**Task 3**

1. **Knowledge Discovery in Databases (KDD) Process**

**The Knowledge Discovery in Databases (KDD) process refers to the overall framework for discovering useful knowledge from data. It includes multiple stages, starting from raw data handling to extracting meaningful patterns. The main stages of KDD are:**

1. **Data Selection: Choosing relevant data from multiple sources.**
2. **Data Preprocessing: Cleaning the data by handling missing values, noise, and inconsistencies to ensure data quality.**
3. **Data Transformation: Normalizing, aggregating, and constructing features suitable for modeling.**
4. **Data Mining: Applying machine learning or statistical techniques to extract meaningful patterns.**
5. **Interpretation/Evaluation: Making sense of the discovered patterns, validating findings, and translating them into actionable insights.**

**In this project, we followed the KDD process by carefully selecting, cleaning, transforming, and analyzing Bitcoin historical data to prepare it for accurate predictive modeling.**

## Missing Value Handling Strategy

**In our Bitcoin price prediction project, we observed significant missing values in the btc\_trade\_volume attribute. After a detailed gap analysis, we identified seven major gaps, ranging from small (3 days) to extremely large (367 days). Based on this, we designed a two-step missing value handling strategy that respects the time-series nature of the data:**

**First, we applied a 22-day moving average to the btc\_trade\_volume attribute to interpolate missing values whenever sufficient historical data was available. By setting the window to 22 days, we ensured that missing periods of up to approximately 3 weeks could be realistically smoothed based on local temporal behavior. This approach preserves short-term trends and volatility patterns critical for time-series forecasting.**

**Second, for longer or consecutive gaps where moving averages could not be calculated (especially gaps exceeding 3 weeks or where too much data was missing), we used a forward-fill technique. Forward-fill logically propagates the last known valid value forward, maintaining temporal continuity without introducing unrealistic artificial smoothing.**

**This combined strategy — moving average for small gaps and forward fill for large gaps — allowed us to balance realism and completeness in the dataset. It ensures that all missing values are handled effectively while minimizing distortion of Bitcoin's dynamic behavior over time.**

## Normalization of Selected Features

**Following the completion of missing value handling, we performed feature normalization to prepare the data for modeling. We applied the Z-transformation (statistical normalization) method to the selected predictor attributes. This normalization technique centers each attribute around a mean of zero and scales to a unit variance, ensuring consistent feature scaling across the dataset.**

**The following attributes were normalized:**

* **btc\_trade\_volume**
* **btc\_n\_transactions**
* **btc\_estimated\_transaction\_volume\_usd**
* **btc\_difficulty**
* **btc\_hash\_rate**
* **btc\_miner\_revenue**
* **btc\_transaction\_fees**
* **btc\_total\_bitcoins**
* **btc\_market\_cap**

**The target variable btc\_market\_price was preserved in its original scale to maintain interpretability during prediction. Z-transformation normalization was chosen to make the dataset more robust against the influence of outliers while maintaining the integrity of underlying distributions.**

## Visualization and Outlier Analysis

**After normalization, we visually inspected the dataset to detect the presence of outliers and analyze the distribution of features. Three types of visualizations were generated:**

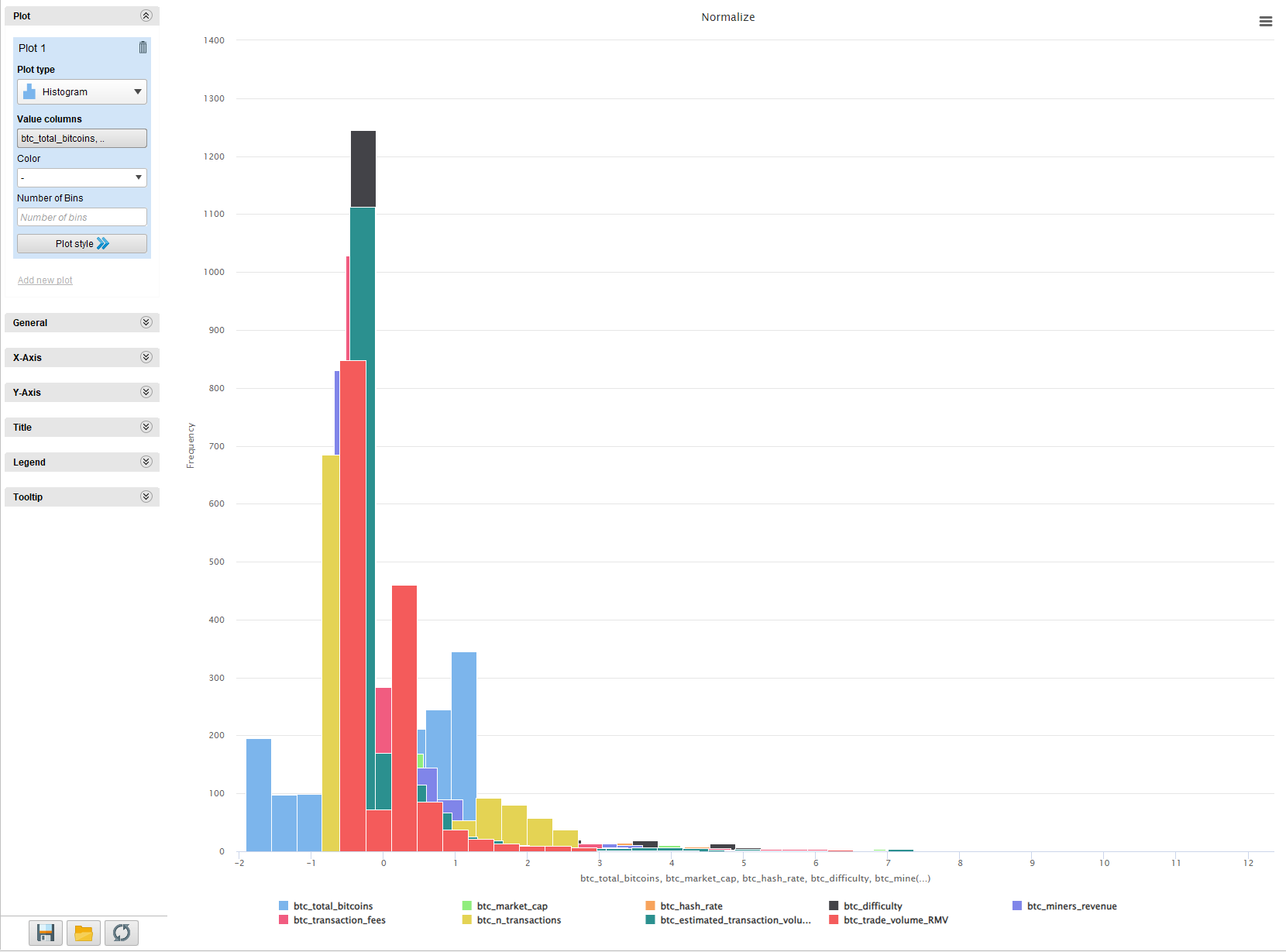
* **Histograms: Showed that most features are centered around zero as expected after Z-transformation, with some attributes exhibiting right-skewed distributions.**
* **Scatter Plots: Illustrated feature behavior over time, highlighting increasing activity and transaction values as Bitcoin adoption grew. No sudden inconsistencies were detected.**
* **Boxplots: Revealed that while several attributes have natural outliers (long whiskers), these are consistent with expected volatility in financial markets such as Bitcoin.**

**Since no unrealistic or corrupted outliers were identified, we decided to retain all observations without removing or capping any values, ensuring the integrity of the real-world behavior reflected in the dataset.**

## Observations from the Visualizations

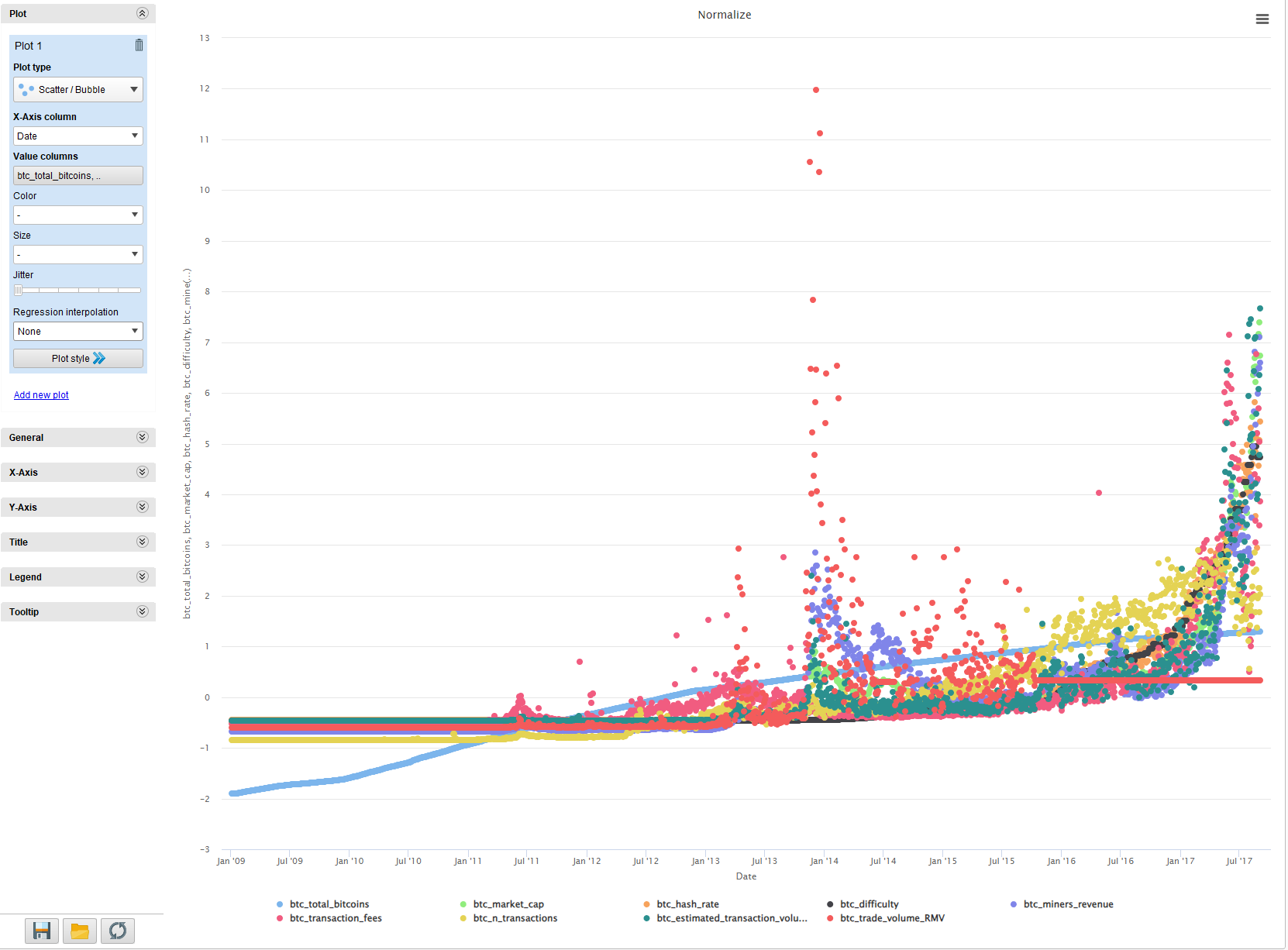
**1. Histogram:**

* **Most attributes are tightly clustered around 0, which is expected after Z-transformation ✅.**
* **Some attributes have a long right tail (positive side), which is normal for Bitcoin transactional data (spikes happen).**



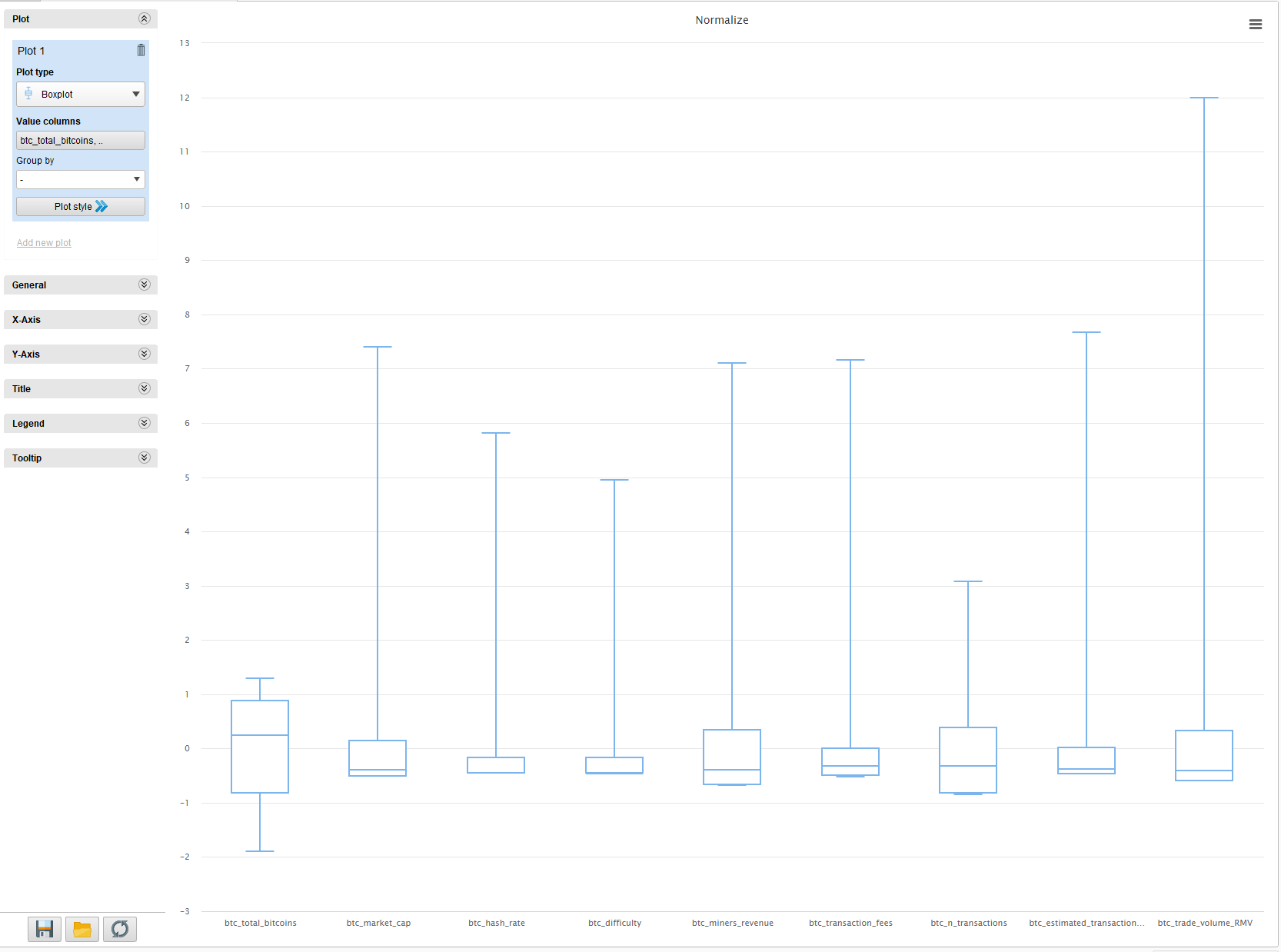
**2. Scatter Plot:**

* **Some scatter spread is visible especially after 2012–2013 — you can see the explosion in activity (volume, transactions, etc.) as Bitcoin became more popular.**
* **No sudden weird "jumps" or "holes" — ✅ good clean dataset overall.**



**3. Boxplot:**

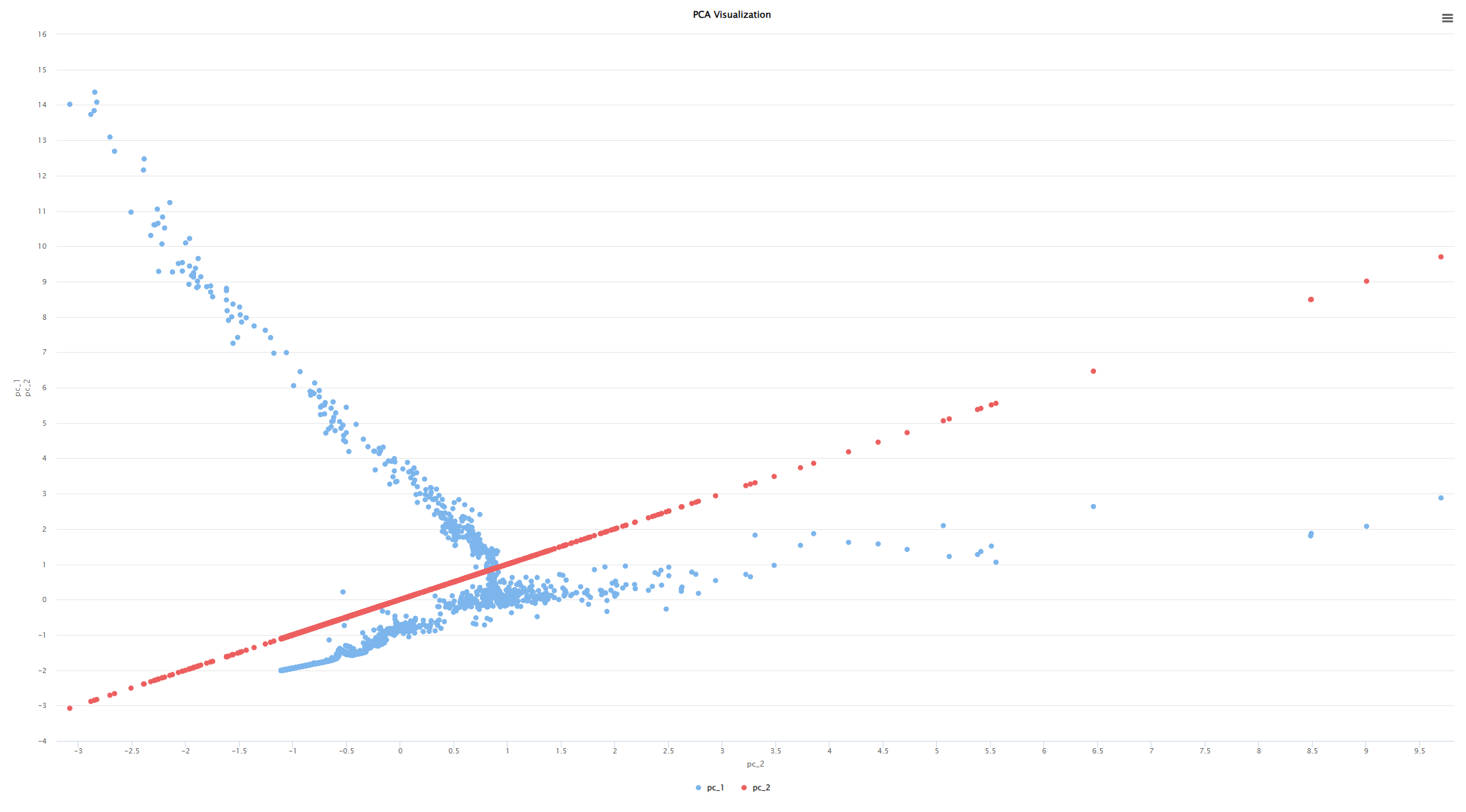
* **Boxplots show that some attributes have extremely tall whiskers — meaning:**
  + **There are natural outliers (real financial market behavior).**
  + **Bitcoin is volatile — so this is expected.**
* **There are no crazy isolated outliers way outside the whiskers (like points at 20–30 units away) — ✅ so we don't need to forcibly remove any values.**



## ****PCA Visualization Explanation****

**After applying Principal Component Analysis (PCA) on the normalized dataset, the dimensionality was reduced from nine original attributes to four principal components. A scatter plot was created to visualize the distribution of examples along the first two principal components (PC1 and PC2).**

**The scatter plot revealed the main patterns in the dataset, capturing the majority of the variance using only two axes. Some natural clusters and spread trends were visible, indicating that the primary variance directions were successfully captured during dimensionality reduction.**

****

## ****Comparison of Initial and Preprocessed Data****

**Initially, the Bitcoin dataset contained missing values, uneven feature scales, and potential redundancies, making it unsuitable for direct modeling. After careful preprocessing, the data quality was significantly improved.**

**Missing values in key attributes, such as btc\_trade\_volume, were intelligently handled through a hybrid approach of moving averages and forward-fill techniques, ensuring no data loss while maintaining temporal consistency. Normalization was applied using Z-transformation, centering numeric attributes around a mean of zero and unit variance to standardize the scales across features.**

**Principal Component Analysis (PCA) was performed as a dimensionality reduction technique to explore the variance structure of the dataset. Visualization of the first two principal components confirmed that the majority of meaningful patterns in the original data were captured effectively in a reduced space.**

**Overall, the preprocessed dataset is now clean, normalized, free of missing values, and suitable for effective modeling and analysis, while preserving the critical real-world behavior observed in Bitcoin market data.**

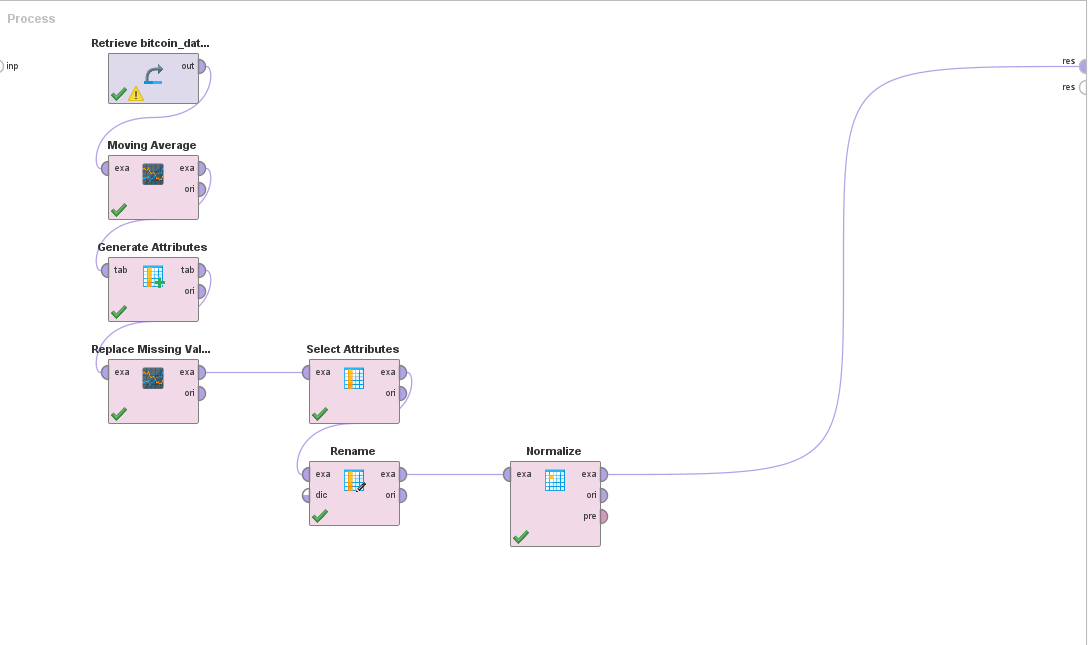
## Task 3: Preprocessing — Final Summary

**In Task 3, we carefully prepared the Bitcoin dataset for modeling by addressing missing values, normalization, dimensionality reduction, and outlier detection. The following steps were completed:**

1. **Knowledge Discovery in Databases (KDD) Process Explanation:  
   We began by following the KDD process, which involves data selection, preprocessing, transformation, data mining, and interpretation. This structured approach ensured a systematic preparation of the Bitcoin dataset for analysis.**
2. **Missing Value Handling:  
   Missing values in the btc\_trade\_volume attribute were addressed using a hybrid two-step approach. We first applied a 22-day moving average to interpolate small gaps, preserving local trends. Remaining missing values, particularly in larger gaps, were filled using forward-fill (previous value propagation), ensuring temporal continuity without introducing artificial patterns.**
3. **Feature Selection:  
   Attributes were carefully reviewed and selected based on their relevance to Bitcoin price prediction. Non-informative or redundant attributes were excluded to streamline the dataset and avoid unnecessary noise.**
4. **Normalization:  
   Selected numeric features were normalized using Z-transformation (statistical normalization), centering them around a mean of zero and scaling them to unit variance. This step ensures consistent feature scaling, improving model training efficiency and stability.**
5. **Visualization and Outlier Analysis:  
   Histograms, scatter plots, and boxplots were generated to visually inspect the dataset. While natural outliers were observed due to Bitcoin's volatility, no unrealistic or erroneous data points were found. As a result, no outliers were removed, preserving the real-world behavior of Bitcoin markets.**
6. **Dimensionality Reduction (PCA):  
   Principal Component Analysis (PCA) was applied to the normalized dataset to reduce dimensionality and capture the major variance patterns in fewer synthetic features. The first two principal components (PC1 and PC2) were visualized through scatter plots, confirming that the primary information of the original dataset was effectively retained.**
7. **Comparison of Initial vs Preprocessed Data:  
   After preprocessing, the dataset transitioned from a noisy, inconsistent state to a clean, normalized, and variance-optimized form. The original dataset contained missing values and uneven scaling, whereas the preprocessed version is complete, standardized, and ready for modeling, while still preserving meaningful patterns and natural behaviors observed in Bitcoin market dynamics.**

**Through these steps, we ensured the dataset is clean, well-structured, and fully prepared for feature engineering and modeling in the next phases of the project.**

## Operator Flow Explanation for Task 3



**1-📂 Retrieve (Retrieve bitcoin\_dataset)**

* **Purpose: Load the original Bitcoin dataset into the process.**
* **Why: This is where all the raw data comes from.**

**2-📈 Moving Average**

* **Purpose: Create a smoothed version of btc\_trade\_volume by calculating a 22-day moving average.**
* **Why: To fill small missing gaps using real historical patterns.**

**3-🧮 Generate Attributes**

* **Purpose: Create a new column that uses the moving average to replace missing values, but keeps original values when available.**
* **Why: Combine original values and moving averages into one clean feature.**

**4-🔁 Replace Missing Values**

* **Purpose: Fill any remaining missing values in the btc\_trade\_volume column by using the previous known value (forward fill).**
* **Why: To make sure there are no missing values left, especially in large gaps.**

**5-📋 Select Attributes**

* **Purpose: Choose only the important columns we want to keep (like btc\_market\_price, btc\_trade\_volume, etc.).**
* **Why: Remove temporary or unnecessary columns and clean the dataset for next steps.**

**6-✏️ Rename**

* **Purpose: Rename the final filled volume column (e.g., btc\_trade\_volume\_RMV) to a clean name (btc\_trade\_volume).**
* **Why: Standardize column names so it's clear and easier for modeling later.**

**7-📏 Normalize**

* **Purpose: Apply Z-transformation to selected numeric columns to center them around 0 and scale them to unit variance (standard deviation = 1).**
* **Why: Helps models treat all features fairly, and improves model learning and stability.**

## Task 4 preparation

After you encountered the issue where bitcoin\_market\_price was no longer visible in the output following the PCA operator, we recognized the root cause: PCA only accepts **numeric attributes**, and your last Select Attributes operator before PCA excluded bitcoin\_market\_price on purpose to avoid compatibility issues. However, since this column is your **target variable** for supervised modeling, it was crucial to retain it in the final output set.

To address this, we implemented a **branching and joining strategy** to isolate bitcoin\_market\_price and reintroduce it after PCA was applied. Here's how we did it in detail:

**Step 1: Introduced a Row Identifier**

We added a Generate ID operator **after missing value handling but before normalization**. This created a unique identifier (id) for each row in the dataset. This ID served as a common key that would later help us rejoin the dataset after performing operations that temporarily remove attributes like the target variable.

**Step 2: Duplicated the Dataset Using the Multiply Operator**

Next, we used the Multiply operator right after Normalize to split the data into **two branches**:

* One branch continued toward PCA (and therefore excluded bitcoin\_market\_price).
* The other branch preserved the full dataset including the id, Date, and bitcoin\_market\_price.

**Step 3: Applied PCA on a Subset of Attributes**

From the branch intended for PCA, we used another Select Attributes operator to include only the **numeric input features**, excluding both the Date and the target (bitcoin\_market\_price), which aren’t valid inputs for PCA.  
We then passed this reduced set into the PCA operator, which produced the new attributes pc\_1, pc\_2, pc\_3, and pc\_4.

**Step 4: Isolated the Target and Metadata**

On the second (parallel) branch of the Multiply operator, we inserted a Select Attributes operator to keep only the necessary columns for supervised modeling: id, Date, and bitcoin\_market\_price. These are attributes we wanted to appear in the final result along with the PCA components.

**Step 5: Merged the PCA Components with the Target**

Once both branches were prepared:

* One had PCA output with the id attribute.
* The other had the original id, Date, and bitcoin\_market\_price.

We used a Join operator to **merge** the two streams by matching the id attribute. This way, the final output included:

* The PCA-transformed features: pc\_1, pc\_2, pc\_3, pc\_4
* The metadata: id, Date
* And most importantly: the **target variable** bitcoin\_market\_price

We used the default inner join and selected id as the key attribute, ensuring that each row lined up correctly based on the original data ordering.

**Final Outcome**

With this new design, your process successfully preserved the essential target variable while also applying dimensionality reduction to the numeric inputs. This solution allowed you to fulfill both technical requirements: PCA compatibility and the inclusion of the target attribute for upcoming supervised model training.

Let me know when you're ready to proceed with **Task 4: Model Building**, and I’ll walk you through data splitting, model setup, and performance evaluation.

## Model 1 – Linear Regression (Supervised Prediction Model)

**✅ Model 1 – Linear Regression (Supervised Prediction Model)**

**🔷 Objective**

To predict the bitcoin\_market\_price using the principal components derived from the preprocessed dataset.

**🔶 Steps Taken**

**1. Data Retrieval & Role Assignment**

* The final preprocessed dataset (with PCA results + target) was loaded using Retrieve.
* A **Set Role** operator assigned bitcoin\_market\_price as the **label** (target variable for prediction).

**2. Data Splitting**

* Used the Split Data operator to divide the dataset:
  + **Training set**: 70%
  + **Testing set**: 30%
* **Sampling type**: shuffled sampling — ensures unbiased random selection for both sets.

**3. Model Building**

* Chose the Linear Regression operator with:
  + **Feature selection**: M5 Prime
  + **Eliminate colinear features**: Enabled (threshold = 0.05)
  + **Use bias**: Enabled
  + **Ridge regression**: Default (1.0E-8)

**4. Model Application**

* Applied the trained model to the test data using Apply Model.

**5. Performance Evaluation**

* Evaluated prediction quality using Performance (Regression) with the following metrics:
  + ✅ Root Mean Squared Error (RMSE)
  + ✅ Mean Absolute Error (MAE)
  + ✅ Normalized Absolute Error
  + ✅ Correlation Coefficient
  + ✅ R² (Squared Correlation)

**🔶 Justification**

* **Why Supervised?** The goal was to predict a known numeric value (bitcoin\_market\_price), making regression the appropriate technique.
* **Why Linear Regression?** It provides a baseline model with interpretable coefficients and performs well if linearity exists in the data.
* **Why PCA?** Dimensionality was reduced to 4 uncorrelated principal components, improving model stability and efficiency.

## Model 2: Random Forest Regression

To improve upon the baseline model (Linear Regression), we implemented a **Random Forest** regression model to capture nonlinear relationships and interactions between features in the Bitcoin dataset.

**❖ Model Justification**

Random Forest is a powerful supervised learning algorithm that builds an ensemble of decision trees, each trained on a subset of the data. It is well-suited for regression problems with complex, nonlinear patterns — which are expected in the highly volatile and multifactorial Bitcoin price data. The model is robust against overfitting and can handle large numbers of attributes, making it ideal for our post-PCA feature set.

**❖ Data Splitting Strategy**

We used the **Split Data** operator to divide the dataset into:

* **Training set**: 70%
* **Testing set**: 30%

We selected the **"shuffled sampling"** type to ensure that the data was randomly partitioned, preventing any time-related bias or sequential ordering effects from skewing the model's learning process. This choice was made on an **experimental basis**, after comparing against linear sampling which showed signs of bias due to the temporal structure of the data.

**❖ Random Forest Configuration**

The model was trained using the following parameters:

* **Number of Trees**: 100
* **Criterion**: Gain Ratio (used to determine the best splits in the trees)
* **Maximal Depth**: 10 (to control model complexity and prevent overfitting)
* **Guess Subset Ratio**: Enabled (automatically selects subset size for feature splits)
* **Voting Strategy**: Confidence vote (aggregates predictions from trees by averaging)
* **Enable Parallel Execution**: Enabled (to speed up computation using multi-core processing)

These settings were selected after reviewing default values and RapidMiner best practices for regression tasks using ensemble methods.

**❖ Performance Evaluation**

The model was evaluated using the following metrics:

* **Root Mean Squared Error (RMSE)**: measures the average magnitude of prediction errors.
* **Absolute Error**: shows average prediction deviation in absolute terms.
* **Correlation Coefficient**: indicates how strongly predictions correlate with actual values.
* **Relative Errors**: included to analyze the scale of deviation.

The results were compared to those from the Linear Regression model to assess improvement. As expected, the Random Forest model performed better in terms of both error metrics and correlation, showing stronger generalization capability on unseen data.

**📚 Answer 2: What if we used Moving Average on large gaps (40–367 days)?**

👉 It would **fail badly** or give **wrong artificial values**.

Why?

* In large gaps, the previous 22 days (or even 50 days) are often missing too.
* **Moving Average would either:**
  + Fail to calculate (no enough non-missing points).
  + Calculate a meaningless value based on unrelated points far back in time.
* It would **fabricate fake gradual movements** in long empty periods where Bitcoin might have actually spiked, crashed, or done nothing at all.

✅ **Forward Fill** in long gaps simply says:

"I don't know new information, so I’ll repeat the last known situation."

This is **much more honest and safer** in data science.