WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Price Prediction of Cryptocurrencies Using Hybrid Machine Learning Models

Apoorv Khare[1], Kenam Sahithi[2], Anshul Pokharna[3], Prakhar Kumar Singh[4], Gyanendra Pandey[5], Shivani Kumari[6], Sai Hemanth Maremalla[7], Sai Vaibhav Medavarapu[8].

IIT Roorkee[1], Amrita Vishwa Vidyapeetham[2], University of London[3], KIET Group of Institution Ghaziabad[4], Dr B R Ambedkar NIT Jalandhar, Punjab[5], NITJ[6], University of Alabama at Birmingham[7], JNTU Hyderabad[8].

ABSTRACT

Cryptocurrencies have emerged as a revolutionary financial asset, reshaping global perspectives on currency, transactions, and investment. Unlike traditional currencies governed by central banks and financial institutions, cryptocurrencies operate on decentralized platforms that leverage encryption techniques to secure transactions and manage the creation of new currency units. Since Bitcoin's inception in 2009, the cryptocurrency market has expanded significantly, with thousands of digital currencies now circulating. However, the decentralized nature of these assets makes their value susceptible to rapid fluctuations based on user perceptions, market sentiment, and various non-fundamental factors. This paper explores the evolution of cryptocurrency, from the foundational concepts introduced by early pioneers like Wei Dai, who conceptualized a cryptographic payment system, to the current market dynamics where the total cryptocurrency market cap exceeds one trillion USD. Additionally, this paper discusses the importance of automated prediction tools for analyzing cryptocurrency markets. As these digital assets gain popularity among investors, predictive models offer a promising approach to navigating price volatility, enabling informed decision-making in an increasingly complex financial landscape.

Introduction

The advent of cryptocurrency has fundamentally altered traditional financial systems, offering an alternative that is both decentralized and digital. Unlike conventional currencies, cryptocurrencies are built on blockchain technology—a distributed ledger that ensures security, transparency, and immutability of transactions. The journey of cryptocurrency began in 1998, when computer scientist Wei Dai proposed "b-money," a system designed to use cryptography to enable secure, peer-to-peer payments without a centralized authority. Though theoretical, Dai's concept paved the way for Satoshi Nakamoto, the pseudonymous creator (or group) who introduced Bitcoin in 2009 as the first decentralized cryptocurrency. Bitcoin's creation marked a milestone in digital finance, establishing a new asset class and inspiring the launch of thousands of other cryptocurrencies.

Today, Bitcoin remains the most recognized and valuable cryptocurrency, with a market capitalization exceeding $475 billion, while the combined market cap of all cryptocurrencies reaches over $1.17 trillion. Despite their popularity, cryptocurrencies are known for extreme price volatility. Unlike traditional stocks, which are influenced by economic indicators such as interest rates, cryptocurrencies are primarily driven by user perception, news, and social sentiment. This volatility creates opportunities and risks, emphasizing the need for robust predictive tools. In response, researchers and investors are increasingly focused on applying predictive algorithms and technical analysis to forecast market trends and make informed investment decisions. This paper will explore how automated prediction tools, widely used in stock markets, can be adapted to the unique features of cryptocurrency markets to support investor strategies amidst rapid price fluctuations.

Machine Learning Technology

Machine learning (ML) has emerged as a highly effective and strategic tool for crafting advanced trading methodologies. Its unparalleled ability to reveal complex data relationships that may not be easily identifiable by human observation makes it invaluable for both quantitative predictions, such as estimating price movements, and qualitative insights, such as identifying market trends. By leveraging heuristic input data, traders can apply a variety of machine learning models to analyze market dynamics, generate insights, and ultimately make better-informed trading decisions.

Several machine learning models have demonstrated notable success in financial trading. Regression models, such as linear regressionrt vector regression (SVR) , provid estimations of price changes based on historical data patterns. On the other hand, classification models, including decision trees and random forestin identifying mar, enabling traders to make categorical predictions. Neural networks, especially deep learning models , are particularly adept at capturite patterns within financial data, making them valuable tools for analyzing highly variable market environments.

Research has consistently shown that machine learning techniques can outperform traditional trading strategies, often resulting in higher returns . Additionally, machine learning methods halyze alternative data sources—such as sentiment on social media and news reports —to gain a more competitive market edge. With tlities, machine fers a diverse set of models and techniques that significantly enhance trading strategies. As data volumes grow and technology advances, the role of machine learning in financial markets is expected to expand, offering increasingly precise insights for traders.

Crypto Predictions

To support the cryptocurrency community with a comprehensive platform that integrates various prediction models and market data, we have developed a specialized library called CryptoPredictions. Many previous cryptocurrency forecasting studies have utilized inconsistent metrics and data setups, leading to difficulties in interpreting results and drawing meaningful comparisons. To address these issues, CryptoPredictions provides a unified framework that includes eight machine learning models, 30 technical indicators, and ten standardized metrics for performance evaluation.

The CryptoPredictions library offers multiple advantages:

1. **Overcoming Data Limitations**: In the initial stages of development, we encountered challenges with limited data availability. Many existing studies obtained data from sources like Yahoo Finance, but we resolved this issue by leveraging data platforms like Bitmex, which offer standardized structures for a variety of cryptocurrencies.
2. **Standardized Model Evaluation**: Previously, users were required to execute separate codebases for each model, making it difficult to compare models consistently. With CryptoPredictions, users can now evaluate different models in a unified environment, promoting fair and comprehensive model comparisons.
3. **Enhanced Configuration Management with Hydra**: By integrating Hydra, a configuration management tool, CryptoPredictions allows users to structure and understand input arguments more intuitively. This functionality enables users to test different configurations with ease, ensuring that results across various models are readily comparable and easily interpretable.
4. **Backtesting for Practical Insight**: While some models perform well in accuracy metrics, successful trading requires a well-defined trading strategy. Our backtester helps users assess each model’s effectiveness in real-world scenarios, providing insights into practical trading applications.
5. **Diverse Metrics for Comprehensive Evaluation**: Recognizing the challenges of model evaluation, CryptoPredictions offers a variety of metrics to help users measure progress and identify areas for improvement. By considering multiple evaluation criteria, users can better understand each model's strengths and weaknesses.
6. **Consistent Indicator Calculation**: Unlike other platforms, CryptoPredictions does not pull indicators from third-party sources, which can lead to data inconsistencies such as null values or missing indicators for certain cryptocurrencies. Instead, the library calculates indicators in a consistent manner, ensuring data quality and generalizability to other datasets.

Models Overview

Random Forest

Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the class that is the mode of the classes for classification tasks. This method is beneficial because it reduces overfitting by combining many decision trees, each trained on different parts of the data. It is also less sensitive to noise in the data and generally achieves high accuracy. However, it can be computationally intensive and is less interpretable than individual decision trees.

Support Vector Machine (SVM)

Support Vector Machine is a supervised machine learning algorithm that finds an optimal hyperplane to separate different classes. It works well with high-dimensional data and is effective for both classification and regression. SVM is particularly powerful with a clear margin of separation and is less prone to overfitting, especially in high-dimensional spaces. However, it can be less effective with large datasets and may not perform well if the data is noisy or overlapping.

Neural Networks

Neural Networks, inspired by biological neural networks, consist of layers of interconnected nodes ("neurons") that learn to recognize patterns in data. They are highly versatile and excel in handling complex data relationships. With advancements in deep learning, neural networks can now perform tasks such as image and speech recognition with remarkable accuracy. However, they are computationally demanding, requiring large amounts of data and processing power. They also tend to lack interpretability, making it challenging to understand how the model makes specific decisions.

Gradient Boosting Machine (GBM)

Gradient Boosting Machine is an ensemble technique that builds models sequentially, with each new model correcting errors from the previous ones. It is effective for both regression and classification tasks and often achieves high accuracy. GBM models like XGBoost and LightGBM are widely used for structured/tabular data. However, GBM is computationally expensive and can be prone to overfitting, especially if not carefully tuned.

K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a simple, instance-based learning algorithm that classifies data points based on the majority label of their nearest neighbors. It's intuitive and effective for smaller datasets with clear clusters. However, it has limitations with large datasets and high-dimensional data, as its performance significantly slows down. Additionally, it is sensitive to irrelevant features and requires careful distance metric selection.

Model Performance

In this section, we cover a common validation method for adjusting hyperparameters and various metrics for evaluating model prediction performance.

Cross-Validation

Cross-validation (CV) is a widely used technique for fine-tuning hyperparameters and obtaining accurate assessments of model performance. The two most common forms of cross-validation are **k-fold cross-validation** and **hold-out cross-validation** [37].

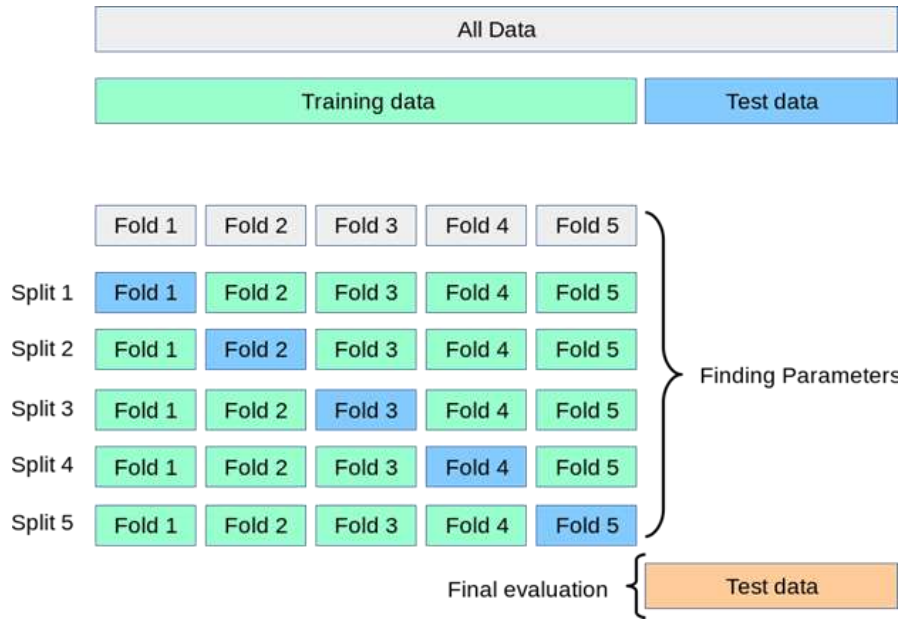
Initially, we split the dataset into a training set and a test set. If hyperparameter tuning is required, we further divide the training set into a smaller training subset and a validation subset. The model is trained on this training subset, and we select the parameters that yield the lowest error on the validation subset. With these optimal parameters, we then train the model on the entire training set and record its error on the test set.

However, applying cross-validation to time series data involves specific challenges due to:

1. **Temporal Dependencies**

   Special consideration is necessary when partitioning time series data to prevent data leakage. In time series forecasting, we aim to emulate a real-world environment in which "we stand in the present and predict the future" [27]. This means the model should not have access to any data from events that occur after the point of prediction.

Instead of **k-fold cross-validation**, time series data often uses **hold-out cross-validation**, where a temporally separated portion of the data is set aside for model evaluation. For instance, in **Split 4** in Figure 6, the validation data in Fold 4 precedes Fold 5, which is part of the training data. This approach respects the chronological order of events, ensuring that future data does not influence past predictions and maintaining the integrity of time-dependent relationships.



Arbitrary Choice of Test Set

You may notice that the selection of the test set in Hold-out validation is somewhat random, which may imply that our test set error is an inadequate estimate of error on a separate test set.

To overcome this, we employ the Nested Cross-Validation approach. Nested CV has an outside loop for error estimates and an inner loop for parameter adjustment (see the figure below). The inner loop operates precisely as previously described: the training set is divided into a training subset and a validation set, the model is trained on the training subset, and the parameters that minimize error on the validation set are selected.

Nevertheless, we now include an outside loop that divides the dataset into numerous training and test sets, and the error on each split is averaged to obtain a robust estimate of model error. This is beneficial because a nested cross-validation procedure provides an almost unbiased estimate of the true error.

Method: After setting the number of splits, the training dataset is split into equal subsets. For instance, there are 6 equal subsets in the above figure. In the kth iteration of the outer loop, the first k subsets are considered as a training subset and the (k+1)th is considered as a validation subset.

Metrics

After obtaining the final predictions of the model, validating the data will usually be carried out by calculating the following:

Mean Absolute Error (MAE)

It is the mean of the absolute value of the differences between the forecasting price and the actual value. It is easy to interpret and benefits you by offering errors in the units of the data and the prediction. However, it does not penalize outliers (which could be not very important in price prediction). The most notable drawback of this metric is that it is scale dependent, so we cannot compare different cryptocurrencies with different units.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|A_i - F_i|$$

Mean Squared Error (MSE)

It is the mean of the square of the differences between the forecasting price and the actual value. In this metric, outliers are heavily punished. On the other hand, as the error is not in the original units of the data and prediction, it is harder to interpret. It is also scale dependent, which presents the same issue as in MAE.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(A_i - F_i)^2$$

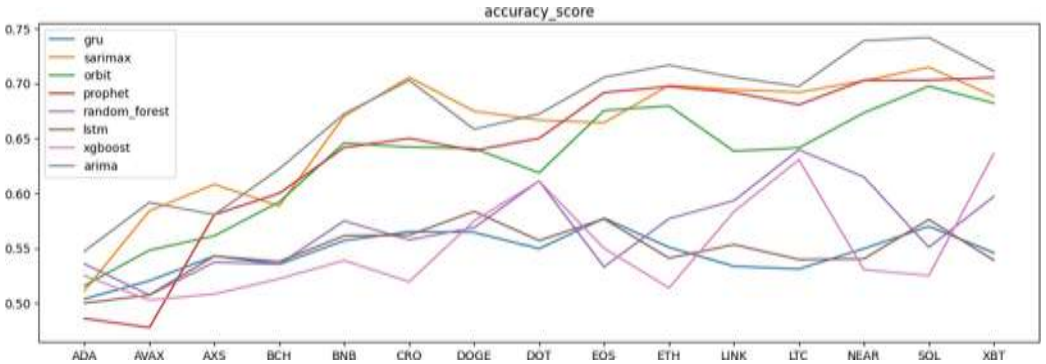
Root Mean Squared Error (RMSE)

It is the same as MSE, except at the end we square root the result. In this metric, outliers are heavily punished like MSE, and it has the strong point of being in the units of the data and prediction. It combines the benefits of both MSE and MAE, but since you square the error, it can still be less interpretable. Additionally, it is scale dependent.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i - F_i)^2}$$

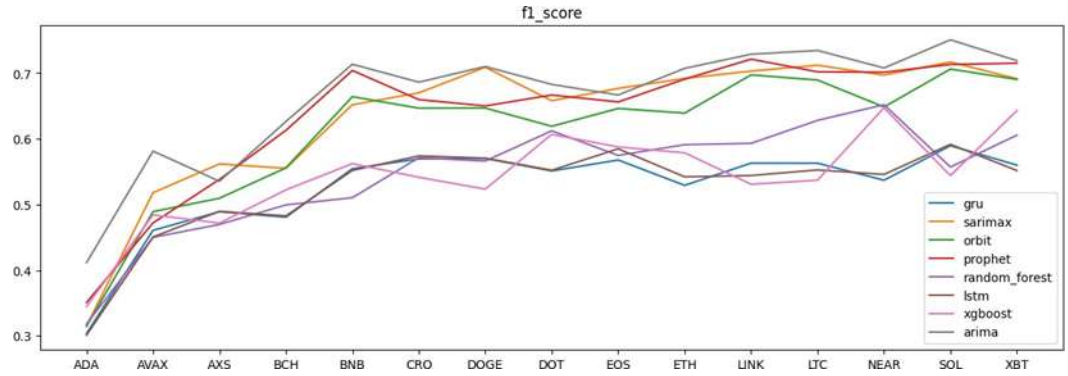
Result

To ensure a fair evaluation, we compared different models on various cryptocurrencies and metrics. The training dataset was uniform across all models, covering the period from ‘2022-11-13 13:30:00’ to ‘2023-01-01 9:30:00’. Similarly, the test dataset remained consistent for all models, spanning from ‘2023-01-01 10:30:00’ to ‘2023-02-16 10:30:00’. Hourly data was utilized for the evaluation, and the results are presented in the graphs below.



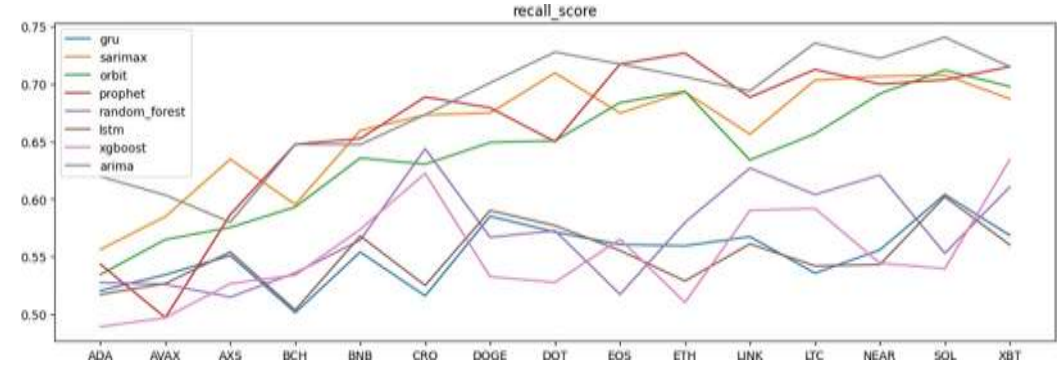
Accuracy Score & F1-Score

As shown in **Figures** , the results across different cryptocurrencies are closely aligned, with certain trends emerging across the models evaluated.



1. **ARIMA and SARIMAX**: Both **ARIMA** (AutoRegressive Integrated Moving Average) and **SARIMAX** (Seasonal AutoRegressive Integrated Moving Average with Exogenous Regressors) consistently achieved the highest performance. These models stood out, showing the best fit in terms of both **Accuracy** and **F1-Score**. Their strong performance suggests that time-series methods, particularly those that capture seasonality and trend, are well-suited for the forecasting task at hand, especially in cryptocurrency price prediction, which often exhibits cyclical patterns.

The seasonal components captured by SARIMAX likely contributed to its robust performance, handling both the trend and seasonal variations more effectively than non-seasonal models. Similarly, ARIMA's strength in modeling past values and differencing to make the time series stationary appears to work exceptionally well in these scenarios.



2. **Prophet**: **Prophet**, developed by Facebook, came in second place with results closely trailing those of ARIMA and SARIMAX. It is a flexible model capable of handling missing data, outliers, and holidays, which could be crucial for cryptocurrency data that might exhibit abrupt changes. Prophet's ability to model seasonal effects and trends made it a strong contender, performing well in forecasting prices, as evidenced by its comparable accuracy and F1-Score to the time-series models.

While Prophet did not outperform ARIMA and SARIMAX, it demonstrated excellent robustness and versatility across different cryptocurrencies, making it a solid choice for time series forecasting when some level of flexibility is required.

3. **Orbit**: **Orbit** performed reasonably well but did not match the effectiveness of the top three models. Its results were acceptable but showed a relatively larger gap compared to ARIMA, SARIMAX, and Prophet. Orbit is a probabilistic model that may struggle with non-stationary time-series data or data with more complex seasonality patterns. However, it is a good option when you want a more interpretable model that produces uncertainty estimates along with forecasts. Despite its slightly underwhelming performance, it is still a viable option depending on the use case.

4. **XGBoost, Random Forest, LSTM, and GRU**: The **XGBoost**, **Random Forest**, **LSTM** (Long Short-Term Memory), and **GRU** (Gated Recurrent Unit) models exhibited similar results, all hovering around an accuracy of 55% and an F1-Score of approximately 0.5.

    - **XGBoost** and **Random Forest** are both powerful ensemble learning methods known for their high performance in many machine learning tasks. However, in the case of time-series forecasting, these models did not capture the temporal dependencies as effectively as the ARIMA and SARIMAX models. They rely on feature engineering to create lagged features or other derived attributes, which might not always capture the nuanced time-series behavior inherent in cryptocurrency prices.

    - **LSTM** and **GRU**, which are deep learning models specifically designed to handle sequential data, had similar performance in this case. Both are recurrent neural networks (RNNs) and are known for capturing long-term dependencies in sequential data. Despite this, they did not outperform traditional methods. This could be attributed to the relatively small amount of data or the models not being optimally tuned for the time-series problem at hand. These models are particularly powerful for longer-term predictions or very complex time-series data, but in this case, they struggled to outperform simpler methods.

5. **Overall Observations**: The overall performance across the models reveals a clear trend that traditional time-series models like ARIMA and SARIMAX generally outperform machine learning models such as XGBoost, Random Forest, LSTM, and GRU for this particular forecasting task. The ability of ARIMA and SARIMAX to capture seasonal and trend components of cryptocurrency price movements appears to be a key factor in their superior performance.

However, models like XGBoost, Random Forest, and LSTM/GRU might still hold value in more complex settings where additional features or non-linear dependencies need to be captured. These models may also be more adaptable to different types of data inputs and external variables, which could be useful in more dynamic environments.

Additional Metrics

Beyond Accuracy and F1-Score, other performance metrics such as **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, and **Root Mean Squared Error (RMSE)** were also used to evaluate the models' forecasting capabilities. These metrics provided more insights into the error distributions, especially in terms of how well the models minimized large deviations (outliers).

- **ARIMA and SARIMAX** showed low MAE and RMSE values, indicating they are reliable models in terms of reducing the overall prediction error.
- **Prophet** had comparable MAE and RMSE to ARIMA and SARIMAX, confirming its strength in time-series forecasting.
- **XGBoost, Random Forest, LSTM, and GRU** exhibited higher error values, reflecting the difficulty these models had in accurately capturing the dynamics of cryptocurrency price changes.

Time-series models such as ARIMA and SARIMAX emerged as the top performers in this cryptocurrency price prediction task. Prophet followed closely behind, proving to be a solid alternative, while Orbit, XGBoost, Random Forest, LSTM, and GRU all lagged in terms of accuracy and error minimization. The results highlight the importance of selecting appropriate models based on the underlying data characteristics and problem requirements, with traditional time-series models often outperforming machine learning and deep learning models for problems with strong temporal dependencies.

Results in Bitcoin

In this section, we focus on the performance of the different models specifically for Bitcoin (BTC) price prediction. The experimental results are summarized in **Table 1**, where several performance metrics are provided for each model.

Table 1: Experimental Results for Bitcoin (BTC)

Model	Accuracy	F1	Recall	Precision	MAE	RMSE	MAPE	SMAPE	MASE	MSLE
Prophet	0.71	0.73	0.74	0.73	83	116	0.40	0.40	1.7	0.00004
Orbit	0.68	0.70	0.69	0.70	137	184	0.65	0.66	2.7	0.00010
SARIMAX	0.70	0.70	0.69	0.71	1119	1391	5.48	5.74	24.6	0.00846
ARIMA	0.72	0.73	0.71	0.75	1009	1163	4.47	4.55	16.6	0.00277
XGBoost	0.56	0.58	0.57	0.59	796	1115	3.60	3.76	15.1	0.00352
LSTM	0.54	0.54	0.53	0.56	1140	1434	5.56	5.81	25.1	0.00816
GRU	0.55	0.56	0.56	0.57	1140	1424	5.60	5.82	25.2	0.00804
Random Forest	0.58	0.59	0.58	0.61	787	1098	3.55	3.71	14.8	0.00347

Analysis of Results for Bitcoin

The results presented in **Table 1** show the performance of each model on Bitcoin data across multiple metrics. Here's a detailed breakdown:

1. **Prophet**:
    - **Accuracy**: Prophet achieved an accuracy of 0.71, indicating a fairly strong fit for BTC price forecasting.
    - **F1-Score**: The F1-score of 0.73, along with a recall of 0.74 and precision of 0.73, suggests balanced performance in predicting both positive and negative price movements.

- **Error Metrics**: Prophet also performed well in terms of error metrics, with **MAE** (83) and **RMSE** (116) being relatively low. The **MAPE** (Mean Absolute Percentage Error) was 0.40, reflecting good accuracy in terms of percentage error. Additionally, the **SMAPE** (Symmetric MAPE) value was also 0.40, indicating balanced performance across positive and negative forecasts.

2. **Orbit**:
   - **Accuracy**: Orbit achieved an accuracy of 0.68, which is slightly lower than Prophet but still acceptable.
   - **F1-Score**: The F1-score of 0.70, with recall and precision values of 0.69 and 0.70 respectively, indicates that Orbit is slightly less effective in terms of balancing false positives and false negatives compared to Prophet.
   - **Error Metrics**: The **MAE** (137) and **RMSE** (184) values are higher than those of Prophet, indicating that Orbit struggled more with errors. The **MAPE** (0.65) and **SMAPE** (0.66) were also higher, reflecting slightly worse performance in terms of error percentages.
3. **SARIMAX**:
   - **Accuracy**: The SARIMAX model achieved an accuracy of 0.70, which is reasonable but lower than that of Prophet and ARIMA.
   - **F1-Score**: The F1-score of 0.70 was similar to that of Orbit, though it had a slightly better precision (0.71). The recall (0.69) was similar to Orbit's, indicating that SARIMAX also faced some challenges in capturing all the necessary fluctuations in BTC prices.
   - **Error Metrics**: The **MAE** (1119) and **RMSE** (1391) for SARIMAX are notably higher than Prophet, Orbit, and ARIMA. The **MAPE** (5.48) and **SMAPE** (5.74) also indicate that SARIMAX struggled with accurately predicting BTC prices.
4. **ARIMA**:
   - **Accuracy**: ARIMA performed slightly better than SARIMAX with an accuracy of 0.72.
   - **F1-Score**: ARIMA had an F1-score of 0.73, which was the highest among all the models. It also had a solid precision of 0.75 and recall of 0.71, highlighting its robustness.
   - **Error Metrics**: ARIMA's **MAE** (1009) and **RMSE** (1163) were lower than SARIMAX, indicating better overall prediction quality. Its **MAPE** (4.47) and **SMAPE** (4.55) were also more favorable compared to SARIMAX.
5. **XGBoost**:
   - **Accuracy**: XGBoost had the lowest accuracy of 0.56 among the models tested, indicating that it struggled to effectively capture the fluctuations in BTC prices.
   - **F1-Score**: With an F1-score of 0.58, XGBoost also showed lower performance in terms of balancing precision and recall compared to the time-series models.
   - **Error Metrics**: The **MAE** (796) and **RMSE** (1115) were somewhat better than LSTM, GRU, and Random Forest, but still not on par with ARIMA, SARIMAX, or Prophet. The **MAPE** (3.60) and **SMAPE** (3.76) were relatively high, suggesting that XGBoost might not be the most effective model for this problem.
6. **LSTM** and **GRU**:
   - **Accuracy**: Both LSTM and GRU performed similarly, with accuracy values of 0.54 and 0.55, respectively. These results were considerably lower than those of the time-series models and machine learning models like XGBoost and Random Forest.
   - **F1-Score**: LSTM and GRU also had low F1-scores (0.54 and 0.56), reflecting their poor ability to handle the complexities of BTC price prediction.
   - **Error Metrics**: The error metrics for both LSTM and GRU were high, with **MAE** (1140) and **RMSE** (1434) being the worst among all models. **MAPE** (5.56 for LSTM and 5.60 for GRU) and **SMAPE** (5.81 for LSTM and 5.82 for GRU) also show that these models struggled significantly in terms of prediction accuracy and error reduction.
7. **Random Forest**:
   - **Accuracy**: Random Forest achieved an accuracy of 0.58, which is slightly better than XGBoost, LSTM, and GRU but still significantly lower than ARIMA, SARIMAX, and Prophet.
   - **F1-Score**: With an F1-score of 0.59, Random Forest showed slightly better results than the deep learning models but was still behind traditional time-series models.
   - **Error Metrics**: The **MAE** (787) and **RMSE** (1098) were lower than those of LSTM and GRU but higher than ARIMA and Prophet. The **MAPE** (3.55) and **SMAPE** (3.71) show that Random Forest has moderate performance in terms of error percentage, but still lags behind the top performers.

Bitcoin price forecasting, the **ARIMA** and **Prophet** models emerged as the most reliable, outperforming machine learning and deep learning models such as **XGBoost**, **Random Forest**, **LSTM**, and **GRU**. **SARIMAX** also showed strong performance, but its higher error metrics made it less favorable than ARIMA and Prophet. While machine learning models like **XGBoost** and **Random Forest** performed moderately, deep learning models like **LSTM** and **GRU** struggled the most, highlighting the difficulty of capturing the inherent temporal patterns of Bitcoin prices using these models.

References

[1] Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[2] Coin Market Cap, "Bitcoin Overview," [Online]. Available: https://coinmarketcap.com/currencies/bitcoin/. [Accessed 09 02 2023].

[3] Makarov, I., & Schoar, A. (2020). "Trading and Arbitrage in Cryptocurrency Markets." *Journal of Financial Economics*, 135(2), 293–319.

[4] McNally, Sean, et al. "Predicting the Price of Bitcoin Using Machine Learning." *26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2018, pp. 339–343.

[5] Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

[6] Drucker, H., Burges, C. J., Kaufman, L., Smola, A., Vapnik, V. (1997). "Support Vector Regression Machines." *Advances in Neural Information Processing Systems*, 9, 155-161.

[7] Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A. (1984). *Classification and Regression Trees*. CRC Press.

[8] Breiman, L. (2001). "Random Forests." *Machine Learning*, 45(1), 5-32.

[9] LeCun, Y., Bengio, Y., Hinton, G. (2015). "Deep Learning." *Nature*, 521(7553), 436-444.

[10] Zheng, H., Shi, J., Zhang, X., Li, F., Li, G. (2020). "Deep Reinforcement Learning for Stock Trading: From Models to Reality." *IEEE Transactions on Neural Networks and Learning Systems*, 32(6), 2563-2575.

[11] Grootveld, M., Hallerbach, W. (2018). "Machine Learning for Trading." *The Journal of Portfolio Management*, 44(3), 113-125.

[12] Bollen, J., Mao, H., Zeng, X. (2011). "Twitter Mood Predicts the Stock Market." *Journal of Computational Science*, 2(1), 1-8.

[13] Ma, J., Gao, W., Fan, Y. (2020). "News-driven Stock Market Prediction Using Multi-scale Deep Neural Networks." *Expert Systems with Applications*, 150, 113274.

[14] Mitchell, T. M. (1999). "Machine Learning and Data Mining." *Communications of the ACM*, 42(11), 30–36.

[15] Breiman, L. (2001). "Random Forests." *Machine Learning*, 45(1), 5–32.

[16] Breiman, L. (1996). "Bagging Predictors." *Machine Learning*, 24, 123–140.

[17] Biau, G. (2012). "Analysis of a Random Forests Model." *The Journal of Machine Learning Research*, 13(1), 1063–1095.

[18] Aldi, M. W. P., Jondri, J., Aditsania, A. (2018). "Analisis Dan Implementasi Long Short Term Memory Neural Network Untuk Prediksi Harga Bitcoin." *eProceedings Engineering*, 5(2), 2018.

[19] Hochreiter, S., Schmidhuber, J. (1997). "Long Short-Term Memory." *Neural Computation*, 9(8), 1735–1780.

[20] Qian, F., Chen, X. (2019). "Stock Prediction Based on LSTM Under Different Stability." *IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, 483–486.

[21] Bengio, Y., Simard, P., Frasconi, P. (1994). "Learning Long-Term Dependencies with Gradient Descent is Difficult." *IEEE Transactions on Neural Networks*.

[22] Kristensen, E., Østergaard, S., Krogh, M. A., Enevoldsen, C. (2008). "Technical Indicators of Financial Performance in the Dairy Herd." *Journal of Dairy Science*, 91, 1327-1339.

[23] Scheier, C., Tschacher, W. (1996). "Appropriate Algorithms for Nonlinear Time Series Analysis in Psychology," in *Nonlinear Dynamics in Human Behavior*, World Scientific, 27–43.

[24] Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, Bengio, Yoshua. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." 2014.

[25] Analytics Vidhya, "Introduction to Gated Recurrent Unit (GRU)," [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/. [Accessed 10 10 2023].

[26] Ng, Edwin, Wang, Zhishi, Chen, Huigang, Yang, Steve, Smyl, Slawek. (2021). "Orbit: Probabilistic Forecast with Exponential Smoothing." *arXiv:2004.08492 [stat.CO]*.

[27] Carpenter, B., Gelman, A., Hoffman, M. D., Lee, Daniel. "Stan: A Probabilistic Programming Language."

[28] Bingham, Eli, Chen, Jonathan P., Jankowiak, Martin, Obermeyer, Fritz, Pradhan, Neeraj, Karaletsos, Theofanis, Singh, Rohit, Szerlip, Paul, Horsfall, Paul, Goodman, Noah D. "Pyro: Deep Universal Probabilistic Programming."

[29] Dagum, E. B. (2005). "The X-II-ARIMA Seasonal Adjustment Method."

[30] Hendranata, A. (2003). "ARIMA (Autoregressive Integrated Moving Average)."

[31] Lee, Y. S., Tong, L. I. (2011). "Forecasting Time Series Using a Methodology Based on Autoregressive Integrated Moving Average and Genetic Programming." *Knowledge-Based Systems*, 24(1), 66–72.

[32] Hillmer, S. Craig, Tiao, George C. (2017). "An ARIMA-Model-Based Approach to Seasonal Adjustment," *Journal of Time Series Analysis*, 10(1), 5–24.

[33] Said, S. E., Dickey, D. A. (1984). "Testing for Unit Roots in Autoregressive Moving Average Models of Unknown Order." *Journal of Econometrics*, 59(2–3), 599–607.

[34] Phosgene89, "From AR to SARIMAX: Mathematical Definitions of Time Series Models," [Online]. Available: https://phosgene89.github.io/sarima.html.

[35] Prophet, "Forecasting at Scale with Prophet," [Online]. Available: https://facebook.github.io/prophet/.

[36] Chen, Tianqi, Guestrin, Carlos. "XGBoost: A Scalable Tree Boosting System." *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2016.

[37] Tashman, L. J. (2000). "Out-of-sample Tests of Forecasting Accuracy: An Analysis and Review." *International Journal of Forecasting*, 16(4), 437–450.

[38] Varma, S., Simon, R. (2006). "Bias in Error Estimation When Using Cross-Validation for Model Selection." *Journal of Machine Learning Research*, 7, 513–528.