Macroeconomic Effects on Bitcoin Price Using Topological Data Analysis and Distance-to-Default Metrics

Eric Schmid

October 2024

Abstract

In this report, we explore the application of Topological Data Analysis (TDA) and financial risk metrics such as the Merton Distance-to-Default (DTD), adapted slightly here, in predicting Bitcoin price quantiles (n=3). Using daily data from Yahoo Finance and FRED, we build a machine learning model that integrates both financial and geometric features. In this study, the target variable is the 30-day moving average of Bitcoin's price, calculated from t-15 to t+15. This target metric captures smoothed price behavior by taking into account both past and future price movements, offering a more stable projection of Bitcoin's price trends. By using this 30-day moving average as a target, the model is able to reduce the impact of short-term volatility, allowing for more stable predictions of medium-term price movements. This approach is particularly beneficial in highly volatile markets like Bitcoin, where daily fluctuations can introduce noise into the predictive process. The analysis leverages Topological Data Analysis (TDA) and financial metrics such as Distance-to-Default (D2D) to enhance the predictive accuracy, offering a more robust understanding of Bitcoin's price behavior under macroeconomic influences. The model is trained using XGBoost with hyperparameter optimization through Bayesian search, achieving an accuracy of 51.62% (which is greater than 33.33% probability for random guessing over three quantiles) and an AUC score of 0.587. To better understand the model's performance, we evaluate entropy per class and plot the confusion matrix, as well as predicted vs actual values over time.

Repository

For full access to the code and related files for this project, please visit the GitHub repository at:

`https://github.com/ericschmid-uchicago/macroeconomic-trends-on-bitcoin`

1 Introduction and Literature Review

Bitcoin is a decentralized digital currency whose market value is highly volatile, creating both opportunities and risks for investors. This volatility makes Bitcoin an ideal candidate for mathematical modeling. Traditional models, such as the Merton DTD model, are commonly used in finance to estimate the distance to default for companies, providing insight into financial risk.

Topological Data Analysis (TDA) is a novel approach that has gained traction in recent years. TDA captures the underlying geometric and topological structures in data, providing distinctive insights that conventional approaches might overlook. Research has shown that TDA can be useful for analyzing high-dimensional time-series data, such as stock market prices and cryptocurrency data [2]. In this paper, we apply TDA to Bitcoin data to extract topological features and use them, along with traditional financial metrics like Treasury yields and Gross Debt, to improve predictions of Bitcoin price changes.

1.1 Bitcoin as a Financial Asset

Bitcoin's decentralized nature and lack of intrinsic value have led to substantial price fluctuations. Its adoption as an investment medium has led to a growing body of literature focused on understanding and predicting its price movements. However, traditional financial indicators such as Treasury yields, inflation rates, and debt levels may still influence Bitcoin, as they reflect the broader economic environment.

1.2 Topological Data Analysis in Finance

TDA has been applied in various fields, including neuroscience, biology, and recently, finance. Persistent homology, one of the key tools in TDA, captures the multi-scale topological features of data by identifying clusters, loops, and voids. These features, called persistence diagrams, are then used to quantify the "shape" of data and provide additional dimensions for analysis. In financial time-series data, TDA can capture subtle, persistent patterns that other methods might overlook.

2 Data Collection and Feature Engineering

2.1 Bitcoin Data from Yahoo Finance

For this project, we used Bitcoin historical price data fetched from Yahoo Finance. The dataset spans from 2014 to 2024, providing sufficient granularity and time depth for meaningful analysis. The dataset includes daily "closing" prices, which serve as the basis for our calculations. The following code fetches and preprocesses the Bitcoin data, ensuring that any timezone information is removed:

```
btc = yf.download("BTC-USD", start="2014-01-01")["Adj Close"]
btc = btc.reset_index()
btc['Date'] = pd.to_datetime(btc['Date']).dt.tz_localize(None)
btc.columns = ['Date', 'BTC']
```

This ensures consistency across the dataset for the next steps.

2.2 FRED Economic Indicators

To capture the broader economic context, we obtained financial data from the Federal Reserve Economic Data (FRED) API. The two primary economic indicators used were the 10-Year Treasury Yield and the U.S. Gross Federal Debt. These metrics reflect the cost of borrowing and the nation's financial obligations, respectively, both of which could influence investor behavior in cryptocurrency markets. These factors were hypothesized to have an impact on Bitcoin's price due to their reflection of macroeconomic conditions:

```
fred = Fred(api_key=fred_api_key)
treasury_10y = fred.get_series("DGS10", observation_start="2014-01-01")
gross_debt = fred.get_series("GFDEBTN", observation_start="2014-01-01")
```

3 Mathematical Models and Methods

3.1 Merton's DTD Model for Bitcoin

The Merton DTD model, traditionally used in corporate finance to assess a company's risk of default, was adapted to the cryptocurrency market in this project. The model calculates the distance to default (DTD) for Bitcoin using the following formula:

$$DTD = \frac{\ln(A/D) + (r - 0.5\sigma^2)T}{\sigma\sqrt{T}}$$

where:

- A is the "asset price" (Bitcoin price),
- D is the "debt" (Gross Federal Debt),
- r is the risk-free interest rate (Treasury yield),
- σ is the volatility of the asset (Bitcoin),
- T is the time horizon.

This metric serves as a proxy for Bitcoin's financial stability and is used as one of the features in our machine learning model.

The Merton Distance-to-Default model is used to assess the risk of an asset's value falling below its liabilities. The DTD is calculated as follows:

```
mert['vol_BTC'] = mert['BTC'].pct_change().rolling(window=30, min_periods=30)
                  .std().shift(1) * np.sqrt(252)
A = mert['BTC'].shift(1)
D = mert['Gross_Debt'].shift(1)
r = mert['Treasury_10Y'].shift(1) / 100
T = 1  # Time horizon in years
mert['btc_d2'] = (np.log(A / D) + (r - 0.5 * mert['vol_BTC'] ** 2) * T) /
                 (mert['vol_BTC'] * np.sqrt(T))
mert['btc_d2'] = mert['btc_d2'].fillna(0)
```

3.2 Topological Data Analysis (TDA)

We performed TDA on the DTD data using Vietoris-Rips persistence. By applying TDA, we captured multi-scale geometric structures in the DTD time series, which are summarized using the bottleneck distance:

```
VR_persistence = VietorisRipsPersistence(metric="euclidean",
                                         homology_dimensions=[0, 1])
persistence_diagrams = VR_persistence.fit_transform(d2_windows)
amplitude = Amplitude(metric='bottleneck')
tda_features = amplitude.fit_transform(persistence_diagrams)
```

These features were padded and aligned with the dataset.

The TDA methodology focuses on transforming time-series data into topological features. For this project, we used the 'gtda' library to compute the persistence diagrams of Bitcoin price data. Persistence diagrams are a graphical representation of the homological features, such as connected components and loops, which persist across multiple scales in the data.

The persistence diagrams are then transformed into numerical features (e.g., persistence entropy and amplitude) that can be fed into the machine learning model. These features help capture long-term trends and structures in the Bitcoin price data that traditional statistical methods might miss.

4 Machine Learning: XGBoost Model

The XGBoost model was chosen due to its efficiency and high performance on structured datasets. It is a gradient boosting algorithm that builds a series of decision trees, where each tree corrects the errors of the previous one.

We split the dataset into training and test sets using a time-series-aware split. The dataset was divided 80% for training and 20% for testing, ensuring that no data leakage occurred. The model used the following features:

- **Bitcoin price:** the daily closing price.
- **Merton DTD:** the calculated distance to default.
- **TDA features:** derived from persistence diagrams.

- **Treasury Yield:** the 10-Year Treasury Yield.
- **Gross Debt:** the U.S. Gross Federal Debt.

5 Target Variable: 30-Day Moving Average with 15-Day Lookahead

In this study, the target variable is based on a centered moving average to represent Bitcoin's price trends over a medium-term window. The target variable is calculated as the 30-day moving average, centered between $t-15$ and $t+15$, where $t=0$ represents the current time. It is important to emphasize that this moving average is only used as a target for prediction purposes and is not used directly by the model as an input feature.

The primary reason for choosing this target variable is to reduce the noise from daily price fluctuations while providing a balanced outlook on price trends over the next 30 days. By calculating the moving average centered on both past $(t-15)$ and future $(t+15)$ data, this target offers a more stable representation of Bitcoin's price trend, helping to smooth out volatility and better inform the model's predictions.

The lookahead window anticipates future price changes, which is especially valuable in highly volatile markets like Bitcoin. However, the centered moving average does not introduce lookahead bias because it is used solely as the target for the model to learn to predict, based on historical data up to time t.

The following Python function demonstrates how the target variable is constructed:

```
def calculate_future_moving_average_and_deciles(mert,
↪  look_ahead_days=15, ma_window=30):
    """
    Calculate target variable based on centered returns.
    At time t=0, the target is based on the moving average from
    ↪  t-15 to t+15.
    """
    # Calculate the centered moving average directly
    mert['future_30d_ma'] = mert['BTC'].rolling(window=ma_window,
    ↪  center=True).mean()

    # Calculate returns relative to current price
    mert['current_price'] = mert['BTC']
    mert['future_return'] = (mert['future_30d_ma'] -
    ↪  mert['current_price']) / mert['current_price']

    # Create target classes based on future returns
    mert['BTC_decile_change'] =
    ↪  pd.qcut(mert['future_return'].dropna(), q=3,
    ↪  labels=False)
```

```
    # Drop rows where we don't have a complete window
    mert = mert.dropna(subset=['BTC_decile_change'])

    # Print alignment check
    print("\nSample alignment check (showing how the centered MA
    ↪  is calculated):")
    debug_df = pd.DataFrame({
        'Date': mert['Date'],
        'Current_Price': mert['BTC'],
        'MA_t-15_to_t+15': mert['future_30d_ma'],
        'Return': mert['future_return'].map('{:.2%}'.format) if
        ↪  not mert['future_return'].isna().all() else
        ↪  mert['future_return'],
        'Target_Class': mert['BTC_decile_change']
    }).head(30)
    print(debug_df.to_string(index=False))

    print("\nTarget class distribution:")
    print(mert['BTC_decile_change'].value_counts())

    return mert
```

This function constructs the target variable by calculating a centered moving average over a 30-day window from $t-15$ to $t+15$, creating a more stable price signal. The function also computes the future return as the percentage difference between the current price and the centered moving average. Based on these future returns, we assign each time step to one of three quantiles (low/negative, medium, or high returns), creating the 'BTC_decile_change' classification that serves as the target for model prediction.

This target construction helps the model focus on medium-term trends while effectively smoothing out daily volatility. The target variable does not directly feed into the model, but instead, it guides the model to predict future price trends based on past and present data.

6 Model Training and Optimization

6.1 Model Setup

We used the XGBoost model to classify Bitcoin price changes into three quantiles (0, 1, 2). The model was trained with Bayesian search over the following parameter space:

```
search_space = {
    'n_estimators': Integer(50, 300),
    'max_depth': Integer(3, 10),
```

```
    'learning_rate': Real(0.01, 0.3, 'log-uniform'),
    'subsample': Real(0.6, 1.0),
    'colsample_bytree': Real(0.6, 1.0),
    'lambda': Real(0.01, 10.0, 'log-uniform'),  # L2 regularization
    'alpha': Real(0.01, 10.0, 'log-uniform')    # L1 regularization
}
```

Bayesian search with Gaussian Processes was used to optimize the hyperparameters.

6.2 Evaluation Metrics

The model's performance was evaluated using AUC and accuracy scores. Additionally, confusion matrices and entropy plots were generated to understand how well the model performed on each class.

7 Results

7.1 Confusion Matrix

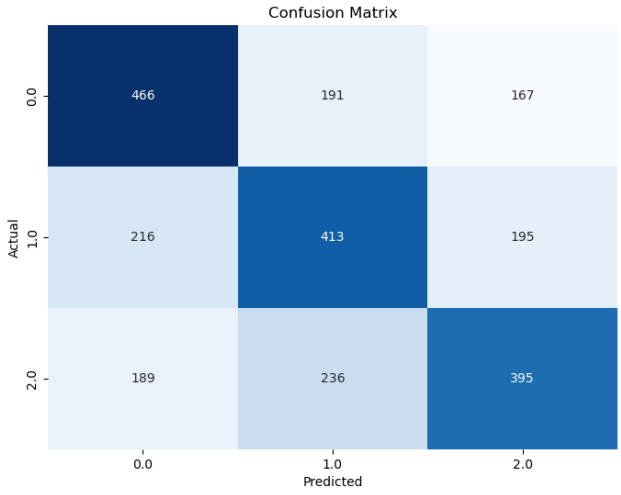

Figure 1: Confusion Matrix for Predicted vs Actual Classes

The matrix compares the actual classes (on the vertical axis) against the predicted classes (on the horizontal axis). Ideally, all predictions should align with the diagonal, indicating that every predicted class matches the actual class. Misclassifications appear off-diagonal.

Class 0: The model predicted class 0 correctly 466 times, though it misclassified 191 instances as class 1 and 167 as class 2. This highlights that while the model performs decently for class 0, it occasionally struggles to separate it from other classes, especially class 1.

Class 1: The model predicted class 1 correctly 413 times but misclassified 216 instances as class 0 and 195 as class 2. Class 1 shows a higher level of misclassification, likely due to overlap in patterns between class 1 and neighboring quantiles.

Class 2: For class 2, the model correctly predicted 395 instances, but misclassified 236 as class 1 and 189 as class 0. This reinforces the confusion between classes 1 and 2 but demonstrates a relatively strong performance in separating class 2 from class 0.

Overall, the confusion matrix shows that while the model captures certain patterns, especially for class 0, there is significant misclassification between classes 1 and 2. This could be mitigated by enhancing feature engineering or adjusting the model's hyperparameters.

7.2 Predicted vs Actual Over Time

Figure 2 compares the predicted and actual class labels over time. While the predicted labels generally track the actual labels, there are several periods where the predicted values deviate, especially during higher volatility intervals.

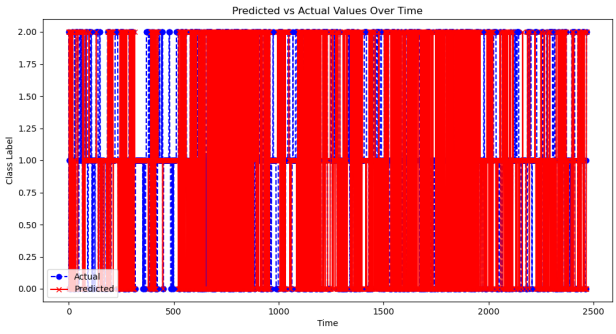

Figure 2: Predicted vs Actual Values Over Time

7.3 Entropy per Class

In Figure 3, the mean entropy values per class are presented. Class 0.0, 1.0, and 2.0 show relatively high and uniform entropy levels, indicating the model's uncertainty in predicting each class. These findings suggest the need for further

refinements, particularly to increase the model's confidence in predicting class 1.0.

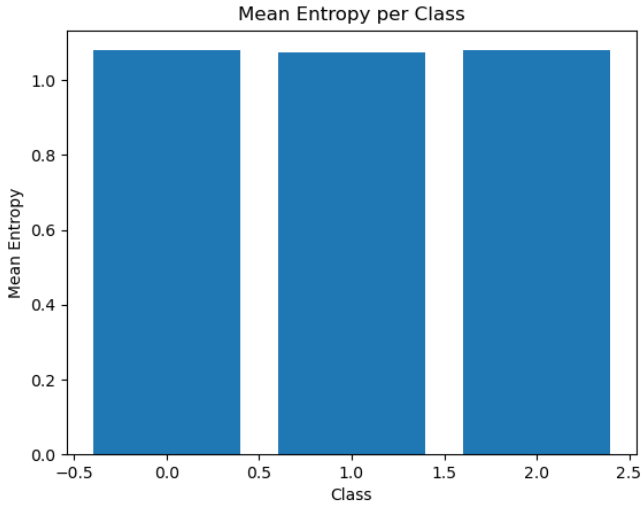

Figure 3: Mean Entropy per Class

8 Conclusion

This study integrates Topological Data Analysis (TDA) with macroeconomic metrics, demonstrating potential for improving Bitcoin price prediction. Despite achieving modest accuracy and AUC scores, the results highlight areas for improvement, particularly in reducing misclassifications between classes 1 and 2 and addressing the model's overall uncertainty. Further research could refine the model by incorporating additional macroeconomic features and improving the regularization strategy to lower entropy and enhance predictive performance.

References

[1] Merton, R. C. (1974). On the Pricing of Corporate Debt: The Risk Structure of Interest Rates. *Journal of Finance*, 29(2), 449-470.

[2] S. W. Akingbade, M. Gidea, M. Manzi, and V. Nateghi, *Why Topological Data Analysis Detects Financial Bubbles?*, arXiv preprint arXiv:2304.06877, 2023. Available at: `https://arxiv.org/abs/2304.06877`.

[3] Scikit-TDA Developers (2023). *gtda: A Python Library for Topological Data Analysis*.