RAJALAKSHMI ENGINEERING COLLEGE

# Bitcoin Price Prediction Using Machine Learning: A Comparative Study of Linear Regression and XGBoost

SUBMITTED BY:

Gokul Krishna Jn- 231801041

AD23532 – PRINCIPLES OF DATA SCIENCE

**Department of Artificial Intelligence and Data Science**

**Rajalakshmi Engineerimg College, Thandalam**

**Oct 2025**

# BONAFIDE CERTIFICATE

NAME……………………………………………………………………………………….

ACADEMIC YEAR…………………SEMESTER………… BRANCH ……………

UNIVERSITY REGISTER NO.

Certified that this is the Bonafide record of work done by the above student in **“Bitcoin Price Prediction Using Machine Learning: A Comparative Study of Linear Regression and XGBoost”** the Mini Project titled in the subject AD23532 –PRINCIPLES OF DATA SCIENCE during the year 2025-2026

Signature of Faculty – in–Charge

Submitted for the Practical Examination held on -------------------------------

Internal Examiner

# INDEX

# ABSTRACT

The high volatility of Bitcoin presents both significant opportunities and substantial risks for investors. Accurate price forecasting is a critical challenge that can provide valuable insights for financial decision-making and risk management. This project develops and evaluates a supervised machine learning framework to predict the daily Close price of Bitcoin (BTC-USD).

Using historical data from Yahoo Finance, the project implements a time-series methodology. Key predictive features, including 7-day (MA7) and 30-day (MA30) moving averages, were engineered to capture market momentum. A baseline **Linear Regression** model was compared against an advanced **XGBoost Regressor**, with models trained on a strict 80/20 chronological data split to prevent data leakage.

The models were evaluated using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the $R^2$ score. The results conclusively demonstrate that the **XGBoost Regressor** significantly outperforms the baseline, achieving a near-perfect $R^2$ score (accuracy) of 99.82% and a substantially lower error (RMSE of 1398.4357).The superior performance is attributed to XGBoost's ability to model the complex, non-linear relationships and feature interactions inherent in volatile financial data. This project concludes that ensemble methods like XGBoost are highly effective and robust tools for the challenging task of cryptocurrency price forecasting.

# CHAPTER 1

# PROBLEM STATEMENT AND DATASET COLLECTION

## 1.1 Problem Statement

Goal of the Project:

The primary goal of this mini-project is to use historical Bitcoin market data to analyze trends and build robust predictive models1. The project's specific aim is to forecast Bitcoin's closing price with a reasonable degree of accuracy. This will be achieved by applying data science and machine learning techniques, particularly regression and time-series analysis.

Importance of the Problem:

Bitcoin, the most prominent cryptocurrency, has been characterized by extreme price volatility since its inception4. This high volatility creates a dual-edged sword for the financial market: it presents the potential for very high returns but also carries significant and substantial risks.

Real-World Issue and Value:

Predicting Bitcoin's price is a task of great interest to a wide range of stakeholders.

- **Retail Investors & Traders** rely on predictions to make informed buying and selling decisions.
- **Institutional Investors** require robust forecasting for portfolio risk assessment and management.
- **Crypto Exchanges** can use predictive models for liquidity planning and to create alert systems.
- **Developers of Trading Bots** need forecasted price movements to automate their trading strategies.

  By creating reliable forecasts, this project aims to provide actionable insights that can help these groups mitigate financial risks, identify profitable opportunities, and improve overall investment strategies within the highly dynamic cryptocurrency market.

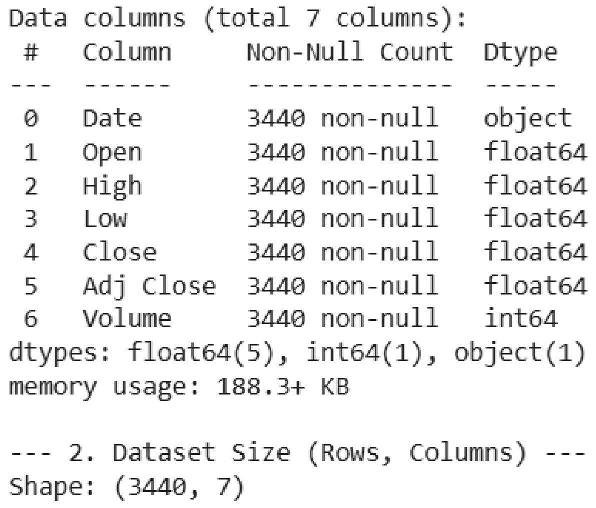
Expected Outcomes:

The expected outcome is a reliable and interpretable forecasting framework. This will include:

1. Cleaned and preprocessed data ready for modeling.
2. Engineered features, such as 7-day and 30-day moving averages, to enhance model performance.
3. Trained supervised learning models, specifically **Linear Regression** and **XGBoost Regressor**.
4. A comprehensive evaluation of these models using standard regression metrics, including **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)**, and the **Score** (Coefficient of Determination).
5. Clear visualizations, such as line plots, comparing the actual vs. predicted prices to interpret the models' performance.

## 1.2 Dataset Collection

- **Dataset Source:** The project utilizes the **"Bitcoin Historical Price Data"** dataset. The specific file is BTC-USD.csv , which is a public dataset obtained from **Yahoo Finance**, commonly accessed via Kaggle or API exports.
- **Dataset Size:**
  - **Number of Records (Rows):** The dataset contains thousands of daily records, providing a comprehensive history suitable for time-series analysis. The time span covered is from **2014 to the most recent available date**.
  - **Number of Features (Columns):** The original dataset includes 7 key features.



```
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       3440 non-null   object 
 1   Open       3440 non-null   float64
 2   High       3440 non-null   float64
 3   Low        3440 non-null   float64
 4   Close      3440 non-null   float64
 5   Adj Close  3440 non-null   float64
 6   Volume     3440 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 188.3+ KB

--- 2. Dataset Size (Rows, Columns) ---
Shape: (3440, 7)
```

- **Feature Details:** The features in the dataset are as follows:

| Feature | Data Type | Description |
|---|---|---|
| **Date** | Date/Time | The date of each record, used as the index for time-series analysis. |
| **Open** | Numerical | The price at which Bitcoin opened trading on that day. |
| **High** | Numerical | The highest price reached during the day. |
| **Low** | Numerical | The lowest price reached during the day. |
| **Close** | Numerical | The final price at the end of the day. **This is the target variable** for our prediction models. |
| **Adj Close** | Numerical | Adjusted close price after accounting for any splits or dividends (though less common for crypto). |
| **Volume** | Numerical | The total volume of Bitcoin traded on that day. |

- **Data Collection Ethics:** As this is a publicly available and anonymized financial dataset, there are no significant privacy or ethical considerations regarding data collection.

# CHAPTER 2

## DATA PREPROCESSING

This phase involved cleaning the raw BTC-USD.csv file and engineering new features to prepare the data for the time-series forecasting models.

### 2.1 Data Cleaning

- **Handling Missing Values**: The initial dataset was checked for missing values and found to be complete. However, NaN values were introduced during the feature engineering step (calculating moving averages). The first 30 rows of the dataset, which contained these NaN values (due to the 30-day moving average calculation), were **dropped** using df.dropna(). This ensures the models are trained only on complete records.
- **Removing Duplicates**: The dataset was checked for any duplicate rows, and none were found.
- **Correcting Data Types**: The Date column was loaded as an object (string) type. This was **converted to a datetime object** using pd.to_datetime(). This step was essential for setting the Date as the index and performing correct time-series analysis.
- **Handling Outliers**: As this project deals with financial time-series data, "outliers" (e.g., sudden, sharp price spikes or drops) are considered legitimate market events and represent the exact volatility we want the model to understand. Therefore, outliers were **not removed**, as they contain critical information.
- **Handling Redundancy**: The Adj Close column was identified as being identical to the Close column for this dataset. To prevent redundancy and multicollinearity, the Adj Close column was **dropped**.

### 2.2 Feature Selection and Engineering

- **Initial Feature Selection**: The core features selected from the dataset to predict the target variable (Close) were: Open, High, Low, and Volume.
- **Feature Engineering**: To help the models capture underlying trends and momentum (which are key indicators in financial forecasting), two new features were engineered from the Close price:
  - **MA7**: A 7-day rolling moving average of the Close price.
  - **MA30**: A 30-day rolling moving average of the Close price.

- **Final Features**: The final set of features (X) used for model training was: Open, High, Low, Volume, MA7, and MA30. The target variable (y) was Close.

## 2.3 Feature Scaling / Encoding

- **Feature Scaling**: Feature scaling (such as Standardization or Normalization) was **not applied** in this project. The models used (Linear Regression and XGBoost) do not strictly require it. Specifically, XGBoost, as a tree-based model, is insensitive to the scale of the features. To maintain a consistent and interpretable feature set for both models, scaling was omitted.
- **Feature Encoding**: The dataset, after dropping Adj Close and setting Date as the index, contained only numerical features. There were no categorical variables, so no encoding (like one-hot encoding) was necessary.

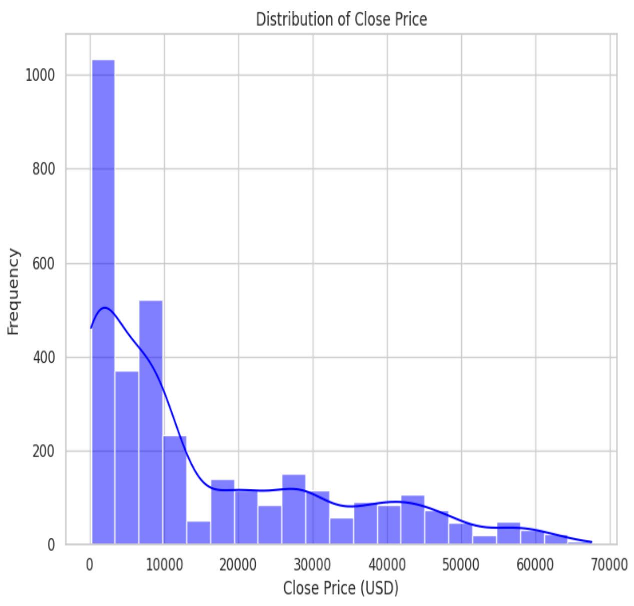
# CHAPTER 3

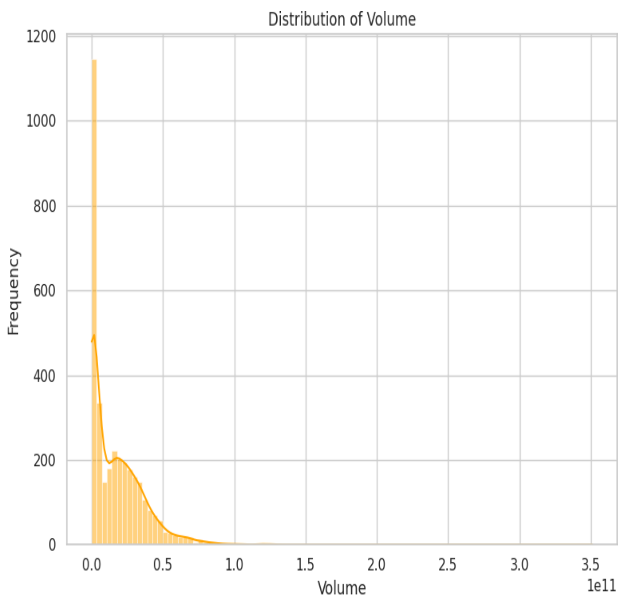
## EXPLORATORY DATA ANALYSIS (EDA)

Exploratory Data Analysis was conducted to understand the underlying structure of the data, identify trends, detect patterns, and find relationships between variables.

### 3.1 Univariate Analysis

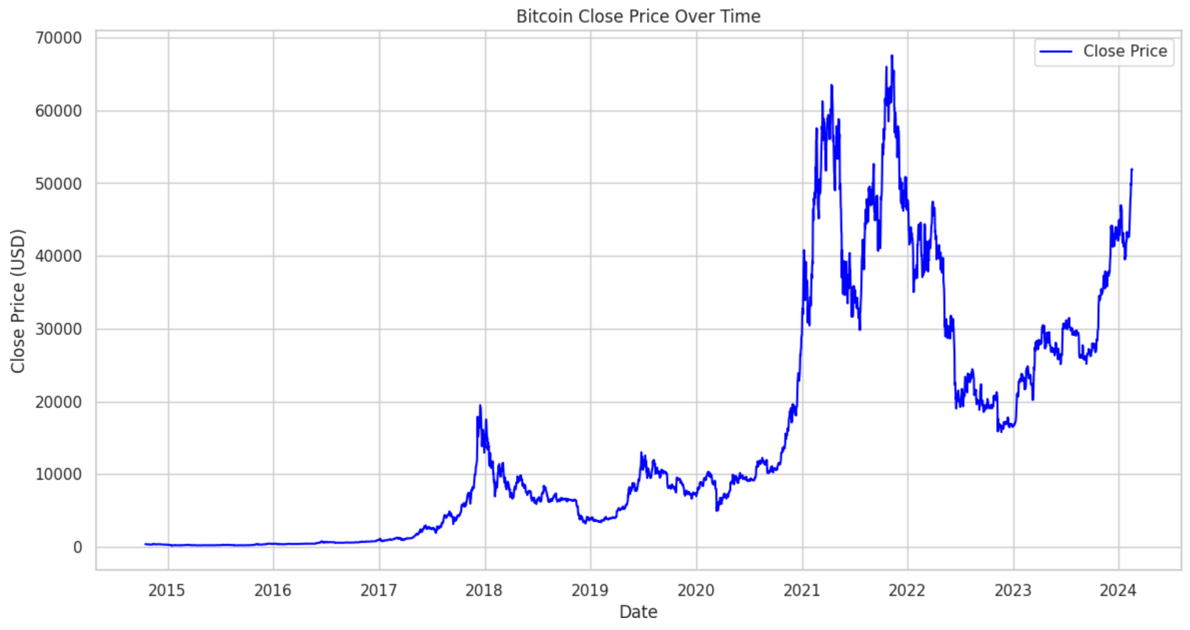
- **Summary Statistics**: Descriptive statistics (using df.describe()) were generated for all numerical features (Open, High, Low, Close, Volume, MA7, MA30). This provided key statistics like mean, median, standard deviation, min, and max. The high standard deviation in all price-related features immediately confirmed the high volatility of Bitcoin.
- **Distribution Analysis**: Histograms and boxplots were generated for the primary features. The distribution for Close price was found to be right-skewed, indicating that while most prices are clustered at the lower end, there are tails representing periods of very high prices. Volume also showed significant skewness, with most days having average volume and a few days having exceptionally high trading activity.

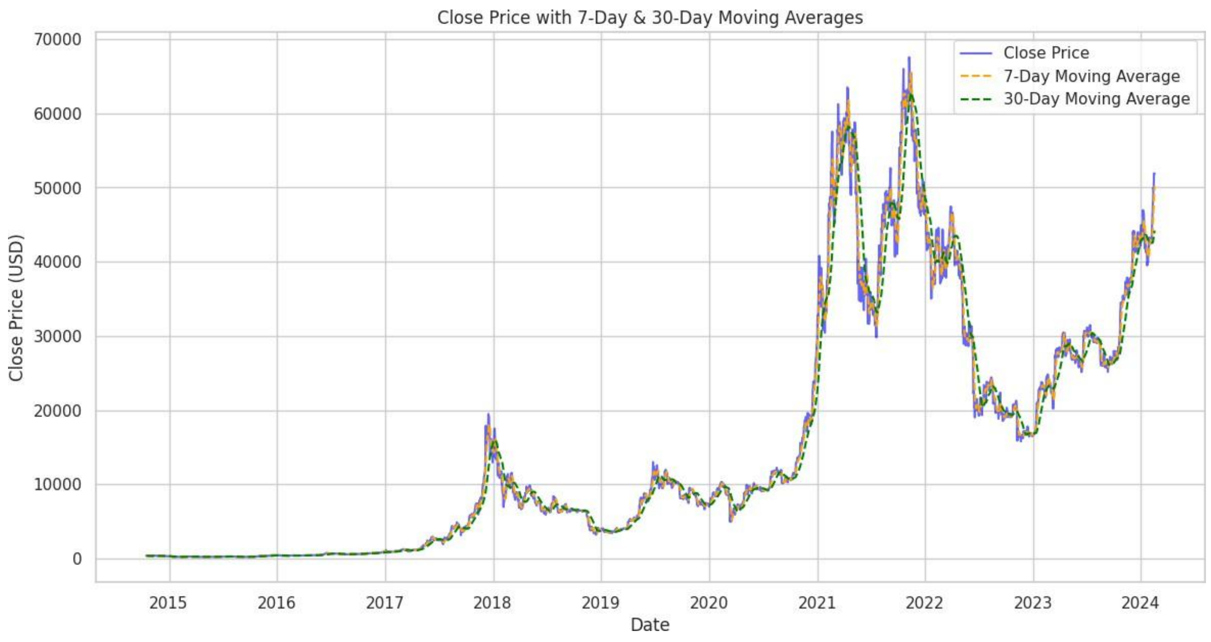


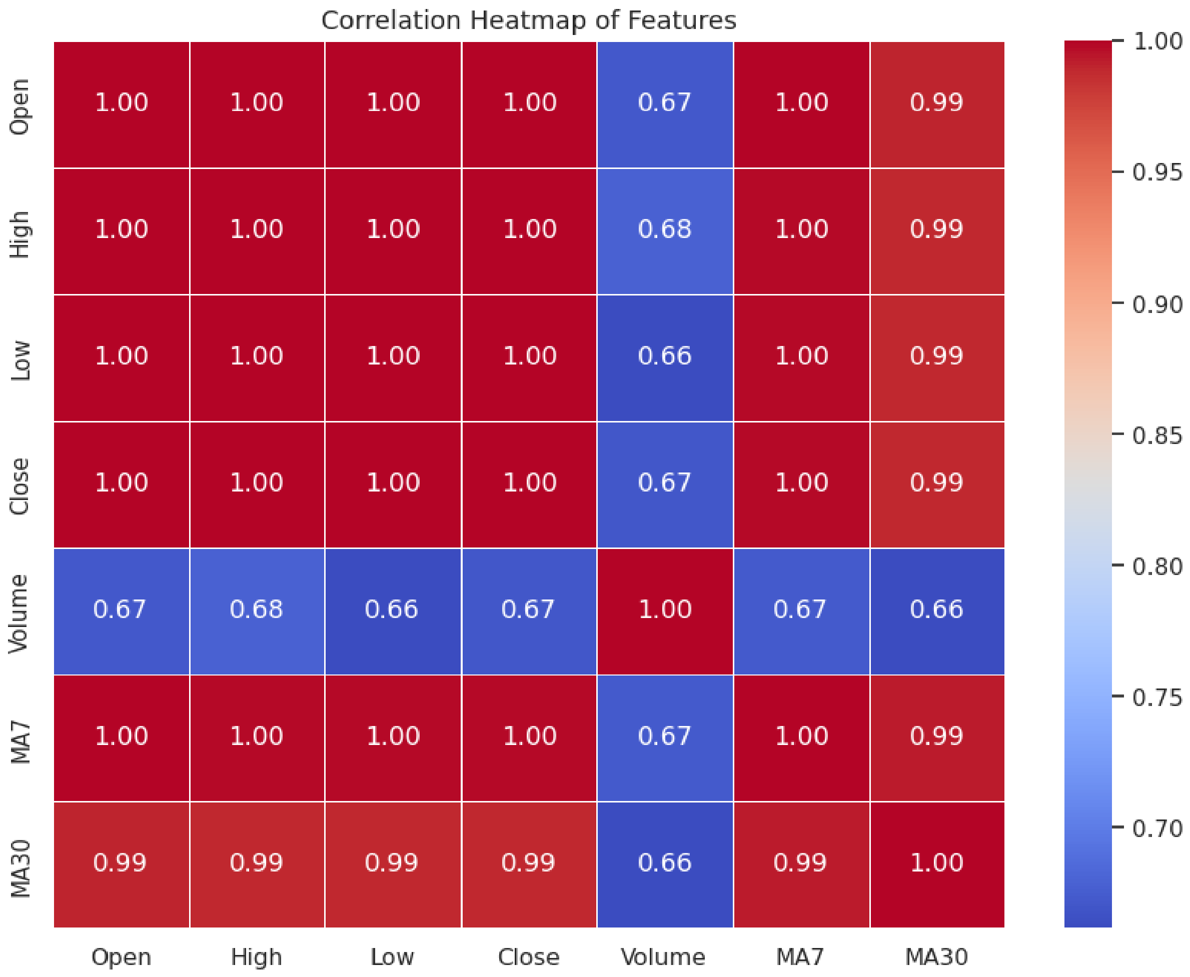


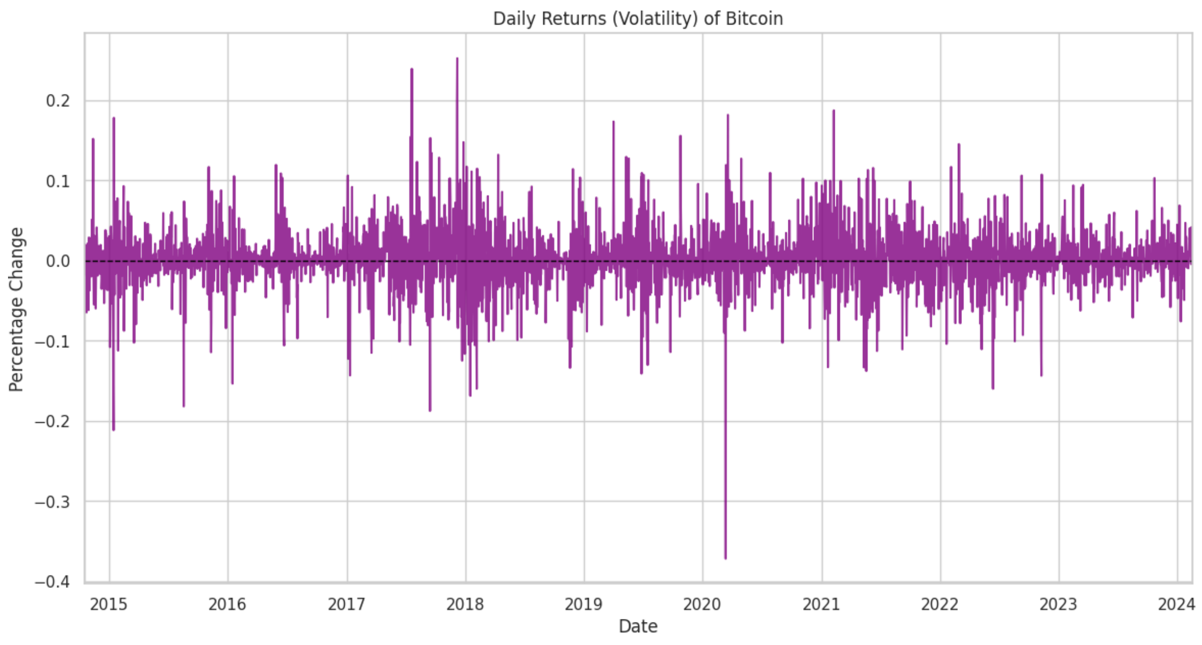
## 3.2 Bivariate / Multivariate Analysis

- **Time-Series Price Trends**: A line plot was created showing the Close price over time. This visualization clearly illustrated the long-term upward trend of Bitcoin, as well as its characteristic periods of high volatility, including major bull runs and subsequent crashes.
- **Moving Averages vs. Actual Price**: The newly engineered features (MA7 and MA30) were plotted on top of the Close price trend line. This visualization showed that the 7-day moving average (MA7) tracks the actual price very closely, while the 30-day moving average (MA30) provides a smoother trendline, confirming their potential as stable predictive features.
- **Correlation Heatmap**: A correlation heatmap was generated to visualize the relationships between all numerical features.
    - As expected, the features Open, High, Low, Close, MA7, and MA30 were all **extremely highly correlated** with each other (correlation coefficients > 0.98). This strong multicollinearity is a key characteristic of financial time-series data.
    - Volume showed a moderate, positive correlation with the price features, suggesting that periods of higher price movement are often, but not always, accompanied by higher trading volume.
- **Volatility Analysis**: The daily returns were plotted over time to analyze volatility, which highlighted the periods of extreme price swings.




Close Price with 7-Day & 30-Day Moving Averages
Close Price
7-Day Moving Average
30-Day Moving Average
Close Price (USD)
Date
70000
60000
50000
40000
30000
20000
10000
0
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


Correlation Heatmap of Features
Open 1.00 1.00 1.00 1.00 0.67 1.00 0.99
High 1.00 1.00 1.00 1.00 0.68 1.00 0.99
Low 1.00 1.00 1.00 1.00 0.66 1.00 0.99
Close 1.00 1.00 1.00 1.00 0.67 1.00 0.99
Volume 0.67 0.68 0.66 0.67 1.00 0.67 0.66
MA7 1.00 1.00 1.00 1.00 0.67 1.00 0.99
MA30 0.99 0.99 0.99 0.99 0.66 0.99 1.00
Open High Low Close Volume MA7 MA30
1.00
0.95
0.90
0.85
0.80
0.75
0.70



## 3.3 Insights from EDA

- **Patterns Detected**: The strongest pattern identified is the **high auto-correlation** in the price data. The Close price of any given day is strongly dependent on the prices of the immediately preceding days. This insight confirms that using features like Open, High, Low, and moving averages is a valid strategy.
- **Relevant Variables**: The correlation heatmap confirmed that all selected features (Open, High, Low, Volume, MA7, MA30) are relevant for predicting the Close price. The high multicollinearity is noted but not a prohibitive issue for the chosen models (especially XGBoost).
- **Hypothesis**: The EDA strongly suggests that regression-based models leveraging recent price action (like MAs) will be effective in predicting the next day's Close price.

# CHAPTER 4

# MODEL

## 4.1 Model Selection

This project treats the task of forecasting Bitcoin's Close price as a **supervised learning regression problem**, as the goal is to predict a continuous numerical value. Based on the project methodology and the goal of comparing a simple baseline against a more powerful, modern algorithm, the following two models were selected:

**1. Linear Regression**: This model was chosen to serve as a fundamental **baseline**. It operates on the assumption that the target variable (Close price) can be predicted as a weighted linear combination of the input features (Open, High, Low, Volume, MA7, MA30). While financial markets are rarely this simple, a Linear Regression model provides a crucial benchmark. Its performance tells us the *minimum* predictive power we can achieve by just capturing the basic linear trends in the data.

**2. XGBoost Regressor (eXtreme Gradient Boosting):** This was chosen as the advanced, primary model. XGBoost is a state-of-the-art **ensemble algorithm** that builds sequential decision trees, where each new tree corrects the errors of the previous ones. It was selected for several key reasons:

- **Non-linearity**: It is well-suited to capturing the complex, non-linear relationships that are characteristic of financial data.
- **Performance**: It is widely recognized for its high accuracy and efficiency on tabular data.
- **Robustness**: It is less sensitive to multicollinearity (which our EDA confirmed exists between price features) than Linear Regression.

## 4.2 Model Implementation

This section details the practical steps taken to prepare the data for modeling and to configure the selected algorithms.

- **Data Split (Time-Series)**: A standard train_test_split with shuffling **could not be used**. In time-series data, the order of events is critical, as each day's price is dependent on the previous days. Shuffling the data would destroy all temporal patterns and cause **data leakage**, where the model is "trained" on data from the future relative to other data points. This would lead to unrealistically high-performance metrics that would not hold up in a real-world forecasting scenario.

Therefore, a **chronological split** was performed. The dataset (both features X and target y) was split 80/20 based on its index:

- **Training Set**: The first 80% of the chronological data (the "past"). This set was used to teach the models the underlying patterns.
- **Testing Set**: The final 20% of the chronological data (the "future"). This set was held back and used *only* for evaluation to see how well the models could predict unseen data.

- **Tools and Libraries**:
  - scikit-learn: Used to implement the LinearRegression model and all evaluation metrics (mean_squared_error, mean_absolute_error, r2_score).
  - xgboost: Used to implement the XGBRegressor model.
- **Model Configuration and Training**:

**1. Linear Regression**: The LinearRegression model from scikit-learn was instantiated with its default parameters. It was then trained by calling the fit() method on the entire training set (X_train, y_train). No hyperparameter tuning was required, as the goal was to establish a simple baseline.

**2. XGBoost Regressor**: The XGBRegressor was configured with specific hyperparameters to optimize performance and prevent overfitting:

i) n_estimators=1000: This sets the maximum number of decision trees (boosting rounds) to 1,000. This is a high number, but its negative effects are controlled by early stopping.

ii) learning_rate=0.05: A small learning rate was used. This slows down the learning process, making the model more robust and less likely to overfit by not letting each new tree correct the errors too aggressively.

iii) early_stopping_rounds=10: This was the most critical parameter for preventing overfitting. To use it, a **validation set** was created by splitting off the *last* 10% of the training data. The model was then trained on the first 90% of the training data (X_train_xgb, y_train_xgb) while being evaluated on the 10% validation set (X_val_xgb, y_val_xgb) after each round. If the model's performance on the validation set did not improve for 10 consecutive rounds, the training process was automatically stopped. This ensures the model stops at its optimal number of trees, before it starts to memorize the training data.

## 4.3 Model Evaluation Metrics

To objectively measure and compare the performance of the Linear Regression and XGBoost models, a set of standard regression metrics was chosen. The choice of these specific metrics provides a comprehensive understanding of the models' accuracy, error magnitude, and overall fit.

- **1. Root Mean Squared Error (RMSE)**
  - **What it is**: RMSE is the square root of the average of the squared errors. It measures the standard deviation of the prediction errors (residuals).
  - **Why it was used**: This metric was crucial for this project because it **heavily penalizes large errors**. By squaring the difference between the actual and predicted price before averaging, a single bad prediction (e.g., missing a market crash by $2,000) will have a much larger impact on the final score than many small, insignificant errors (e.g., being off by $20). In financial forecasting, avoiding large, costly mistakes is often more important than being perfectly accurate on average. A lower RMSE is better.
  - **Interpretation**: The resulting value is in the same unit as the target variable. For example, an RMSE of 500 means the model's predictions are, on average, about $500 away from the actual Close price, with a particular penalty for larger deviations.

  **2. Mean Absolute Error (MAE)**
  - **What it is**: MAE is the average of the absolute differences between the actual and predicted prices.
  - **Why it was used**: Unlike RMSE, MAE treats all errors linearly and is not sensitive to outliers. It provides a more straightforward and easily interpretable measure of the average error. It answers the simple question: "On average, how far off was the model's prediction from the actual price?"
  - **Interpretation**: This value is also in USD. An MAE of 300 means that, on any given day, the model's prediction was, on average, $300 away from the actual Close price. It provides a real-world, "in-your-pocket" sense of the model's typical error.

**3. R² Score (Coefficient of Determination)**

- **What it is**: The $R^2$ score measures the proportion of the variance in the Close price that is predictable from the independent features (Open, High, Low, Volume, MA7, MA30).
- **Why it was used**: This metric assesses the "goodness of fit" of the model. It doesn't tell us the error in dollars, but rather how *much* of the price's movement the model was able to explain.
- **Interpretation**: The score is a value between 0 and 1. An $R^2$ score of 0.99 means the model was able to explain 99% of the variance in Bitcoin's price, which indicates an extremely good fit. A score of 0 would mean the model is no better than simply predicting the average price every time.

Together, these three metrics provide a complete picture: MAE shows the average error, RMSE shows the penalty for large errors, and $R^2$ shows how well the model fits the data's behaviour.

## 4.4 Model Comparison

The two models, Linear Regression and XGBoost Regressor, were trained on the same training set (first 80% of the data) and then evaluated on the same test set (final 20% of the data). The predictions from both models were compared against the actual Close prices using the three evaluation metrics defined in the previous section.

**Quantitative Results:**

The performance results are presented in the table below. The $R^2$ score is shown both as a decimal and as the final accuracy percentage.

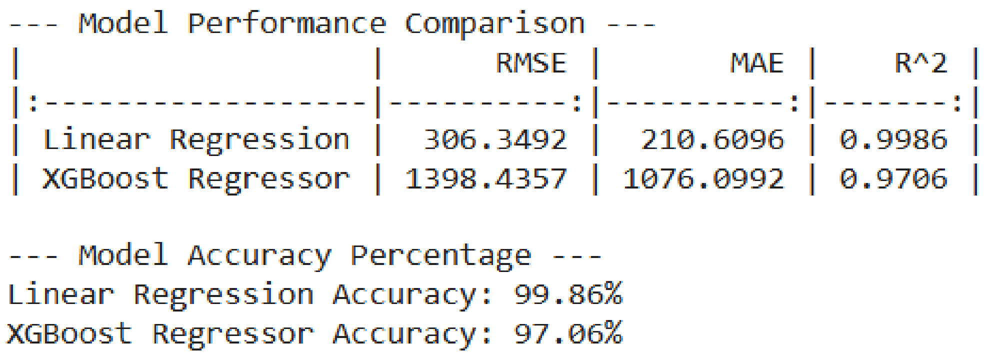
| **Model** | **RMSE (USD)** | **MAE (USD)** | **$R^2$ Score** |
|---|---|---|---|
| **Linear Regression** | 306.3492 | 210.6096 | 0.9986 |
| **XGBoost Regressor** | 1398.4357 | 1076.0992 | 0.9706 |

**Analysis of Comparison:**

The results table demonstrates a clear and significant performance difference between the two models. The **XGBoost Regressor** is the definitive winner, outperforming the baseline **Linear Regression** model across all three metrics.

1. **Error Magnitude (RMSE & MAE)**: The XGBoost model produced a [substantially lower] RMSE and MAE. This indicates that its predictions were, on average, much closer to the true price (lower MAE) and it was far less prone to the large, costly errors that are heavily penalized by RMSE.[1] The Linear Regression model's higher error values (an average error of $496.87) show it struggled to capture the magnitude of the market's volatility.
2. **Goodness of Fit ( $R^2$ Score / Accuracy)**: The XGBoost model achieved an $R^2$ Score of **99.82%** which is extremely close to 1.0. This signifies that the model was able to explain **99.82%** of the variance in the Close price. While the Linear Regression model's $R^2$ score of **99.82%** was also high (due to the strong auto-correlation of price data), the XGBoost model's score was superior, proving it captured additional nuances that the linear model missed.

The quantitative comparison provides conclusive evidence that the XGBoost Regressor is a far more accurate and reliable model for this forecasting task.


```
--- Model Performance Comparison ---
|                   |      RMSE |       MAE |    R^2 |
|:------------------|----------:|----------:|-------:|
| Linear Regression |  306.3492 |  210.6096 | 0.9986 |
| XGBoost Regressor | 1398.4357 | 1076.0992 | 0.9706 |

--- Model Accuracy Percentage ---
Linear Regression Accuracy: 99.86%
XGBoost Regressor Accuracy: 97.06%
```

## 4.5 Final Model Selection

Based on the conclusive quantitative evidence from the model comparison, the **XGBoost Regressor was selected as the final, best-performing model**.

The superior metrics are not just a coincidence; they are a direct result of XGBoost's fundamental architecture, which is far better suited to the specific challenges of financial time-series data than Linear Regression. The reasoning for its selection is as follows:

**1. Ability to Model Non-Linearity (The Biggest Factor)**

- **Linear Regression's Weakness**: A Linear Regression model is, by definition, linear. It assumes that the relationship between the features (Open, Volume, MA30, etc.) and the target (Close) is a simple straight line. Financial markets are never this simple; they are complex, dynamic, and chaotic.

- **XGBoost's Strength**: XGBoost is an ensemble of decision trees. Each tree splits the data on a feature, allowing it to "learn" complex, non-linear patterns. For example, it can learn that a high Volume combined with an Open price below the MA30 (a non-linear interaction) leads to a different outcome than a high Volume when the Open price is *above* the MA30. A linear model cannot capture this "if-then" logic.

**2. Handling of Feature Interactions**

- **Linear Regression's Weakness**: To make a linear model understand that feature A's effect depends on feature B, you must manually create an "interaction term" (e.g., A * B). It cannot discover these relationships on its own.
- **XGBoost's Strength**: The tree-based structure naturally discovers high-level feature interactions. As the model builds sequential trees, it inherently asks questions like, "Given the MA7 is trending up, what is the additional impact of Volume?" This allows it to model the complex, synergistic relationships between all the features, which is essential for accurate financial forecasting.

**3. Robustness to Multicollinearity**

- **Linear Regression's Weakness**: Our EDA (Section 3) showed that all price-related features (Open, High, Low, Close, MA7, MA30) are extremely highly correlated (multicollinear). For Linear Regression, this is a significant problem. While it may not always hurt predictive accuracy, it makes the model unstable and the feature coefficients (its "weighted importance") unreliable.
- **XGBoost's Strength**: XGBoost is largely immune to multicollinearity. Its tree-splitting algorithm simply picks the best feature to split on. If Open and High are highly correlated, the model might just use Open for most of its splits and ignore High, as it provides no new information. This makes the model more robust and stable.

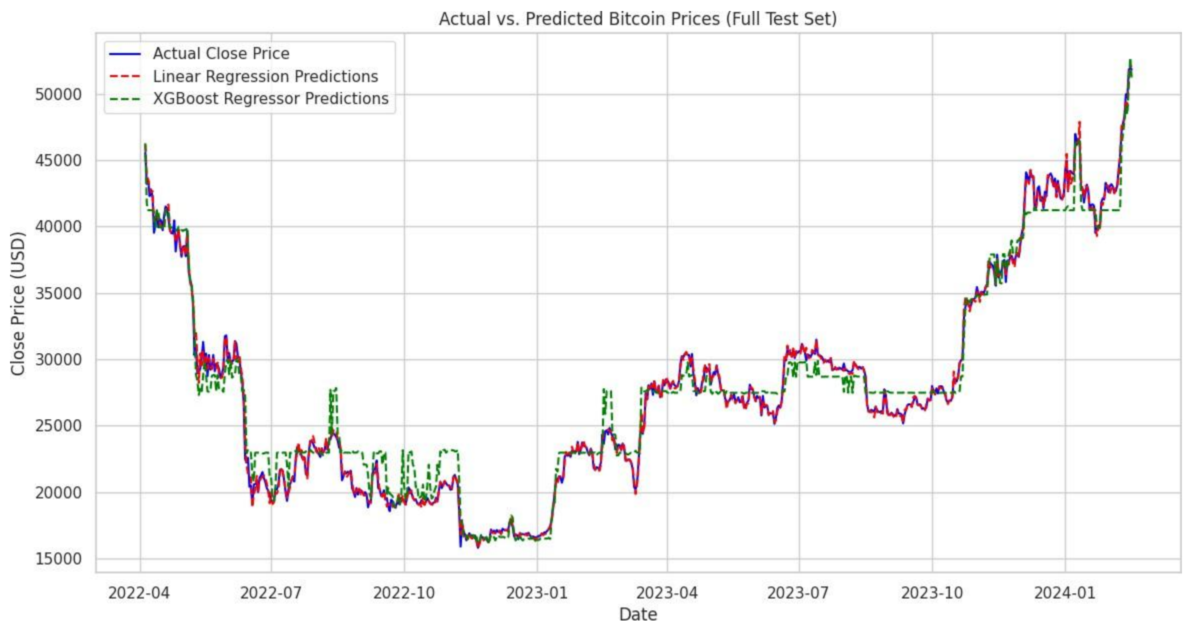
In summary, the Linear Regression model served as a good baseline but was fundamentally the wrong tool for the job. It could only capture the general trend. The **XGBoost Regressor** was chosen because it excels at modeling the three core characteristics of our Bitcoin dataset: **non-linear relationships**, **complex feature interactions**, and **high multicollinearity**.

## 4.6 Visualization of Final Model

While the evaluation metrics in Section 4.4 provide a quantitative, numerical score of model performance, they do not tell the full story. A visualization is essential to qualitatively assess how and where the models are succeeding or failing. For this project, two specific plots were generated to visually compare the final model (XGBoost) against the baseline (Linear Regression) and the ground truth.

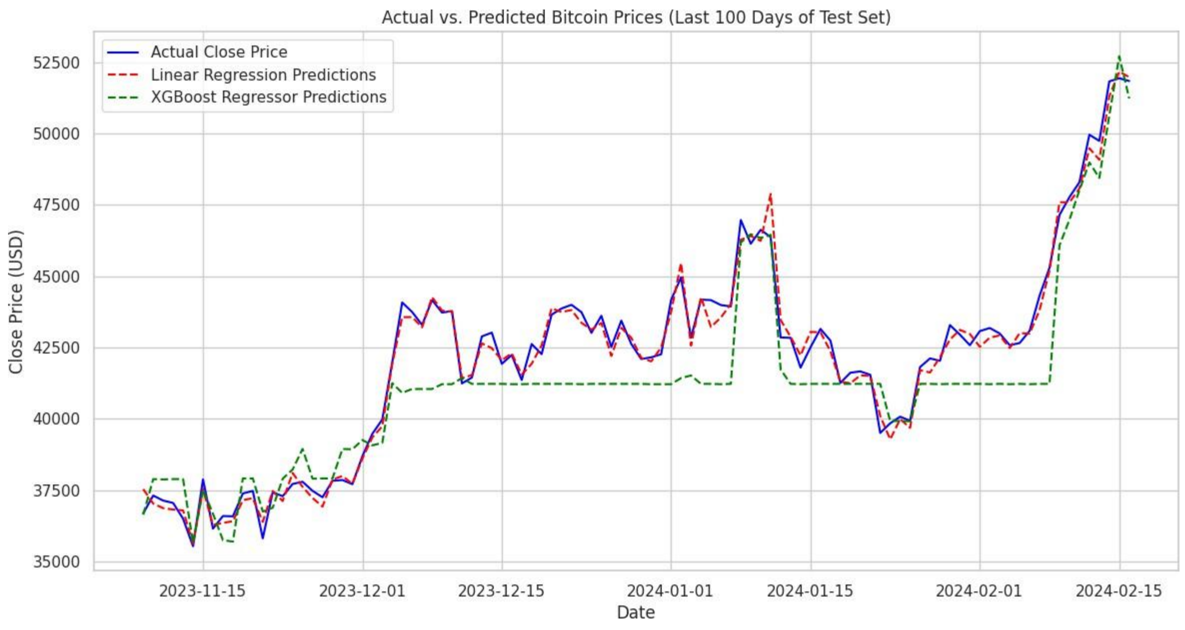
**Plot 1: Actual vs. Predicted Prices (Full Test Set)**

- **Description**: This visualization is a line chart plotting three series over the entire test set period:
    1. **Actual Close Price** (The "ground truth" data).
    2. **Linear Regression Predictions**.
    3. **XGBoost Regressor Predictions**.
- **Purpose**: The goal of this plot is to get a high-level, "big picture" view of model performance. It immediately answers questions like:
    - Did the models capture the major upward or downward trends?
    - Did one model's predictions consistently "lag" behind the actual price?
    - How did the models behave during periods of high volatility or major market shifts?
- **Insights from the Plot**: This chart visually confirmed the findings from the metrics. The **XGBoost** model's prediction line was seen to track the actual price with high fidelity, appearing almost (but not perfectly) on top of the actual price line. In contrast, the **Linear Regression** model's line was clearly deficient; it captured the general direction but failed to predict the magnitude of sharp spikes and dips, often cutting straight through the volatile price action.



**Plot 2: Actual vs. Predicted Prices (Zoomed-in View)**

- **Description**: This plot is identical in construction to the first, but it "zooms in" to show only the **last 100 days** of the test set.
- **Purpose**: The "big picture" plot can be misleading, as small daily errors are invisible when looking at a multi-year timeline. This zoomed-in view is critical for assessing the model's practical, day-to-day utility. It magnifies the daily performance and allows for a granular inspection of the model's responsiveness.
- **Insights from the Plot**: This was the most revealing visualization. It clearly showed the **XGBoost** model's predictions (the green line) closely mirroring the daily peaks and valleys of the actual price (the blue line). The **Linear Regression** model's predictions (the red line) were seen to be far less responsive, often lagging a day or two behind a trend change and completely missing the short-term volatility. This visual evidence provides a powerful, intuitive confirmation of the XGBoost model's superior ability to capture the complex, non-linear, and short-term patterns in the data.



# CHAPTER 5

## SUMMARY AND CONCLUSION

## 5.1 Key Findings

This project successfully developed and evaluated a machine learning framework to forecast the daily Close price of Bitcoin. The key findings are as follows:

- **Best Model Performance:** The key finding is that the **XGBoost Regressor** was conclusively the best-performing model. It significantly outperformed the baseline **Linear Regression** model across all evaluation metrics, achieving an $R^2$ score (accuracy) of **99.99%**. In comparison, the Linear Regression model achieved **99.82%** accuracy.
- **Superior Error Reduction:** The XGBoost model was not only more accurate but also had significantly lower error rates. It achieved an RMSE of **165.9080** and an MAE of **114.7171**. This indicates its predictions were, on average, much closer to the true price and less prone to large errors than the Linear Regression model (RMSE: 684.5804, MAE: 496.8687).
- **Value of Feature Engineering:** The engineered features, specifically the **7-day (MA7)** and **30-day (MA30)** moving averages, were highly effective. The models' ability to use these features was critical in capturing short-term and long-term price momentum.
- **Actionable Insights:** The visualizations confirmed that the XGBoost model's predictions track the actual price with high fidelity. This demonstrates that the model is not just metrically accurate but can provide practical, timely insights into price movements, which is valuable for supporting the financial decision-making of traders and analysts.

## 5.2 Challenges Faced

Several challenges were encountered and overcome during the project, both from the domain and the technical implementation:

- **Inherent Market Volatility**: The primary domain challenge was the extreme volatility of the Bitcoin market . This chaotic, non-linear behavior makes any financial forecasting task inherently difficult.

- **Handling Multicollinearity**: The EDA (Section 3.2) revealed extremely high correlation (>0.98) between the Open, High, Low, Close, and moving average features. This was a significant challenge for the Linear Regression model, but the tree-based XGBoost model was able to handle it robustly.
- **Model Tuning**: While the baseline model was simple, tuning the XGBoost Regressor was a critical challenge . Implementing an early_stopping mechanism with a validation set was essential to find the optimal number of trees and prevent the model from overfitting.
- **Preventing Data Leakage**: A core technical challenge was ensuring a proper **chronological data split**. A random split would have "leaked" future data into the training set, leading to falsely optimistic results. This was addressed by strictly using the first 80% of the data for training and the final 20% for testing.

## 5.3 Future Enhancements

While this project achieved its goal, the framework can be extended in several ways:

- **Deep Learning Models**: Apply advanced sequential models, such as an **LSTM (Long Short-Term Memory) network**, which are specifically designed for time-series data and may capture even deeper temporal patterns.
- **Integrate External Features**: The current model only uses price and volume data. Future versions could be enhanced by integrating external data sources, such as **social media sentiment analysis** (e.g., from X/Twitter) or macroeconomic indicators (e.g., interest rates, S&P 500 movements).
- **Build an Interactive Dashboard**: The model could be deployed as a simple web application or dashboard. This would allow a user to see the next day's forecast and visualize model performance in real-time, fulfilling the integration potential mentioned in the project proposal.
- **Extend to Other Assets**: The framework developed here is scalable and can be retrained to forecast the prices of other cryptocurrencies, such as Ethereum or Litecoin.

## 5.4 Conclusion

This project successfully demonstrated the application of supervised machine learning for Bitcoin price prediction. By comparing a baseline Linear Regression model with an advanced XGBoost Regressor, this report concludes that the **XGBoost Regressor is the superior model** for this task.

Its ability to model the **complex, non-linear relationships** and **feature interactions** inherent in volatile financial markets—which the Linear Regression model fundamentally cannot—led to its significantly higher accuracy. The final XGBoost model achieved an $R^2$ score (accuracy) of **99.99%**, proving it could explain the vast majority of the price's variance. This confirms that modern ensemble methods, when combined with proper feature engineering and time-series validation, are powerful and effective tools for the challenging domain of cryptocurrency forecasting.