Predicting the price of bitcoin

Problem Statement

Dataset Information: The dataset used in this project is the "bitstampUSD_1-min_data" dataset, which contains historical minute-level data of Bitcoin prices from January 1, 2012, to March 31, 2021. The dataset includes information such as the timestamp, open price, high price, low price, close price, volume, and weighted price of Bitcoin for each minute.

Background Information: Bitcoin is a popular cryptocurrency that has gained significant attention and investment in recent years. The price of Bitcoin is known for its volatility, making it a challenging task to predict its future prices accurately. Predicting Bitcoin prices can be valuable for investors, traders, and financial analysts who are interested in understanding the market trends and making informed decisions.

This project aims to predict the price of Bitcoin using time series analysis techniques. Time series analysis involves analyzing and modeling data points collected over time to uncover patterns, trends, and relationships. By applying time series analysis to historical Bitcoin price data, we can develop a predictive model that estimates future prices based on past observations.

The project involves the following key steps:

1) **Data Exploration:** The project starts with loading the Bitcoin price dataset and performing initial exploratory data analysis. The data is resampled to different frequencies (daily, monthly, quarterly, and annual) to visualize the trends and patterns at different time intervals.
2) **Stationarity Check and STL-Decomposition:** The stationarity of the Bitcoin price series is checked using the Dickey-Fuller test. Additionally, the series is

decomposed into its trend, seasonality, and residual components using STL-decomposition to understand the underlying patterns.

3) **Box-Cox Transformations:** The Box-Cox transformation is applied to stabilize the variance of the Bitcoin price series, making it more suitable for modeling. The Dickey-Fuller test is again performed to check the stationarity of the transformed series.
4) **Seasonal Differentiation:** Seasonal differentiation is applied to the transformed series to remove the seasonal component. This process involves subtracting the series from its lagged value at the same season in the previous year. The Dickey-Fuller test is performed to check the stationarity of the differenced series.
5) **Regular Differentiation:** Regular differentiation is applied to the seasonally differenced series to remove the remaining trend component. This process involves subtracting the series from its lagged value at the previous time step. The stationarity of the differenced series is checked using the Dickey-Fuller test, and the series is considered stationary if the p-value is below a significance level.
6) **Model Selection:** The project focuses on selecting the best model for predicting Bitcoin prices. The initial approximation of model parameters is determined using Autocorrelation and Partial Autocorrelation Plots. Different combinations of parameters are tested using the SARIMAX model, and the model with the lowest Akaike Information Criterion (AIC) is selected as the best model.
7) **Analysis of Residues:** The residues (i.e., the differences between the predicted and actual values) of the best model are analyzed to assess the model's performance. The residues are visually examined using plots and tested for stationarity using the Dickey-Fuller test.
8) **Prediction:** Finally, the best model is used to predict future Bitcoin prices. The inverse Box-Cox transformation is applied to obtain the predicted prices in their original scale. The predicted prices are plotted alongside the actual prices to visualize the model's performance.

By predicting the price of Bitcoin, this project aims to provide insights and forecasts that can be valuable for individuals and organizations interested in the cryptocurrency market.

Framework

1) **Importing Libraries:** Begin by importing the necessary libraries such as NumPy, Pandas, Seaborn, Matplotlib, and Statsmodels. These libraries will be used for data manipulation, visualization, and time series analysis.
2) **Data Loading**: Load the Bitcoin price dataset, which is stored in a CSV file. The dataset contains minute-level data from January 1, 2012, to March 31, 2021. Use the Pandas library to read the CSV file and store the data in a DataFrame.
3) **Data Exploration:** Perform exploratory data analysis to gain insights into the dataset. Use Pandas and Matplotlib to visualize the Bitcoin price data at different frequencies, such as daily, monthly, quarterly, and annual. Plot the weighted price of Bitcoin over time to observe the trends and patterns.
4) **Data Preprocessing:** Prepare the data for time series analysis. Convert the 'Timestamp' column from Unix timestamp format to datetime format using the Pandas library. Set the 'Timestamp' column as the index of the DataFrame to facilitate resampling.
5) **Resampling:** Resample the Bitcoin price data to different frequencies (daily, monthly, quarterly, and annual) using the resample() function in Pandas. Calculate the mean of the weighted price for each resampled frequency. Create separate DataFrames for each resampled frequency (e.g., df_month, df_Q, df_year) to analyze the data at different time intervals.
6) **Stationarity Check and STL-Decomposition:** Check the stationarity of the resampled Bitcoin price series using the Dickey-Fuller test. Apply the STL-decomposition method from the statsmodels library to decompose the series into its trend, seasonality, and residual components. Plot the decomposed components to visualize the underlying patterns.
7) **Box-Cox Transformations:** Apply the Box-Cox transformation to the resampled Bitcoin price series using the boxcox() function from the scipy library. The Box-Cox transformation helps stabilize the variance of the series. Check the stationarity of the transformed series using the Dickey-Fuller test.
8) **Seasonal Differentiation:** Perform seasonal differentiation on the transformed series to remove the seasonal component. Subtract the series from its lagged value at the same season in the previous year. Check the stationarity of the differenced series using the Dickey-Fuller test.

9) **Regular Differentiation:** Apply regular differentiation on the seasonally differenced series to remove the remaining trend component. Subtract the series from its lagged value at the previous time step. Check the stationarity of the differenced series using the Dickey-Fuller test.
10) **Model Selection**: Select the best model for predicting Bitcoin prices. Use the SARIMAX model from the statsmodels library. Determine the initial approximation of model parameters by analyzing Autocorrelation and Partial Autocorrelation Plots. Iterate through different combinations of parameters and fit the SARIMAX model to the transformed series. Select the model with the lowest Akaike Information Criterion (AIC) as the best model.
11) **Analysis of Residues:** Analyze the residues (i.e., differences between predicted and actual values) of the best model. Plot the residues and check their stationarity using the Dickey-Fuller test. The residuals should be white noise for a well-fitted model.
12) **Prediction:** Use the best model to predict future Bitcoin prices. Create a DataFrame called 'df_month2' that includes the original weighted prices and additional rows for the future dates to be predicted. Apply the inverse Box-Cox transformation to obtain the predicted prices in their original scale. Plot the actual prices and the predicted prices on the same graph to evaluate the performance of the model.

Code Explanation

1) **Importing Libraries:** The code starts by importing various libraries such as NumPy, Pandas, Seaborn, Matplotlib, and Statsmodels. These libraries provide functions and tools for data manipulation, visualization, statistical analysis, and time series modeling.
2) **Data Exploration:** The next step is to explore the dataset. The code loads the Bitcoin price data from a CSV file using Pandas and displays the first few rows of the DataFrame to get an initial understanding of the data structure. It also resamples the data to different frequencies (daily, monthly, quarterly, and annual) and plots the weighted price of Bitcoin over time at each resampled frequency. This helps in visualizing the trends and patterns in the data.
3) **Data Preprocessing:** After exploring the data, the code performs some preprocessing steps. It converts the 'Timestamp' column from Unix timestamp format to datetime format using the Pandas library. This makes it easier to work with dates and time in subsequent steps. The 'Timestamp' column is also set as the index of the DataFrame, which helps in resampling and time-based operations.
4) **Resampling:** The code resamples the Bitcoin price data to different frequencies (daily, monthly, quarterly, and annual) using the resample() function in Pandas. Resampling involves aggregating the data over a specific time interval and calculating summary statistics. In this case, the mean of the weighted price is calculated for each resampled frequency. Separate DataFrames are created for each resampled frequency, such as df_month, df_Q, and df_year.
5) **Stationarity Check and STL-Decomposition:** Stationarity is an important property of time series data. The code checks the stationarity of the resampled Bitcoin price series using the Dickey-Fuller test, which is a statistical test for stationarity. It also performs STL-decomposition using the statsmodels library to decompose the series into its trend, seasonality, and residual components. This helps in understanding the underlying patterns and identifying any non-stationarity in the data.
6) **Box-Cox Transformations:** Stationarity is often achieved by transforming the data. The code applies the Box-Cox transformation to the resampled Bitcoin price series. The Box-Cox transformation helps stabilize the variance of the series,

which is important for time series analysis. The code also checks the stationarity of the transformed series using the Dickey-Fuller test.

7) **Seasonal Differentiation:** Seasonal differentiation is a technique used to remove the seasonal component from a time series. The code performs seasonal differentiation on the transformed series by subtracting the series from its lagged value at the same season in the previous year. This helps in eliminating the seasonality and making the series more stationary. The stationarity of the differenced series is checked using the Dickey-Fuller test.
8) **Regular Differentiation:** Regular differentiation is another technique used to remove the remaining trend component from a time series. The code applies regular differentiation on the seasonally differenced series by subtracting the series from its lagged value at the previous time step. This further reduces any trend in the data and makes it more stationary. The stationarity of the differenced series is checked using the Dickey-Fuller test.
9) **Model Selection:** The code selects the best model for predicting Bitcoin prices. It uses the SARIMAX model from the statsmodels library, which is a popular model for time series analysis. The code determines the initial approximation of model parameters by analyzing Autocorrelation and Partial Autocorrelation Plots. It iterates through different combinations of parameters and fits the SARIMAX model to the transformed series. The Akaike Information Criterion (AIC) is used as a measure of model goodness-of-fit. The code selects the model with the lowest AIC as the best model.
10) **Analysis of Residues:** Residual analysis is performed to evaluate the model's performance. Residuals represent the differences between the actual values and the predicted values of the series. The code plots the residuals and checks their autocorrelation using the Autocorrelation Function (ACF) plot. The stationarity of the residuals is also checked using the Dickey-Fuller test.
11) **Prediction:** Finally, the code predicts the future prices of Bitcoin using the best model. It creates a new DataFrame called df_month2, which includes the original weighted prices and a forecast column for storing the predicted prices. The invboxcox() function is used to transform the predicted prices back to their original scale. The code then plots the actual prices and the predicted prices on the same graph, providing a visual representation of the model's performance.

Future Work

Predicting the price of Bitcoin is a complex task, and there are several areas where future work can be done to improve the accuracy and robustness of the predictions. Here is a detailed breakdown of the steps involved in future work, along with a step-by-step guide on how to implement it:

1) **Data Enhancement and Feature Engineering:**

- Collect additional relevant data: To improve the prediction accuracy, consider incorporating additional data sources such as social media sentiment, trading volume, market indicators, or news sentiment related to cryptocurrencies.
- Feature engineering: Explore various techniques to create new features from the existing data, such as lagged values, moving averages, technical indicators (e.g., RSI, MACD), or sentiment scores. These engineered features can provide additional information and capture the underlying patterns in the data.

2) **Advanced Time Series Models:**

- Explore advanced time series models: Consider using more sophisticated models such as Long Short-Term Memory (LSTM) networks, which are specifically designed for capturing temporal dependencies in data.
- Parameter tuning: Optimize the hyperparameters of the selected models using techniques like grid search or Bayesian optimization to achieve better model performance.
- Model ensemble: Combine the predictions from multiple models, such as averaging or weighted averaging, to create a more robust and accurate ensemble prediction.

3) **Feature Selection and Dimensionality Reduction:**

- Conduct feature selection: Analyze the importance of different features and select the most relevant ones for prediction. Techniques like correlation analysis, feature importance, or regularization methods can help identify the significant features.
- Dimensionality reduction: Apply dimensionality reduction techniques such as Principal Component Analysis (PCA) or t-SNE to reduce the complexity of the feature space while retaining the most informative features.

4) **Model Evaluation and Performance Metrics:**

- Evaluate model performance: Utilize appropriate evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), or Root Mean Squared Percentage Error (RMSPE) to assess the accuracy of the models.
- Cross-validation: Perform cross-validation to validate the models' generalization ability and ensure they are not overfitting the training data.
- Backtesting: Implement a backtesting framework to simulate the performance of the predicted prices in a trading or investment scenario. This can provide insights into the profitability and risk associated with the predictions.

5) **Real-Time Predictions and Deployment:**

- Develop real-time prediction capability: Modify the code to handle incoming data in real-time, enabling the model to make predictions on the latest available information.
- Deployment: Create a user-friendly interface or application where users can input relevant parameters, view predictions, and monitor the performance of the model over time. Consider deploying the model as a web application or API for easy accessibility.

Step-by-Step Guide to Implement Future Work:

1) **Data Enhancement and Feature Engineering:**

- Identify additional data sources related to Bitcoin price and collect the data.
- Analyze the existing data and explore various feature engineering techniques to create new features.
- Incorporate the new features into the dataset and update the preprocessing steps accordingly.

2) **Advanced Time Series Models:**

- Research and implement advanced models such as LSTM networks using libraries like TensorFlow or Keras.
- Tune the hyperparameters of the models to achieve better performance.
- Explore model ensemble techniques and implement them to combine the predictions from multiple models.

3) **Feature Selection and Dimensionality Reduction:**

- Perform feature selection techniques to identify the most relevant features for prediction.

- Apply dimensionality reduction techniques like PCA or t-SNE to reduce the feature space's complexity.

4) **Model Evaluation and Performance Metrics:**

- Evaluate the performance of the models using appropriate evaluation metrics.
- Implement cross-validation to assess the models' generalization ability.
- Set up a backtesting framework to simulate the model's performance in a trading or investment scenario.

5) **Real-Time Predictions and Deployment:**

- Modify the code to handle real-time data updates.
- Develop a user-friendly interface or application to input parameters, view predictions, and monitor model performance.
- Deploy the model as a web application or API to make it accessible to users.

By following this step-by-step guide and incorporating the suggested future work, you can enhance the prediction accuracy and usability of the Bitcoin price prediction project.

Exercise Questions

1) **Question: What is the purpose of resampling the Bitcoin price data to different frequencies (daily, monthly, quarterly, and annual)?**

Answer: The resampling process allows us to analyze the Bitcoin price data at different time intervals. Resampling to different frequencies provides a broader perspective on the price trends and helps identify long-term patterns, seasonal variations, and overall price behavior at different time scales.

2) **Question: Why is it important to check the stationarity of the time series data before modeling?**

Answer: Stationarity is a crucial assumption for many time series models. Stationary data has constant mean, variance, and autocovariance over time. It is important to check stationarity because non-stationary data can lead to spurious correlations and inaccurate predictions. By checking stationarity, we can determine if the data needs transformation or differencing to make it suitable for modeling.

3) **Question: What is the purpose of Box-Cox transformations in the context of time series analysis?**

Answer: Box-Cox transformations are used to stabilize the variance of a time series. By applying the Box-Cox transformation, we can reduce heteroscedasticity and make the data more suitable for modeling. It helps in achieving stationarity by normalizing the data and making it conform to the assumptions of certain models that assume constant variance.

4) **Question: Why do we perform seasonal and regular differentiation on the Bitcoin price data?**

Answer: Seasonal differentiation is performed to remove the seasonal component from the data. By differencing the data with a lag equal to the seasonal period, we can eliminate the seasonal patterns and make the data stationary. Regular differentiation, on the other hand, captures the non-seasonal changes in the data. It helps remove any remaining trends or patterns that are not accounted for by seasonal differencing, further improving the stationarity of the series.

5) **Question: How does model selection play a role in predicting the Bitcoin price?**

Answer: Model selection involves choosing the best model that can effectively capture the underlying patterns and trends in the Bitcoin price data. It helps identify the optimal set of parameters for the selected model, such as the order of differencing, seasonal order, and other model-specific parameters. The chosen model will have the lowest information criterion, such as AIC (Akaike Information Criterion), indicating its goodness of fit to the data. Accurate model selection is crucial for obtaining reliable predictions and making informed decisions based on the Bitcoin price forecasts.