Model Selection: 3. cross\_validate\_model(model, X, y, cv): This function should perform cross-validation on the given model using the provided data and return the evaluation results.

1. select\_best\_model(models, eval\_results): This function should select the best model from a list of models based on their evaluation results.

Functions for Investigating Residuals: 5. plot\_residuals(model, X\_train, y\_train, X\_test, y\_test): This function should plot the residuals of the given model using the provided training and testing data.

Functions for Forecasting: 6. forecast(model, X): This function should use the provided model to forecast the values of the provided data and return the forecasted values.

Functions for Hypothesis Testing: 7. evaluate\_hypothesis(data, hypothesis, metrics): This function should apply the selected models for the specified hypothesis and evaluate their performance using the specified evaluation metrics on the testing data. It should return the evaluation results.

Overall Framework:

1. Split the data into training and testing sets.
2. For each hypothesis, apply the selected models and evaluate their performance using relevant metrics.
3. Perform cross-validation and model selection to choose the best model for each hypothesis.
4. Investigate the residuals of the best models for each hypothesis to ensure their validity.
5. Forecast using the selected models.
6. Communicate the results and insights gained from the chosen model, including its strengths, limitations, and potential improvements for future analysis.

Non normal data models

1. Robust statistical methods: These methods are designed to be less sensitive to outliers and non-normality in the data. Examples include robust regression, robust estimators of location and scale, and robust time series models like the autoregressive conditional duration (ACD) model.
2. Copula-based models: Copulas are used to model the dependence structure between multiple variables without making assumptions about their marginal distributions. They can be particularly useful when dealing with non-normal financial data, such as modeling the joint distribution of stock returns and other financial variables.
3. Regime-switching models: These models allow for changes in the underlying data-generating process, which can be useful when dealing with non-normal data that exhibits different behaviors in different periods. Examples include Markov-switching models and hidden Markov models.
4. Wavelet-based methods: Wavelet analysis can be used to decompose a time series into different frequency components, which can help capture non-normal features in the data. Wavelet-based methods can be applied to various tasks, such as denoising, forecasting, and volatility estimation.
5. Bayesian methods: Bayesian techniques can be used to model non-normal data by specifying flexible prior distributions and updating them with observed data. Examples include Bayesian hierarchical models, Bayesian state-space models, and Bayesian non-parametric methods like Gaussian process regression.