Project Title: Machine Learning Bitcoin Price Prediction

Name: Trevor Huffstetler (801472945)

Primary Paper:

Authors: Chen, Z., Li, C., & Sun, W

Title: Bitcoin price prediction using machine learning: An approach to sample dimension engineering

Year:2020

Journal Name: *Journal of Computational and Applied Mathematics*, *365*, 112395.

Link: https://doi.org/10.1016/j.cam.2019.112395

1. Introduction

1.1 Problem Statement:

This paper addresses the challenge of accurately predicting Bitcoin prices using machine learning; specifically, how sample frequency and feature engineering impact model accuracy. Accurately predicting Bitcoin price is challenging due to the following reasons: inherent volatility of the cryptocurrency market, its susceptibility to social factors, and the elusive nature of Bitcoin. Current research often focuses on achieving high prediction accuracy while overlooking the impact of different feature sets and modeling techniques. This aligns with the problem highlighted by Chen et al. (2020), who notes that few studies focus on the feasibility of applying different modeling techniques to samples with varying data structures and dimensional features for Bitcoin price prediction

1.2 Motivation and Significance:

Existing Bitcoin forecasting research usually focuses on a singular frequency, neglecting a clear understanding of how these sample dimensions drive model performance. This gap can lead to choosing the wrong model or unintentionally overfitting data. This project aims to address this by building complementary datasets with different frequencies and testing various machine learning models. Understanding the influence of sample frequency and feature count will have practical implications for building more effective Bitcoin price prediction models.

1.3 Brief Overview of the Approach:

This study replicates the approach used by Chen et al (2020) to predict Bitcoin price using machine learning. Two distinct datasets were created: a high-frequency dataset (minute-

by-minute) and a low-frequency dataset (daily). Various machine learning models (Logistic Regression, LDA, SVM, Random Forest) are tested on these datasets to assess the impact of sample frequency on performance. A dimensionally-aware approach to feature extraction was employed, using high-dimensional features for the low-frequency data and low-dimensional technical indicators for the high-frequency data. The prediction target is a binary classification (price up or down).

2. Background

2.1 Summary of Related Research:

Ortu et al. (2022): Explored deep learning models trained on trading indicators and social media sentiment (Reddit and GitHub). This approach showed improved accuracy by incorporating social media sentiment as a measure for public interest.

- **Pros:** Demonstrated the potential value of incorporating social media data for improved prediction accuracy.
- **Cons:** Relies on paid APIs for high-quality social media data, limiting reproducibility. Deep learning models are prone to overfitting in volatile markets and require frequent retraining.
- **Relation to Main Method:** This research highlights the potential of external factors (like sentiment) but also underscores the challenges of using complex models and noisy data, which my study aims to address by focusing on data frequency and feature selection.

Chen (2023): Investigated the performance of LSTM and Random Forest models using 47 variables across 8 categories on daily data, split into two distinct time periods for training and evaluation.

- **Pros:** Temporally split data for more robust evaluation.
- **Cons:** Exclusively used daily data, ignoring potential short-term patterns. Did not explore the impact of different data frequencies or sample dimension engineering.
- **Relation to Main Method:** This study acknowledges the importance of temporal considerations but highlights the gap in understanding the role of data frequency, which my research directly investigates by using both daily and minute-level data.

3. Methods

3.1 Details of the Algorithms and Methods:

Data Collection:

Bitcoin price data was sourced from Kaggle (Kaggle, n.d.),

- This dataset consisted of Open, Close, High, Low, Volume Bitcoin price information for every minute between 2012 to 2025
- Additionally, network-level data from Blockchain.com (Blockchain.com, n.d.) and Google Trends data (Google Trends, n.d.) were collected.

Dataset Creation:

Using the Kagle dataset, two separate datasets were created based on frequency: a low-frequency daily dataset and a high-frequency minute-level dataset.

- High-Frequency Dataset:
    - The Kaggle dataset was already linked to a minute time period frequency so no additional modification was needed to create the high frequency dataset
- Low-Frequency Dataset:
    - **To create a daily frequency dataset** the OHLV minute dataset was resampled and aggregated to a daily frequency by applying the following aggregations:
        - 'open':'first',
        - 'high' : 'max',
        - 'low' : 'min',
        - 'close' : 'last',
        - 'volume' : 'sum'
    - The daily dataset was then joined to the Blockchain and Google Trends
    - **The blockchain data** was recorded at three-day intervals, with each metric starting on a different offset. To address this, I created a continuous daily date index spanning the entire time range, then aligned each blockchain metric to this index using forward fill to ensure consistency across all features
    - **Google trend** data was only available on a monthly frequency, so these values were aligned to the start of the month and forward-filled to daily values.
- **The binary classification target variable was created to measure price movements**

- **A 1 signified an upward price movement in the next 5 minutes for the minute dataset and 1 day for the daily dataset**
- **A 0 signified a downward price movement in the next 5 minutes for the minute dataset and 1 day for the daily dataset**

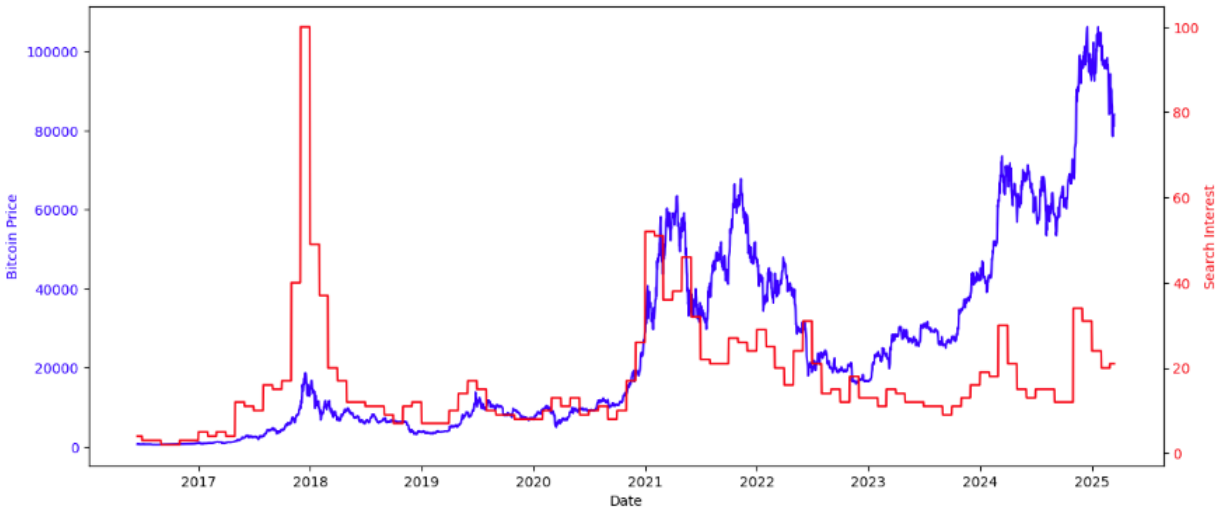
Feature Engineering:

There are 3 types of features that were applied to the daily frequency dataset. The first is Blockchain Network features, such as block size, transaction difficulty, and hash rate. They reflect the health status of the Bitcoin blockchain, which is critical for understanding network stability—key factors that influence Bitcoin's value.

The second are Bitcoin Trading and Technical features. This includes indicators like number of transactions, transaction volume, and mempool size. These features give us insight into the demand and transaction costs.

The third feature is Bitcoin Attention, which is derived from Google Trends data. Fluctuations in search interest often correlate with changes in demand and market sentiment, providing us with a valuable leading indicator for price movement. The visualization below suggests that there is strong connection between bitcoin price and Google search trends - a proxy for social sentiment. However, while the actual search interest values (0-100) don't perfectly align with the price, the changes in search interest over time tend to be more indicative of positive price movement than consistently high values alone.

Plotting Google Trends against Bitcoin ***created in Python using Seaborn**



- **Low-Frequency (Daily) Data:** High-dimensional features will be used, including blockchain activity (block size, transaction difficulty, hash rate), public attention (Google Trends), and macroeconomic signals
- **High-Frequency (Minute) Data:** Low-dimensional technical trading indicators (momentum, volatility, and moving averages) were used.

1. **Blockchain Network**
    1. Block Size
    2. Transaction Difficulty
    3. Hash Rate
2. **Bitcoin Trading and Technical (StockStats Python Library)**
    1. Number of Transactions
    2. Estimated Transaction Volume
    3. Mempool Size
    4. Mempool Transaction Count
    5. Transaction Fees
    6. Market Capitalization
3. **Bitcoin Attention**
    1. Google Trends

Machine Learning Models:

The high frequency and low frequency datasets were used to train 4 models – Linear Discriminant Analysis (LDA), Logistic Regression, Random Forest (RF), and Support Vector Machine (SVM). A GridSearchCV was used to identify the optimal parameters, and a Sequential Feature Selector was used to identify the optimal features for each model

LDA

Linear Discriminant Analysis is a supervised learning method for classification and to reduce dimensionality. Input features are projected onto a lower-dimensional space to maximize class separability by minimizing within-class variance.

Logistic Regression

Logistic Regression is a statistical method that can be used for binary classification tasks. It models the probability that an instance belongs to a particular class. It assumes a linear relationship between the independent variables and the log-odds of the dependent outcome. The table below shows which parameters were applied to this model.

	C	penalty	solver
logreg_best_params	100	l2	liblinear

SVM

Support Vector Machines are a supervised learning model designed for binary classification. It finds the hyperplane that best separates the classes in feature space. The table below shows which parameters were applied to this model.

	C	gamma	kernel
SVM_best_params	10	scale	rbf

Random Forest

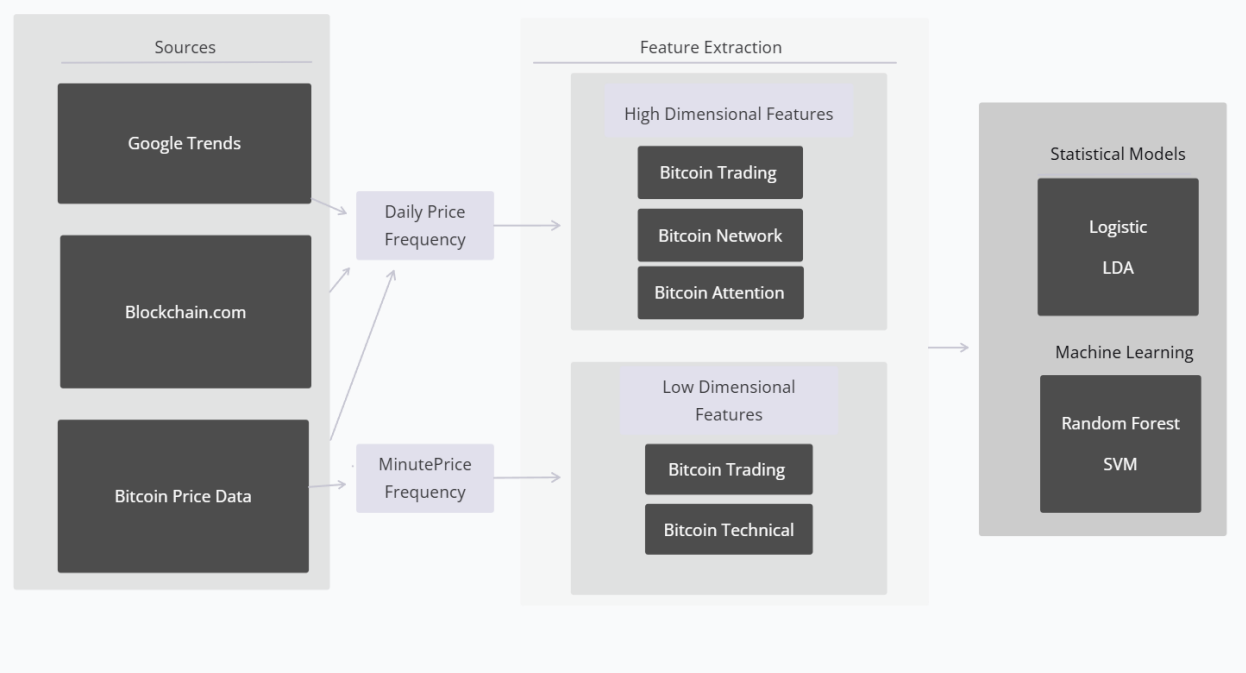
Random Forest is a type of ensemble learning method that constructs a collection of decision trees. Each collection of decision trees is trained on bootstrapped samples with randomized feature subsets. The table below shows which parameters were applied to this model.

	bootstrap	max_depth	max_features	min_samples_split	n_estimators
RF_best_params	False	5	sqrt	10	100

Training and Evaluation:

Grid search will be used for hyperparameter tuning. To maintain the sequential nature of financial data, a **TimeSeriesSplit** was used. This technique ensures that each training set precedes the next in order to prevent lookahead bias. The model is trained on historical data, and each step in the split maintains the integrity of the time sequence. Model performance will be evaluated using a confusion matrix, precision, recall, and F1-score.

3.2 Overall Framework Figure:



1. The diagram shows where the data was sourced, how it was divided into separate datasets, what the features were for each dataset, and which models were used to predict price movement.

4. Experiments

4.1 Reproduction of the Paper's results:

Daily Frequency Dataset Results:

In the original paper they achieved an average accuracy of 65% for the LDA and Logistic Regression models and an average accuracy of 55% of the Random Forest and SVM machine learning models for the daily frequency datasets. This aligned to their hypothesis "that properly selected high-dimensional feature sets can compensate for the simplicity of models in Bitcoin daily price prediction" Chen, Z., Li, C., & Sun, W. (2020). When reproducing this approach, I found the opposite to be true: the LDA and Logistic Regression performed worse with an average accuracy of 51% compared to an average accuracy of 60% for the Random Forest and SVM Models.

Minute Frequency Dataset Results:

In the original paper they achieved an average accuracy of 65% for the LDA and Logistic Regression models and an average accuracy of 55% of the Random Forest and SVM machine learning models for the daily frequency datasets. Once again aligning to their hypothesis "that properly selected high-dimensional feature sets can compensate for the simplicity of models in Bitcoin daily price prediction" Chen, Z., Li, C., & Sun, W. (2020). When reproducing this approach, the LDA and Logistic Regression performed with an average accuracy of 55% compared to an average of 54% for the Random Forest and SVM Models.

4.2 Own Experiments with Other Data:

Although I closely followed the algorithm of the original paper, I did experiments with a few different features and approaches. The first deviation from the original paper is in how I engineered my features for the daily dataset. After my initial testing of the models, I received around 50% accuracy for all the models, so I developed a function that allowed me to apply several additional features that enabled me to measure how each feature changed over a given window.

Add additional features and then choose optimal ones

```
[ ]: #currently most of my high dimensional features are absolute values and do not contain change indicators.
     # so next step is to add some momentum and change indicators and drop the raw OCHLV price data

[646]: def calculate_pct_chg(df, column_name, window):
           """input the dataframe and the columm name you want to compute the percent change on as well as what window to calculate on"""
           new_column_name = f"{column_name}_pct_chg_{window}"
           df[new_column_name] =   df[column_name].pct_change(window).shift(1)
           print(new_column_name)

       def calculate_pct_chg_roll(df, column_name, window):
           """input the dataframe and the columm name you want to compute the rolling percent change on as well as what window to calculate on"""
           new_column_name = f"{column_name}_pct_chg_roll_{window}"
           df[new_column_name] =   df[column_name].pct_change().rolling(window).mean().shift(1)
           print(new_column_name)

[648]: columns_pct_list = ['avg-block-size', 'hash-rate', 'difficulty',
              'estimated-transaction-volume-usd', 'market-cap', 'mempool-count',
              'mempool-size', 'n-transactions', 'transaction-fees-usd',
              'bitcoin_search_interest','open','close','high','low','volume']

[650]: for col in columns_pct_list:
           calculate_pct_chg(high_dimension_df, col, 2)
           calculate_pct_chg(high_dimension_df, col, 5)
           calculate_pct_chg_roll(high_dimension_df, col, 2)
           calculate_pct_chg_roll(high_dimension_df, col, 5)

       avg-block-size_pct_chg_2
       avg-block-size_pct_chg_5
       avg-block-size_pct_chg_roll_2
       avg-block-size_pct_chg_roll_5
       hash-rate_pct_chg_2
       hash-rate_pct_chg_5
       hash-rate_pct_chg_roll_2
       hash-rate_pct_chg_roll_5
       difficulty_pct_chg_2
       difficulty_pct_chg_5
       difficulty_pct_chg_roll_2
       difficulty_pct_chg_roll_5
       estimated-transaction-volume-usd_pct_chg_2
       estimated-transaction-volume-usd_pct_chg_5
       estimated-transaction-volume-usd_pct_chg_roll_2
       estimated-transaction-volume-usd_pct_chg_roll_5
       market-cap_pct_chg_2
       market-cap_pct_chg_5
```

The second experiment was to see if I could lengthen the timeframe that the models trained on. Chen, Z., Li, C., & Sun, W. (2020)'s time period for training their models was from February 2017 to February 2019 and July 17,2017 to January 17, 2018 for the daily and minute datasets, respectively. Whereas my models were trained on January 2015 to April 2025 for both the daily and minute datasets. After receiving an initial accuracy of 50% for all models, I adjusted my time period to July 17th, 2017 to January 17, 2018 for both the daily and minute datasets. This greatly increased my accuracy scores closer to 60%.

4.3 Thoughts about the Results and the Solution:

Chen, Z., Li, C., & Sun, W. (2020) hypothesized that LDA and Logistic Regression will perform better with lower-frequency data, while SVM and Random Forest are expected to show better performance with higher-frequency data. As stated before, my results show the opposite: Random Forest and Support Vector Machine models outperformed the LDA and Logistic Regression by about 4% on the daily dataset.

Low Frequency Dataset Results:

```
logreg best accuracy: 0.560
logreg best params: {'model__C': 0.01, 'model__penalty': 'l1', 'model__solver': 'saga'}
LDA best accuracy: 0.460
LDA best params: {}
SVM best accuracy: 0.580
SVM best params: {'model__C': 1, 'model__gamma': 'scale', 'model__kernel': 'linear'}
RF best accuracy: 0.600
RF best params: {'model__bootstrap': True, 'model__max_depth': 20, 'model__max_features': 'sqrt', 'model__min_samples_split': 2, 'model__n_estimator
s': 200}
```

High Frequency Dataset Results:

```
logreg best accuracy: 0.537
logreg best params: {'model__C': 0.01, 'model__penalty': 'l1', 'model__solver': 'liblinear'}
logreg selected features: ['open', 'volume']

                    0             1  accuracy      macro avg   weighted avg
precision    0.547266      0.468590   0.54634       0.507928       0.511632
recall       0.988562      0.012183   0.54634       0.500372       0.546340
f1-score     0.704514      0.023748   0.54634       0.364131       0.396182
support    72476.000000  60002.000000   0.54634  132478.000000  132478.000000
LDA best accuracy: 0.536
LDA best params: {}
LDA selected features: ['low', 'volume']

                    0             1  accuracy      macro avg   weighted avg
precision    0.547238      0.466365  0.546294       0.506802       0.510609
recall       0.988617      0.012016  0.546294       0.500317       0.546294
f1-score     0.704505      0.023429  0.546294       0.363967       0.396032
support    72476.000000  60002.000000  0.546294  132478.000000  132478.000000
```

While I do believe frequency and dimensionality play a key role in determining which models to use, based on my results, I am not able to confirm that simpler statistical models excel with lower frequency datasets with higher dimensionality. What my results suggest is that the period on which the data is trained is more important than which dimensions to use. I believe this comes down to the nature of temporal data, especially volatile data such as financial markets. Market price movement is affected by different features at different points in time. For example, in a bullish market, social sentiment and upward price movement is likely to have a greater affect on performance than in a bear market.

4.4 Challenges:

There were a few key challenges that might have affected the results of this experiment. The first was the frequency of google trend data. Historical Google Trend data is only available on a monthly frequency which means any variation in search trends is not captured until the next month. Sentiment can change quickly, and considering the results published by Ortu, M., Uras, N., Conversano, C., Bartolucci, S., & Destefanis, G. (2022) social sentiment is a strong indicator of price movement.

The second challenge was aligning the Blockchain data to the correct daily index. Each blockchain indicator has a different starting timepoint and was offset by 3 days, which

means a price movement on one given day might not have the correct Blockchain variables associated with it.

5. Conclusions

5.1 Concluding Remarks:

This project was not able to reproduce the paper's results and the models achieved lower accuracy scores than expected. However, temporal data—specifically financial data—is difficult to predict. I learned that parameters must be high tuned and the time period on which to train and test the models must be carefully chosen. I believe my scores would significantly improve if higher quality Google Trend data was obtained and if more research went into choosing an appropriate time-period. Results provide a baseline for further analysis and optimization. While this experiment did not successfully underscore the importance of considering sample dimensions in the design of prediction models, it highlighted the importance of choosing the correct parameters and time-period to improve accuracy for all models.

5.2 Possible Future Work:

Future steps involve further hyperparameter tuning and testing of the models. Exploring a wider range of features and potentially incorporating social sentiment data could be considered. Additionally, including a LTSM model in this experiment might improve accuracy scores since this model excels at time series/temporal data.

6. Contributions

Previous Work:

The architecture and approach were taken from the paper "Bitcoin price prediction using machine learning: An approach to sample dimension engineering" by Chen, Z., Li, C., & Sun, W. (2020). The approach of leveraging a low-frequency and high-frequency dataset was taken from this work. Additionally, use of the Blockchain Network, Bitcoin Attention, and Bitcoin Trading high-dimensional features were taken from this paper.

My Contributions:

I created and trained the LDA, Logistic Regression, Random Forest, and Support Vector Machines in a python environment using SKLearn.

Additionally, I performed all the data-preprocessing for each dataset. This included reindexing datapoints, removing nulls, and joining data sources together.

I created all the code needed to reproduce the results of the paper.

References:

Chen, Z., Li, C., & Sun, W. (2020). ***Bitcoin price prediction using machine learning: An approach to sample dimension engineering*****.** ***Journal of Computational and Applied Mathematics, 365*****, 112395.** https://doi.org/10.1016/j.cam.2019.112395

Chen, J. (2023). ***Analysis of Bitcoin price prediction using machine learning*****.** ***Journal of Risk and Financial Management, 16*****(1), 51.** https://doi.org/10.3390/jrfm16010051

Ortu, M., Uras, N., Conversano, C., Bartolucci, S., & Destefanis, G. (2022). ***On technical trading and social media indicators for cryptocurrency price classification through deep learning*****.** ***Expert Systems with Applications, 198*****, 116804.** https://doi.org/10.1016/j.eswa.2022.116804

Kaggle. (n.d.). ***Bitcoin historical data*****. Kaggle. Retrieved April 2025, from** https://www.kaggle.com/datasets/mczielinski/bitcoin-historical-data

Google Trends. (n.d.). ***Bitcoin keyword search interest*****. Retrieved April 2025, from** https://trends.google.com/trends/

Blockchain.com. (n.d.). ***Blockchain explorer charts and statistics*****. Retrieved April 2025, from** https://www.blockchain.com/explorer/charts

PyPI. (n.d.). ***StockStats: A Python package for financial indicators*****. Retrieved April 2025, from** https://pypi.org/project/stockstats/