Fast Finance Analyzer

David Kennet
Wentworth Institute of Technology

This project looks at how machine learning can be used to predict short-term price movements in both stocks and cryptocurrencies. A multithreaded C++ fetcher was used to collect real-time data on six assets: Bitcoin, Ethereum, Solana, Apple, Microsoft, and Alphabet. With features like rolling averages, volatility, and price change percentages, models such as Ridge Regression, Random Forest, and K-Means clustering were built to measure prediction performance and spot behavioral patterns. The results show that while there is some potential in using simple indicators to forecast price direction, there are clear limits especially in fast-moving markets like crypto.

Keywords - Machine Learning, Financial Prediction, Cryptocurrency, Stock Analysis, Volatility

I. Introduction (*Heading 1*)

In recent years, machine learning has become an essential tool for finance companies, where it is used to detect patterns, manage risk, and inform trading decisions. This project dives into whether short-term price movements in both stocks and cryptocurrencies can be predicted using machine learning strategies. Cryptocurrencies, in particular, are known to be extremely volatile, making them the perfect topic for evaluating the limitations of predictive modeling.

This project examines real-time data from six financial assets: three stocks (Apple, Microsoft, and Alphabet) and three cryptocurrencies (Bitcoin, Ethereum, and Solana). Using a multithreaded C++ data fetcher, hourly and daily data were collected and analyzed. Feature engineering in Python included technical indicators such as rolling averages, volatility, and price change percentage. The objective was not only to test whether machine learning could predict short-term price behavior, but also to identify which indicators were most useful, how different assets respond to market trends, and how the behavior of cryptocurrencies compares to that of traditional stocks.

By applying models like Ridge Regression, Random Forest Classifier, and KMeans clustering, this report evaluates both the predictive potential and the challenges of modeling financial markets with limited feature sets. This work contributes to a better understanding of how machine learning can (or cannot) achieve meaningful signals in noisy, constantly changing financial environments.

II. Datasets

A. Source of dataset (Heading 2)

The datasets used in this project were generated in real-time using a custom-built C++ multithreaded data fetcher. Rather than relying on preexisting datasets, financial data was collected directly from two credible APIs: CoinGecko for cryptocurrency data and Alpha Vantage for stock market data. The cryptocurrency data (Bitcoin, Ethereum, and Solana) was obtained in hourly intervals straight from CoinGecko, while the stock data (Apple, Microsoft, and Alphabet) was fetched from Alpha Vantage using a registered API key.

The C++ fetcher utilized cURL and multithreading to request and save the data concurrently, providing fast and efficient retrieval of multiple assets. Data was gathered during the month of April 2025, and all responses were saved in CSV format for further analysis in Python. These datasets reflect up-to-date market behavior and offer a strong foundation for evaluating short-term price prediction using machine learning.

B. Character of the datasets

The project used six separate datasets: three for cryptocurrencies (Bitcoin, Ethereum, Solana) and three for stocks (Apple, Microsoft, Alphabet). Each dataset was stored in CSV format and consisted of tabular time-series data.

Asset Type	Asset	Frequency	Rows	Key Columns	Units
Cryptocurrency	Bitcoin	Hourly	~721	Timestamp, Price, Volume	USD, UTC timestamp
Cryptocurrency	Ethereum	Hourly	~721	Timestamp, Price, Volume	USD, UTC timestamp
Cryptocurrency	Solana	Hourly	~721	Timestamp, Price, Volume	USD, UTC timestamp
Stock	Apple	Daily	~100	Date, Open, High, Low, Close, Volume	USD, Shares
Stock	Microsoft	Daily	~100	Date, Open, High, Low, Close, Volume	USD, Shares
Stock	Alphabet	Daily	~100	Date, Open, High, Low, Close, Volume	USD, Shares

All cryptocurrency data included timestamps in Unix milliseconds, which were converted to human-readable datetime format using Python's pd.to_datetime() function. For stock data, the format was already standardized with ISO date strings.

New Features include:

Feature Name	Description	Applied To	Formula / Logic	Units
rolling_avg	Rolling average of price	All assets	24-period (crypto), 5-period (stocks) using .rolling().mean()	USD
volatility	Rolling standard deviation of price	All assets	Same window as rolling_avg, via .rolling().std()	USD
price_change	Percentage change in price	All assets	(price_t - price_t-1) / price_t-1 using .pct_change()	Decimal (unitless)

979-8-3315-0993-4/25/$31.00 ©2025 IEEE

Feature Name	Description	Applied To	Formula / Logic	Units
	between intervals			
target	Binary label for price direction (up/down)	BTC only	1 if next price > current price, else 0 via .shift()	Binary (0 or 1)
cluster	Cluster label from KMeans (grouped behavior)	BTC only	Unsupervised clustering of rolling_avg and volatility	Integer (0–2)

These engineered features were created to enhance the predictive power of the dataset and support machine learning models. All features were calculated using standard Python libraries (Pandas, NumPy, Scikit-learn). Feature creation helped uncover underlying structure in asset behavior, enabled regression and classification tasks, and supported exploratory analysis through clustering.

III. Methodology

A. Data Collection with Multithreaded C++

To collect real-time financial data efficiently, a multithreaded C++ data fetcher was developed. This system used the cURL library to request data from CoinGecko and Alpha Vantage. Separate threads were spawned to fetch data for each asset concurrently, significantly reducing total runtime. The output was parsed using the nlohmann/json library and saved in CSV format for further analysis.

- **Assumptions**: APIs provide accurate and real-time data
- **Advantages**: Fast, efficient, and custom-controlled; ideal for fetching multiple datasets in parallel.
- **Disadvantages**: Requires low-level code management and error handling; way harder to scale than Python scripts.
- **Why chosen**: Demonstrates technical skill and allows full control over request logic.
- **Extras**: Timestamps and API keys were managed through environment variables to keep the code secure and reusable.

B. Feature Engineering in Python

Once the CSVs were generated, data was cleaned and enhanced in Python using Pandas and NumPy.

- **Methods Used**:
  - rolling_avg: 24-hour (crypto) or 5-day (stocks) rolling average using .rolling().mean()
  - volatility: Rolling standard deviation via .rolling().std()
  - price_change: Computed with .pct_change()
  - target: Binary price direction for classification (1 if price went up)
- **Why chosen**: These are common and effective technical indicators used in financial forecasting.
- **Assumptions**: Past movement holds some predictive value for the near future.

C. Ridge Regression

Ridge Regression is a linear regression model that minimizes the sum of squared residuals with a penalty term on the size of coefficients.

- **Assumptions**: Linear relationship between features and target.
- **Advantages**: Reduces overfitting
- **Disadvantages**: Can underperform when nonlinear patterns dominate.
- **Why chosen**: Simple and interpretable baseline for predicting actual price values.
- **Python function**: sklearn.linear_model.Ridge()
- **Adjustments**: Used default alpha, could be optimized with cross-validation in future work.

D. Random Forest Classifier

Random Forest is a learning method based on decision trees, used here to predict price direction (up/down) as a classification problem.

- **Assumptions**: Data patterns can be captured by tree splits; no strong linearity required.
- **Advantages**: Handles nonlinearities, interpretable in terms of feature importance.
- **Disadvantages**: Can overfit if not tuned; not ideal for time series unless changed accordingly.
- **Why chosen**: Well-suited for classifying short-term directional movement.
- **Python function**: sklearn.ensemble.RandomForestClassifier()
- **Result**: Achieved ~47% accuracy, indicating the difficulty of predicting crypto with limited features.

E. KMeans Clustering

KMeans is an unsupervised learning algorithm used to group similar data points based on selected features.

- **Assumptions**: Clusters are spherical and equally sized.
- **Advantages**: Quick and easy to implement; helpful for exploring patterns.
- **Disadvantages**: Sensitive to initial cluster centers; assumes Euclidean distance is meaningful.
- **Why chosen**: Useful for identifying behavioral regimes in BTC based on volatility and rolling average.
- **Python function**: sklearn.cluster.KMeans()
- **Extras**: Data was scaled with StandardScaler() before clustering to ensure fair feature weighting.

IV. Results.

A. Price Trend Visualizations

To understand how different assets behave over time, price trend plots were created for both cryptocurrencies (Bitcoin, Ethereum, Solana) and stocks (Apple, Microsoft, Alphabet). Cryptocurrencies were plotted using hourly data, and stocks were plotted with daily closing prices.

Figure 1. Crypto price trends (Bitcoin, Ethereum, Solana; separated because of the bitcoins scale)

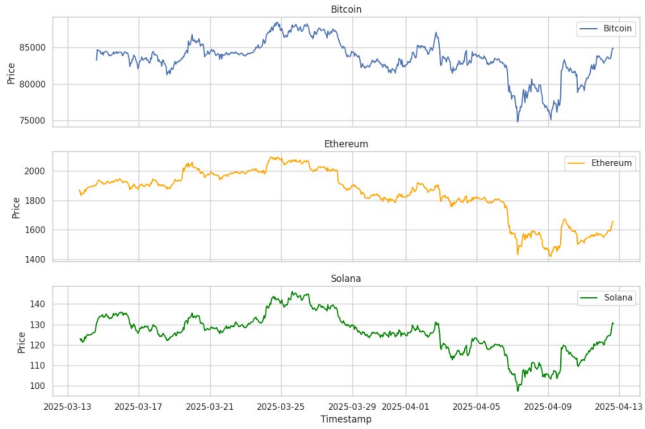
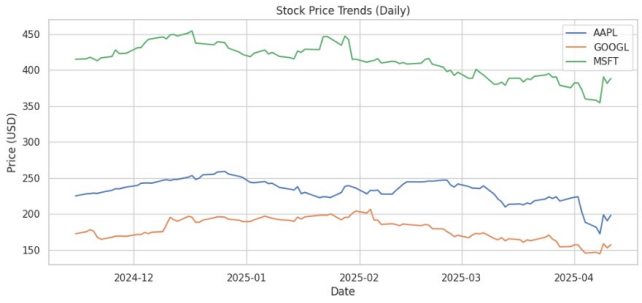


Figure 2. Stock price trends (Apple, Microsoft, Alphabet)

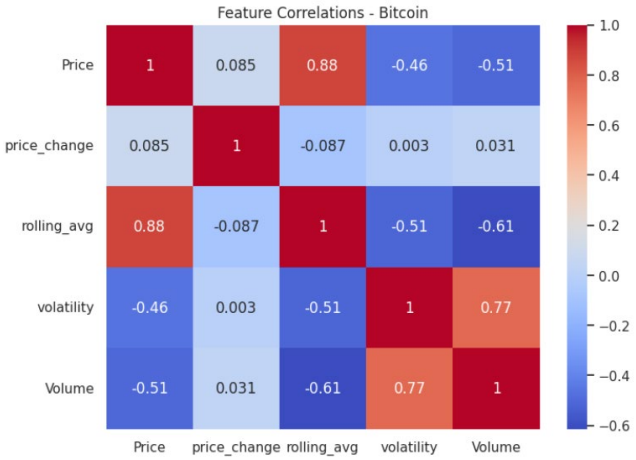


These plots reveal that cryptocurrency assets show significantly more short-term fluctuation than traditional stocks, with Bitcoin and Solana having sharp jumps and corrections. Stocks on the other hand tend to follow smoother trends.

B. Feature Correlation Analysis

A correlation heatmap was generated using engineered features for Bitcoin to identify which indicators were most strongly related to price movements.

Figure 3. Heatmap of feature correlations (BTC)



The analysis showed that rolling_avg is highly correlated with price (0.88), while volatility and volume had moderate negative correlations. This helped inform which features were most meaningful in the predictive models.

C. Regression Results – Ridge Regression

Ridge Regression was applied to predict the actual Bitcoin price using rolling_avg and volatility. The model achieved a **Mean Squared Error (MSE)** of **1,613,306.75**.

This result suggests that while the model captures general price trends, it struggles to make highly accurate short-term predictions due to the noisy nature of financial time series.

D. Classification Results – Random Forest

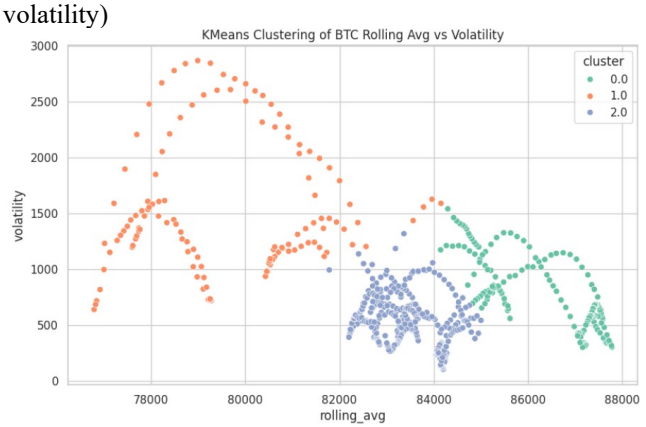
A Random Forest Classifier was used to predict whether the Bitcoin price would increase in the next hour. The model achieved an accuracy of approximately **47%**, which is a little worse than randomly guessing.

This result highlights the difficulty in predicting short-term price direction with only a small set of technical indicators. It also shows that while these features offer some value, more complex patterns, or external inputs such as news may be needed to improve performance.

E. Clustering Results – KMeans

KMeans clustering was applied to Bitcoin's rolling average and volatility to identify behavior patterns. The algorithm grouped data into three clusters, revealing different "volatility regimes."

Figure 4. KMeans clustering result for BTC (rolling_avg vs volatility)



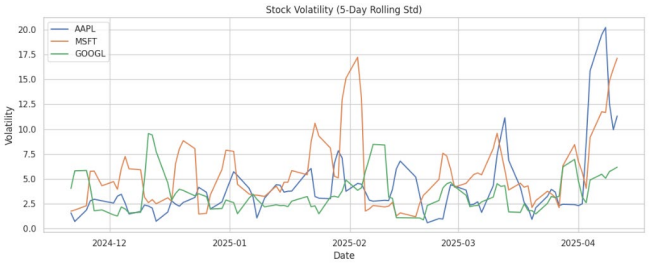
This analysis helped visualize how Bitcoin behaves under different market conditions.

F. Volatility Comparison Across Assets

Volatility was calculated using a 24-period rolling standard deviation for crypto and a 5-day rolling standard deviation for stocks.

Figure 5. Volatility of BTC, ETH, and SOL (separate because bitcoin scale was making Ethereum and Solana look flat)

Figure 6. Volatility of AAPL, MSFT, and GOOGL



These visualizations confirmed that cryptocurrencies consistently experience higher short-term volatility than stocks. Even after scaling and normalization, the crypto plots show larger fluctuations in rolling volatility.

V. Discussion

While the models and visualizations produced helpful insights, the results also showed several limitations and areas for future improvement.

The most notable weakness was the Random Forest Classifier's performance in predicting short-term price direction. Despite being a strong ensemble model, it achieved only ~47% accuracy. This likely stems from the limited number of input features and the highly stochastic nature of crypto markets. In the short term, price direction is influenced by many external factors such as news, social sentiment, and global economic indicators, which were not included in this project.

Similarly, the Ridge Regression model captured overall price trends but returned a relatively high MSE, reinforcing that linear models struggle to account for the volatility.

Another limitation is the use of only two features (rolling_avg and volatility) for most models. While this keeps the models interpretable, it restricts their depth and predictive power. In the future having technical indicators such as Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) would be extremely beneficial.

The KMeans clustering was visually effective in separating market regimes, but the assumptions of equal-sized, spherical clusters may not fully represent real market behavior. A more advanced clustering algorithm like DBSCAN or Gaussian Mixture Models would be a better alternative.

While the project shows that simple models can detect broad patterns in financial data, better predictions would require a better feature set.

VI. Conclusion

This project set out to explore whether machine learning can be effectively applied to predict short-term price movements in financial markets, using both stocks and cryptocurrencies. A multithreaded C++ program was created to fetch real-time data from CoinGecko and Alpha Vantage. Python was used for data preprocessing, feature engineering, and machine learning analysis.

Through regression, classification, and clustering techniques, the project showed multiple insights. Ridge Regression showed that rolling averages and volatility can capture general price trends, while Random Forest struggled to accurately predict short-term price direction with limited features, reinforcing the complexity of financial market behavior. KMeans clustering successfully identified

different volatility regimes, demonstrating the value of unsupervised learning in understanding asset behavior.

Perhaps the most significant finding was the difference in volatility and predictability between cryptocurrencies and stocks. Cryptocurrencies displayed much higher short-term volatility, which limited the performance of simple predictive models but also put a spotlight on the potential for future work using more complex indicators.

Overall, this project demonstrates that while machine learning offers useful tools for financial data analysis, short-term price prediction especially in crypto markets remains a highly challenging task. However, with more features, more time-series validation, and predictive performance this project can be greatly improved.

ACKNOWLEDGMENT *(Heading 5)*

I want to give a big thank you to Dr. Pang for giving me so much support on this project. It was an honor being in your class. You have taught me so much and this project only taught me more. The skills I have learned will be a steppingstone for future successes.

REFERENCES

[1] CoinGecko, "API documentation," [Online]. Available: https://www.coingecko.com/en/api. [Accessed: Apr. 10, 2025].

[2] Alpha Vantage, "Stock API documentation," [Online]. Available: https://www.alphavantage.co/documentation/. [Accessed: Apr. 10, 2025].

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.