Cryptocurrency Final Report

Dylan Kim, Siddharth Patel, Zihan Yu, Weijie Guan

DS 3000

Mohit Singhal

Abstract

This project aimed to predict cryptocurrency price volatility using machine learning models. The models were based on the market capitalization and trading volume of the coins. We implemented a linear regression, polynomial regression, and an interaction model in an effort to capture a potential correlation between these features. The interaction polynomial regression was our best-performing model; however, it explained approximately 2% of the volatility. Our analysis depicted violations of critical assumptions such as linearity and homoscedasticity, which suggests that crypto volatility is influenced by factors not captured in our data. In the end, we encountered the challenges of predicting such a volatile asset class. For a more accurate analysis, we may find more features or models useful.

Introduction

As cryptocurrencies continue to garner more popularity, we have seen just how volatile some coins can be. This volatility poses a real risk for the everyday investor. Additionally, the stability issues raise concerns over the feasibility of integrating cryptocurrencies into our day-to-day transactions. Analyzing the relationship between a coin's market cap and its volatility can provide real insights into how investors can mitigate risk. In theory, larger coins should be more stable than lower market cap coins. We aimed to predict volatility using machine learning models. There are two main questions our analysis tackles:

1. Are larger and established coins (like BTC and ETH) less volatile compared to smaller or emerging coins?
2. What statistical signals explain short-term volatility, and how can we use these driven insights to predict how liquidity and market size relate to risk?

Data Description

Here is the flow of our data processing pipeline:

1. Fetch data from CoinMarketCap API to get raw cryptocurrency data
2. Clean the data to get rid of stablecoins and other unwanted values
3. Visualize the data using a variety of plotting libraries
4. Analyze data through machine learning models

To begin analyzing the data, we used the cryptocurrency dataset through the CoinMarketCap API. This process involved collecting the raw data, performing data cleaning, and saving the cleaned datasets for analysis. The cleaning process removed invalid values such as NaN, n/a, and 0 (where applicable). This addressed stablecoins, which artificially maintain low volatility by

design, and excluded records that can't be analyzed, such as cryptocurrencies with a market cap ≤ 0 or volatility ≤ 0.

Next, we focused on addressing our key questions. This involved creating new features such as volatility (calculated as the absolute value of percent_change_24h) and volume_to_mcap_ratio (calculated as 24h volume divided by market cap). Both of these helped with the analysis of usage patterns. We categorized cryptocurrencies into market cap categories using Small (<$1B), Medium ($1B-$10B), Large ($10B-$100B), and Mega (>$100B). Further analysis included generating basic statistics and creating visualizations such as scatterplots to identify relationships between market cap and volatility. We also used box plots (using Seaborn and Matplotlib) to visualize volatility distributions across market cap categories. These steps provided insights into whether larger cryptocurrencies show more price stability and how volatility patterns differ across coin sizes.

Finally, the cleaned data were prepared for machine learning by selecting relevant features for predictive volatility modeling. Due to the wide range of values in our dataset (market caps span from millions to trillions and volatility from <1% to >100%), we applied logarithmic transformations to market capitalization and volatility. This stabilizes variance and makes relationships more linear, which is important for improving regression model performance.

	name	symbol	rank	market_cap	price	volume_24h	\
0	Bitcoin	BTC	1	2.227144e+12	111693.244287	2.698098e+10	
1	Ethereum	ETH	2	4.782503e+11	3962.362753	1.518473e+10	
3	XRP	XRP	4	1.572487e+11	2.620057	3.941539e+09	
4	BNB	BNB	5	1.551535e+11	1114.770320	2.305617e+09	
5	Solana	SOL	6	1.066220e+11	194.050283	3.325281e+09	

	percent_change_24h	circulating_supply	volatility	volume_to_mcap_ratio	\
0	1.008372	1.993983e+07	1.008372	0.012115	
1	1.329040	1.206983e+08	1.329040	0.031751	
3	5.825413	6.001728e+10	5.825413	0.025066	
4	0.614897	1.391798e+08	0.614897	0.014860	
5	1.422965	5.494555e+08	1.422965	0.031188	

	cap_category
0	Mega
1	Mega
3	Mega
4	Mega
5	Mega

Methods

To validate our ML approach, we carefully considered the mathematical foundations, the underlying assumptions, and the possible pitfalls that could affect our project. Our main research question was, "Can we predict cryptocurrency price volatility based on market fundamentals like market capitalization and trading volume?" Guided by this, we initially developed models using Linear Regression and Polynomial Regression to explore relationships in our dataset.

For our first ML model, we utilized Linear Regression to estimate the relationship between the independent variable (log-transformed market capitalization) and the dependent variable (log-transformed volatility). The method is widely used because of its simplicity and interpretability. However, Linear Regression relies on several critical assumptions: linearity of the relationship between predictors and the response variable, independence of residuals, constant variance of residuals, and normality of residuals. We log-transformed both variables because cryptocurrency market caps range from millions to over 2 trillion dollars, spanning more than nine orders of magnitude. When we assessed our dataset, we found significant violations of these assumptions, including nonlinear relationships and inconsistent residual patterns. This model yielded an R^2 of only 0.0144, which means it explained just 1.44% of the variance in volatility, thus limiting the accuracy and reliability of its predictions, as shown in the Results section.

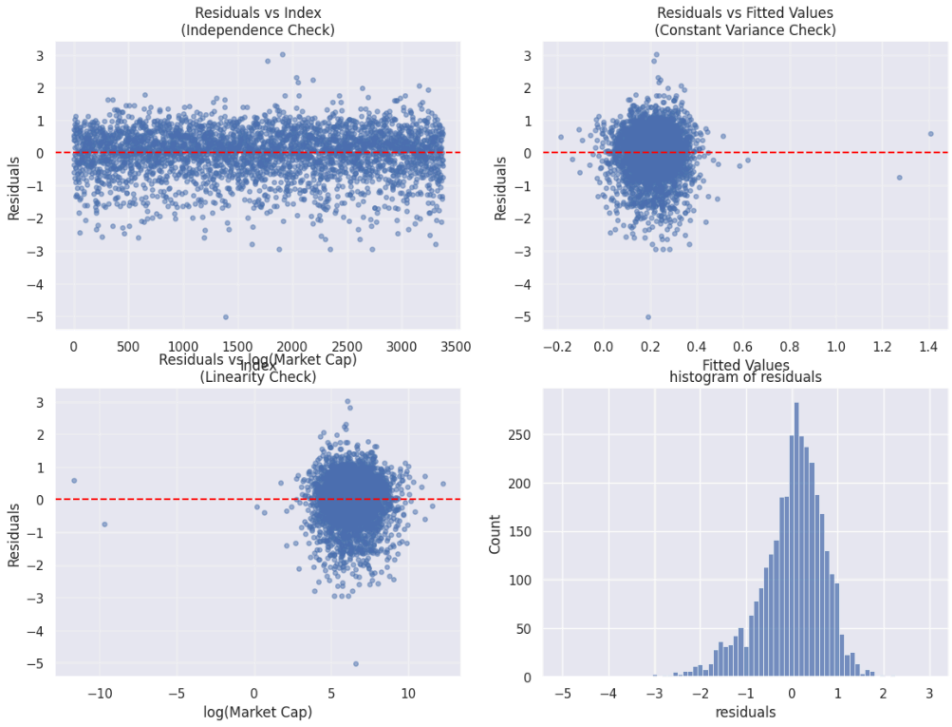
To handle the nonlinear relationship, we applied Polynomial Regression (degree 4) for our second ML model. We proceeded with polynomial regression by adding terms up to the fourth power, that is, x^2, x^3, and x^4, to capture curvature in the data. While the polynomial model performed somewhat better, with R^2 improving from 0.0144 to 0.0215, it explained only 2.15% of the volatility variance. It was a 49% improvement over linear regression, though far from good, hence showing the limitations of polynomial regression in efficiently modeling the complexities of our dataset, as will be shown in the Results section.

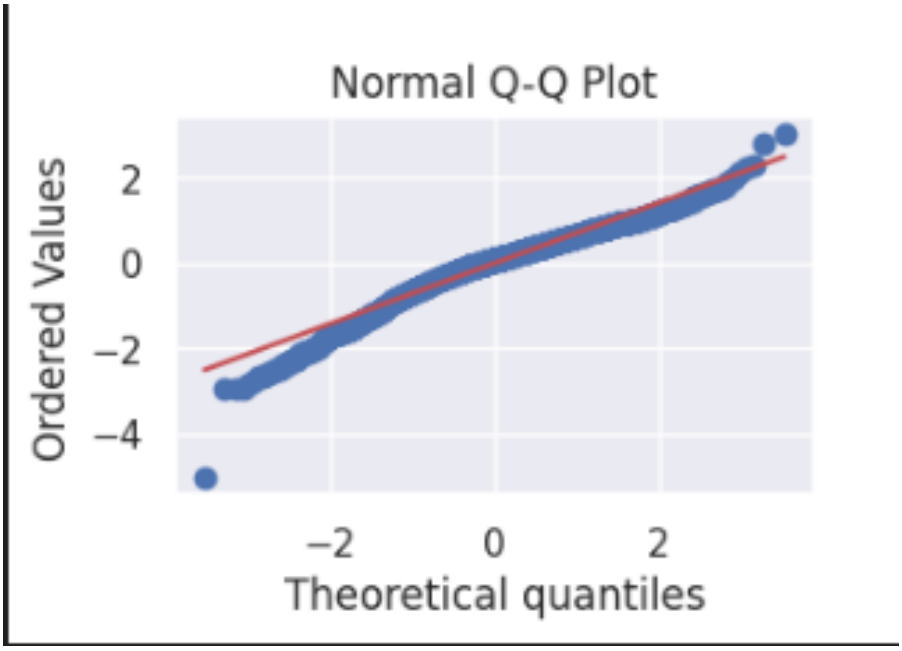
Recognizing these challenges, we incorporated trading volume as our final ML model to refine our approach. We created a binary dummy variable called high_volume that splits cryptocurrencies based on whether their volume-to-market-cap ratio is above or below the median. Then we added interaction terms between this dummy variable and each polynomial term, testing whether the market cap-volatility relationship differs for high-liquidity versus low-liquidity coins. For this revised analysis, we focused on key features including market capitalization, trading volume, and the interactions between them. This model achieved the highest R^2 of 0.0236 (2.36%), a marginal improvement of 0.0021 over Model 2. However, this minimal gain of only 10% relative improvement came at the cost of doubling the feature count from 4 to 8. This suggests the interaction terms captured noise rather than genuine patterns. Converting continuous volume data into a binary variable likely caused some information loss.

All three models were compared using a 70-30 train-test split on 3,376 cryptocurrencies, and performance was evaluated using Mean Squared Error and R^2.

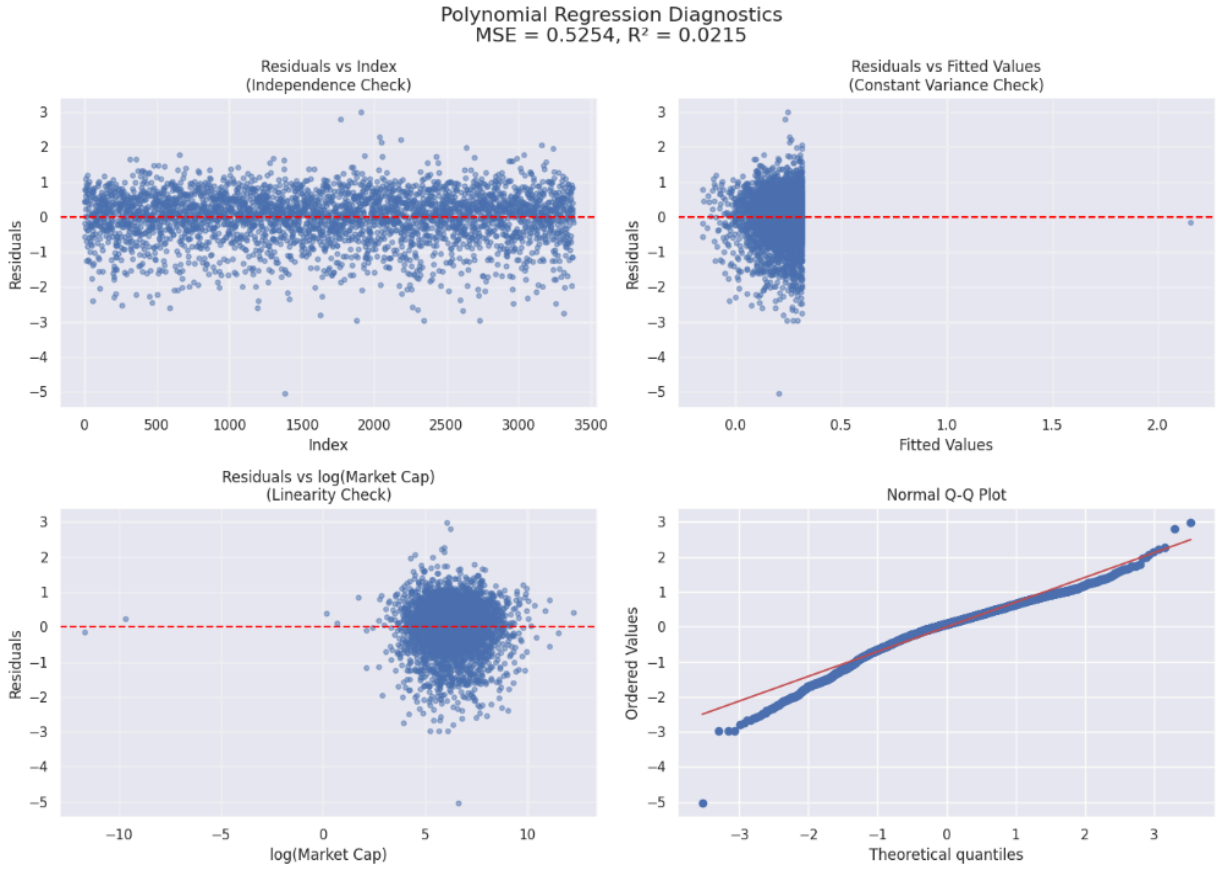
Results

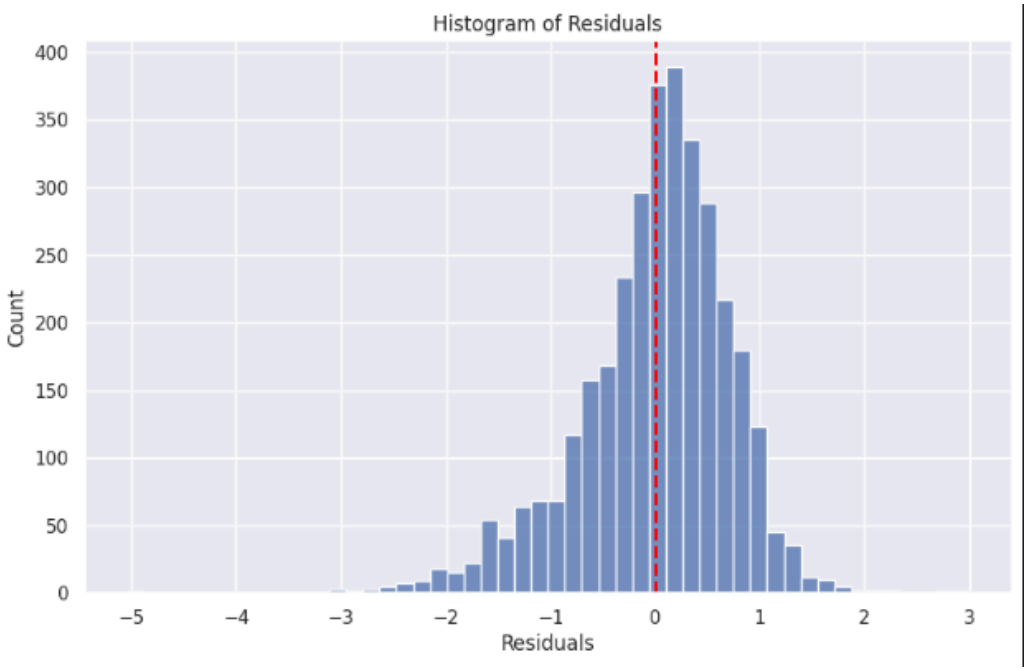
Firstly, we analyzed Model 1: $\log_{10}(\text{Market Cap}) \rightarrow \log_{10}(\text{Volatility})$ using a Linear Regression model. We transformed the Market Cap and Volatility with log because larger coins add a heavy right skewness to the dataset. For a linear regression, there are a few assumptions required for optimal performance. Specifically, these assumptions are linearity, independence, homoscedasticity, and normality. The Residuals vs Index plot shows a random scatter of points, which means the independence assumption appears to be met. Next, we look at the check for constant variance or homoscedasticity. In our residuals vs fitted values plot, we see that the spread of the residuals is clustered around x = 0.2. Because of this, we determined that our data is heteroscedastic. In our linearity check, we see a similar cluster to that of the constant variance check. This ultimately suggests a non-linearity in the relationship we are modeling. Lastly, our histogram plot of residuals shows a bell-shaped curve centered around zero, which indicates a normal distribution. Additionally, for the most part, our Q-Q plot confirms these findings further. There is a very slight deviation in both the lower tail and upper tail, which may suggest there are outliers that the linear regression did not capture. The linear regression model was our worst performing model with an R^2 score of 0.0144 and an MSE of 0.518. Simply put, market cap alone is not enough to determine volatility in cryptocurrencies.

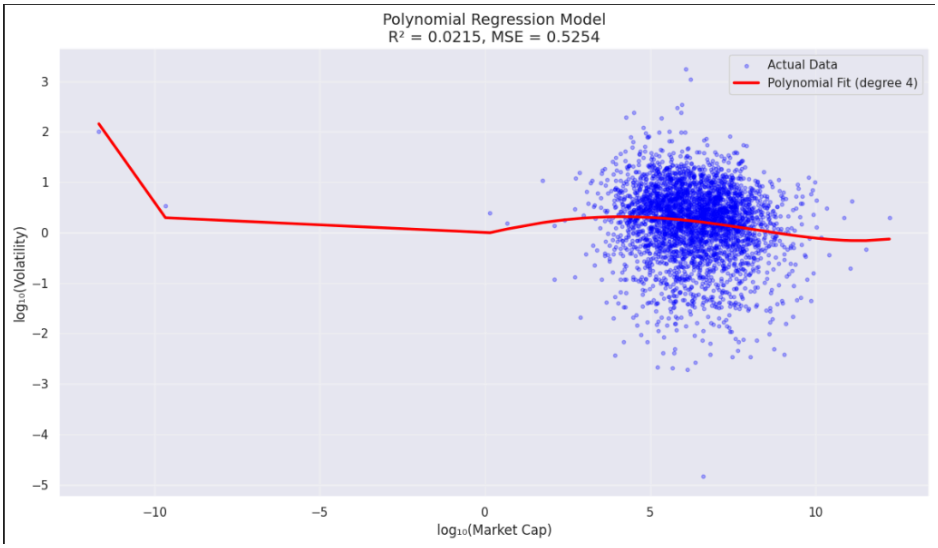




In an attempt to improve our model, we opted for a polynomial regression model. In our residuals vs index plot, we see that the residuals are randomly scattered, indicating the observations are independent. Next, in our constant variance check, we see that the heteroscedasticity problem persists as a funnel-like shape is clearly present. In our linearity check, yet again, we see a more clustered pattern than a random scatter. This leads us to believe that there are unexplained factors in our polynomial regression model. Lastly, our Q-Q plot shows a similar structure to that of the linear regression Q-Q plot. While the middle of the data mainly follows the line, the tails both have a slight deviation. Additionally, our model slightly outperformed the linear regression model, achieving an R^2 score of 0.0215. Our MSE of 0.525, however, remained relatively similar to that of the linear regression model.


Polynomial Regression Diagnostics
MSE = 0.5254, R² = 0.0215
Residuals vs Index
(Independence Check)
Residuals
Index
Residuals vs Fitted Values
(Constant Variance Check)
Residuals
Fitted Values
Residuals vs log(Market Cap)
(Linearity Check)
Residuals
log(Market Cap)
Normal Q-Q Plot
Ordered Values
Theoretical quantiles


Histogram of Residuals
Count
Residuals



Thirdly, we analyzed an interaction polynomial regression model. We created a “high volume” dummy variable in an attempt to teach the model how to detect different curves for higher volume vs lower volume assets. Ultimately, as seen below, the model, while the best performing, still had a very weak performance. The mean squared error of 0.524 implies that our predictions poorly represent the true volatility of the coins. We also see that our R^2 value of 0.0236 only explains 2.36% of the volatility. These results align directly with the concern of these models, as it is clear that volatility cannot be meaningfully predicted from market cap.

```
np.random.seed(42)

X_market_cap = np.array(cleaned_df['market_cap'])
x1 = np.log10(X_market_cap).reshape(-1, 1)
y = np.log10(cleaned_df['volatility'])

# create dummy variable: high_volume (1 if above median, 0 if below)
median_volume_ratio = cleaned_df['volume_to_mcap_ratio'].median()
X_high_volume = (cleaned_df['volume_to_mcap_ratio'] > median_volume_ratio).astype(int).values.reshape(-1, 1)

print(f"high volume (above median): {X_high_volume.sum()} cryptocurrencies")
print(f"low volume (below median): {len(X_high_volume) - X_high_volume.sum()} cryptocurrencies")

# degree 4 for the quartic equation
poly = PolynomialFeatures(degree=4)

# sets the X_poly with bias column in the beginning
X_market_cap_poly_bias = poly.fit_transform(x1)

# remove the bias column
X_market_cap_poly = X_market_cap_poly_bias[:, 1:]

# combines X_market_cap_poly with interaction terms and dummy variables
X_poly = np.concatenate([
    X_market_cap_poly,
    X_high_volume * x1,
    X_high_volume * (x1**2),
    X_high_volume * (x1**3),
    X_high_volume * (x1**4)
], axis=1)

print(f"\nFinal feature matrix shape: {X_poly.shape}")
print("First 5 rows:")
print(X_poly[0:5])

crossval = train_test_split(X_poly, y, test_size=0.3, random_state=3)
Xtrain, Xtest, ytrain, ytest = crossval

# finds the line of best fit
m = line_of_best_fit(Xtrain, ytrain)
model = linreg_predict(Xtest, ytest, m)

# prints out MSE and R^2 values
print(f"\nCross-Validation Results:")
print(f'MSE for the interaction polynomial model = {model["mse"]:.3f}')
print(f'R^2 for the interaction polynomial model = {model["r2"]:.4f}')
```

```
high volume (above median): 1688 cryptocurrencies
low volume (below median): 1688 cryptocurrencies

Final feature matrix shape: (3376, 8)
First 5 rows:
[[1.22389329e+01 1.49791479e+02 1.83328787e+03 2.24374873e+04
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.15278644e+01 1.32891659e+02 1.53195703e+03 1.76601930e+04
  1.15278644e+01 1.32891659e+02 1.53195703e+03 1.76601930e+04]
 [1.10909589e+01 1.23009370e+02 1.36429188e+03 1.51313052e+04
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.10626765e+01 1.22382811e+02 1.35388145e+03 1.49775525e+04
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.08642961e+01 1.18032930e+02 1.28234470e+03 1.39317726e+04
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]

Cross-Validation Results:
MSE for the interaction polynomial model = 0.524
R^2 for the interaction polynomial model = 0.0236
```

Discussion

Predicting cryptocurrency volatility using market capitalization and trading volume proved to be extremely challenging, highlighting limitations within our various strategies. Notably, across all the models, we saw extremely low R^2 values. Although we increased model complexity to use polynomial terms, interaction terms, and dummy variables (Models 2 and 3), we only saw marginal improvements overall. Model 2's R^2 of 0.0215 was 49% better than Model 1's 0.0114, suggesting polynomial terms captured some non-linearity. Out of the models, model 3 performed the best with an R^2 of 0.0236, indicating that converting continuous volume data to a binary dummy variable added predictive information. However, it still only explained 2% of the variation in volatility, leaving an astronomical 98% unexplained. Our findings confirm that market capitalization and binary volume indicators alone are not enough to predict the volatility of cryptocurrencies. This likely stems from not addressing complex interactions within the market and its trends.

Future work might include:

- Utilizing continuous volume features (volume_24h, volume_to_mcap_ratio) instead of binary dummies
- Exploring other predictors, such as market sentiment and trends, to better explain the volatility of cryptocurrencies
- Implementing other modeling methods, such as ones that utilize Time Series

Despite our models' limited predictive power, ethical implications must be addressed:

1. This research is purely academic. Our models' poor performance ($R^2 < 2.4\%$) demonstrates that cryptocurrency markets are unpredictable using simple metrics.
2. Our analysis uses publicly available data, which cannot detect market manipulation, off-exchange trading, or real-time dynamics.

To develop more effective models of cryptocurrency volatility in future work, it is crucial to understand the limitations of our models and how they can be potentially improved upon. Our results suggest that accurately predicting volatility may require an approach that can account for temporal patterns in volatility, market trends, and other factors beyond simple market metrics like market cap. Our attempts underscore the challenges of modeling noisy financial data, where simple market predictors and standard regression techniques are often insufficient in capturing real-world patterns.