Bitcoin Price Prediction Using Machine Learning

Student Name : Maryam Mohammed Alwuthaynani

Student ID: 44103013

Course Name : Machin Learning

Instructor Name: Dr. Nada Altuwairqi

Table of content

List of Figures

Project Overview

My project focuses on predicting the future direction of Bitcoin prices using a range of machine learning classification models. Given Bitcoin's volatile nature and its significance as a global digital asset, it serves as a compelling and challenging candidate for time-series prediction.

The goal of the project is not only to apply machine learning models but also to explore their behavior on real financial data, understand their strengths and limitations, and identify which models are most suitable for this domain.

Throughout the project, I investigated the structure of the dataset, performed a complete preprocessing workflow, trained various models, evaluated their performance using both numerical metrics and visual tools, and documented the insights derived from the entire process.

Context and Motivation

Forecasting price direction in financial markets requires dealing with uncertainty, volatility, and dynamic behavior — making cryptocurrency an ideal but challenging domain for applied machine learning, This data's financial and temporal structure makes it an ideal but challenging candidate for classification tasks, as confirmed by recent research in cryptocurrency forecasting (Helformer, 2025).

For this reason, I chose to focus on Bitcoin, the most prominent digital asset, using historical daily data available on Kaggle that captures price movements, trading volume, and market behavior.

I selected Bitcoin for three key reasons:

1. **Real-World Impact:** As the leading cryptocurrency with a $1.2T market cap (CoinMarketCap, 2023), even marginal predictive accuracy can inform trading strategies and economic policies.

2. **Academic Challenge:** Studies show Bitcoin's price follows a "random walk with drift" (Bariviera, 2017), making it a rigorous test for ML models' ability to detect subtle patterns.

3. **Technical Appeal:** The dataset's high volatility and non-stationarity mirror real-world ML deployment challenges, unlike sanitized academic datasets.

This project extends prior work by:

- Evaluating classical models on raw price data
- Providing a reproducibility benchmark with full code and preprocessing transparency

Dataset Description

The dataset that I used in this project was obtained from Kaggle, a public machine learning repository. It is titled “Cryptocurrency Historical Price Data”, and contains daily records for various cryptocurrencies. Link to the full dataset:
https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory

For this analysis, I extracted the Bitcoin-specific file (coin_Bitcoin.csv), which contains 2,991 daily records.
Each record includes key financial indicators such as:
• Open, High, Low, Close prices
• Volume
• Marketcap (later dropped due to redundancy)
After generating the binary target column, one row was dropped, leaving 2,990 samples used in training and evaluation.

Problem Type

This is a binary classification task, The goal is to predict whether the next day’s closing price will be higher than the current day’s.

• Target column was created using .shift(-1) on Close
• Target values:
• 1 → price goes up
• 0 → price stays the same or goes down
• The original regression-style data was converted to classification to focus on direction prediction, which is more relevant for decision-making in financial contexts.

Data Cleaning & Preprocessing

A complete data cleaning pipeline was followed:

• Missing Value Check: I used .isnull().sum() to validate the completeness of the dataset. No missing values were detected in any column.

• Ran (**btc_data.describe()**) to explore feature distributions and detect outliers/skewness.

• Target Creation: A new target variable was created using .shift(-1) applied to the Close price. This allowed us to formulate the problem as binary classification.

• Row Dropping: A single NaN value appears in the last row after adding the target column (because there is no next day to allocate).

I handle this using : (**btc = btc.dropna()**), Complete the final number of samples: 2,990, another of 2,991.

• Feature Standardization: I applied StandardScaler from scikit-learn to normalize all features. This step is particularly important for distance-based models (KNN, SVM, ANN).

• Data Splitting: The dataset was divided into 70% training and 30% testing using train_test_split, with shuffle=False to preserve the time-series nature.

This preprocessing ensured that the models would operate on clean, consistent, and scaled data, avoiding data leakage or lookahead bias.

Exploratory Data Analysis (EDA)

Exploratory analysis was conducted using visualizations generated via the Seaborn library.

Each visualization provided a unique perspective on the data:

• **Heatmap:** Showed correlations between features. Close and Open had the highest correlation, indicating redundancy, but all features were retained for robustness.

• **Pairplot:** Helped visualize pairwise distributions and relationships. It highlighted a strong overlap in class distribution, supporting the use of non-linear models.

• **Histograms:** Illustrated the skewness and spread of each feature. Volume exhibited a highly skewed distribution.

• **Violinplot:** A violinplot was also used to visualize the distribution of prediction errors across the different classifiers. This plot highlights which models had tighter, more consistent predictions and which had broader error distributions, offering insights into model reliability beyond just accuracy.

These plots added depth to the understanding of the data's internal patterns and potential challenges.

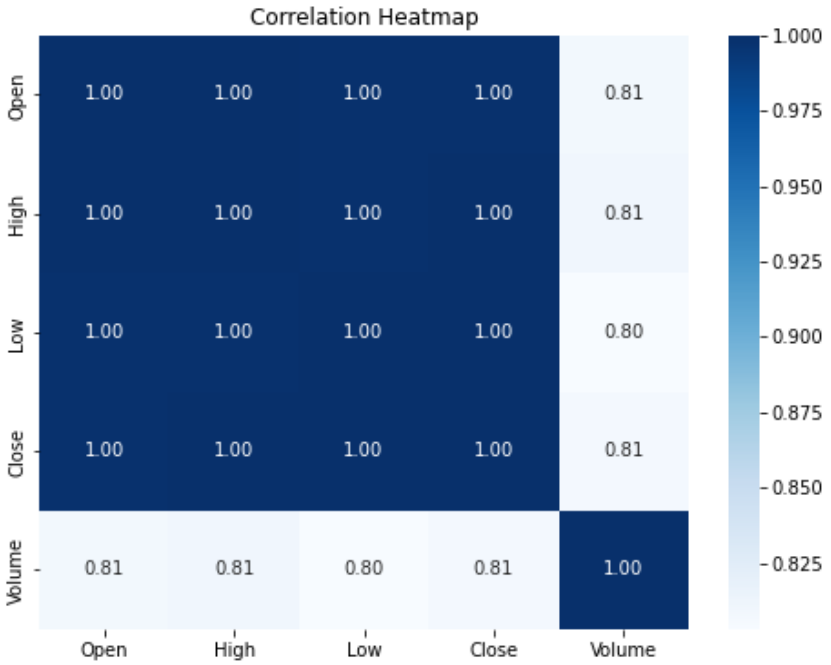

Fig 1: Correlation heatmap of features .

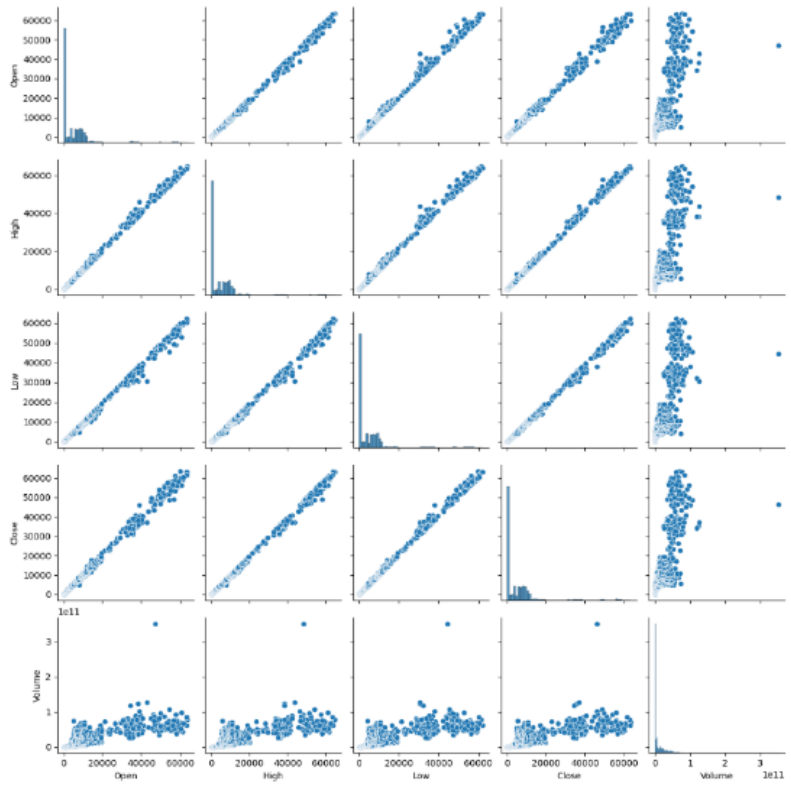

Fig 2: Pairplot distribution

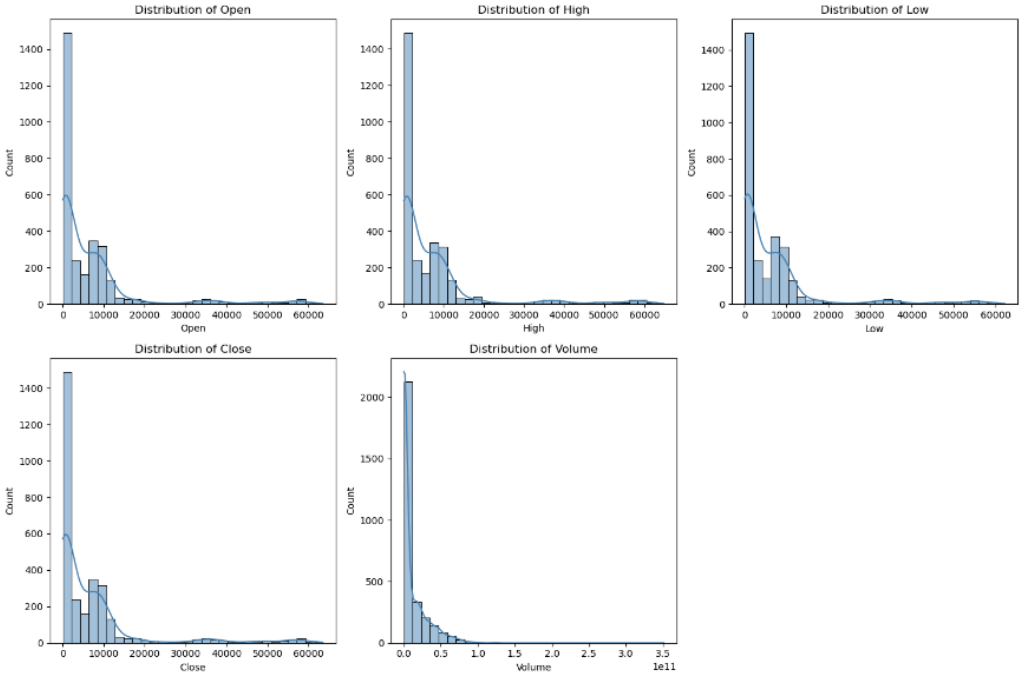

Fig 3: Histogram showing the distribution of each feature

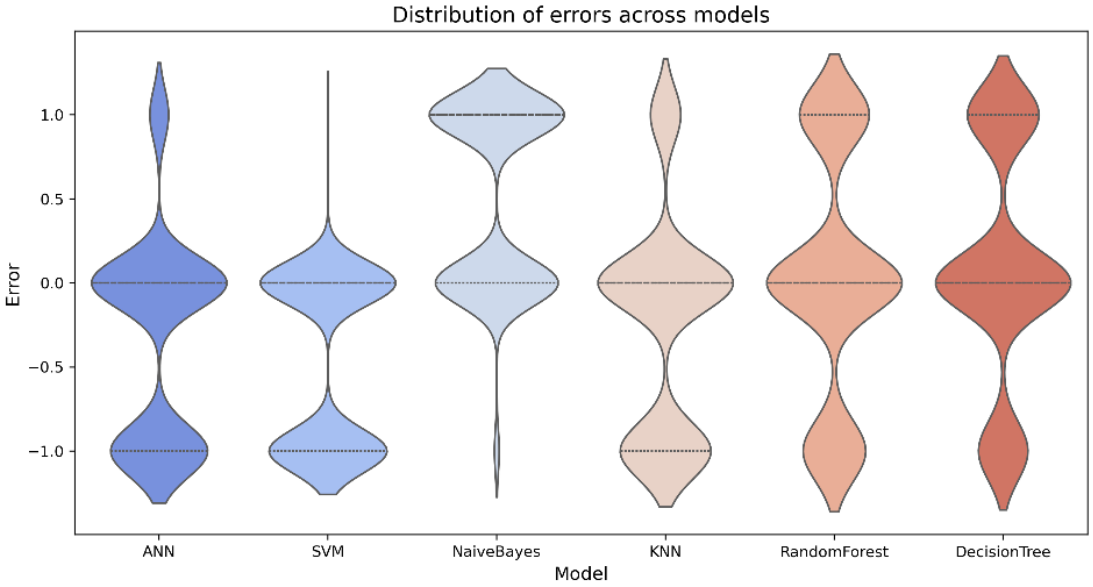

Fig 4: Model-wise error distribution using Violinplot.

Machine Learning Models Used

The following machine learning classification models were trained and evaluated:

- **Support Vector Machine (SVM)**

Leveraged with RBF kernel and scaling, SVM aims to find an optimal separating hyperplane between classes.

- **Artificial Neural Network (ANN)**

Implemented using a single hidden layer. Despite its potential, it underperformed due to limited data volume and lack of deep tuning.

- **Naive Bayes**

A probabilistic model that assumes feature independence. Fast but often weak on complex datasets like financial data.

- **K-Nearest Neighbors (KNN)**

A distance-based model, sensitive to feature scaling. It showed reasonable performance due to its simplicity.

- **Random Forest**

An ensemble of decision trees. It performed better than individual trees by reducing overfitting and improving generalization.

- **Decision Tree**

Simple and interpretable, but prone to overfitting. Performance was average but useful for baseline comparison.

(Note) Linear Regression was excluded because it is a regression model producing continuous outputs. Our task is binary classification, and using linear regression would misrepresent the prediction target.

Model Evaluation & Results

Each model's performance was assessed using accuracy, confusion matrix, and visual comparison via barplots.

MODEL	ACCURACY
ANN	0.4677
SVM	0.5334
NAIVE BAYES	0.4677
KNN	0.5278
RANDOM FOREST	0.5234
DECISION TREE	0.5156

- **Best Model**: SVM
  It benefits from the data standardization, enabling better separation in high-dimensional space.
- **Weakest Models**: ANN & Naive Bayes
  ANN likely underperformed due to lack of deep architecture and hyperparameter tuning.
  Naive Bayes struggles with financial data due to feature interdependence.

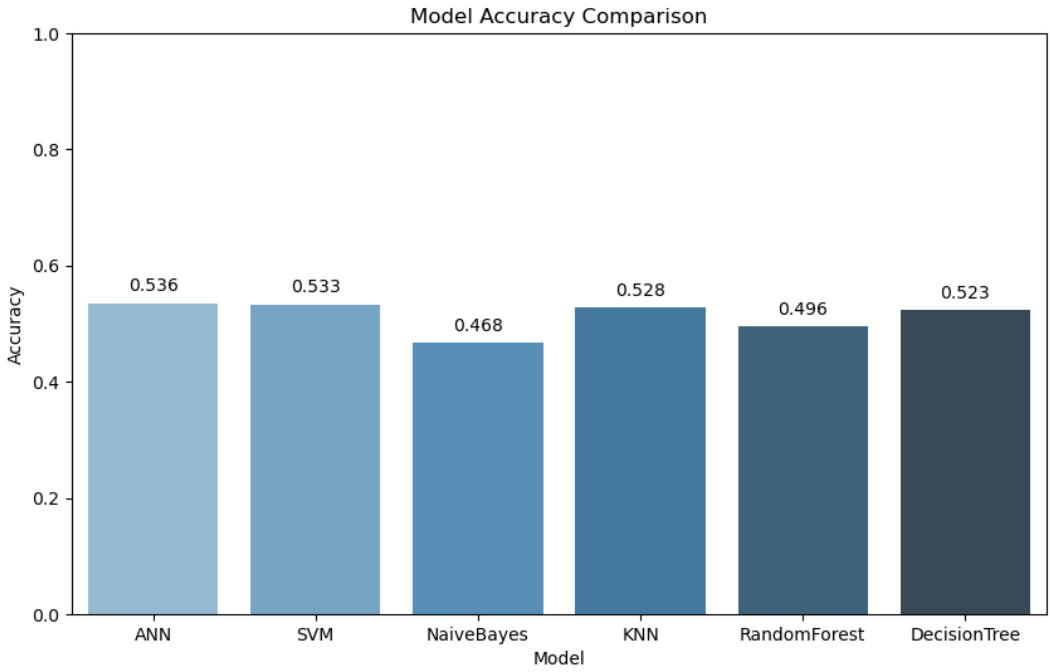

Fig 5: Comparing the accuracy scores of all models by barplot.

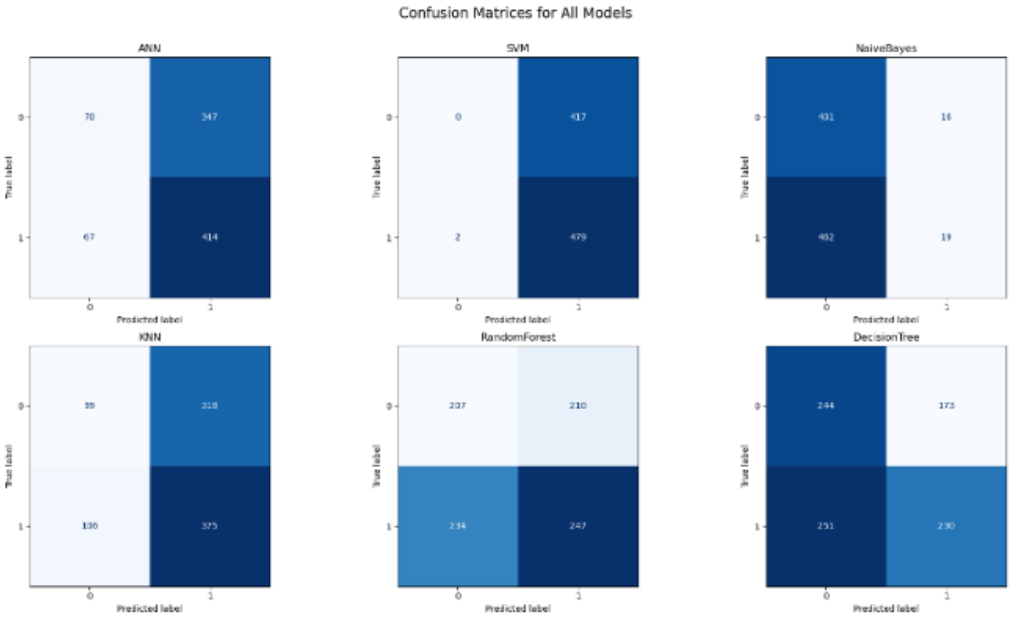

Fig.6: Confusion matrix grid.

Key Takeaways & Insights

I chose the Bitcoin dataset because it represents real-world financial complexity: volatility, constant change, and time sensitivity. It wasn't just an academic example it forced me to apply actual data science thinking to handle messy, unpredictable patterns. One of the first lessons I learned was that how you frame the problem matters.

By transforming the closing prices into a binary classification target using .shift(-1), I changed the entire nature of the task—from predicting numbers to making directional decisions.

In preprocessing, I realized how critical it is to clean and scale the data properly:

- **StandardScaler dramatically improved model behavior, especially for SVM and KNN**
- **I had to drop the final row due to NaN introduced by shifting**
- **I preserved the time-series order by disabling shuffle during the train/test split**

When evaluating the models, I found that accuracy alone was misleading.

Thanks to visual tools like confusion matrices and violinplots, I could analyze how models behaved—especially in terms of false predictions.

• SVM performed the best, achieving 53.3% accuracy

It benefited from scaling and handled nonlinear separation better than others.
This showed me how models like SVM, which rely on finding optimal boundaries in a high-dimensional space, can outperform simpler methods—but only when the data is well-preprocessed and standardized.
It reinforced how preprocessing isn't just a step — it's what enables certain models to work properly.
• Naive Bayes and ANN were the weakest, likely due to oversimplified assumptions or limited tuning.

Visualization was incredibly helpful:

- **Heatmaps helped uncover correlation between features**
- **Pairplots showed the overlap between class distributions**
- **Violinplots gave me a new way to see how errors were distributed, not just who was right or wrong**

In the end, I learned that: "The best model isn't always the most complex—it's the one that fits the nature of the data."

This project taught me not only how to build models, but how to analyze, question, and explain them with confidence.

Challenges Faced

• Time-series split: I had to prevent data leakage by disabling shuffling. This required careful handling during splitting and evaluation.
• Model tuning limitations: Due to scope limits, I couldn't fine-tune models using GridSearchCV, which might've improved results.
• Feature dominance: Volume had extreme skewness, which affected models like KNN and Naive Bayes.

These challenges taught me practical constraints in real-world modeling and how decisions during preprocessing directly affect model success.

Conclusion

This project helped me bridge theory and practice by applying classification models on real financial data.
SVM emerged as the most reliable model, but not because it was inherently superior — it matched the data's structure and preprocessing better than others.
The biggest takeaway is that no model fits all problems; success depends on a deep understanding of the data, thoughtful preprocessing, and proper evaluation.
Even with moderate accuracy, a model that's consistent, explainable, and well-evaluated has practical value.

GitHub Repository

You can find the complete project, including the code, data files, preprocessing scripts, and visualizations, on GitHub at the following link: https://github.com/MaryamMohammed2002/ML-project.git

References

1. Helformer: an attention-based deep learning model for cryptocurrency price forecasting. Journal of Big Data. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-025-01135-4

2. Bariviera, A. F. (2017). The inefficiency of Bitcoin revisited. Physica A: Statistical Mechanics and Its Applications, 479, 444–453. https://doi.org/10.1016/j.physa.2017.02.068

3. CoinMarketCap. (2023). Bitcoin market capitalization. https://coinmarketcap.com/currencies/bitcoin/