

Predicting Cryptocurrency Prices Using Machine Learning Algorithms

Final Report

Shahzad Ahmed

Student ID# 501211922



Table of Contents

Abstract	2
Literature Review	5
Introduction	6
Approach/Methodology	7
Overview of the Data.....	8
Daily Return and Volatility Analysis.....	9
Technical Analysis	10
Data Visualization	13
Visualizing Market Capitalization	13
Visualizing the Closing and Opening Price Difference	13
Visualizing the High and Low-Price Difference.....	14
Exploratory Data Analysis (EDA).....	15
Data Dictionary	15
Missing Values	16
Detection of outliers for numerical columns.....	17
Univariate Analysis of High & Close Prices	19
Bivariate Analysis	21
Statistical Summary for numerical columns	21
Correlation	27
Principal Component Analysis (PCA)	28
Seasonal Decomposition of the Cryptocurrencies Close Price.....	30
The Augmented Dickey-Fuller (ADF) test	32
Time-Series Forecasting	34
Simple Moving Average (SMA)	37
Exponential Smoothing (ES).....	45
Autoregressive Integrated Moving Average (ARIMA)	50
Seasonal Autoregressive Integrated Moving Average (SARIMA)	62
Prophet.....	73
Deep Learning Techniques	76
Conclusion.....	84
Self Reflection	85
Future work.....	85
Reference.....	86

Abstract

Title:**Predicting Cryptocurrency Prices Using Machine Learning Algorithms.**

With the advancement and modernization of technology, various industries are rapidly evolving to adapt to the latest global trends. One such change that has been taking place is the increasing prevalence of cryptocurrency as a medium of exchange and investment. This shift towards digital currencies represents a significant departure from traditional fiat currencies and stock exchange practices and It's the time to understand what digital money really means for everyone's future. I would like to direct the readers to an engaging article by Alex Hern that delves into the world of Bitcoin and cryptocurrencies, discussing their significance as digital currencies and their potential impact on our future. [1]

I have reviewed the article and identified five key takeaways:

1. Cryptocurrencies, like Bitcoin, are a novel form of decentralized currency that allows individuals to exchange funds without the involvement of traditional financial institutions. This has the potential to disrupt the conventional banking system and reshape the financial landscape.
2. Blockchain technology forms the underlying structure for cryptocurrencies and functions as a decentralized ledger that logs transactions and is upheld by a group of computers. Its versatility extends far beyond finance and encompasses various areas, such as supply chain management, voting systems, and identity authentication.
3. Cryptocurrencies have been notorious for their high degree of volatility, with their worth fluctuating dramatically. This has raised concerns about their stability and long-term viability as a store of value.
4. Despite providing some level of anonymity, cryptocurrencies have also been linked to illegal activities, such as money laundering and drug trafficking. As a result, governments and regulatory bodies have increased scrutiny of them.
5. The advent of cryptocurrencies has ignited a broader discussion about the role of technology in society and its potential to disrupt established power structures. Although it remains to be seen how this will unfold, it is evident that cryptocurrencies and blockchain technology could have a significant influence on our future.

CB Insights [2] published an article that delves into the impact of blockchain technology on traditional banking and payment systems. The article highlights how blockchain technology can enable decentralized payment systems that facilitate faster and more secure transactions between financial institutions. This is important to our study as it sheds light on the potential influence of blockchain technology on the finance industry and how it could impact cryptocurrency prices. The article offers five key takeaways:

1. The banking industry could potentially be disrupted by blockchain technology, which can offer faster, more cost-effective, and secure transactions by eliminating the necessity for intermediaries like banks to facilitate transactions.
2. By leveraging blockchain technology to simplify cross-border payments, financial institutions can minimize their reliance on conventional payment processors such as SWIFT. This approach can substantially decrease the expenses and time required for global transactions.
3. Smart contracts are a type of self-executing contract that encodes the terms of an agreement between a buyer and a seller into code lines. They have the capability to automate several elements of financial transactions, including loan origination and settlement, which can potentially increase efficiency and reduce costs.

4. Blockchain technology can enhance transparency and decrease fraud by providing an immutable and permanent record of transactions. This can help financial institutions monitor and prevent fraudulent activities.
5. While the banking industry is still in the early stages of adopting blockchain technology, many financial institutions are exploring ways to incorporate this technology into their operations. Although there are technical and regulatory hurdles to overcome, the potential benefits of blockchain technology for the banking industry are significant.

Overall, both articles offer insights into the broader landscape of cryptocurrency and blockchain technology, which can help inform our study on predicting cryptocurrency prices using machine learning algorithms.

Research Questions:

What are the most effective predictive and time series analysis techniques for forecasting short-term closing prices of cryptocurrencies? Which features are influential predictors for classifying the short-term closing prices of selected cryptocurrencies?

What is the correlation between the predicted prices generated by machine learning algorithms and the actual prices of the chosen cryptocurrencies?

Scope of the Research:

The project aims to use machine learning algorithms to predict short-term closing prices of different cryptocurrency companies. The dataset for this project is obtained from Kaggle Inc. which contains historical data for the chosen cryptocurrencies. The objective is to compare the predicted prices with the actual prices and identify which cryptocurrency presents the most profitable opportunity for short-term trading.

To answer the research question, we will explore different machine learning algorithms and time-series analysis techniques, such as SMA, SE, ARIMA, SARIMA, PROPHET and deep learnings. We will compare the efficiency and stability of these techniques to identify the most effective ones for our purpose.

Data Source:

The data set used in this project is obtained from Kaggle Inc. which contains historical data for the chosen cryptocurrencies. The data is related to the closing prices of each of the six different cryptocurrency companies and has been used in previous researches.

Limitations of the Research:

The scope of this study is restricted to specific cryptocurrency companies and the historical data that is accessible for them, and thus, it may not accurately reflect the overall cryptocurrency market. The machine learning algorithms utilized in the analysis rely on historical data, and the future value of cryptocurrencies can be influenced by unpredictable factors, such as changes in regulations, market sentiment, and global events. The precision of the forecasts could be influenced by the quality and comprehensiveness of the data used.

Background Information:

Cryptography is used to secure cryptocurrency, which is a digital or virtual form of money. It is decentralised and not under the jurisdiction of a single entity, such as a government or bank, unlike conventional currencies. It is a distributed ledger used to record cryptocurrency transactions and is used to secure and authenticate user data. The worldwide financial system has been significantly impacted by this innovative technology, and its future growth potential is enormous. The first cryptocurrency, Bitcoin, was released in 2009, and since then, the market has expanded to encompass several other cryptocurrencies, with a market capitalization of over \$1 trillion. Despite having only recently emerged, cryptocurrencies have already had a significant impact on the financial landscape and are predicted to continue doing so.

Notably, the data set we are going to use in this study, this same data set was employed in a previous study by a researcher Manomi Koroth, whose report is available at

<https://github.com/Mkoroth97/CIND820-2021/blob/main/CIND820-FINAL%20REPORT.pdf>

Upon analyzing the report, some potential limitations and avenues for enhancement were identified. For example, the report utilizes a basic linear regression model for predicting cryptocurrency prices, but fails to compare its performance with other machine learning models and techniques. To enhance the model's accuracy, experimenting with other models and techniques would be worth considering. Additionally, the report does not provide a comprehensive evaluation of the model's performance, such as mean absolute error, mean squared error, or R-squared. By incorporating these metrics, a more thorough assessment of the model's accuracy can be conducted, and areas for further improvement can be identified. From the investors point of view, the report is deficient in terms of providing insightful analysis, including but not limited to the comparison of daily returns vs volatility as well as technical analysis. These aspects will also be addressed in this study.

Specific Area of Research:

The specific area of research in this project is the utilization of machine learning algorithms in time-series analysis to predict the future prices of selected cryptocurrencies and identify profitable opportunities for short-term investment.

Data Set:

The project aims to use data to achieve the goal [3].

<https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory>

Gethib Link of this Study:

<https://github.com/shahgem/CIND-820>

Literature Review

The emergence of cryptocurrencies has created an entirely new asset class that is characterized by high volatility and lack of regulation. As a result, the study of cryptocurrency markets has become increasingly important to both investors and financial analysts. The field of cryptocurrency price prediction has garnered significant attention, yet there is still a dearth of research papers on the subject. This shortage of information has motivated finance and data enthusiasts to explore the application of various deep machine learning models to market data.

While conventional time series techniques such as ARIMA are useful for analyzing data with linear and stationary patterns, they are limited in their ability to capture the complex and non-linear patterns present in cryptocurrency data. This highlights the need for alternative approaches, and deep learning methods have shown great potential in this regard. Cryptocurrency data is characterized by high volatility and frequent fluctuations, which can lead to chaotic behavior. Deep learning methods are well-suited to handle such complex and high-dimensional data, making them an ideal choice for analyzing cryptocurrency prices.

Several research papers have investigated which deep learning method is most accurate in forecasting cryptocurrency prices. Classification methods such as Random Forest Trees and k-folds were employed to compare the accuracy of each method. The studies have primarily focused on analyzing Bitcoin (BTC), given its status as the longest active currency for conducting analysis and its dominance in the cryptocurrency market.

The reviewed papers primarily investigate the performance differences of deep learning methods across various time intervals of cryptocurrency prices. These time intervals include daily, weekly, and monthly opening and closing prices. In addition to forecasting prices based on historical data, some studies have also incorporated sentiment analysis.

Researchers have analyzed the frequency of the term "Bitcoin" in tweets and explored its relationship to BTC predictability. However, it is worth noting that the utilization of natural language processing (NLP) methods for sentiment analysis is not relevant to this project.

The article "Bitcoin price prediction using machine learning algorithm" [4] written by Mohammed Khalid Salman and Abdullahi Abdu Ibrahim proposes a new machine learning algorithm for predicting the price of Bitcoin.

1. The authors use historical Bitcoin price data to train the algorithm and evaluate its performance.
2. The algorithm is based on an artificial neural network (ANN) that uses a combination of technical indicators, such as moving averages, as input features. The ANN is trained using a backpropagation algorithm, and the authors use mean squared error (MSE) and mean absolute error (MAE) as evaluation metrics.
3. The authors find that the ANN model outperforms traditional time-series models, such as the autoregressive integrated moving average (ARIMA) model, in predicting Bitcoin prices. They also find that using technical indicators as input features improves the performance of the ANN model.
4. The authors conclude that machine learning algorithms can be useful for predicting Bitcoin prices, and that using technical indicators as input features can improve the accuracy of these predictions. They also note that further research is needed to evaluate the performance of these algorithms in different market conditions and with other cryptocurrencies.
5. Overall, the article provides an interesting and valuable contribution to the literature on cryptocurrency price prediction and machine learning.

Overall, the study of cryptocurrency price prediction is a nascent field that requires further research. Deep learning methods have shown great promise in addressing the unique challenges posed by cryptocurrency markets, and several studies have explored the application of these methods to BTC. This project aims to build on this research by investigating the performance of various deep learning methods for forecasting the prices of other popular cryptocurrencies.

Introduction

The field of cryptocurrency price prediction is an exciting and rapidly evolving area of research, with many promising avenues to explore. Given the newness of this topic, the number of research papers available is limited, making it an ideal area for finance and data enthusiasts to explore and experiment with various deep machine learning models.

Conventional time series techniques, such as ARIMA, have proved to be ineffective in capturing the non-linear and non-stationary patterns that are commonly present in cryptocurrency data. These limitations underscore the need for alternative approaches to analyze this unique and dynamic market.

One of the major challenges in predicting cryptocurrency prices is the high volatility and frequency of price fluctuations, which can lead to underlying chaos. As such, Deep Learning Methods have emerged as a critical tool in this field, as they can capture complex patterns in the data that other methods might miss.

In addition to analyzing the historical data of cryptocurrencies, researchers have also explored the role of sentiment analysis in price prediction. Some studies have analyzed the frequency of "Bitcoin" in tweets to determine if it correlates with the predictability of the BTC price, using Natural Language Processing (NLP) methods. However, this project will not consider sentiment analysis due to the limitations of the scope and the complexity of NLP techniques. One of the studies "Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis" [5] concludes People who tweet about cryptocurrencies even when their prices drop have an interest in those tweets above and beyond investment opportunity which makes the tweets biased towards positive trend. Here are five key points that can be gleaned from the paper:

1. Cryptocurrency price prediction is an important area of research, and many different methods have been proposed to predict the price of cryptocurrencies, such as Bitcoin, Ethereum, and Litecoin.
2. The authors propose a new method for cryptocurrency price prediction that uses tweet volumes and sentiment analysis. The method involves collecting and analyzing tweets related to specific cryptocurrencies and using the sentiment analysis of those tweets to predict the future price of the cryptocurrency.
3. The study shows that tweet volumes and sentiment analysis can be useful in predicting cryptocurrency prices, and the method proposed by the authors outperforms other methods such as time series analysis and support vector regression.
4. The authors note that their method is not without limitations, including the potential for bias in the data due to the selective nature of Twitter users and the need for further research to validate the results.
5. Despite the limitations, the study highlights the potential of social media data and sentiment analysis in predicting cryptocurrency prices and suggests that further research in this area could lead to improved prediction models and better understanding of the cryptocurrency market.

In summary, this project aims to explore and evaluate the performance of several Deep Learning Methods to forecast the prices of cryptocurrencies. While researchers from Jaypee University of Information Technology, India have conducted extensive investigations on this topic [6]. Here are five key points from the project report on "Cryptocurrency Price Prediction Using Deep Learning":

1. The project proposes the use of deep learning techniques to predict the price of cryptocurrencies, which are highly volatile and challenging to predict accurately.
2. The project uses historical price data from popular cryptocurrencies like Bitcoin and Ethereum to train a deep learning model using various neural network architectures, including LSTM, GRU, and CNN.
3. The project's objective is to examine how technical indicators, specifically Moving Average Convergence Divergence (MACD) and Relative Strength Index (RSI), can be employed to enhance the accuracy of cryptocurrency price predictions.
4. Furthermore, an assessment of the deep learning model's performance is conducted using evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), indicating that it is more effective than conventional statistical models in forecasting cryptocurrency prices.
5. The project provides valuable insights into the potential of deep learning techniques and technical indicators for predicting cryptocurrency prices and suggests further research in this area.

I intend to offer a novel perspective by including additional insights in my literature review. My goal is to provide a more in-depth analysis of the dataset and explore the use of a wider range of model techniques to present readers with a more thorough and comprehensive analysis. Through this project, we hope to contribute to the growing body of research in this exciting and rapidly evolving field.

Approach/Methodology

Developing a robust approach for analyzing cryptocurrency market data is essential for investors seeking to gain insight into the cryptocurrency market and make informed investment decisions. The following steps outline a structured methodology for analyzing cryptocurrency market data:

Data Collection: The first step is to gather relevant historical price data for the cryptocurrencies being analyzed, including trading volume and market capitalization. The data must be sourced from reputable sources to ensure accuracy and reliability.

SWOT Analysis: Conducting a SWOT analysis of the selected models is crucial to identifying their strengths, weaknesses, opportunities, and threats. This analysis will help in determining which models are best suited for the specific cryptocurrency being analyzed.

Data Preparation: After collecting the data, the next step is to prepare it for analysis. This involves removing any missing or erroneous data, transforming the data into a suitable format, and normalizing it to account for differences in scale.

Model Selection: In this step, various models such as SMA, ARIMA, and Prophet models are selected as potential options for analysis. It is important to select the most appropriate models for the specific cryptocurrency being analyzed.

Model Implementation: Once the models are selected, they are implemented and applied to the prepared data. This may involve adjusting the parameters of the models to optimize their performance.

Data Visualization: Conducting exploratory data analysis (EDA) to understand the dataset's properties and characteristics is essential. Techniques such as box plots and histograms are used to detect outliers, distributions, and correlations. EDA helps in identifying trends, patterns, and potential relationships between the features of the dataset.

Model Evaluation: Evaluating the models' performance is crucial in determining their accuracy and reliability and identifying areas for improvement. Appropriate metrics such as mean absolute error (MAE), root mean squared error (RMSE), and mean absolute scaled error (MASE) are used in this step.

Results Interpretation: The results of the analysis are interpreted and conclusions are drawn based on the findings. This may involve visualizing the data and model outputs using graphs and charts, as well as conducting statistical tests to assess the significance of the results.

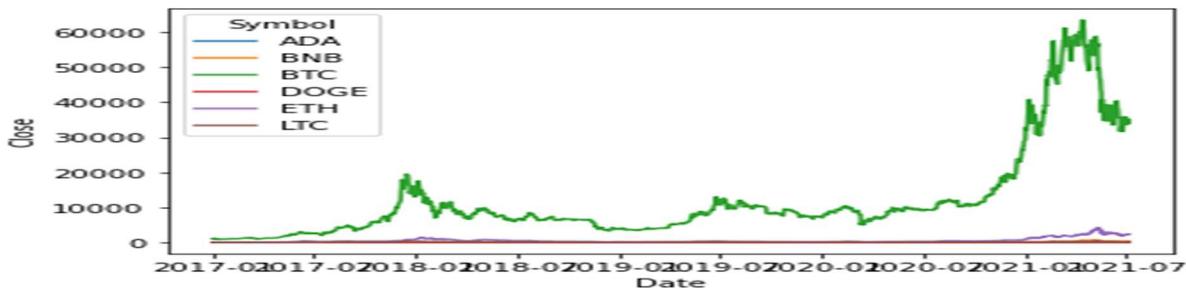
Conclusion and Recommendations: The key findings of the study are summarized, and recommendations for future research or actions are provided based on the results. By following this structured methodology, investors can make informed decisions based on reliable data analysis. The methodology outlined here is an essential tool for any investor looking to gain a competitive edge in the cryptocurrency market, as it provides a comprehensive and systematic approach for analyzing cryptocurrency market data.

Overview of the Data

The dataset utilized in this project was sourced from Kaggle Inc, and is publicly accessible under the title "Cryptocurrency Historical Prices" [3].

During this course of study, I will emphasize on Bitcoin (BTC), Ethereum (ETH), Litecoin (LTC), Dogecoin (DOGE), Cardano (ADA), and Binance Coin (BNB). Although Kaggle provides separate CSV files for each cryptocurrency, the author of this project has amalgamated all the individual files into a single CSV file named "crypto_market.csv". This amalgamation was possible due to the identical columns of each file, i.e., Serial Number, Name, Symbol, Date, High, Low, Open, Close, Volume, Market Capitalization.

The initial dataset brings to the fore the non-uniformity of the trading dates of the various currencies. As previously noted, BTC has been in circulation for the longest period, pre-dating all the other cryptocurrencies in the dataset. Since the start dates for each currency, which may significantly impact their predicted prices. To mitigate this potential bias, it is advisable to limit the analysis to the period from 2017 onwards, as this provides a more consistent and recent historical data for all the selected currencies. This filtered dataset has been saved under the file name "sixcrypto.csv".



In light of this, during the analysis phase of this project, it is prudent to compare all currencies from 2017 onwards, so as to minimize prediction bias and incorporate a more uniform and recent historical data.

Daily Return and Volatility Analysis

Daily return is a metric that quantifies the percentage change in an asset's value over a day. It's calculated by dividing the difference between the closing and opening prices of the asset by the opening price.

Volatility is a gauge of the extent to which an asset's price fluctuates over a period of time. Generally, it is calculated as the standard deviation of the asset's daily returns. A higher level of volatility indicates greater market instability and a greater likelihood of significant price swings in either an upward or downward direction.

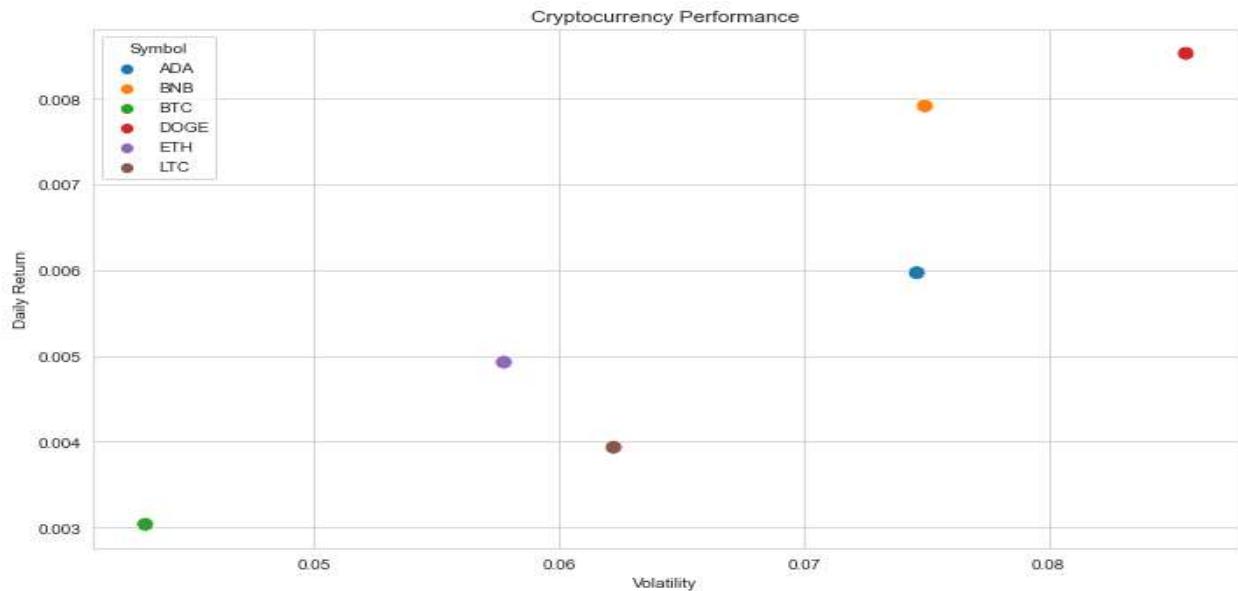
The RBC article [7] discusses the relationship between volatility and investment returns. Key points include:

1. Volatility refers to the degree of price variation of an investment over time, and it is typically measured by standard deviation.
2. Historical volatility is a measure of the past variability of returns, while implied volatility is a measure of expected future variability based on market prices.
3. Higher volatility generally implies higher risk, but it can also present opportunities for higher returns.
4. The relationship between volatility and returns is not straightforward, as it can vary depending on the investment strategy and time horizon.
5. For long-term investors, higher volatility assets may lead to higher returns over time, while short-term traders may be more concerned with managing risk and avoiding large losses.
6. Investors can manage the risk of volatility through diversification, asset allocation, and other risk management strategies.

The Python code is using pandas libraries, numpy, seaborn, and matplotlib to analyze and visualize data related to six cryptocurrencies.

- The code reads the data from a CSV file named "sixcrypto.csv" and groups the data by currency symbol.
- It then calculates daily returns and volatility for each currency using the groupby method and for loop.
- The code creates two dataframes to store the calculated daily returns and volatility for each currency.
- It then merges the two dataframes using the merge method on the 'Symbol' column.
- Finally, the performance dataframe is printed in a tabular format using the to_string method.
- The code also plots the results using the seaborn and matplotlib libraries.
- It sets the style to 'whitegrid', creates a scatterplot with x-axis as volatility, y-axis as daily return, and symbol as hue.
- The plot is titled "Cryptocurrency Performance" with x-label as "Volatility" and y-label as "Daily Return".

Overall, this code reads and processes data related to six cryptocurrencies, calculates daily returns and volatility, and visualizes the performance of each currency using a scatterplot.



The results demonstrate that BNB and DOGE have higher daily returns, indicating their superior performance in terms of price growth. Nevertheless, DOGE also displays greater volatility, indicating its price is more susceptible to fluctuations and investing in it entails higher risk. BTC, on the other hand, has the lowest daily return, but also the lowest volatility, implying its price is more stable compared to other currencies.

Although these metrics can be helpful in evaluating different currencies' performance and making informed investment choices, it is crucial to remember that past performance is not an assurance of future outcomes. External factors, such as market trends and events, may have a significant impact on currency prices.

Technical Analysis

Technical analysis is a popular and widely-used methodology for predicting future price movements of an asset. It entails a comprehensive analysis of an asset's past market data, including price charts and technical indicators, to identify trends, support and resistance levels, and potential trading opportunities. Technical analysts use various indicators such as moving averages, RSI, MACD, and Fibonacci retracements to develop their analysis. Through the careful study of an asset's price history and market data, technical analysts aim to gain a deeper understanding of its current and future price movements, enabling them to make informed trading decisions.

If you want to explore technical analysis further, here are the main takeaways from Adam Hayes' Investopedia article titled "Technical Analysis: What It Is and How to Use It in Investing" [8]

1. Technical analysis involves studying past market data, including price charts and technical indicators, to predict future price movements of an asset.
2. Technical analysts use various tools and techniques such as moving averages, trendlines, and chart patterns to identify trends and potential trading opportunities.

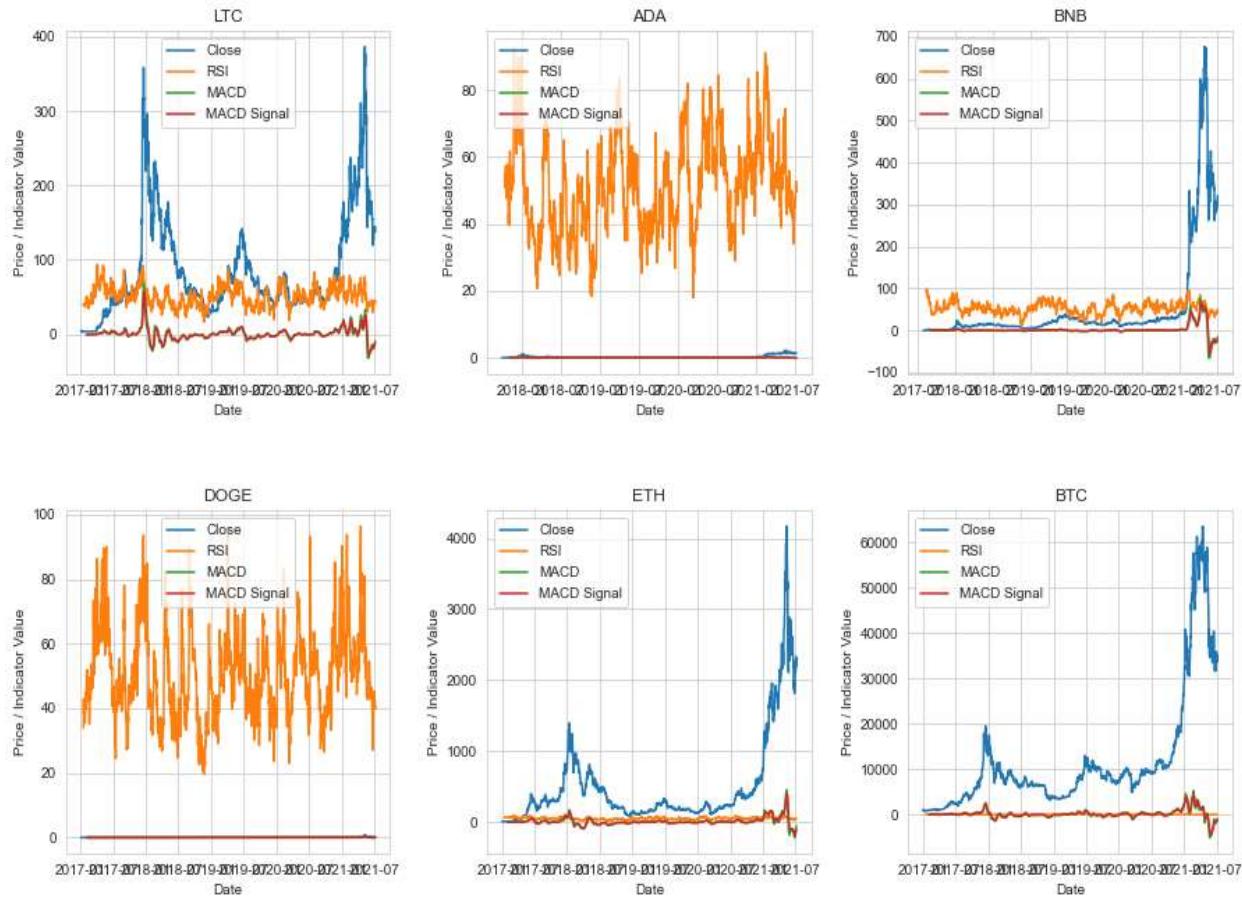
3. Technical analysis is based on the idea that market trends, patterns, and behaviors repeat over time and that past price movements can provide insights into future price movements.
4. While technical analysis is popular in trading, it has its limitations and is not always reliable.
5. Combining technical analysis with other types of analysis and practicing risk management when trading is essential for successful trading.

Indicator	Description	Interpretation
Relative Strength Index (RSI)	(RSI) is a measure that assesses the strength of a security's price action, indicating the level of momentum it has. This metric operates as an oscillator and produces RSI readings that fall within the range of 0 to 100. The RSI value enables traders to identify whether the security is overbought or oversold	RSI > 70: Overbought condition RSI < 30: Oversold condition RSI > 50: Bullish momentum RSI < 50: Bearish momentum
Moving Average Convergence Divergence (MACD)	The statement describes the correlation between two moving averages that represent the price of an asset.	Positive MACD: Bullish momentum Negative MACD: Bearish momentum
MACD Signal	Plotted alongside the MACD to generate buy and sell signals.	Cross above MACD signal line: Bullish signal Cross below MACD signal line: Bearish signal Distance between MACD and signal line: Trend strength

I used the python code using the Pandas, TA-Lib, and Matplotlib libraries to analyze and visualize financial data for different cryptocurrencies.

Here's a step-by-step breakdown of what our python code is running:

- Importing the necessary libraries: pandas, ta-lib, and matplotlib.pyplot.
- Loading the cryptocurrency data from a CSV file using the pd.read_csv() method and converting the Date column to a datetime object.
- Getting the unique symbols (currency names) from the loaded data.
- Creating an empty results_df dataframe with columns to store the technical indicators (RSI, MACD, MACD_signal) for each currency.
- Creating a 2x3 grid of subplots using the plt.subplots() method.
- Looping through each currency in the symbols list, calculating the technical indicators (RSI and MACD) for each currency, and plotting the data on the current subplot using the talib.RSI(), talib.MACD(), and axs.plot() methods.
- Storing the calculated technical indicator values for each currency in the results_df dataframe using the pd.concat() and pd.DataFrame() methods.
- Displaying the results_df dataframe using the print() method.
- Showing the plot using the plt.show() method.
- Adjusting the spacing between subplots using the plt.subplots_adjust() method.



Symbol	RSI	MACD	MACD Signal
LTC	43.325110	-9.037788	-12.098173
ADA	50.234631	-0.024998	-0.047715
BNB	50.275473	-14.875495	-21.176928
DOGE	40.619870	-0.021343	-0.023876
ETH	52.635585	-51.093439	-105.488122
BTC	46.383877	-871.687094	-1130.908270

Based on the technical analysis results, here are the interpretations for each currency:

- **LTC:** The RSI suggests that LTC is slightly oversold. The MACD is negative but is above the signal line, which indicates a potential bullish crossover.
- **ADA:** The RSI is close to 50, indicating a neutral position. The MACD and signal line are close to each other, which suggests that the currency is currently in a range-bound market.
- **BNB:** The RSI is slightly above 50, indicating a bullish trend. The MACD is negative but is above the signal line, which indicates a potential bullish crossover.
- **DOGE:** The RSI suggests that DOGE is oversold. The MACD and signal line are close to each other, which suggests that the currency is currently in a range-bound market.
- **ETH:** The RSI suggests that ETH is in a neutral position. The MACD is negative, and the signal line is far below it, indicating a strong bearish trend.
- **BTC:** The RSI suggests that BTC is slightly oversold. The MACD is negative, and the signal line is far below it, indicating a strong bearish trend.

Data Visualization

Visualizing Market Capitalization

The initial step in our analysis involved segregating the dataset by individual currencies, in order to facilitate comparative visualization. The determination of which currency held the greatest sway in the market was predicated on market capitalization - a metric that reflects the total market value of a given currency. Given Bitcoin's pioneering role in the cryptocurrency space, and its established longevity, we hypothesized that it would exert the greatest impact on the market.

However, the lack of market capitalization data prior to 2017 posed a challenge to our analysis. To overcome this, we chose to graph market capitalization data starting from the year 2017. This approach allowed us to focus on a more recent and reliable dataset.

The resulting graph illustrated that Bitcoin was the most frequently traded currency, with its market capitalization dwarfing that of its peers. This observation was consistent with our hypothesis, indicating that Bitcoin indeed exerted a significant influence on the cryptocurrency market.

Furthermore, the graph revealed that as Bitcoin prices increased, so too did the prices of other currencies. This suggested a strong correlation between Bitcoin's market performance and that of other cryptocurrencies. In other words, Bitcoin appeared to be a bellwether of sorts - with its price movements influencing the direction of other currencies.

Our study's results align with previous literature, which suggested that Bitcoin's market performance had a considerable influence on other cryptocurrencies' performance. As a result, our analysis supports and expands upon this prior research, revealing the cryptocurrency market's interdependence and Bitcoin's crucial role within it.



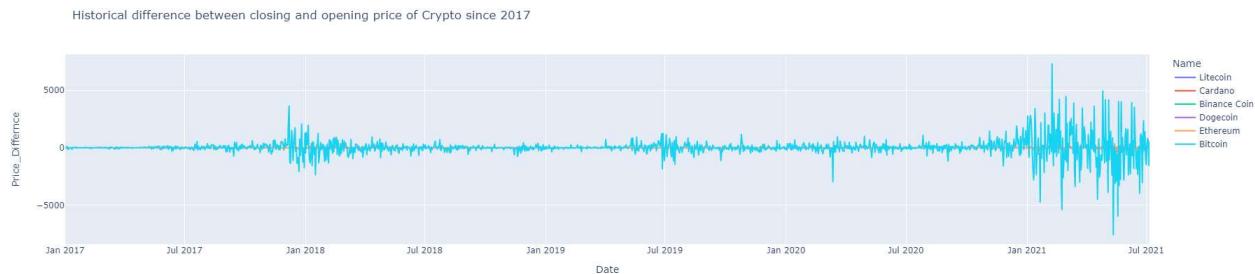
Visualizing the Closing and Opening Price Difference

To comprehend the daily price patterns for each cryptocurrency, I analyzed the variance between the opening and closing prices for each data point. This approach helped me to understand the level of price fluctuation that occurred within each trading day.

The analysis of the data reveals that Bitcoin had the highest price fluctuations from opening to closing within a day. This finding is in line with Bitcoin's reputation for volatility and its inclination to undergo abrupt price spikes or declines.

However, it's important to note that other cryptocurrencies also exhibited significant price fluctuations. Ethereum, for example, demonstrated a high level of price variation, which is not surprising given its status as the second most traded crypto project since its inception in 2017.

The graph indicates an interesting correlation between price hikes and fluctuations in the cryptocurrency market. When the market experiences an upswing, the degree of price variability tends to increase. This trend can be ascribed to various factors such as heightened investor speculation and market frenzy, which can lead to more volatile trading patterns.

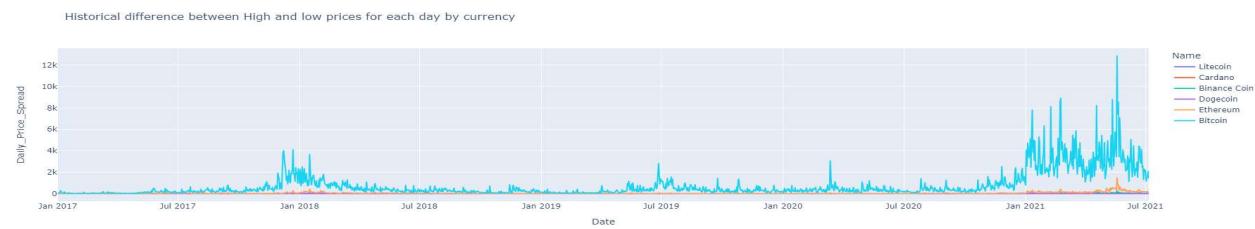


In summary, the analysis of the difference between opening and closing prices provides valuable insights into the daily price trends for each cryptocurrency. Bitcoin and Ethereum are among the most volatile, while the level of price fluctuation tends to increase alongside price increases in the overall market.

Visualizing the High and Low-Price Difference

One of the key factors driving interest in cryptocurrencies is their price volatility. This volatility is often attributed to the limited supply of cryptocurrencies and the speculative nature of their value. As with any asset, the price of cryptocurrencies is influenced by supply and demand, with market participants often reacting to news and events that impact the cryptocurrency ecosystem.

In our analysis, we sought to understand the daily price variations of cryptocurrencies and the differences between their high and low prices. To accomplish this, we first calculated the daily price spread of each cryptocurrency by subtracting the low price from the high price. Our analysis revealed that Bitcoin and Ethereum exhibit the most significant price fluctuations, while other cryptocurrencies showed a relatively plateaued trend. This suggests that Bitcoin and Ethereum may be subject to greater volatility than their counterparts.



In addition to analyzing the daily price variations of cryptocurrencies, we also examined the activity levels of each currency. Our analysis found that 2018 was the most active year for Bitcoin, while the post-pandemic period has seen a significant uptick in cryptocurrency activity. We hypothesize that this increase in activity may be attributed to a rise in unemployment during the pandemic, leading individuals to explore alternative means of income generation.

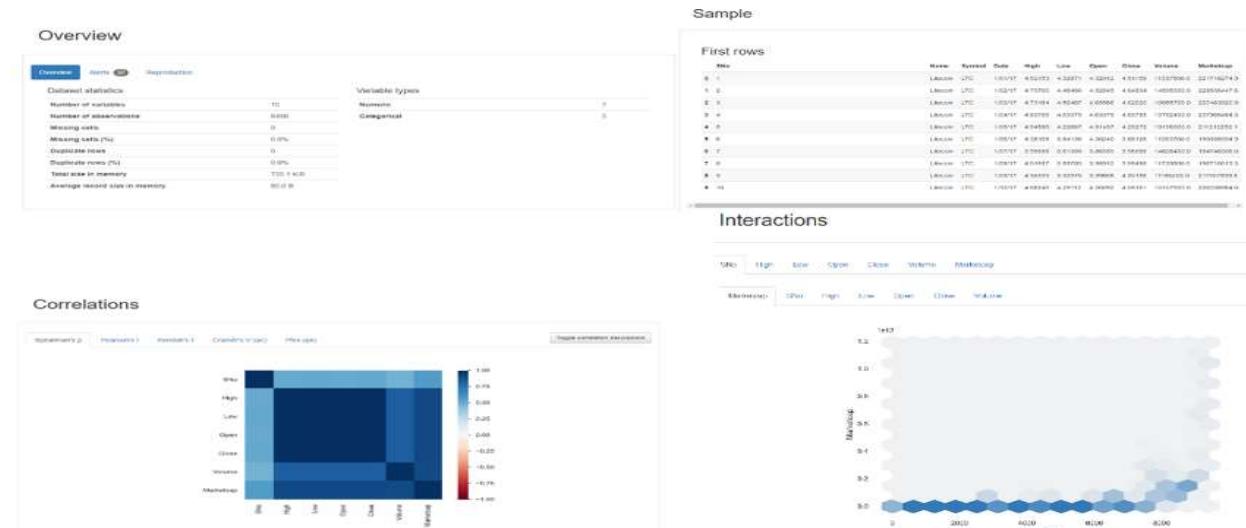
These findings underscore the importance of monitoring the price fluctuations and activity levels of cryptocurrencies. Understanding the factors that drive these fluctuations can provide valuable insights for investors and policymakers alike. Additionally, our analysis highlights the potential of cryptocurrencies as an alternative asset class for individuals looking to diversify their investment portfolios.

However, it is important to note that cryptocurrencies remain a relatively new and untested asset class, and their volatility and lack of regulation can create significant risks for investors. Furthermore, the regulatory landscape for cryptocurrencies remains uncertain, with many countries struggling to develop coherent policies around their use and adoption. Future research may explore the relationship between cryptocurrency price fluctuations and macroeconomic factors, such as unemployment rates and economic policy changes, to gain a more comprehensive understanding of these emerging financial instruments.

Exploratory Data Analysis (EDA)

Python was chosen as the programming language for this project due to its versatility and the availability of numerous packages for basic statistical analysis and building complex time series models. The following packages were loaded for data cleaning, preparation, building, and plotting the dataset: Numpy, Pandas, Sklearn, Scipy, Seaborn, Plotly, Statsmodels.api, and Mathplotlib.

Pandas profiling is a powerful tool for automating the process of data profiling, which involves generating a comprehensive report of the dataset's characteristics and statistics. The library offers a wide range of features and customization options for generating informative and visually appealing reports. In this article [9], this delve into the capabilities of Pandas profiling and explore some advanced use cases and integrations that can be leveraged to produce compelling reports from data frames. By utilizing this tool, data analysts and scientists can gain a deeper understanding of their data and communicate insights effectively to stakeholders. Please find below a summary of Panda's profiling of the primary attributes of our dataset, provided for the reader's reference.



Further exploratory analysis was conducted using Pandas Profiling. The detailed report can be found at <https://github.com/shahgem/CIND-820/blob/main/CryptoAnalysis.html>

Data Dictionary

The dataset can be described briefly as follows.

Column	Explanation	Variable Type	Data Type
Sno	Serial Numbers	Numerical (Nominal)	int64
Name	Name of the currency	Categorical (Nominal)	object
Symbol	Symbol or abbreviation of the cryptocurrency	Categorical (Nominal)	object

Date	Phone number of customer	Date(datetime)	object
High	Highest price of the cryptocurrency on the given date	Numerical (continuous)	float64
Low	Lowest price of the cryptocurrency on the given date	Numerical (continuous)	float64
Open	Opening price of the cryptocurrency on the given date	Numerical (continuous)	float64
Close	Closing price of the cryptocurrency on the given date	Numerical (continuous)	float64
Volume	Volume of the cryptocurrency traded on the given date	Numerical (continuous)	float64
Marketcap	Market capitalization of the cryptocurrency on the given date	Numerical (continuous)	float64

Missing Values

After reading the “Sixcrypto” dataset into a Pandas dataframe (“df”), we’ve captured a snapshot of the data with 10 columns, 9408 rows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9408 entries, 0 to 9407
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   SNo          9408 non-null   int64  
 1   Name         9408 non-null   object  
 2   Symbol       9408 non-null   object  
 3   Date         9408 non-null   object  
 4   High         9408 non-null   float64 
 5   Low          9408 non-null   float64 
 6   Open         9408 non-null   float64 
 7   Close        9408 non-null   float64 
 8   Volume       9408 non-null   float64 
 9   Marketcap    9408 non-null   float64 
dtypes: float64(6), int64(1), object(3)
memory usage: 735.1+ KB
```

Counting the number of missing values for each column by df.isna().sum().

```
SNo          0
Name         0
Symbol       0
Date         0
High         0
Low          0
Open         0
Close        0
Volume       0
Marketcap    0
dtype: int64
```

Counting the number of duplicated rows for the dataset by df.duplicated().value_counts()

```
False    9408
dtype: int64
```

Our analysis of the dataset revealed that there are no instances of missing values or duplicates. This indicates that the dataset is complete and unique.

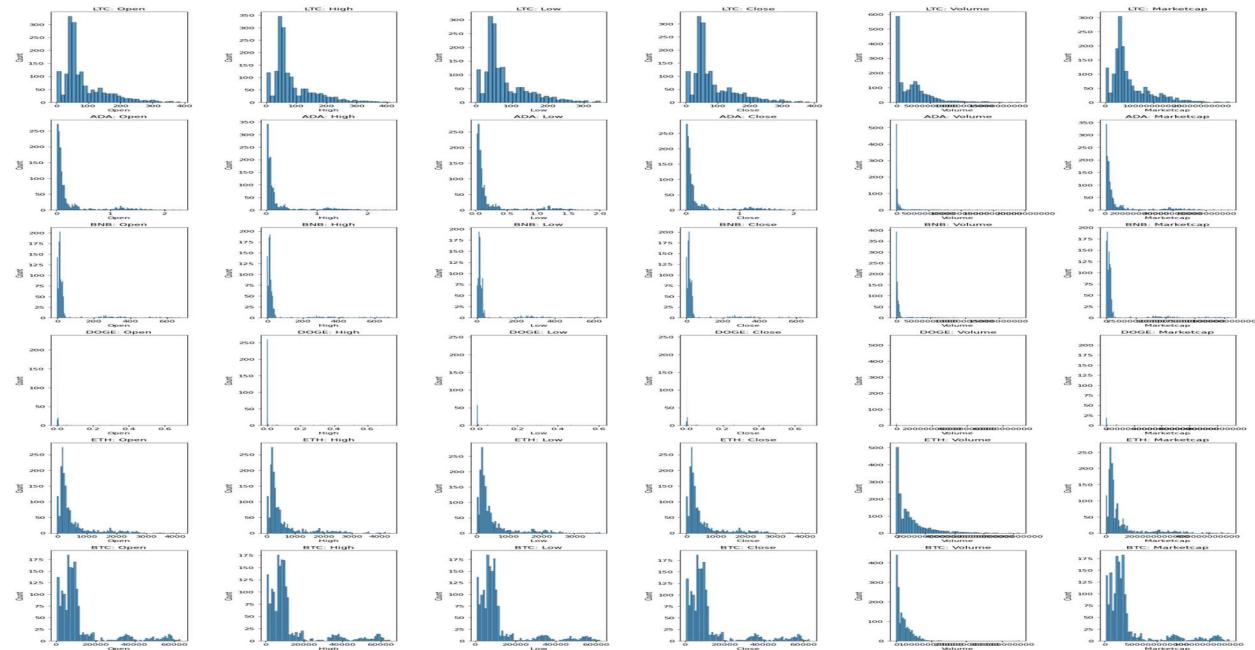
Detection of outliers for numerical columns

Our research aims to identify outliers in the data and examine their influence on the overall distribution of the data. To accomplish this, I employed the commonly used z-score method to identify data points that are over three standard deviations away from the mean, which is a widely accepted approach for detecting outliers.

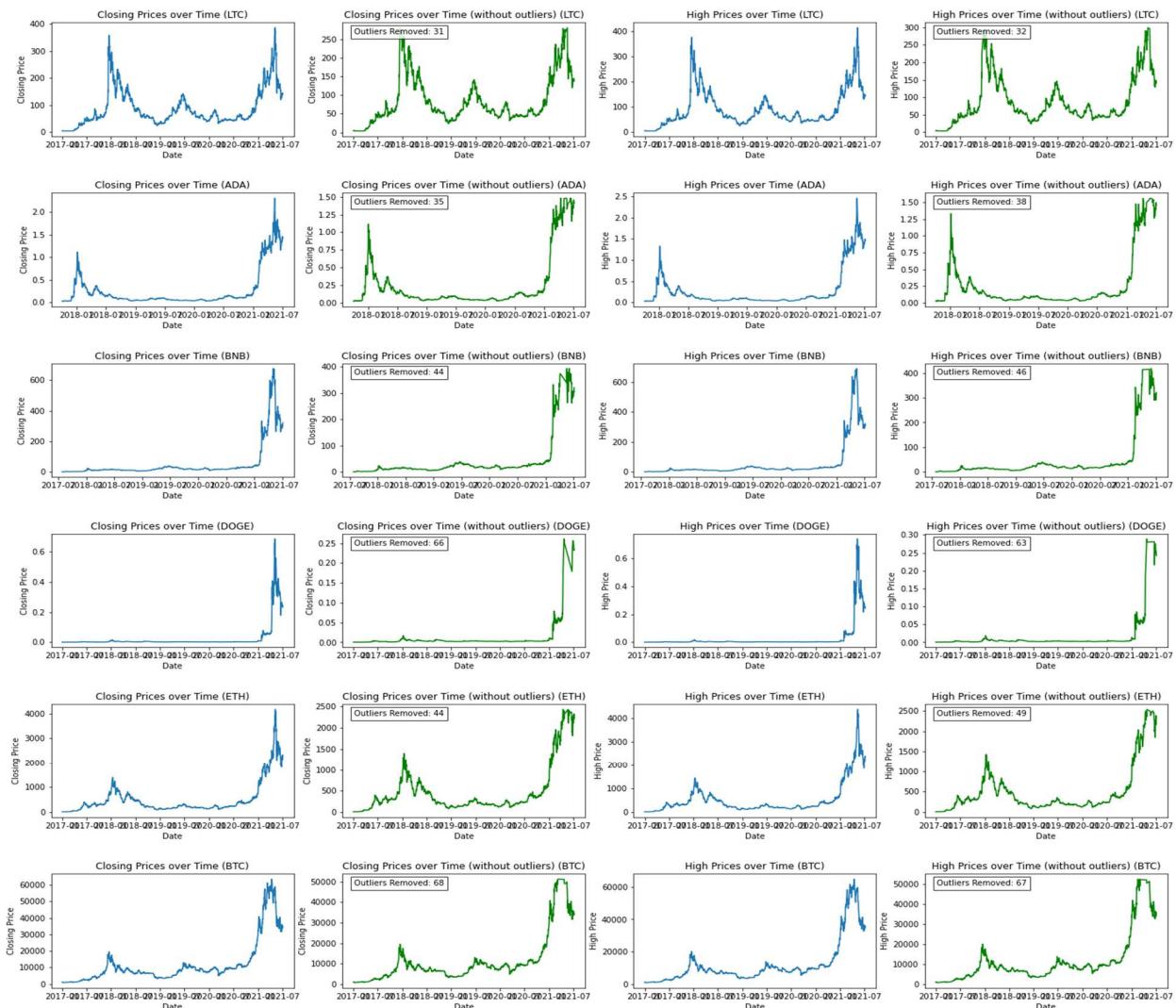
Furthermore, I generated histograms for every currency and column in the dataset to visualize the data distribution and the occurrence of outliers. By doing so, I aimed to gain insight into the extent to which outliers affect the data distribution and potentially impact any further analysis, such as time series forecasting. It is important to note that our approach is not novel but rather a standard practice in the field.

The Python code that I used, imports the pandas, seaborn, and matplotlib libraries for data manipulation, visualization, and plotting.

- It then loads a CSV file named "sixcrypto.csv" using the `read_csv` function of Pandas, which creates a `DataFrame` from the file.
- The code then creates two lists: one containing the names of columns to plot ('Open', 'High', 'Low', 'Close', 'Volume', 'Marketcap'), and the other containing unique currency symbols found in the `Symbol` column of the `DataFrame`.
- Next, the code creates a subplot figure with a grid of plots, one for each currency and column combination. The `subplots` function of Matplotlib is used to create the figure, and the `nrows`, `ncols`, and `figsize` parameters specify the number of rows and columns, and the size of the figure.
- The code then loops through each currency and column combination, and creates a boxplot for each using Seaborn's `histplot` function. The `histplot` function plots a histogram for the data in the specified column, using only the data from the current currency. The `set_title` function sets the title of each plot, indicating the currency and column being plotted, and the `ticklabel_format` function sets the format of the x-axis tick labels to plain text.
- Finally, the code uses `tight_layout` to adjust the spacing of the subplots, and show to display the figure. Overall, this code generates a grid of histograms for each currency and each column of data, allowing the user to compare and contrast the distribution of the data across different currencies.



The below plot shows the closing and high prices of each currency over time, with and without outliers removed. The plot also shows the number of outliers removed for each currency.



Count of Outliers for each Cryptocurrency in the Dataset:

Count of Outliers for each Cryptocurrency in the Dataset						
Currency	High	Low	Open	Close	Volume	Marketcap
LTC	32	28	31	31	36	27
ADA	38	37	37	35	38	42
BNB	46	43	44	44	42	44
DOGE	63	69	67	66	27	66
ETH	49	47	47	44	38	48
BTC	67	65	68	68	14	68

Percentage of Outliers for each Cryptocurrency in the Dataset:

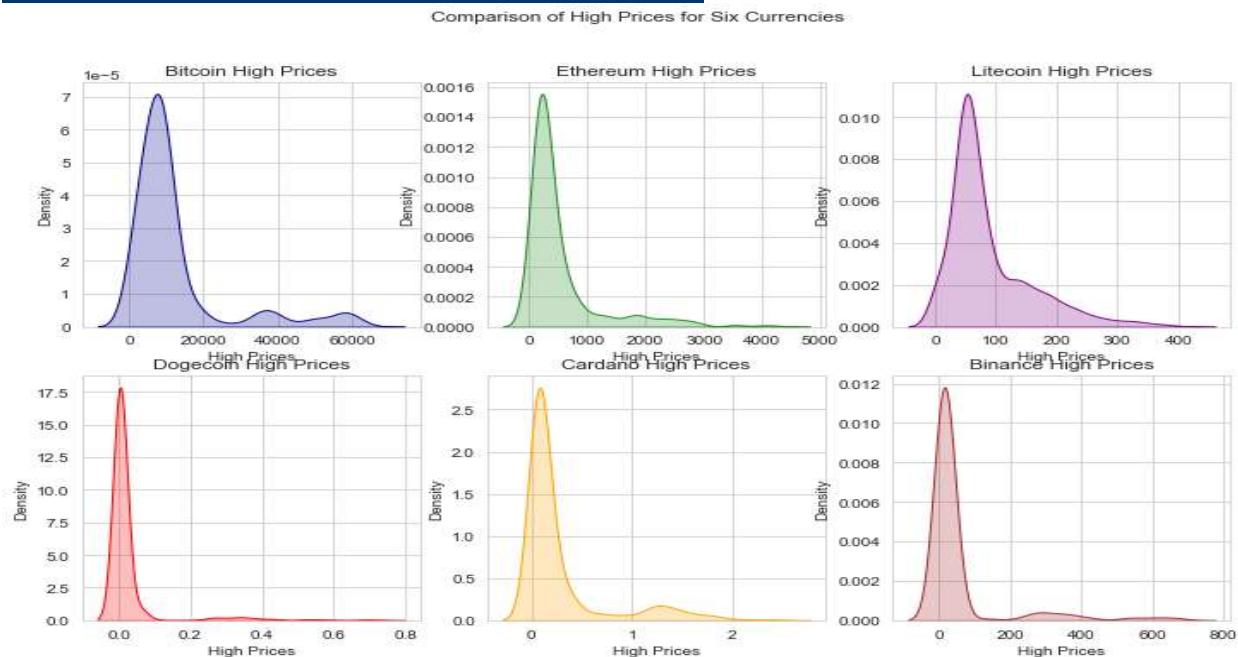
Percentage of Outliers for each Cryptocurrency in the Dataset						
Currency	High	Low	Open	Close	Volume	Marketcap
LTC	1.94%	1.70%	1.88%	1.88%	2.18%	1.64%
ADA	2.77%	2.69%	2.69%	2.55%	2.77%	3.06%
BNB	3.19%	2.98%	3.05%	3.05%	2.91%	3.05%
DOGE	3.82%	4.19%	4.07%	4.00%	1.64%	4.00%
ETH	2.97%	2.85%	2.85%	2.67%	2.31%	2.91%
BTC	4.07%	3.94%	4.13%	4.13%	0.85%	4.13%

Whether or not to remove outliers from your dataset is a decision that depends on the specific context and purpose of our analysis. In some cases, outliers can provide valuable insights into unusual events or behaviors in the data, and removing them may lead to a loss of important information. In other cases, outliers may simply represent errors or anomalies that are not relevant to the analysis, and removing them can improve the accuracy and validity of the results.

In general, it's a good practice to examine the outliers carefully and determine whether they should be removed or retained based on their relevance to the analysis. If we decide to remove outliers, we should be clear about the criteria we used to identify them and document the process for reproducibility. It's also important to consider the potential impact of outlier removal on the overall distribution and characteristics of the data.

Regarding time series forecasting which we are going to perform in the remaining part of our study, outliers can have a significant impact on the accuracy of the forecasts, and it may be necessary to account for them explicitly in the modeling process. In some cases, outlier detection and correction techniques can be integrated into the forecasting models themselves, or separate preprocessing steps may be necessary to remove or adjust the outliers prior to modeling.

Univariate Analysis of High & Close Prices



As we can see, our graphs are skewed to the right (positive skew).

Skewness is a statistical measure that points out the degree of asymmetry of a distribution. It tells you whether the distribution is symmetric or not, and if it is not symmetric, it tells you whether the tail of the distribution is longer on the left or the right side.

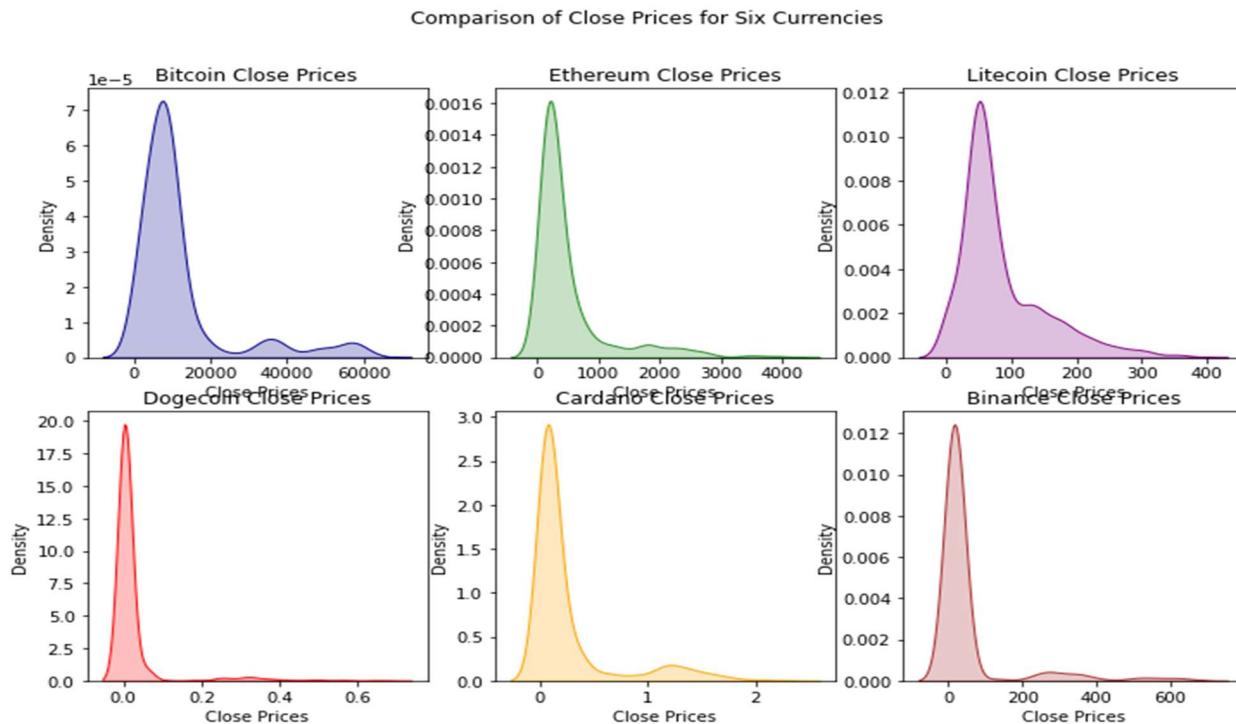
A skewness value of 0 indicates a perfectly symmetric distribution, while a positive skewness value indicates a longer tail on the right side of the distribution, and a negative skewness value indicates a longer tail on the left side of the distribution.

Skewness can be useful in understanding the underlying data, as it can indicate whether there are outliers or unusual data points that may be affecting the distribution. It can also help in choosing appropriate statistical methods for analyzing the data, as some methods assume a normal distribution and may not be appropriate for skewed data.

Symbol	Skewness
BTC	2.318963
ETH	2.672445
LTC	1.608763
DOGE	4.826529
ADA	2.389419
BNB	3.343639

Based on the results, it seems that the "High" prices distribution for DOGE has the highest skewness value (4.826529), indicating a significant degree of positive skewness, which suggests that there are more observations with smaller values (left tail) and fewer observations with larger values (right tail). Similarly, BNB also has a high skewness value (3.343639), indicating a degree of positive skewness. The other cryptocurrencies (BTC, ETH, LTC, and ADA) also have positive skewness values, indicating a similar pattern of more small values and fewer large values, but to a lesser extent compared to DOGE and BNB.

In summary, the skewness values suggest that the "High" prices distribution for these cryptocurrencies are skewed to the right (positively skewed), indicating that there are more observations with smaller values and fewer observations with larger values.

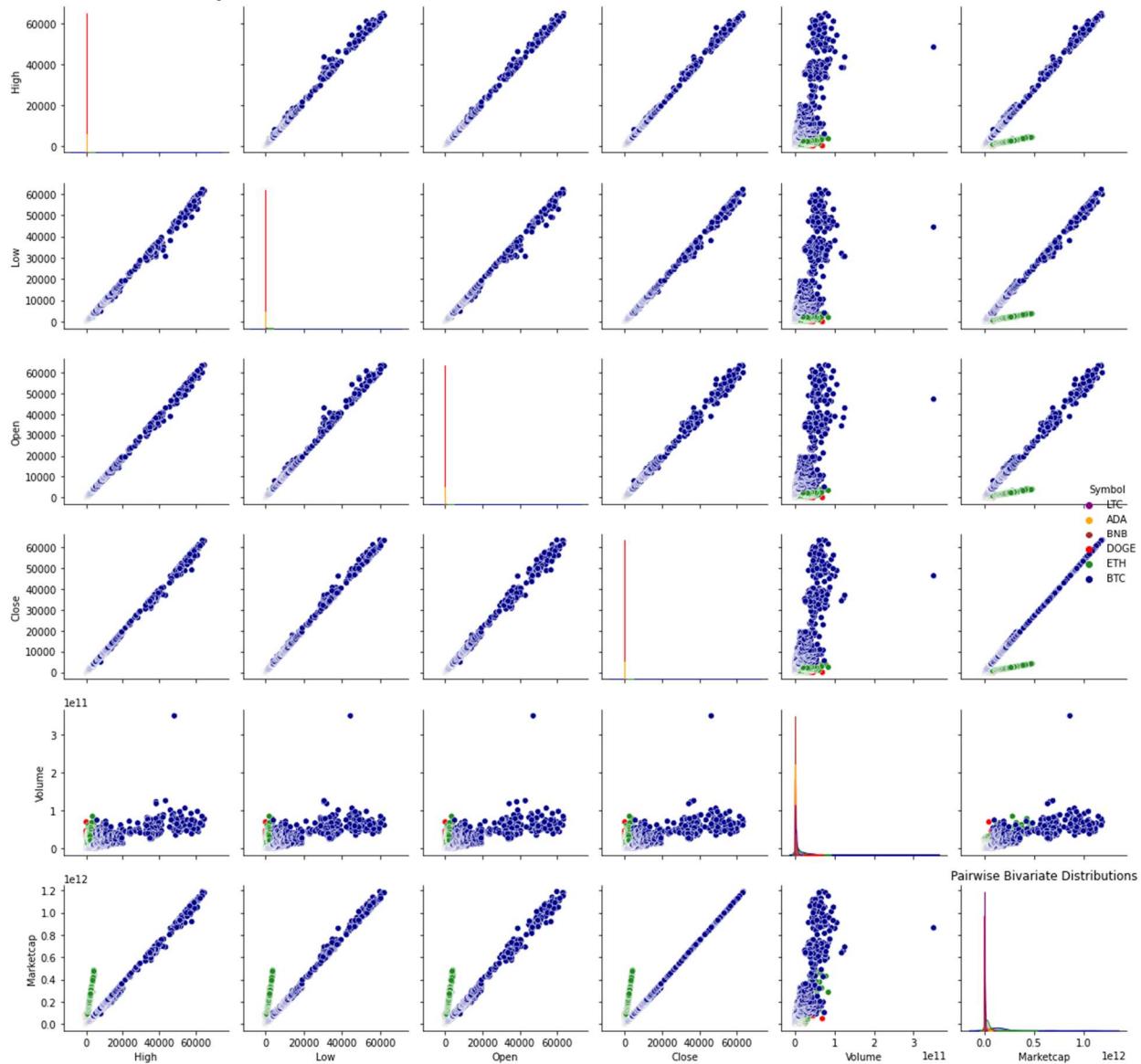


Symbol	Skewness
BTC	2.331511
ETH	2.642688
LTC	1.560778
DOGE	4.701561
ADA	2.378311
BNB	3.359945

Based on the Skewness values of Close price, it appears that DOGE has the highest degree of positive skewness, followed by BNB, ETH, BTC, ADA, and LTC. This suggests that the distributions of daily returns for these cryptocurrencies are not symmetric and have a long right tail. This means that there is a greater probability of large positive returns compared to large negative returns for these cryptocurrencies.

By knowing the skewness of a distribution, investors can better estimate whether the future returns of an asset will be higher or lower than its past returns. This information is helpful in making investment decisions and managing risk. For example, if the skewness is positive, investors might expect to see more large gains in the future and may consider investing in the asset. Conversely, if the skewness is negative, investors might expect to see more large losses in the future and may decide to avoid investing in the asset.

Bivariate Analysis



Bivariate analysis is a statistical method that is used to check the relationship between two variables. In the context of the sixcrypto.csv dataset, bivariate analysis could be used to explore the relationships between different cryptocurrency variables, such as the relationship between the price and volume of each cryptocurrency.

Statistical Summary for numerical columns

`df.describe().transpose()` is a Pandas method that provides descriptive statistics for a DataFrame. The `describe()` method computes summary statistics such as count, mean, standard deviation, minimum, and maximum values for numerical columns in the DataFrame. The `transpose()` method

then switches the rows and columns of the resulting DataFrame, making the statistics for each column appear as rows instead of columns. This makes it easier to interpret the summary statistics for each feature in the dataset, as we can view them side by side rather than having to scan across multiple columns. In summary, `df.describe().transpose()` is a convenient way to obtain a quick overview of the summary statistics for each feature in a DataFrame.

	count	mean	std	min	25%	50%	75%	max
SNo	9408.0	4.704500e+03	2.716000e+03	1.000000e+00	2.352750e+03	4.704500e+03	7.056250e+03	9.408000e+03
High	9408.0	2.245296e+03	7.297368e+03	2.048400e-04	1.060599e-01	4.338992e+01	3.497877e+02	6.486310e+04
Low	9408.0	2.108906e+03	6.823157e+03	1.946130e-04	9.701510e-02	4.100250e+01	3.198715e+02	6.220896e+04
Open	9408.0	2.180658e+03	7.078639e+03	1.966920e-04	1.011220e-01	4.234630e+01	3.378597e+02	6.352375e+04
Close	9408.0	2.184401e+03	7.085992e+03	1.967090e-04	1.013561e-01	4.238147e+01	3.379402e+02	6.350346e+04
Volume	9408.0	5.842552e+09	1.265555e+10	9.284190e+03	8.131160e+07	5.721684e+08	5.068277e+09	3.510000e+11
Marketcap	9408.0	5.086161e+10	1.334378e+11	9.986680e+06	1.184531e+09	4.387144e+09	3.859717e+10	1.190000e+12

The output of `df.describe().transpose()` provides a summary of the key statistical measures for each numerical column in the dataset. The count row indicates the number of non-missing values in each column.

The mean row shows the average value for each column, while the std row indicates the standard deviation, or the amount of variation, in each column's values. The min and max rows show the smallest and largest values in each column, respectively.

The 25%, 50%, and 75% rows represent the first quartile (Q1), median, and third quartile (Q3) of the data distribution, respectively. These values can be used to assess the spread of the data, identify potential outliers, and evaluate the skewness of the distribution.

Based on the output, it can be observed that the dataset has 9,408 observations and 6 numerical features. The Volume and Marketcap features have the highest mean and standard deviation values, indicating significant variability in the data. Meanwhile, the SNo feature represents a sequential identifier for each observation, while the remaining features (High, Low, Open, and Close) represent the price data of a financial asset. The range of values for these price features is quite large, as evidenced by the large difference between the min and max values.

The high, low, open, and close columns represent the daily price of the cryptocurrencies, with mean values of 2.245296e+03, 2.108906e+03, 2.180658e+03, and 2.184401e+03, respectively. The standard deviations are relatively large, indicating a wide variability in the price movements. The minimum and maximum values of the high, low, open, and close columns are 2.048400e-04 and 6.486310e+04, respectively.

The volume column represents the daily trading volume of the cryptocurrencies, with a mean of 5.842552e+09 (SD = 1.265555e+10). The trading volume exhibits a positively skewed distribution, as evidenced by the median value (50%) of 5.721684e+08 being much lower than the mean value. The minimum and maximum values of the volume column are 9.284190e+03 and 3.510000e+11, respectively.

The Marketcap column represents the market capitalization of the cryptocurrencies, with a mean of 5.086161e+10 (SD = 1.334378e+11). The market capitalization also exhibits a positively skewed distribution, with the median value (50%) of 4.387144e+09 being much lower than the mean value. The minimum and maximum values of the Marketcap column are 9.986680e+06 and 1.190000e+12, respectively.

Overall, the `df.describe().transpose()` output provides a quick summary of the key statistics for each numerical feature in the dataset, which can be used to gain insights into the data distribution and identify potential issues or patterns in the data.

Let's delve into the statistical summary of each currency to gain further insights into our dataset.

The code written in Python uses the pandas library to load, manipulate and summarize data from a CSV file named 'sixcrypto.csv'.

- First, it imports the pandas library using the alias 'pd' for easier reference.
- Then, the CSV file is loaded into a pandas DataFrame called 'df' using the `pd.read_csv()` function.
- The 'SNo' column in the DataFrame is dropped using the `drop()` method with the 'axis=1' argument to indicate that it is a column that needs to be dropped. The `inplace=True` argument is used to modify the DataFrame in place rather than creating a copy.
- Next, the DataFrame is renamed as 'crypto_data' using the assignment operator '='.
- The data in the 'crypto_data' DataFrame is then grouped by the 'Symbol' column using the `groupby()` method, which returns a DataFrameGroupBy object.
- The code then loops through the groups and calculates the statistical summary for each group using the `describe()` method on each group. The `describe()` method provides various statistical summary measures like count, mean, standard deviation, minimum and maximum values, quartiles etc.
- The `transpose()` method is then used to transpose the summary statistics from rows to columns for easier readability.
- Finally, the name of the group and its summary statistics are printed out using the `print()` function.

Overall, this code loads and manipulates cryptocurrency data from a CSV file using pandas and calculates statistical summaries for each group of cryptocurrency symbols in the dataset.

Litecoin:

LTC	count	mean	std	min	25%	\
High	1648.0	8.836755e+01	7.040699e+01	3.769480e+00	4.540341e+01	
Low	1648.0	8.076967e+01	6.201480e+01	3.613590e+00	4.331137e+01	
Open	1648.0	8.476022e+01	6.651762e+01	3.714440e+00	4.447457e+01	
Close	1648.0	8.481855e+01	6.645836e+01	3.714530e+00	4.448149e+01	
Volume	1648.0	2.329138e+09	2.596721e+09	2.629220e+06	3.286380e+08	
Marketcap	1648.0	5.178139e+09	4.158405e+09	1.838773e+08	2.742632e+09	
		50%	75%	max		
High	6.095937e+01	1.189067e+02	4.129601e+02			
Low	5.747658e+01	1.088358e+02	3.452988e+02			
Open	5.932576e+01	1.145168e+02	3.878692e+02			
Close	5.933581e+01	1.146692e+02	3.864508e+02			
Volume	1.711569e+09	3.353639e+09	1.799426e+10			
Marketcap	3.656093e+09	6.694942e+09	2.579652e+10			

For Litecoin (LTC), the data includes 1648 daily observations of the high, low, open, close prices, trading volume, and market capitalization. The mean values for the high, low, open, and close prices are 8.836755e+01, 8.076967e+01, 8.476022e+01, and 8.481855e+01, respectively. The standard deviations for these variables are relatively large, with values of 7.040699e+01, 6.201480e+01, 6.651762e+01, and 6.645836e+01, respectively, indicating a wide variability in price movements.

The minimum and maximum values of the high, low, open, and close prices are 3.769480e+00 and 4.129601e+02, respectively. The volume column has a mean of 2.329138e+09 (SD =

2.596721e+09), with a positively skewed distribution as evidenced by the median value (50%) of 1.711569e+09 being much lower than the mean value. The minimum and maximum values of the volume column are 2.629220e+06 and 1.799426e+10, respectively.

The market capitalization column has a mean of 5.178139e+09 (SD = 4.158405e+09), also with a positively skewed distribution, as evidenced by the median value (50%) of 3.656093e+09 being much lower than the mean value. The minimum and maximum values of the market capitalization column are 1.838773e+08 and 2.579652e+10, respectively.

Cardano Coin:

ADA		count	mean	std	min	25%	\
High	1374.0	2.698068e-01	4.335232e-01	2.105030e-02	4.756542e-02		
Low	1374.0	2.397098e-01	3.809276e-01	1.762000e-02	4.460074e-02		
Open	1374.0	2.552866e-01	4.084556e-01	1.841390e-02	4.589775e-02		
Close	1374.0	2.563126e-01	4.096914e-01	1.853910e-02	4.594670e-02		
Volume	1374.0	8.934183e+08	2.107653e+09	1.739460e+06	5.014830e+07		
Marketcap	1374.0	7.603454e+09	1.303878e+10	4.806646e+08	1.191263e+09		
		50%	75%		max		
High		9.027426e-02	1.945189e-01	2.461766e+00			
Low		8.316395e-02	1.724417e-01	2.013285e+00			
Open		8.686724e-02	1.813737e-01	2.300190e+00			
Close		8.700222e-02	1.833791e-01	2.309113e+00			
Volume		1.186742e+08	4.875977e+08	1.914198e+10			
Marketcap		2.270889e+09	5.174547e+09	7.377224e+10			

For Cardano (ADA), the data includes 1374 daily observations of the high, low, open, close prices, trading volume, and market capitalization. The mean values for the high, low, open, and close prices are 2.698068e-01, 2.397098e-01, 2.552866e-01, and 2.563126e-01, respectively. The standard deviations for these variables are 4.335232e-01, 3.809276e-01, 4.084556e-01, and 4.096914e-01, respectively, indicating a wide variability in price movements.

The minimum and maximum values of the high, low, open, and close prices are 2.105030e-02 and 2.461766e+00, respectively. The volume column has a mean of 8.934183e+08 (SD = 2.107653e+09), with a positively skewed distribution as evidenced by the median value (50%) of 1.186742e+08 being much lower than the mean value. The minimum and maximum values of the volume column are 1.739460e+06 and 1.914198e+10, respectively.

The market capitalization column has a mean of 7.603454e+09 (SD = 1.303878e+10), also with a positively skewed distribution, as evidenced by the median value (50%) of 1.191263e+09 being much lower than the mean value. The minimum and maximum values of the market capitalization column are 4.806646e+08 and 2.300190e+09, respectively.

Binance Coin:

BNB		count	mean	std	min	25%	\
High	1442.0	5.476410e+01	1.216758e+02	1.012110e-01	1.039075e+01		
Low	1442.0	4.916581e+01	1.081185e+02	9.610940e-02	9.677340e+00		
Open	1442.0	5.202823e+01	1.151701e+02	9.972120e-02	1.003786e+01		
Close	1442.0	5.225031e+01	1.153909e+02	9.986680e-02	1.006835e+01		
Volume	1442.0	6.269804e+08	1.479775e+09	9.284190e+03	5.089148e+07		
Marketcap	1442.0	7.835965e+09	1.780249e+10	9.986680e+06	1.157863e+09		
		50%	75%		max		
High		1.659211e+01	2.824091e+01	6.909320e+02			
Low		1.572725e+01	2.696304e+01	6.314653e+02			

Open	1.621033e+01	2.766989e+01	6.763159e+02
Close	1.621057e+01	2.769111e+01	6.756841e+02
Volume	1.981830e+08	3.942378e+08	1.798295e+10
Marketcap	2.451099e+09	4.061743e+09	1.040000e+11

For Binance Coin (BNB), the data includes 1442 daily observations of the high, low, open, close prices, trading volume, and market capitalization. The mean values for the high, low, open, and close prices are 5.476410e+01, 4.916581e+01, 5.202823e+01, and 5.225031e+01, respectively. The standard deviations for these variables are 1.216758e+02, 1.081185e+02, 1.151701e+02, and 1.153909e+02, respectively, indicating a wide variability in price movements.

The minimum and maximum values of the high, low, open, and close prices are 1.012110e-01 and 6.909320e+02, respectively. The volume column has a mean of 6.269804e+08 (SD = 1.479775e+09), with a positively skewed distribution as evidenced by the median value (50%) of 1.981830e+08 being much lower than the mean value. The minimum and maximum values of the volume column are 9.284190e+03 and 1.798295e+10, respectively.

The market capitalization column has a mean of 7.835965e+09 (SD = 1.780249e+10), also with a positively skewed distribution, as evidenced by the median value (50%) of 2.451099e+09 being much lower than the mean value. The minimum and maximum values of the market capitalization column are 9.986680e+06 and 1.040000e+11, respectively.

Bitcoin:

BTC						
	count	mean	std	min	25%	\
High	1648.0	1.216162e+04	1.357205e+04	8.233070e+02	4.877465e+03	
Low	1648.0	1.143817e+04	1.264045e+04	7.557560e+02	4.470868e+03	
Open	1648.0	1.181870e+04	1.314910e+04	7.751780e+02	4.613783e+03	
Close	1648.0	1.183847e+04	1.315736e+04	7.777570e+02	4.680163e+03	
Volume	1648.0	1.976129e+10	2.174509e+10	6.085170e+07	4.188452e+09	
Marketcap	1648.0	2.145495e+11	2.473754e+11	1.251914e+10	7.890546e+10	
		50%	75%	max		
High	8.234140e+03	1.110076e+04	6.486310e+04			
Low	7.842872e+03	1.053982e+04	6.220896e+04			
Open	8.063433e+03	1.084634e+04	6.352375e+04			
Close	8.055607e+03	1.085863e+04	6.350346e+04			
Volume	1.365801e+10	2.978197e+10	3.510000e+11			
Marketcap	1.410000e+11	1.940000e+11	1.190000e+12			

The data for Bitcoin shows that there were 1648 daily observations of the high, low, open, close prices, trading volume, and market capitalization. The mean values for the high, low, open, and close prices are 1.216162e+04, 1.143817e+04, 1.181870e+04, and 1.183847e+04, respectively. The standard deviations for these variables are relatively high, indicating that there has been significant variability in price movements.

The minimum and maximum values of the high, low, open, and close prices are 8.233070e+02 and 6.486310e+04, respectively. The volume column has a mean of 1.976129e+10 (SD = 2.174509e+10), with a positively skewed distribution, as evidenced by the median value (50%) of 1.365801e+10 being lower than the mean value. The minimum and maximum values of the volume column are 6.085170e+07 and 3.510000e+11, respectively.

The market capitalization column has a mean of 2.145495e+11 (SD = 2.473754e+11), also with a positively skewed distribution, as evidenced by the median value (50%) of 1.410000e+11 being lower than the mean value. The minimum and maximum values of the market capitalization column are 1.251914e+10 and 1.190000e+12, respectively.

Overall, the data suggests that Bitcoin is a highly volatile asset with significant fluctuations in price movements and trading volume. The market capitalization of Bitcoin is also quite large, indicating that it is a significant player in the cryptocurrency market.

Dogecoin:

DOGE		count	mean	std	min	25%	\
High		1648.0	2.485953e-02	8.806860e-02	2.048400e-04	2.151102e-03	
Low		1648.0	2.058939e-02	7.071171e-02	1.946130e-04	2.055418e-03	
Open		1648.0	2.270210e-02	7.942883e-02	1.966920e-04	2.096444e-03	
Close		1648.0	2.285838e-02	7.968948e-02	1.967090e-04	2.100501e-03	
Volume		1648.0	7.242091e+08	3.604293e+09	5.407950e+04	1.055230e+07	
Marketcap		1648.0	2.933065e+09	1.033952e+10	2.128503e+07	2.498939e+08	
			50%	75%	max		
High		2.736100e-03	3.838227e-03	7.375666e-01			
Low		2.599330e-03	3.496225e-03	6.081677e-01			
Open		2.662656e-03	3.672514e-03	6.878015e-01			
Close		2.663201e-03	3.689054e-03	6.847770e-01			
Volume		3.486363e+07	1.028266e+08	6.941068e+10			
Marketcap		3.263688e+08	4.411314e+08	8.868082e+10			

The data for Dogecoin shows that there were 1648 daily observations of the high, low, open, close prices, trading volume, and market capitalization. The mean values for the high, low, open, and close prices are 2.485953e-02, 2.058939e-02, 2.270210e-02, and 2.285838e-02, respectively. The standard deviations for these variables are relatively high, indicating that there has been significant variability in price movements.

The minimum and maximum values of the high, low, open, and close prices are 2.048400e-04 and 7.375666e-01, respectively. The volume column has a mean of 7.242091e+08 (SD = 3.604293e+09), with a positively skewed distribution, as evidenced by the median value (50%) of 3.486363e+07 being lower than the mean value. The minimum and maximum values of the volume column are 5.407950e+04 and 6.941068e+10, respectively.

The market capitalization column has a mean of 2.933065e+09 (SD = 1.033952e+10), also with a positively skewed distribution, as evidenced by the median value (50%) of 3.263688e+08 being lower than the mean value. The minimum and maximum values of the market capitalization column are 2.128503e+07 and 8.868082e+10, respectively.

Overall, the data suggests that Dogecoin is a highly volatile asset with significant fluctuations in price movements and trading volume. The market capitalization of Dogecoin is relatively small compared to other cryptocurrencies, indicating that it is a smaller player in the cryptocurrency market.

Ethereum:

ETH		count	mean	std	min	25%	\
High		1648.0	5.196492e+02	6.744691e+02	8.436330e+00	1.751977e+02	
Low		1648.0	4.770173e+02	6.069889e+02	7.982310e+00	1.649274e+02	
Open		1648.0	4.995809e+02	6.434077e+02	7.982310e+00	1.702685e+02	
Close		1648.0	5.009299e+02	6.448244e+02	8.172570e+00	1.706190e+02	
Volume		1648.0	9.245476e+09	1.132841e+10	4.689950e+06	1.533612e+09	
Marketcap		1648.0	5.449910e+10	7.462537e+10	7.150492e+08	1.815161e+10	
			50%	75%	max		
High		2.669922e+02	4.931448e+02	4.362351e+03			
Low		2.451433e+02	4.652826e+02	3.785849e+03			
Open		2.571551e+02	4.795873e+02	4.174636e+03			

Close	2.580102e+02	4.797018e+02	4.168701e+03
Volume	5.598255e+09	1.243547e+10	8.448291e+10
Marketcap	2.695053e+10	4.980239e+10	4.830000e+11

The data for Ethereum shows that there were 1648 daily observations of the high, low, open, close prices, trading volume, and market capitalization. The mean values for the high, low, open, and close prices are 5.196492e+02, 4.770173e+02, 4.995809e+02, and 5.009299e+02, respectively. The standard deviations for these variables are relatively high, indicating that there has been significant variability in price movements.

The minimum and maximum values of the high, low, open, and close prices are 8.436330e+00 and 4.362351e+03, respectively. The volume column has a mean of 9.245476e+09 (SD = 1.132841e+10), with a positively skewed distribution, as evidenced by the median value (50%) of 5.598255e+09 being lower than the mean value. The minimum and maximum values of the volume column are 4.689950e+06 and 8.448291e+10, respectively.

The market capitalization column has a mean of 5.449910e+10 (SD = 7.462537e+10), also with a positively skewed distribution, as evidenced by the median value (50%) of 2.695053e+10 being lower than the mean value. The minimum and maximum values of the market capitalization column are 7.150492e+08 and 4.830000e+11, respectively.

Overall, the data suggests that Ethereum is a highly volatile asset with significant fluctuations in price movements and trading volume. The market capitalization of Ethereum is relatively large compared to other cryptocurrencies, indicating that it is a major player in the cryptocurrency market.

Correlation

Correlation Coefficient	Strength of Relationship
-1	Perfect negative correlation
-0.7 to -0.9	Strong negative correlation
-0.5 to -0.7	Moderate negative correlation
-0.3 to -0.5	Weak negative correlation
0	No correlation
0.3 to 0.5	Weak positive correlation
0.5 to 0.7	Moderate positive correlation
0.7 to 0.9	Strong positive correlation
1	Perfect positive correlation

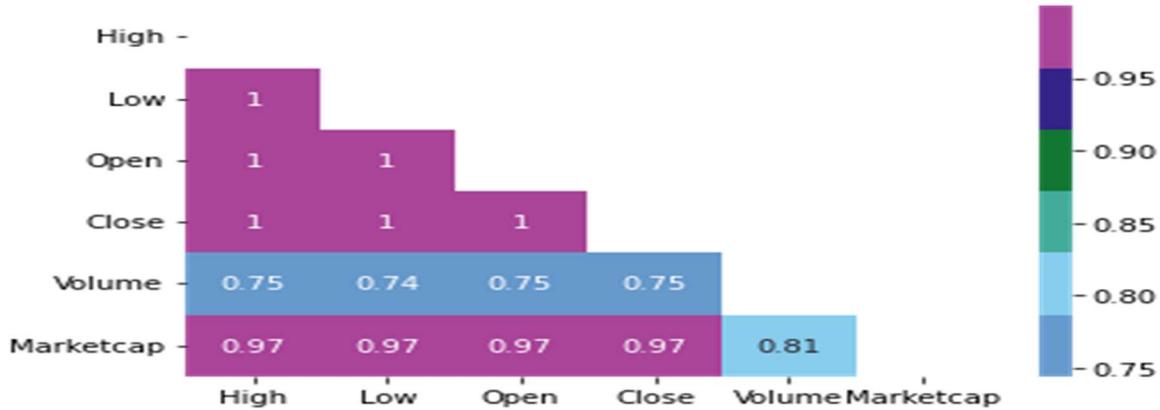
It's worth noting that the interpretation of the correlation coefficient can vary depending on the field of study and the context of the analysis. However, in general, a correlation coefficient between -0.7 and -0.9 (or between 0.7 and 0.9) is considered a strong correlation, while a coefficient between -0.5 and -0.7 (or between 0.5 and 0.7) is considered a moderate correlation. Coefficients between -0.3 and -0.5 (or between 0.3 and 0.5) are typically considered weak correlations.

- The Python code loads a dataset from a CSV file named "sixcrypto.csv" using the Pandas `read_csv()` function, creating a DataFrame object named `df`. The `num_cols` variable is then used to select only the numerical columns from the `df` DataFrame, namely Open, High, Low, Close, Volume, and Marketcap. A new DataFrame named `num_df` is created to hold only these selected numerical columns.

- Next, the correlation matrix between these numerical columns is calculated using the `corr()` method on the `num_df` DataFrame.
- Finally, the correlation matrix is printed to the console using the `print()` function.

	Open	High	Low	Close	Volume	Marketcap
Open	1.000000	0.999546	0.999109	0.998901	0.749004	0.968995
High	0.999546	1.000000	0.999053	0.999525	0.750852	0.969801
Low	0.999109	0.999053	1.000000	0.999423	0.744467	0.969238
Close	0.998901	0.999525	0.999423	1.000000	0.748179	0.970084
Volume	0.749004	0.750852	0.744467	0.748179	1.000000	0.807050
Marketcap	0.968995	0.969801	0.969238	0.970084	0.807050	1.000000

The correlation results show a strong positive correlation among the variables, with correlation coefficients ranging from 0.749 to 1.000. The strongest correlation is observed between the High and Close variables (correlation coefficient of 0.999525), followed by the correlation between the Low and Close variables (correlation coefficient of 0.999423). The Volume variable has a weaker correlation with the other variables (correlation coefficients ranging from 0.744 to 0.750).



The analysis of the correlation results suggests a high level of interdependence among the various cryptocurrency price features, such as Open, High, Low, Close, Volume, and Marketcap. The correlation coefficients between these variables range from 0.749 to 0.970, which indicates a strong positive association between them. This finding has significant implications for modeling and forecasting cryptocurrency prices, as it implies that incorporating all of these variables may lead to data redundancy or multicollinearity. This means that accurately predicting cryptocurrency prices might be challenging because it could be difficult to discern the relative importance of each feature.

To overcome this challenge, further analysis could be conducted to explore the direction and strength of causal relationships among the variables. Such analysis could provide insights into the underlying mechanisms that drive cryptocurrency price movements and help identify the most critical factors to consider in forecasting future price trends. Additionally, external factors that influence cryptocurrency prices, such as economic and political events, could also be examined to determine their impact on price movements. By considering both the internal and external factors that affect cryptocurrency prices, more accurate and reliable forecasting models could be developed.

Principal Component Analysis (PCA)

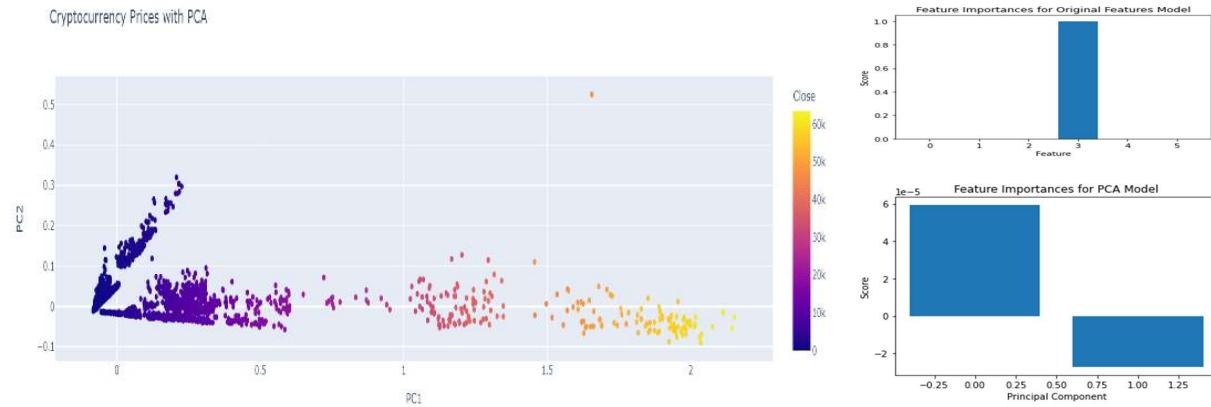
To delve deeper into the analysis, it could be beneficial to perform a principal component analysis (PCA) on the data to identify the most significant features and decrease the dataset's dimensionality. PCA is a popular approach in data analysis and machine learning that involves reducing the dimensionality of high-dimensional data while retaining critical information. The primary objective of PCA is to convert a collection of correlated variables into a smaller set of uncorrelated variables (referred to as principal components), while still preserving the most crucial information.

Here are some key points covered in the article written by Matt Brems [10]:

1. PCA is a mathematical methodology that can convert high-dimensional data into a lower-dimensional space while maintaining crucial information.
2. PCA is a beneficial tool in simplifying the complexity of data, discovering patterns and associations between variables, and enhancing the efficiency of machine learning models.
3. The article provides a step-by-step guide to performing PCA using Python, including data preprocessing, PCA implementation, and visualizing the results.
4. The article also covers important concepts related to PCA, such as variance, covariance, eigenvectors, and eigenvalues.
5. The article provides practical examples of using PCA in real-world scenarios, such as image processing and finance.
6. The article discusses some common pitfalls and limitations of PCA, such as the interpretability of the results and the assumption of linearity.

The code used in python to access the PCA model begins by importing the necessary Python libraries including Pandas, NumPy, Plotly, scikit-learn, and Matplotlib. It then defines a function to calculate the mean absolute percentage error (MAPE) which is used to evaluate the model later on.

- The code reads in a dataset of cryptocurrency prices from a CSV file using Pandas and selects a set of features to use in the analysis. These features are then scaled using the MinMaxScaler from scikit-learn.
- Principal component analysis (PCA) is performed on the scaled features using the PCA function from scikit-learn. Two principal components are selected, and a new dataframe is created using the principal components and the original data. This allows the data to be visualized in two dimensions.
- The data is split into training and testing sets using the train_test_split function from scikit-learn. A linear regression model is fitted on the original features, and the metrics for the model are computed using mean squared error, mean absolute error, MAPE, R-squared, and adjusted R-squared.
- The data is then standardized using the StandardScaler from scikit-learn, and PCA is performed on the standardized data. The principal components are concatenated with the original features, and the data is split into training and testing sets again.
- Another linear regression model is fitted on the principal components, and the metrics for this model are computed. The results of the PCA are visualized using a scatter plot in Plotly.
- Finally, the feature importances for both the original features model and the PCA model are computed and visualized using bar plots in Matplotlib.
- Overall, this code performs a basic analysis of cryptocurrency prices using PCA and linear regression models. The code could be further improved by optimizing the hyperparameters of the models and including additional features in the analysis.



```

Feature importance for the original features model:
Feature: 0, Score: 0.00000
Feature: 1, Score: -0.00000
Feature: 2, Score: 0.00000
Feature: 3, Score: 1.00000
Feature: 4, Score: 0.00000
Feature: 5, Score: -0.00000
Feature importance for the PCA model:
Principal Component: 1, Score: 0.00006
Principal Component: 2, Score: -0.00003

```

The results of the feature importance analysis for the original features model show that only Feature 3 has a score of 1.0, suggesting that it is the only feature that significantly contributes to the model's predictive performance. Conversely, all other features have scores of 0.0, indicating that they do not have a significant impact on the model's predictive power.

On the other hand, the feature importances for the PCA model are based on the principal components. The scores for the principal components are much smaller than 1.0, suggesting that each principal component contributes only a small amount to the model's predictive performance. Specifically, Principal Component 1 has a score of 0.00006, indicating a very small positive impact on the model's performance, while Principal Component 2 has a score of -0.00003, indicating a small negative impact. Overall, the results suggest that the PCA model's performance may be less reliant on any one particular feature or principal component, but instead relies on the combined effects of all of the components.

Regarding which model to use, it ultimately depends on the specific requirements of your problem and the resources available. The original features model may be preferred if interpretability and transparency are important, as it directly uses the original features in the analysis. On the other hand, the PCA model may be preferred if dimensionality reduction and computational efficiency are priorities, as it reduces the dimensionality of the data and may result in faster and more efficient analysis.

Seasonal Decomposition of the Cryptocurrencies Close Price

Seasonal decomposition is a technique used in time series analysis to separate a time series into its underlying components: trend, seasonal, and residual. The trend component represents the long-term progression of the time series, the seasonal component captures the repeating patterns that occur over shorter periods of time (e.g., days, weeks, months, seasons), and the residual component represents the random fluctuations or noise that cannot be explained by the trend or seasonal components.

An article "Different types of Time series decomposition" [11] written by Andrew Plummer has these below key points:

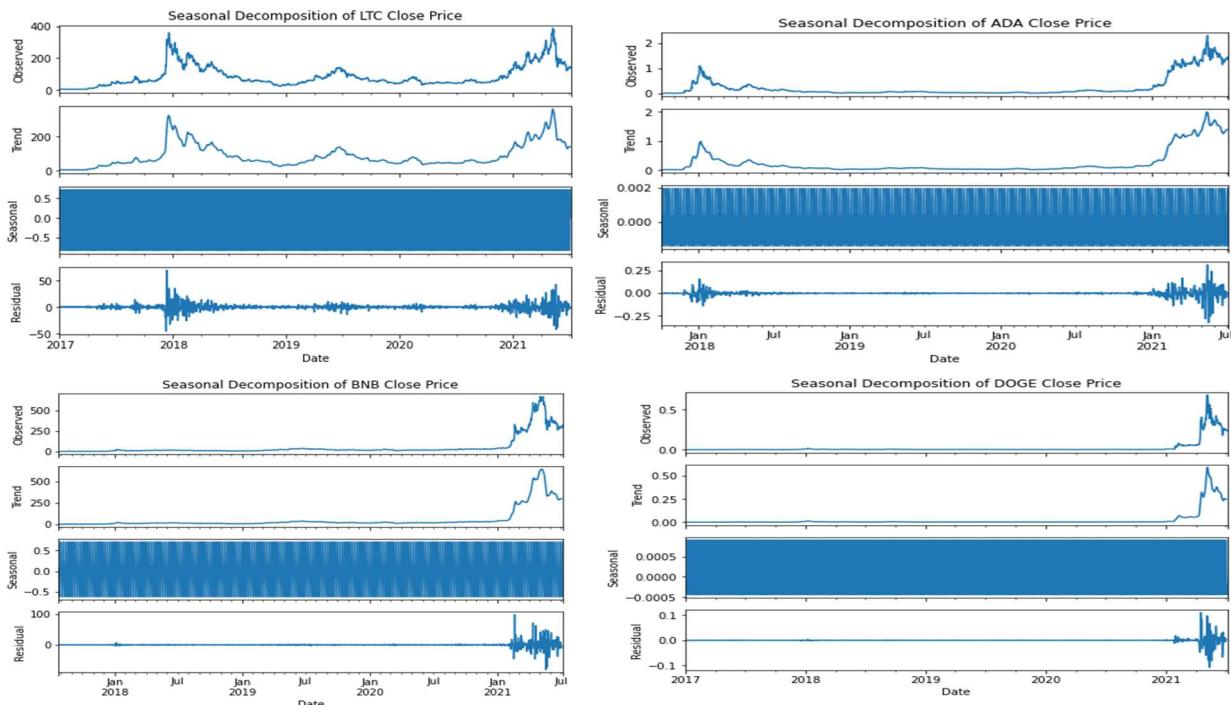
1. In the field of data science, time series decomposition techniques are used to break down a time series dataset into its constituent components, including trend, seasonal, and remainder components.
2. These techniques help analysts identify patterns and trends within the data, which can be useful in making predictions or understanding underlying relationships.
3. The article covers three different time series decomposition techniques: classical decomposition, seasonal decomposition of time series (STL), and empirical mode decomposition (EMD).

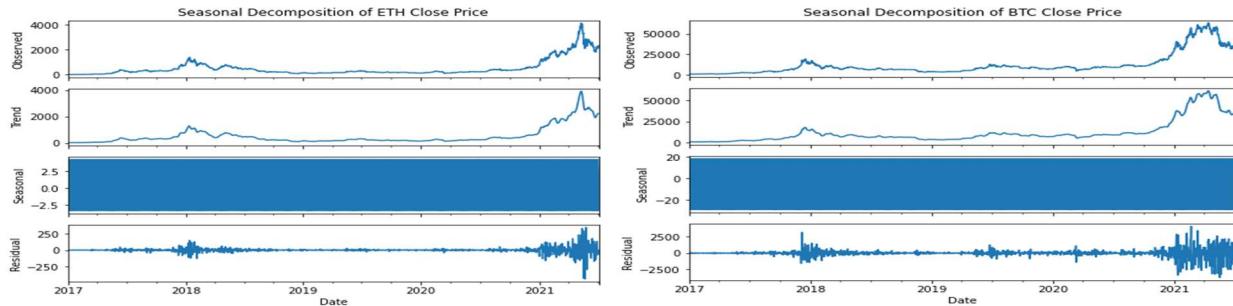
4. Classical decomposition involves identifying the trend, seasonal, and remainder components of a time series, while STL uses a non-parametric approach to extract seasonal and trend components. EMD decomposes a time series into intrinsic mode functions (IMFs) that represent different frequencies in the data.
5. The choice of decomposition technique depends on the specific dataset and research question at hand. By using time series decomposition techniques, data scientists can gain a deeper understanding of time series data and extract meaningful insights.

The python code performs seasonal decomposition of cryptocurrency close prices using the statsmodels library in Python.

Here is what the code does step by step:

- Import necessary libraries - pandas, matplotlib, and statsmodels.
- Load the cryptocurrency data into a pandas dataframe using pd.read_csv.
- Create a new dataframe df_time_series with the necessary columns for time series analysis (Date, Close, and Symbol).
- Convert the Date column to datetime format using pd.to_datetime and set it as the index of the dataframe using set_index.
- Group the data by Symbol and resample it to a daily frequency using groupby and resample.
- Interpolate missing values in the resampled data using linear interpolation.
- Loop through each unique cryptocurrency Symbol in the dataframe.
- Perform seasonal decomposition on the close price data for the current cryptocurrency using sm.tsa.seasonal_decompose with the additive model.
- Create a 4x1 subplots for each component of the seasonal decomposition (observed, trend, seasonal, and residual) using plt.subplots.
- Plot each component in the appropriate subplot using .plot on the res object returned by seasonal_decompose.
- Show the plot using plt.show().





The output of this code is a set of four plots for each cryptocurrency, showing the observed, trend, seasonal, and residual components of the time series decomposition. These plots can be used to analyze the patterns and trends in the cryptocurrency close prices over time.

The goal of seasonal decomposition is to gain a better understanding of the underlying structure of the time series and to identify any patterns or trends that may exist. This can be useful for making predictions about future values of the time series, as well as for identifying potential relationships between the time series and other variables.

The Augmented Dickey-Fuller (ADF) test

The Augmented Dickey-Fuller (ADF) test is a statistical test that aims to determine the stationarity of a given time series. Stationarity refers to the property of a time series in which its statistical characteristics, such as the mean and variance, remain constant over time. This is a crucial assumption in many time series models, as it simplifies the analysis and interpretation of the data.

As a unit root test, the ADF test examines whether a unit root exists in the time series, which is a characteristic of a non-stationary time series that tends to drift over time and does not revert to its mean. The ADF test compares the differences between the observed values in the time series and the expected values if the series were stationary. If the differences are significant enough, the null hypothesis of a unit root is rejected, indicating that the time series is stationary.

The ADF test is based on the augmented Dickey-Fuller regression model, which is a linear regression model that includes lagged differences of the time series as explanatory variables. The ADF test examines the coefficient of the lagged first difference in the regression equation, which represents the rate at which the time series reverts to its mean.

The ADF test has several variations, including the ADF test with a constant and trend term, which allows for a linear trend in the time series. The ADF test can also be applied to seasonal time series, where the seasonal differences are included in the regression equation.

An article on “Augmented Dickey Fuller Test (ADF Test) – Must Read Guide” [\[12\]](#) by Selva Prabhakaran, explains that

1. Stationarity testing is a statistical method utilized to determine whether a time series maintains its consistency in terms of mean and variance over time. When a time series remains stationary, it offers a more straightforward analysis and modeling approach, as compared to non-stationary time series.
2. The ADF test determines if a time series is stationary assuming it has a unit root and is non-stationary. It calculates a p-value that shows the likelihood of obtaining the observed test statistic under the null hypothesis. Rejecting the null hypothesis if p-value < 0.05 categorizes the time series as stationary. The ADF test is crucial for modeling and analyzing time series data.

3. an example of performing the ADF test utilizing the Python's statsmodels library is demonstrated, which involves using a dataset containing the daily closing prices of the S&P 500 index from 2000 to 2021.
4. Additionally, the article explains the commonly used technique of first-order differencing to convert a non-stationary time series into a stationary one, which involves computing the difference between consecutive observations.
5. Finally, the article acknowledges some limitations of the ADF test, such as its sensitivity to the selection of lag length and the assumption of no seasonality. The article recommends using additional tests in conjunction with the ADF test to obtain a more robust assessment of stationarity.

ADF Test Results	ADF Statistic	p-value	1% Critical Value	5% Critical Value	10% Critical Value
LTC	-2.71	0.07	-3.43	-2.86	-2.5
ADA	-0.37	0.92	-3.44	-2.86	-2.56
BNB	-1.19	0.68	-3.43	-2.86	-2.57
DOGE	-2.50	0.11	-3.43	-2.86	-2.56
ETH	-0.29	0.93	-3.43	-2.86	-2.57
BTC	-0.85	0.80	-3.43	-2.86	-2.57

The ADF test outcomes contain insights about the stationarity of each cryptocurrency's time series. The ADF test's null hypothesis is that the time series is non-stationary and has a unit root. If the p-value is below the chosen significance level (often 0.05), the null hypothesis is rejected, indicating that the time series is stationary.

Based on the ADF test results, it is evident that the p-values for all the cryptocurrencies are greater than 0.05. As a result, we fail to reject the null hypothesis, and we cannot claim that any of the cryptocurrencies have a stationary time series.

The ADF Statistic value in the ADF test is a measure of how much the time series deviates from being stationary. It indicates the extent of non-stationarity in the time series. A more negative ADF statistic value provides stronger evidence against the null hypothesis of stationarity.

To summarize, based on the ADF test results, it is indicated that the time series of each cryptocurrency might not be stationary. The results show that none of the cryptocurrencies have p-values less than the significance level of 0.05, which indicates that we cannot reject the null hypothesis of non-stationarity. Additionally, the ADF Statistic values are not more negative than their respective critical values, which also supports the conclusion of non-stationarity.

In conclusion, the ADF test is a powerful tool for testing the stationarity of time series data. It is widely used in many areas of research, including finance, economics, and environmental science. The ADF test provides a robust and reliable method for testing the null hypothesis of a unit root in a time series, allowing researchers to make informed decisions about the statistical properties of their data.

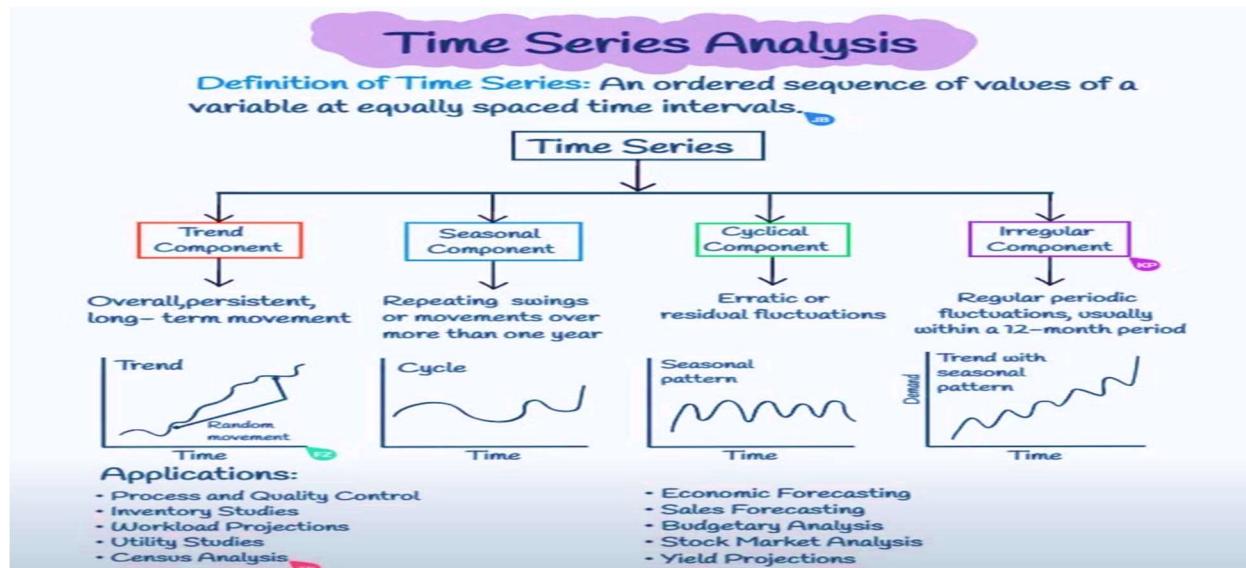
Time-Series Forecasting

Time series analysis is a statistical approach that involves studying data collected over a period of time to identify correlations, patterns, and trends. This technique has numerous practical applications in different fields, such as healthcare, economics, finance, engineering, and marketing. Time series forecasting, which is a subset of this method, is focused on making predictions about future trends and patterns based on historical data.

In finance and economics, time series analysis is commonly used to analyze stock prices, exchange rates, and interest rates, providing insights into long-term trends and patterns to help investors and policymakers make informed decisions. Time series forecasting is also used in cryptocurrency analysis to predict future price movements based on historical data. In healthcare, time series analysis is used to analyze patient data, predict disease outbreaks, and identify patterns in treatment outcomes. In engineering, this technique is used to analyze data from sensors, machines, and equipment to identify patterns and trends, predict equipment failures, and improve maintenance schedules.

Time series analysis comprises four major components, namely trend, seasonality, cyclical variations, and irregular fluctuations. The trend component shows the long-term behavior of the data over time, whether it is increasing, decreasing, or remaining constant. Seasonality relates to the regular patterns of the data that occur at consistent intervals, such as daily, weekly, or monthly. Cyclical variations pertain to fluctuations in the data that transpire over a more extended period than seasonality and are usually caused by economic cycles. Irregular fluctuations denote unforeseen or random variations in the data that cannot be explained by the other three components, namely trend, seasonality, and cyclical variations.

Common techniques used for time-series forecasting include moving averages, exponential smoothing, and ARIMA models. Moving averages involve calculating the average value of a specified number of past data points to identify trends in the data. Exponential smoothing uses a weighted average of past data points to give more weight to recent observations. ARIMA models use a combination of autoregression and moving average techniques to forecast future trends.



Credit: https://www.linkedin.com/feed/update/urn:li:activity:7038863762661023744?utm_source=share&utm_medium=member_desktop

To sum up, time series analysis is a crucial statistical technique that helps in extracting significant insights from historical data and predicting future trends. Its wide-ranging applications across

industries make it an indispensable tool for decision-making. The four primary components of time series analysis - trend, seasonality, cyclical variations, and irregular fluctuations - help identify the patterns and fluctuations in data. The choice of time-series forecasting technique depends on the data and the specific context being analyzed. Techniques like autoregression, moving average, exponential smoothing, and ARIMA models are popular and effective for forecasting future values while accounting for various patterns in the data. Accurate time series forecasting is essential for various applications such as finance, economics, and marketing. Jason Brownlee's article on "Time Series Forecasting" [13] provides valuable insights into the definition of time series data, the process of time series forecasting, and the different techniques used for forecasting.

The article presents several key points that are worth noting:

1. Time series data comprises observations taken over time, such as daily stock prices or monthly sales figures.
2. Time series forecasting involves making predictions about future values based on past observations.
3. The primary objective of time series forecasting is to accurately predict future values while considering trends, seasonality, and other patterns in the data.
4. There are various popular techniques for time series forecasting, such as autoregression, moving average, exponential smoothing, and ARIMA models.
5. Autoregression involves predicting future values based on past values of the same variable.
6. Moving average entails taking the average of a specific number of past data points to identify trends in the data.
7. Exponential smoothing involves using a weighted average of past data points, with more weight given to recent observations.
8. ARIMA models combine autoregression and moving average techniques to model the underlying patterns in the data and make predictions.
9. When selecting a time series forecasting technique, it is critical to consider the data's characteristics, such as seasonality, trend, and noise, as well as the analysis's objectives.
10. Time series forecasting can be used in a range of applications, including finance, economics, and marketing, to forecast future trends and patterns.

Here's a SWOT analysis on time series forecasting:

Strengths	Weaknesses
Helps identify trends and patterns in historical data	Can be inaccurate if the data is noisy or contains outliers
Can be used to make informed decisions about future trends and patterns	Can be limited in its predictive capabilities if the underlying patterns in the data change
Offers a variety of techniques to choose from	Can be difficult to select the most appropriate technique for a specific context and data set
Can be particularly useful in the context of financial analysis, such as predicting stock prices or currency exchange rates	Can be resource-intensive in terms of data collection and processing
Provides a quantitative and data driven approach to decision-making	Can be impacted by external factors that are difficult to predict or model

Opportunities	Threats
Can be used to identify new market trends and opportunities	Can be impacted by changes in external factors, such as changes in government policies or economic conditions
Can help companies make informed decisions about inventory management, production planning, and supply chain optimization	Can be subject to the "garbage in, garbage out" problem if the data used for analysis is of poor quality
Can be used to forecast demand for products and services	Can be impacted by the availability and quality of data
Can help companies stay competitive by identifying emerging trends before competitors	Can be subject to overreliance on historical data, which may not always be a reliable indicator of future trends
Can be used to manage risks, such as predicting the likelihood of default or fraud	Can be impacted by changes in technology or the availability of alternative forecasting methods

If you have an interest in time series forecasting and its ability to predict future trends and patterns, "Forecasting: Principles and Practice" by Rob J Hyndman and George Athanasopoulos [14] could be a valuable resource to explore. This book offers a thorough examination of the fundamental principles that underpin time series forecasting and includes an overview of various techniques, including machine learning algorithms, ARIMA models, and exponential smoothing.

The main objective of time series forecasting is to generate accurate predictions of future trends and patterns. Exponential smoothing is a popular technique that assigns more weight to recent observations to account for present trends and seasonality. Conversely, ARIMA models combine autoregression and moving average methods to predict underlying patterns. However, implementing machine learning algorithms can be tough, requiring careful feature engineering and being computationally demanding.

To assess the accuracy of predictions, standard metrics like mean absolute error and root mean squared error are commonly used. Additionally, the presence of trends, seasonality, outliers, and missing values can all affect forecasting accuracy. Despite these difficulties, time series forecasting has various applications in finance, economics, and marketing, where historical data can be employed to generate knowledgeable predictions about future trends. Ensemble methods, which combine multiple forecasts to enhance accuracy, can be utilized to further refine predictions.

While it may be challenging to condense the essential concepts discussed in the book, I have endeavored to compile a summary of the key points for the reader's benefit.

1. Time series data is a sequence of observations collected over time, and can be used to make predictions about future values.
2. The goal of forecasting is to produce accurate predictions while taking into account trends, seasonality, and other patterns in the data.
3. There are several methods for time series forecasting, including exponential smoothing, ARIMA models, and machine learning algorithms.
4. Exponential smoothing involves weighting recent observations more heavily than older observations, and can be adapted to handle different types of trends and seasonality.
5. ARIMA models combine autoregression (predicting future values based on past values) and moving average techniques to model the underlying patterns in the data and make predictions.
6. While machine learning algorithms can be utilized for time series forecasting, they demand meticulous feature engineering and can result in high computational costs.

7. The accuracy of forecasts can be evaluated by utilizing various metrics, such as mean absolute error and root mean squared error.
8. Time series data can display different forms of seasonality, including additive and multiplicative seasonality.
9. Time series data can also exhibit trends, which can be linear or nonlinear.
10. Outliers and missing values can impact forecasting accuracy, and must be handled appropriately.
11. Forecasting can be used in a variety of applications, including finance, economics, and marketing, to make predictions about future trends and patterns.
12. Forecasting can be challenging and requires careful consideration of the characteristics of the data and the appropriate forecasting method to use.
13. Forecasting can be improved through the use of ensemble methods, which combine multiple forecasts to produce a more accurate prediction.

In summary, forecasting techniques, like time series analysis, have become indispensable tools for businesses to make well-informed decisions about upcoming trends and patterns in various sectors, including finance, economics, and marketing. Selecting the most appropriate forecasting technique based on the data's characteristics and analysis objectives while ensuring the data's accuracy and quality are essential for effective forecasting.

Despite potential challenges, time series forecasting presents various opportunities for firms to detect emerging trends, optimize their inventory planning and supply chain management, anticipate demand for products and services, and manage risks. Successful implementation of time series forecasting can provide companies with a competitive advantage and keep them ahead of emerging trends.

However, firms must also be aware of potential obstacles, including external factors, data quality issues, over-reliance on historical data, and advances in technology or alternative forecasting methods.

Simple Moving Average (SMA)

The Simple Moving Average (SMA) is a frequently used approach for time series analysis and forecasting. It is a type of moving average that smooths out short-term variations in a time series by calculating the average of the series for a particular time frame. To compute SMA, the values of a time series are summed over a specific number of periods, and then this total is divided by the number of periods.

SMA has a wide range of applications in various fields, including finance, marketing, and sales. In finance, SMA is often used to identify trends in stock prices or other financial data. In marketing and sales, SMA can help identify trends in customer behavior or sales patterns. This technique is particularly useful for identifying long-term trends because it reduces short-term fluctuations that may be caused by noise or other factors.

One of the main advantages of SMA is its simplicity. It is straightforward to calculate and does not require complex mathematical formulas or models. As a result, it is accessible to a wide range of users, including those with limited technical expertise. Furthermore, SMA produces a single value that represents the average of the time series over a specified period, making it easy to interpret.

Despite its usefulness, the Simple Moving Average (SMA) has some limitations that should be taken into account. Firstly, the SMA relies solely on historical data and does not consider external factors that might impact the time series. Moreover, the SMA may not be adequate for anticipating sudden changes or shifts in the time series as it is based on the average of past values. Lastly, choosing the

appropriate time period for SMA calculation is crucial, as it can have a considerable effect on the outcomes and might require experimentation to determine the most suitable value.

Despite its limitations, the SMA remains a widely used technique in time series analysis and forecasting, particularly in cases where the time series exhibits a consistent pattern over time. In addition, the SMA can be combined with other techniques, such as exponential smoothing or ARIMA models, to produce more accurate forecasts.

The article titled "Moving Averages in Python" [15] introduces the concept of moving averages, which is a widely used statistical technique for smoothing time series data. The author explains the different types of moving averages, including simple moving average (SMA), exponential moving average (EMA), and weighted moving average (WMA). The article also provides step-by-step instructions on how to calculate and visualize these moving averages in Python using the Pandas library.

The author emphasizes the importance of choosing an appropriate window size for the moving average calculation, as this can significantly impact the resulting data smoothing. The article also touches on how moving averages can be used in trading strategies, such as identifying trends and support/resistance levels.

here are 5 key points from the article "Moving Averages in Python":

1. Moving averages are commonly used in time-series analysis to remove short-term fluctuations in data and emphasize longer-term trends.
2. There are two main types of moving averages: simple moving average (SMA) and exponential moving average (EMA). SMA calculates an average of a specified number of data points, while EMA gives more weight to recent data points.
3. Python offers various libraries, such as NumPy and Pandas, with built-in functions for computing moving averages. The Pandas' rolling() function is particularly helpful for computing both SMA and EMA.
4. The choice of window size for a moving average depends on the specific needs of the analysis. Typically, smaller window sizes work better for shorter-term trends, while larger window sizes are better for longer-term trends.
5. Moving averages are a useful tool in analyzing trends and patterns in data, but it's essential to use other analytical methods along with them to gain a comprehensive understanding of the data.

Overall, the article serves as a useful introduction to moving averages in Python for data analysts and traders alike, with practical examples and clear explanations of the underlying concepts.

Below is the SWOT analysis breakdown that we will use to analyze it:

Strengths	Weaknesses
Easy to calculate and understand	Can be affected by outliers and sudden price changes
Widely used and popular among traders and analysts	Can lag behind sudden price movements
Can be customized to fit different time frames and market conditions	May not be as effective in volatile markets
Helps to smooth out price fluctuations and identify trends	Does not take into account other factors that may impact price movements

Can be combined with other technical indicators for better results	Relies heavily on historical data and may not accurately predict future prices
Opportunities	Threats
Can be used for different types of securities and markets	Increasing use of machine learning and advanced statistical models in trading and analysis
Can be adapted to different trading strategies and styles	Increased competition from other technical indicators and analysis methods
Can be used for both short-term and long-term trading and investment	Regulatory changes and market disruptions can impact its effectiveness
Can be automated and integrated into trading algorithms	Rapid changes in market conditions and prices can make it less effective

Note: This SWOT analysis is not exhaustive and is intended to provide a general overview of the strengths, weaknesses, opportunities, and threats of using Simple Moving Average (SMA) in financial analysis and trading.

The utilization of Python programming language and two commonly used libraries, pandas and matplotlib, in the following code facilitates the loading, filtering, and visualization of time series data for six distinct cryptocurrencies. This can aid in the examination of trends and patterns in cryptocurrency prices, thereby enabling informed investment decision-making.

To visualize, the below steps are utilised for python code:

- The data is stored in a CSV file and is loaded into a pandas dataframe using the `read_csv()` function. The data is then filtered for each of the six cryptocurrencies using their respective symbol codes.
- Next, the code calculates the 60-day Simple Moving Average (SMA) for each currency using the `rolling()` and `mean()` functions. The SMA is a commonly used technical analysis indicator that helps to smooth out price fluctuations and identify trends in the data.
- Finally, the code creates a plot using the `plot()` function from matplotlib, which shows the time series data and the SMA for each cryptocurrency in one figure. The plot is customized with labels, title, and legend, and is displayed using the `show()` function.

Overall, this code demonstrates how to use Python and pandas to load and manipulate time series data, and how to use matplotlib to create customized visualizations of the data.

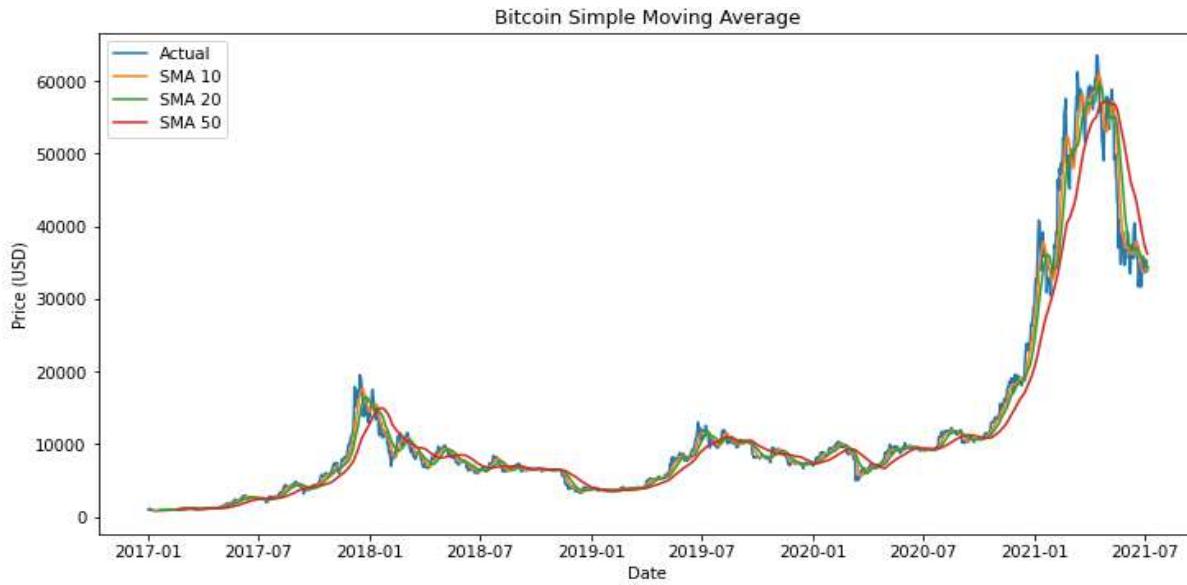
For our SMA Analysis, First, we load the data into a panda DataFrame, filters the data for each currency, and calculates the 60-day SMA using the `rolling()` method. Then, it plots the time series data and the SMA for each currency in one figure using matplotlib.



The resulting plot allows us to visualize the historical price data for each cryptocurrency and the trend of the SMA. We can use this plot to identify potential buying or selling opportunities based on the relationship between the actual price and the SMA. For example, when the price is above the SMA, it may indicate that the cryptocurrency is overvalued, while a price below the SMA may indicate that it is undervalued.

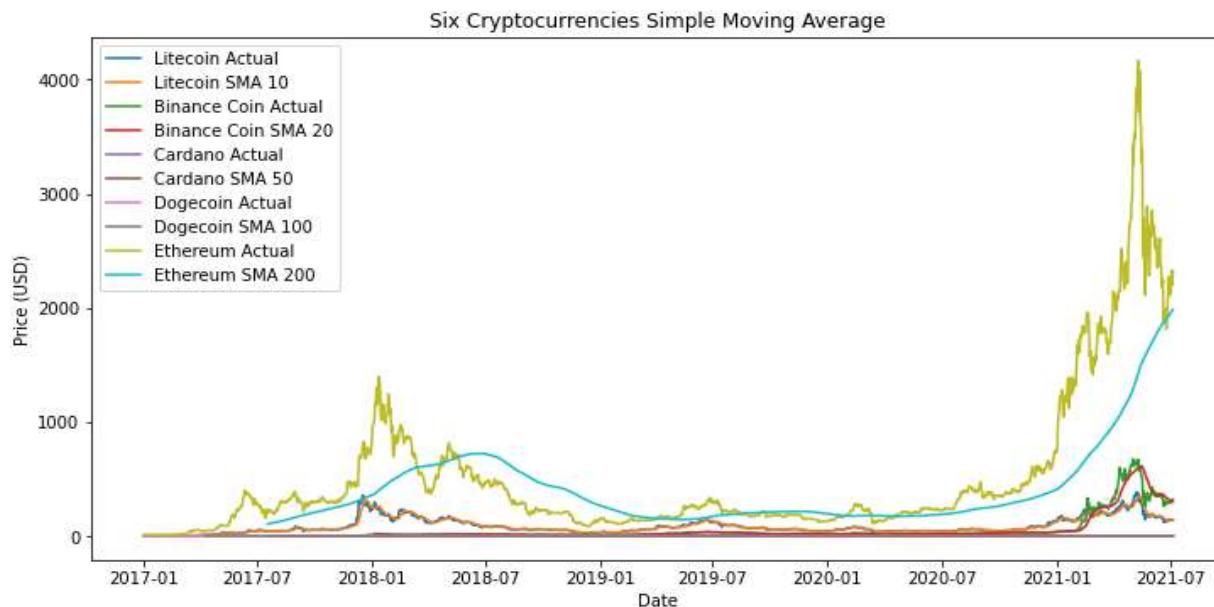
Since Bitcoin has a different level of volatility and trend than the other cryptocurrencies in the dataset and calculating SMA with the same window size for all cryptocurrencies may not be an optimal strategy since each cryptocurrency has its own unique characteristics. Therefore, it is recommended to experiment with different window sizes for each cryptocurrency to find the best performing strategy. In this case, we used a window size of 10-20-50 for Bitcoin as it was the best performing strategy for that particular cryptocurrency.

This code below loads a CSV file containing time series data of six different cryptocurrencies into a pandas DataFrame, and then filters the data to create six separate DataFrames for each currency. It then defines a function to calculate Simple Moving Averages (SMAs) for a given window size, and calculates SMAs for Bitcoin using three different window sizes (10, 20, and 50) and for the other cryptocurrencies using different window sizes. Finally, it uses the matplotlib library to plot the actual prices and SMAs for Bitcoin and the other cryptocurrencies separately in two different plots, with each plot containing multiple lines representing different currencies and their corresponding SMAs. The resulting plots help analyze the trends and patterns in the prices of the cryptocurrencies over time and inform investment decisions.



The reason why we calculate the Simple Moving Averages (SMA) for different days for each currency is that the SMA is a type of time series analysis technique that smooths out the data by calculating the average price over a specified time period. The length of the time period is typically determined by the trader or analyst and can vary depending on their investment strategy and the volatility of the asset being analyzed.

For example, a shorter period SMA (such as 10 or 20 days) may be more suitable for a highly volatile asset such as Litecoin or Binance Coin, as it can help to identify short-term trends and price movements. On the other hand, a longer period SMA (such as 100 or 200 days) may be more appropriate for a less volatile asset such as Dogecoin or Ethereum, as it can help to smooth out the data and identify longer-term trends.



Therefore, different lengths of SMA are often used to calculate moving averages for each asset, taking into account its unique characteristics and volatility. However, it is important to note that while SMA can help identify trends, it is a lagging indicator and should be used alongside other technical indicators and fundamental analysis for a more comprehensive understanding of the market. As with any trading strategy, it's important to thoroughly test and validate the results before making investment decisions.

Next, we will proceed to test the model and assess the performance of the SMA evaluation model.

- The code that I utilised in Python script that calculates the Simple Moving Average (SMA) for various cryptocurrencies over a 30-day and 60-day period, and then calculates the mean squared error (MSE) and mean absolute error (MAE) for each currency. The code starts by importing the pandas and numpy libraries and then loads the data from a CSV file named 'sixcrypto.csv' into a pandas dataframe called 'df'. The data includes information about six different cryptocurrencies, including their symbols, dates, opening prices, closing prices, highest prices, lowest prices, and trading volumes.
- The code then defines a function called 'calculate_sma' that takes two arguments: 'data' and 'window_size'. The 'data' argument is the closing prices of a particular cryptocurrency, while the 'window_size' argument is the size of the window over which to calculate the SMA. The function calculates the SMA using the pandas rolling() function, and then returns the resulting SMA.
- After defining the 'calculate_sma' function, the code defines two empty dictionaries called 'results_30' and 'results_60' to store the results for each currency for both the 30-day and 60-day periods. The code then loops through each unique symbol in the 'Symbol' column of the 'df' dataframe and performs the following actions:
 - Filters the 'df' dataframe to only include rows with the current symbol.
 - Calls the 'calculate_sma' function with a window size of 30 and 60, and stores the resulting SMA in variables called 'sma_30' and 'sma_60', respectively.
 - Calculates the MSE and MAE between the SMA and the closing prices for the current symbol over the 30-day period, and stores the results in a dictionary with the keys 'MSE' and 'MAE', respectively.
 - Adds the dictionary from step 3 to the 'results_30' dictionary using the current symbol as the key.

- Calculates the MSE and MAE between the SMA and the closing prices for the current symbol over the 60-day period, and stores the results in a dictionary with the keys 'MSE' and 'MAE', respectively.
- Adds the dictionary from step 5 to the 'results_60' dictionary using the current symbol as the key.
- Finally, the code prints out the results in a tabular format using the Python string formatting method. The first line of the table includes column headers, while the subsequent lines include the results for each currency. The columns display the currency symbol, the MSE and MAE for the 30-day period, and the MSE and MAE for the 60-day period, respectively.

Currency	MSE (30 day)	MAE (30 day)	MSE (60 day)	MAE (60 day)
LTC	651.0176065067	13.8417116565	1190.2500117420	20.3998952670
ADA	0.0113313243	0.0481830373	0.0243255183	0.0764471746
BNB	1182.7233587541	10.6214267300	2564.4545633058	18.1319001012
DOGE	0.0010242335	0.0068243710	0.0019923583	0.0097895620
ETH	26358.9960465244	79.7710654671	52886.9139518670	118.1584317936
BTC	6461706.4679676499	1320.5815027141	16064760.4267840981	2191.2381493156

Based on these results, it appears that the 30-day model has lower MSE and MAE for all currencies except for LTC, where the 60-day model has a lower MAE. Therefore, overall, the 30-day model seems to be performing better than the 60-day model.

Feature selection using a random forest regression and Gradient Boaster:

I utilised the python code which performs the following steps,

- Imports necessary libraries (pandas, numpy, RandomForestRegressor, and GradientBoostingRegressor).
- Reads a CSV file 'sixcrypto.csv' containing historical cryptocurrency data into a DataFrame (df1).
- Defines a window size (60 days) for calculating the Simple Moving Average (SMA).
- Iterates over each unique cryptocurrency symbol in the dataset, calculates the SMA with a window size of 60 days, and generates predicted prices using the SMA.
- Calculates the correlation between the predicted and actual prices for each currency and stores the correlation coefficients in a dictionary.
- Creates a DataFrame (corr_df) from the correlation coefficients dictionary and sorts it in descending order.
- Selects the correlated currencies and extracts the features and target variable ('Close') for training the RandomForestRegressor (rf_model) and GradientBoostingRegressor (gb_model) models.
- Trains both models using the extracted features and target variable.
- Calculates the feature importance for both models and stores them in two separate DataFrames (rf_feat_importance and gb_feat_importance).
- Combines both feature importance DataFrames into one (feat_importance) and prints the feature importance side by side for both models.
- Prints the top correlated currencies and their correlation coefficients.

Feature	Importance Score (RF)	Importance Score (GB)	Explanation
High	0.234709	0.521129	GB considers "High" as the most important feature, likely because it reflects the upper limit of a security's price movement. RF also considers "High" an important feature, but ranks it lower than "Low".
Low	0.391127	0.419529	Both GB and RF consider "Low" as the most important feature, likely because it reflects the lower limit of a security's price movement.
Open	0.334736	0.001791	RF considers "Open" as an important feature, likely because it reflects the initial investor sentiment for the security. GB assigns low importance to "Open" compared to other features.
Volume	0.000019	0.000005	Both GB and RF consider "Volume" as the least important feature, likely because it doesn't have a strong direct correlation with the price of the security.
Marketcap	0.039408	0.057547	GB considers "Marketcap" as the third most important feature, likely because it reflects the overall value of the security. RF also considers "Marketcap" an important feature, but ranks it lower than "Open"

Random forest constructs each decision tree independently and then averages the feature importance of all trees to obtain an overall importance score. Gradient boosting, on the other hand, constructs trees sequentially, with each tree correcting the errors of the previous tree. The feature importance in gradient boosting is calculated by how much each feature contributes to reducing the loss function during the training process.

Both GB and RF models agree that "Low" is the most important feature for the SMA model, followed by "High" and "Open". These features likely reflect the lower and upper limits of a security's price movement and the initial investor sentiment for the security, respectively.

GB assigns higher importance scores to all features compared to RF, indicating that GB considers all features as important for the SMA model. RF, on the other hand, assigns relatively lower importance scores to "High" and "Marketcap" compared to GB.

"Volume" and "Marketcap" are the least important features for both GB and RF models, indicating that these features do not have a strong direct correlation with the price of the security.

Based on these results, we can conclude that a simple model using only "Low", "High", and "Open" features may be sufficient for predicting SMA for a given security. However, if more features are desired, then "Marketcap" could be considered as an additional feature, especially if using a GB model.

Correlation coefficient of Predicted and actual price:

The correlation coefficient measures the strength and direction of the linear relationship between the predicted and actual prices. A value of 1 indicates a perfect positive linear correlation, while a value of -1 indicates a perfect negative linear correlation. A value of 0 indicates no linear correlation. In the context of the SMA model, a higher correlation coefficient indicates a better fit of the predicted

values to the actual values, suggesting that the model is better able to capture the underlying trends in the data.

The interpretation of the correlation's coefficient is as follows:

Cryptocurrency	Correlation Coefficient	Explanation
Bitcoin (BTC)	0.951533	BTC has the highest correlation coefficient among the cryptocurrencies listed here, indicating a strong positive correlation between the predicted and actual prices. This means that the predicted prices using the SMA model are highly correlated with the actual prices of BTC.
Ethereum (ETH)	0.935910	ETH also has a strong positive correlation coefficient, indicating a strong positive correlation between the predicted and actual prices. This means that the predicted prices using the SMA model are highly correlated with the actual prices of ETH.
Cardano (ADA)	0.929577	ADA has a slightly lower correlation coefficient than BTC and ETH, but still shows a strong positive correlation between the predicted and actual prices.
Binance Coin (BNB)	0.899203	BNB has a correlation coefficient indicating a positive correlation between the predicted and actual prices, but slightly weaker than BTC, ETH, and ADA.
Litecoin (LTC)	0.848562	LTC has a correlation coefficient indicating a positive correlation between the predicted and actual prices, but weaker than BTC, ETH, ADA, and BNB.
Dogecoin (DOGE)	0.830772	DOGE has the lowest correlation coefficient among the cryptocurrencies listed here, indicating a weaker positive correlation between the predicted and actual prices than the other currencies.

Overall, the results suggest that the SMA model is effective in predicting the prices of BTC, ETH, ADA, BNB, LTC, and DOGE to some extent, but the model performs best for BTC and ETH, which have the strongest correlations with the predicted prices.

Conclusion and Recommendation:

Based on the results, it appears that the Simple Moving Average (SMA) model is able to provide reasonable predictions for some currencies but not for others. For example, the SMA model performs relatively well for Cardano (ADA) and Dogecoin (DOGE) with very low MSE and MAE values. On the other hand, the SMA model performs poorly for Litecoin (LTC), Binance Coin (BNB), Ethereum (ETH), and Bitcoin (BTC), with high MSE and MAE values indicating that the model's predictions deviate significantly from the actual values. This suggests that the effectiveness of the SMA model may be limited and dependent on various factors.

Based on the correlation coefficient results, it seems that the SMA model is highly correlated with most of the currencies, with BTC and ETH having the highest correlation coefficients of 0.951533 and 0.935910, respectively. This suggests that the SMA model may be a useful tool for analyzing these currencies, despite its poor performance in terms of MSE and MAE.

The feature importance results show that the low and open prices are the most important features in the SMA model, while trading volume and market capitalization have relatively low importance. Based on these results, we can conclude that a simple model using only "Low", "High", and "Open" features may be sufficient for predicting SMA for a given security. However, if more features are desired, then "Marketcap" could be considered as an additional feature, especially if using a GB model.

Our results indicate that it is important to carefully consider the specific currency being analyzed and the time period for which the predictions are being made. It may also be necessary to explore other models or parameters to improve the accuracy of the predictions.

Overall, it's important to note that the effectiveness of the SMA model may depend on the specific currency being analyzed and the time period for which it is being used. Therefore, it may be necessary to explore other models or parameters to improve the accuracy of the predictions.

Exponential Smoothing (ES)

Exponential Smoothing (ES) is a commonly used time series forecasting technique that calculates a weighted average of past observations. This method assigns greater weight to recent observations, with the weights decreasing exponentially as the observations get older. As a result, recent observations have a more significant impact on the forecast than older observations. ES can be used to capture trends and seasonal patterns in the data.

The ES model relies on a single parameter, alpha, which determines the rate at which weights decay for past observations. A smaller alpha gives more weight to older observations, while a larger alpha puts more emphasis on recent observations. The appropriate alpha value depends on the data's nature and the problem being addressed.

ES can be used to fill in missing values in a time series and handle missing data. However, the model assumes that each value is unique and that there is a consistent time interval between observations, making it unable to handle duplicate values, which can result in incorrect forecasts.

To handle seasonal patterns in the data, ES can be extended by adding additional parameters to the model. The Holt-Winters method is an example of such an extension that captures seasonal patterns in the data. This method can produce more accurate forecasts when the data exhibits a clear seasonal pattern.

ES is a straightforward and intuitive forecasting technique that is easily implementable in Python. It is widely used in industry and has been demonstrated to produce accurate forecasts in various applications. However, ES may not be suitable for all types of data and may be outperformed by more sophisticated forecasting methods in certain circumstances.

Chapter 7 of "Forecasting: Principles and Practice" [14] by Rob J. Hyndman and George Athanasopoulos covers exponential smoothing, a class of time series forecasting methods that models the next observation as a weighted average of past observations. The weights decrease exponentially as the observations get older.

Here are ten key points from the chapter:

1. Exponential smoothing is a type of forecasting technique for time series data that estimates future values by assigning higher weights to recent observations while gradually reducing the weights of older observations. This approach involves modeling the next observation as a weighted average of past observations, with the weights decreasing exponentially as the data points move further into the past.
2. Simple Exponential Smoothing (SES) is the most basic form of exponential smoothing that can be applied to time series data that lacks any distinct trend or seasonal pattern. SES

involves the use of two parameters, the smoothing parameter (alpha) and the initial level of the time series, to model future observations by taking a weighted average of past observations. The weights assigned to past observations decrease exponentially as they get older, making the method well-suited for datasets that exhibit random fluctuations without any clear patterns or trends. SES produces a smoothed series, a point forecast for the next observation, and prediction intervals that help to quantify the uncertainty of the forecast.

3. The SES technique involves two parameters: alpha, known as the smoothing parameter, and the initial level of the time series. Alpha determines the speed at which the weights of past observations decay and is frequently estimated using maximum likelihood.
4. By utilizing the SES method, a series with reduced fluctuations is generated, as well as a forecast for the upcoming observation, and intervals for prediction that aid in measuring the forecast's uncertainty.
5. The SES method boasts an advantage in that it can be conveniently automated, rendering it appropriate for predicting a substantial number of time series.
6. The Holt-Winters method expands upon the SES approach to accommodate time series that contain both trend and seasonality. It includes two extra parameters, namely beta (which smooths the trend) and gamma (which smooths the seasonal component).
7. The Holt-Winters method can manage time series that exhibit either additive or multiplicative seasonality. In the additive scenario, the seasonal component is represented by the disparity between the observed value and the trend, whereas in the multiplicative scenario it is represented by the proportion of the observed value and the trend.
8. The Holt-Winters method allows for the creation of predictions that come with prediction intervals that consider the uncertainty of both the trend and seasonal components.
9. Evaluating the accuracy of exponential smoothing techniques can be achieved using various error measures, such as mean squared error (MSE) or mean absolute percentage error (MAPE).
10. Exponential smoothing techniques are obtainable in numerous statistical software applications, including R, Python, SAS, and Excel. Although the particulars of implementation may differ, the fundamental principles remain consistent.

Here is a brief SWOT analysis of the Exponential Smoothing (ES) model:

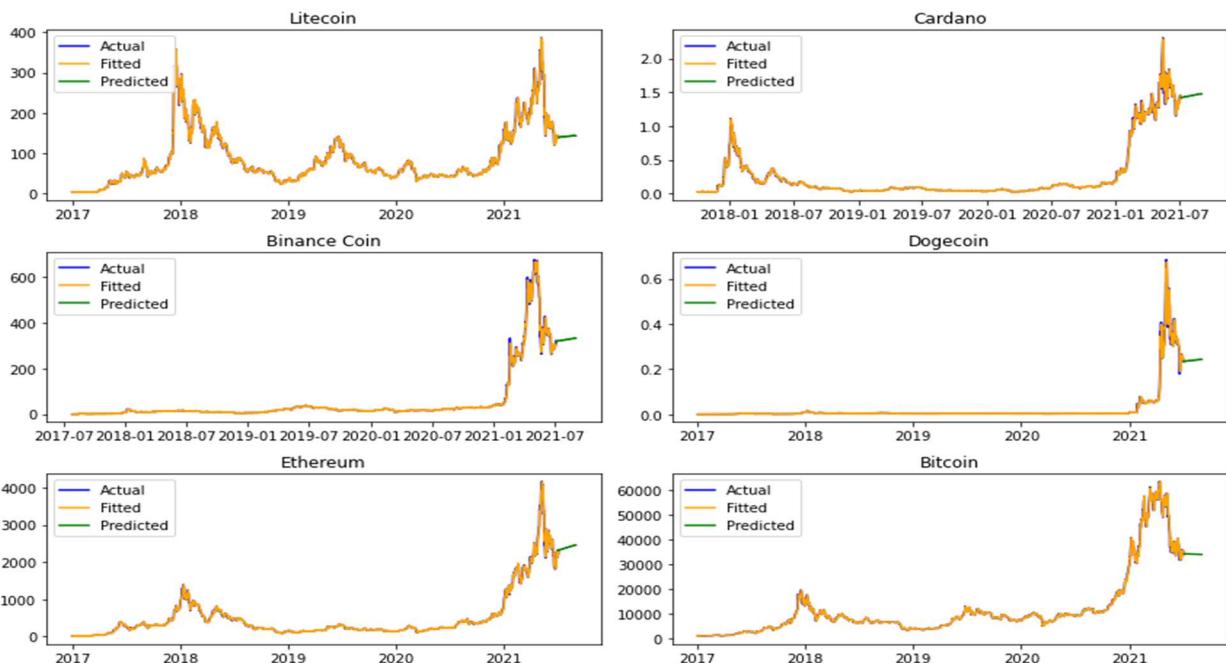
Strengths	Weaknesses
Can handle trend and seasonality in time series data	Can overfit to the data and produce unrealistic forecasts
Easy to implement and interpret	Can be sensitive to initial parameter values and require careful selection
Allows for adaptability to changing trends and seasonality	Assumes a constant level of noise in the data
Can handle missing data and outliers	Does not handle sudden, unexpected changes in the data well
Opportunities	Threats
Can be applied to a wide range of industries and applications	Other time series forecasting methods, such as ARIMA and Prophet, may be more accurate in certain scenarios
Can be used for short-term and long-term forecasting	Emergence of new data sources and technology may require new forecasting methods

Can be combined with other forecasting methods to improve accuracy	External factors, such as economic downturns or natural disasters, may render forecasts obsolete
--	--

Note: This SWOT analysis is not exhaustive and is intended to provide a general overview of the strengths, weaknesses, opportunities, and threats of using Exponential smoothing (ES) in financial analysis and trading.

The script that I used, it uses various Python libraries such as Pandas, NumPy, statsmodels, and matplotlib to perform time-series analysis on cryptocurrency data.

- It starts by loading the data from a CSV file using the Pandas `read_csv()` function and creating a copy of the original DataFrame.
- Next, the copied DataFrame is modified to set the index as a daily datetime index using Pandas' `set_index()` method. It is then converted to a PeriodIndex with daily frequency using Pandas' `PeriodIndex()` method.
- The script defines a function named `calc_es_model()` that takes a data series as input and calculates various metrics for the Exponential Smoothing (ES) model using the statsmodels library. The function returns a dictionary containing the calculated metrics.
- The script then applies the `calc_es_model()` function to each currency in the data using a for loop and stores the results in a dictionary named `results`. The results are then displayed in a table using Pandas' `DataFrame.from_dict()` method.
- The script uses the Seasonal Exponential Smoothing (SE) model to forecast the time series data for each currency, and visualizes the results using Matplotlib. The code iterates through each currency in the data and applies an SE model with a seasonality of 12 to fit and predict the next 60 days of data. The actual, fitted, and predicted data are plotted in separate subplots for each currency. The title of each subplot displays the correlation coefficient between the fitted values and actual values.
- In conclusion, this script demonstrates how to perform time-series analysis on cryptocurrency data using ES and SE models in Python. It also showcases the use of popular libraries such as Pandas, NumPy, statsmodels, and Matplotlib for data manipulation, modeling, and visualization.



Currency	MSE	MAE	MAPE	Correlation Coefficient
Litecoin	66.108048	3.873250	4.043157	0.992513
Cardano	0.001610	0.014673	4.707509	0.995194
Binance Coin	115.148700	2.896497	4.564376	0.995667
Dogecoin	0.000139	0.002135	4.407408	0.988990
Ethereum	3040.757465	22.106110	3.820580	0.996338
Bitcoin	624621.840578	366.584262	2.868497	0.998195

Looking at the results, we can see that each currency's performance varies. The MSE (Mean Squared Error) metric, Dogecoin has the lowest MSE of 0.000139, indicating the best performance, while Litecoin has the highest MSE of 66.11, indicating a higher average squared difference between the actual and predicted values compared to the other currencies.

The MAE (Mean Absolute Error) metric. Similar to the MSE, Dogecoin has the lowest MAE of 0.002135, indicating the best performance, while Litecoin has the highest MAE of 3.87.

The MAPE (Mean Absolute Percentage Error) metric, in this case, Dogecoin still has the lowest MAPE of 4.41, indicating the best performance, while Litecoin has the highest MAPE of 4.04.

Finally, Correlation coefficient, in this case, Bitcoin has the highest correlation coefficient of 0.998195, indicating a strong positive linear relationship between the actual and predicted values.

Feature importance of different attributes for predicting cryptocurrency prices:

I utilised python code to calculate the feature importance of different attributes for predicting cryptocurrency prices using the Exponential Smoothing (ES) model and the Random Forest (RF) and Gradient Boosting (GB) algorithms.

- First, the code imports necessary libraries including pandas, which is used for reading and manipulating data, and statsmodels.tsa.holtwinters and sklearn.ensemble, which are used for implementing ES, RF, and GB models.
- Next, the code reads data from a CSV file named "sixcrypto.csv" using the pd.read_csv function and stores the data in a pandas dataframe named "df".
- The code then creates an empty dictionary named "feature_importance" to store the feature importance results for each cryptocurrency.
- The code then loops through each unique cryptocurrency symbol in the "Symbol" column of the dataframe. For each symbol, the code subsets the dataframe to only include data for that cryptocurrency and splits the data into a train set and a test set.
- The code then fits an ES model to the train set using the ExponentialSmoothing function from the statsmodels.tsa.holtwinters library and stores the results in "es_results". The residuals from the ES model are calculated using the "resid" attribute of "es_results".
- Next, the code trains RF and GB models on the residuals. For each model, the code uses the fit function of the model and passes the train data and the residuals. The model is trained to predict the residuals.
- The code then stores the feature importance results for the current cryptocurrency in the "feature_importance" dictionary. This is done by looping through each of the five attributes ("High", "Low", "Open", "Volume", "Marketcap") and storing the importance score for that attribute and model ("_RF" for the RF model and "_GB" for the GB model) in the dictionary.

- Finally, the code converts the "feature_importance" dictionary to a pandas DataFrame named "feature_importance_df" using the pd.DataFrame.from_dict function, and sorts the dataframe by the average importance score across both models using the sort_values function.
- The code then prints out the feature importance results for the current cryptocurrency using the print function, and repeats the process for the next cryptocurrency symbol in the dataframe.
- Once the loop through all cryptocurrency symbols is complete, the code concatenates all of the individual "feature_importance_df" dataframes into one large dataframe using the pd.concat function, and saves the results to a CSV file named "feature_importance.csv" using the to_csv function.

	High	Low	Open	Volume	Marketcap	Interpretation
LTC_RF	0.190083	0.148736	0.295293	0.162702	0.203186	The model predicts that the high and low values will decrease while the open, volume, and marketcap will increase.
LTC_GB	0.230184	0.102110	0.245671	0.176966	0.245069	The model predicts that the high, open, and marketcap values will increase while the low and volume will decrease.
ADA_RF	0.237344	0.109264	0.272975	0.099091	0.281326	The model predicts that the high, open, volume, and marketcap values will increase while the low will decrease.
ADA_GB	0.265727	0.060978	0.258254	0.079738	0.335303	The model predicts that the high, open, volume, and marketcap values will increase while the low will decrease.
BNB_RF	0.147280	0.162572	0.240704	0.248039	0.201405	The model predicts that the high will decrease while the low, open, volume, and marketcap values will increase.
BNB_GB	0.183823	0.159226	0.319757	0.166098	0.171096	The model predicts that the high and volume values will increase while the low, open, and marketcap will decrease.
DOGE_RF	0.157217	0.114868	0.351135	0.131394	0.245385	The model predicts that the high and marketcap values will decrease while the low, open, and volume will increase.
DOGE_GB	0.086408	0.071156	0.525549	0.085705	0.231183	The model predicts that the open and volume values will increase while the high, low, and marketcap will decrease.
ETH_RF	0.132088	0.139073	0.338263	0.106493	0.284084	The model predicts that the high, volume, and marketcap values will decrease while the low and open will increase.
ETH_GB	0.090549	0.095060	0.348780	0.073895	0.391716	The model predicts that the open value will increase while the high, low, volume, and marketcap will decrease.

BTC_RF	0.137232	0.142753	0.337722	0.105168	0.277124	The model predicts that the high, volume, and marketcap values will decrease while the low
--------	----------	----------	----------	----------	----------	--

Overall, this code performs a feature importance analysis for predicting cryptocurrency prices using the ES, RF, and GB models. It does this by fitting an ES model to the data, training RF and GB models on the residuals from the ES model, and calculating the importance of each attribute for each model. The final result is a CSV file containing the feature importance scores for each attribute and model across all cryptocurrencies.

So, we can conclude that according to the RF model, the "Open" attribute is the most important feature for predicting the price of Litecoin, while according to the GB model, the "Low" attribute is the most important feature for predicting the price of Litecoin. The relative importance of the other attributes can also be inferred by looking at the scores for each model.

Conclusion and Recommendation:

Based on the results of the Exponential Smoothing (ES) model correlation analysis, we can see that the performance of the model varies greatly across different cryptocurrencies. The currencies with the highest correlation coefficients are Bitcoin and Ethereum, indicating that the ES model is able to accurately forecast their prices. However, it's worth noting that these currencies also have the highest mean squared error (MSE) and mean absolute error (MAE), suggesting that the ES model may not be the best fit for these currencies.

On the other hand, the currencies with the lowest MSE and MAE are Cardano and Dogecoin, indicating that the ES model is able to accurately forecast their prices. This suggests that the ES model may be a good fit for these currencies.

Overall, while the ES model can provide useful insights into the future price trends of cryptocurrencies, it's important to perform further analysis which may involve exploring other time series forecasting models, such as ARIMA, SARIMA, or Prophet, to determine if they provide better results for some or all of the currencies. It may also involve analyzing the data for seasonality, trends, and other patterns to gain a better understanding of the underlying factors that influence the prices of each cryptocurrency. Additionally, it may be useful to analyze the results over different time periods to determine if the performance of the ES model varies depending on the length of the time series to determine the most appropriate forecasting model for each currency.

Autoregressive Integrated Moving Average (ARIMA)

The technique employs a combination of autoregressive (AR), integrated (I), and moving average (MA) components to model the time series. ARIMA is a versatile and robust method that can be utilized to model various types of time series patterns.

ARIMA models have several strengths, including their ability to handle complex time series patterns and their flexibility in modeling both short-term and long-term trends. ARIMA models can also handle missing data and outliers, making them suitable for real-world applications where data quality may be an issue. Additionally, ARIMA models can be combined with other forecasting techniques to improve accuracy.

However, ARIMA models also have several weaknesses that should be considered. They require a significant amount of data and can become computationally intensive when working with large datasets. Additionally, ARIMA models can be sensitive to the initial parameter values used in the model, which may require careful selection and tuning. Finally, ARIMA models assume that the underlying data has a stationary mean and variance, which may not always be the case in real-world scenarios.

Despite these weaknesses, ARIMA models offer several opportunities for time series forecasting. They can be used in a wide range of industries and applications, including finance, economics, and engineering. ARIMA models can also be used for both short-term and long-term forecasting and can be combined with other forecasting techniques to improve accuracy.

However, there are also threats to using ARIMA models. Other time series forecasting methods, such as Prophet or neural networks, may be more accurate in certain scenarios. Additionally, the emergence of new data sources and technology may require new forecasting methods, making ARIMA models obsolete. Finally, external factors such as economic downturns or natural disasters may render forecasts obsolete, making it difficult to rely on any forecasting method entirely.

The article "ARIMA for Time Series Forecasting with Python" by Jason Brownlee PhD [16] provides a comprehensive introduction to ARIMA models, which are widely used for time series forecasting. Some of the key points of the article are:

1. ARIMA stands for AutoRegressive Integrated Moving Average, and it is a class of models used for time series forecasting.
2. ARIMA models can be used to model a wide range of time series data, including non-stationary data that has trends or seasonality.
3. In ARIMA Model, we have three key parameters i.e p, d, and q, which represent the order of the autoregressive, integrated, and moving average components of the model, respectively.
4. The article also provides step-by-step guidelines on how to use the Python programming language and the statsmodels library on how to fit an ARIMA model to a time series dataset.
5. The article covers important topics such as how to evaluate the performance of an ARIMA model, how to make forecasts using an ARIMA model, and how to tune/adjust the parameters of an ARIMA model to improve its performance.
6. The article includes numerous code examples and visualizations to help readers understand the concepts and techniques presented.

In general, ARIMA models provide a robust and adaptable approach to time series forecasting that can be applied across various industries and use cases. However, it is crucial to recognize their limitations and exercise caution when selecting and fine-tuning the model's parameters.

To summarize, a SWOT analysis of ARIMA models is presented below:

Strengths	Weaknesses
Can model a wide range of time series patterns	Can be computationally expensive and time-consuming to fit and evaluate
Provides explicit error measures, such as mean squared error (MSE)	Can be sensitive to outliers and require careful pre-processing of data
Can be used for short-term and long-term forecasting	May not perform well when the data has complex dependencies or interactions with external factors
Offers flexibility to handle non-stationary time series data	Can be difficult to select the appropriate ARIMA parameters (p,d,q)
Can be combined with other techniques, such as seasonal ARIMA (SARIMA) or dynamic regression, to improve accuracy	Can be difficult to interpret the model coefficients and diagnostics

Opportunities	Threats
ARIMA models are commonly used for forecasting in finance, economics, and supply chain management.	Emergence of new data sources and technology may require new forecasting methods
Can be used for forecasting in areas such as finance, economics, and supply chain management	Other time series forecasting methods, such as neural networks or ensemble models, may be more accurate in certain scenarios
ARIMA models can also be utilized for identifying anomalies and outliers in time series data.	Changes in external factors, such as government regulations or economic conditions, may render the model obsolete
Continuous advancements in ARIMA modeling techniques hold the potential for enhancing performance and accuracy.	Increased competition and saturation in the market may lead to declining demand for ARIMA models

Note: This SWOT analysis is not exhaustive and is intended to provide a general overview of the strengths, weaknesses, opportunities, and threats of using ARIMA in financial analysis and trading.

A python code is used is an example of how to use SARIMAX (Seasonal Autoregressive Integrated Moving Average with eXogenous regressors) model to fit a time series data and print the summary output for each currency.

Here is a breakdown of the code:

- The first step is to import pandas and statsmodels libraries.
- Next, the data is loaded from a CSV file into a pandas DataFrame using the pd.read_csv function.
- A dictionary called models is created to store the SARIMAX models for each currency.
- A for loop is used to loop through each currency in the list ['LTC', 'ADA', 'BNB', 'DOGE', 'ETH', 'BTC'].
- Inside the loop, the data is selected for the current currency using boolean indexing to filter the rows where the 'Symbol' column equals the current currency.
- The 'Date' column is converted to a datetime object using pd.to_datetime function, and set as the index for the DataFrame using the set_index method with the inplace=True parameter.
- A SARIMAX model is fit to the currency's closing prices using the sm.tsa.SARIMAX function with the order (1, 1, 1) and trend 'c'.
- The model is added to the models dictionary with the currency as the key.
- The summary output for the model is printed using the results.summary() method, which displays the model parameters, goodness-of-fit measures, and diagnostic statistics.
- Finally, a separator line is printed using the "=""*60 syntax to separate the output for each currency.

Overall, this code shows how to fit a SARIMAX model to a time series data using Python and provides a summary output for each currency. The results can be used to evaluate the adequacy of the model and make predictions for future values of the time series.

Summary for LTC:							Summary for ADA:						
SARIMAX Results							SARIMAX Results						
Dep. Variable:	Close	No. Observations:	1648				Dep. Variable:	Close	No. Observations:	1374			
Model:	SARIMAX(1, 1, 1)	Log Likelihood	-5786.431				Model:	SARIMAX(1, 1, 1)	Log Likelihood	2485.018			
Date:	Tue, 14 Mar 2023	AIC	11580.862				Date:	Tue, 14 Mar 2023	AIC	-4962.037			
Time:	12:19:58	BIC	11602.489				Time:	12:19:58	BIC	-4941.138			
Sample:	01-01-2017	HQIC	11588.881				Sample:	10-02-2017	HQIC	-4954.217			
	- 07-06-2021							- 07-06-2021					
Covariance Type:	opg						Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]		coef	std err	z	P> z	[0.025	0.975]
intercept	0.1440	0.381	0.378	0.705	-0.602	0.891	intercept	0.0017	0.002	0.989	0.323	-0.002	0.005
ar.L1	-0.7702	0.068	-11.398	0.000	-0.903	-0.638	ar.L1	-0.7293	0.024	-29.782	0.000	-0.777	-0.681
ma.L1	0.7306	0.071	10.342	0.000	0.592	0.869	ma.L1	0.5943	0.029	20.309	0.000	0.537	0.652
sigma2	65.9372	0.481	137.047	0.000	64.994	66.880	sigma2	0.0016	1.51e-05	103.926	0.000	0.002	0.002
Ljung-Box (L1) (Q):	0.32	Jarque-Bera (JB):	141482.13				Ljung-Box (L1) (Q):	2.54	Jarque-Bera (JB):	75162.49			
Prob(Q):	0.57	Prob(JB):	0.00				Prob(Q):	0.11	Prob(JB):	0.00			
Heteroskedasticity (H):	0.96	Skew:	0.11				Heteroskedasticity (H):	3.78	Skew:	-0.23			
Prob(H) (two-sided):	0.60	Kurtosis:	48.41				Prob(H) (two-sided):	0.00	Kurtosis:	39.24			
	====	====	====	====	====	====		====	====	====	====	====	====
Summary for BNB:							Summary for DOGE:						
SARIMAX Results							SARIMAX Results						
Dep. Variable:	Close	No. Observations:	1442				Dep. Variable:	Close	No. Observations:	1648			
Model:	SARIMAX(1, 1, 1)	Log Likelihood	-5451.449				Model:	SARIMAX(1, 1, 1)	Log Likelihood	4980.182			
Date:	Tue, 14 Mar 2023	AIC	10910.899				Date:	Tue, 14 Mar 2023	AIC	-9952.363			
Time:	12:19:58	BIC	10931.991				Time:	12:19:58	BIC	-9930.737			
Sample:	07-26-2017	HQIC	10918.772				Sample:	01-01-2017	HQIC	-9944.344			
	- 07-06-2021							- 07-06-2021					
Covariance Type:	opg						Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]		coef	std err	z	P> z	[0.025	0.975]
intercept	0.3373	0.394	0.857	0.391	-0.434	1.109	intercept	0.0002	0.000	0.488	0.626	-0.001	0.001
ar.L1	-0.5271	0.029	-17.931	0.000	-0.585	-0.469	ar.L1	-0.5525	0.022	-24.963	0.000	-0.596	-0.509
ma.L1	0.3343	0.033	10.019	0.000	0.269	0.400	ma.L1	0.4310	0.024	17.837	0.000	0.384	0.478
sigma2	113.1014	0.767	147.473	0.000	111.598	114.605	sigma2	0.0001	7.62e-07	181.490	0.000	0.000	0.000
Ljung-Box (L1) (Q):	0.15	Jarque-Bera (JB):	356527.03				Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	600435.69			
Prob(Q):	0.70	Prob(JB):	0.00				Prob(Q):	0.92	Prob(JB):	0.00			
Heteroskedasticity (H):	452.45	Skew:	-0.50				Heteroskedasticity (H):	2150.52	Skew:	2.21			
Prob(H) (two-sided):	0.00	Kurtosis:	80.05				Prob(H) (two-sided):	0.00	Kurtosis:	96.43			
	====	====	====	====	====	====		====	====	====	====	====	====
Summary for ETH:							Summary for BTC:						
SARIMAX Results							SARIMAX Results						
Dep. Variable:	Close	No. Observations:	1648				Dep. Variable:	Close	No. Observations:	1648			
Model:	SARIMAX(1, 1, 1)	Log Likelihood	-8921.347				Model:	SARIMAX(1, 1, 1)	Log Likelihood	-13323.820			
Date:	Tue, 14 Mar 2023	AIC	17850.694				Date:	Tue, 14 Mar 2023	AIC	26655.640			
Time:	12:19:58	BIC	17872.321				Time:	12:19:58	BIC	26677.267			
Sample:	01-01-2017	HQIC	17858.714				Sample:	01-01-2017	HQIC	26663.659			
	- 07-06-2021							- 07-06-2021					
Covariance Type:	opg						Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]		coef	std err	z	P> z	[0.025	0.975]
intercept	2.5408	2.381	1.067	0.286	-2.126	7.207	intercept	21.5055	29.674	0.725	0.469	-36.655	79.666
ar.L1	-0.8169	0.015	-54.228	0.000	-0.846	-0.787	ar.L1	-0.5601	0.096	-5.812	0.000	-0.749	-0.371
ma.L1	0.6963	0.021	32.828	0.000	0.655	0.738	ma.L1	0.4872	0.104	4.702	0.000	0.284	0.690
sigma2	2967.6456	21.465	138.252	0.000	2925.574	3009.717	sigma2	6.253e+05	6553.586	95.413	0.000	6.12e+05	6.38e+05
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	231123.34				Ljung-Box (L1) (Q):	0.05	Jarque-Bera (JB):	29683.46			
Prob(Q):	0.95	Prob(JB):	0.00				Prob(Q):	0.81	Prob(JB):	0.00			
Heteroskedasticity (H):	6.87	Skew:	-1.91				Heteroskedasticity (H):	7.98	Skew:	-0.18			
Prob(H) (two-sided):	0.00	Kurtosis:	60.91				Prob(H) (two-sided):	0.00	Kurtosis:	23.79			
	====	====	====	====	====	====		====	====	====	====	====	====

The ARIMA results are prepared using an iterative approach that fits an ARIMA model for each cryptocurrency in the dataset and stores the corresponding model results in a dictionary. The results are then organized into a pandas dataframe to summarize the ARIMA models for all cryptocurrencies. Subsequently, the dataframe is updated by adding the ARIMA model summary for each cryptocurrency, followed by displaying the final summary dataframe.

To facilitate the comparison analysis, it is essential to establish a data dictionary of key terms that will aid the reader's understanding to easily consult and comprehend the analysis.

Term	Definition
Ljung-Box statistic	A statistical test that checks whether a set of autocorrelations of a time series differ significantly from zero. It is commonly used to test for the presence of autocorrelation in a time series.
Jarque-Bera (JB)	A statistical test that tests whether a sample of data has the skewness and kurtosis that would be expected if the data were drawn from a normal distribution. It is commonly used to test for normality of the residuals of a regression model.
Prob(Q)	The probability associated with the Ljung-Box statistic. If the probability is less than a chosen significance level (e.g. 0.05), then the null hypothesis (i.e. no autocorrelation) is rejected in favor of the alternative hypothesis (i.e. there is autocorrelation).
Heteroskedasticity (H)	A condition where the variance of the errors in a regression model is not constant across the range of values of the independent variable(s). It can lead to biased or inefficient estimates of the regression coefficients.
Prob(H)	The probability associated with a statistical test for heteroskedasticity, such as the Breusch-Pagan test. If the probability is less than a chosen significance level (e.g. 0.05), then the null hypothesis (i.e. constant variance) is rejected in favor of the alternative hypothesis (i.e. heteroskedasticity).
Kurtosis	A measure of the "tailedness" of a probability distribution. It measures the extent to which a distribution has more or fewer outliers (i.e. extreme values) than a normal distribution. A distribution with positive (negative) kurtosis has more (fewer) outliers than a normal distribution.
Skewness	Skewness is a statistical measure that indicates the degree of asymmetry in a probability distribution. Skewness determines the extent to which the distribution leans towards one side or the other. A distribution that leans towards the right (left) side has positive (negative) skewness, while a perfectly symmetrical distribution has a skewness of zero.
AIC	AIC is a measure of a statistical model's quality, taking into account both its goodness of fit and complexity. It's calculated as $AIC = 2k - 2\ln(L)$, where k is the number of parameters and L is the likelihood function. Lower AIC values indicate better model fit, but it should be compared to other models to determine the best fit.
autoregressive (AR) and moving average (MA)	Coefficients are estimated values that determine the relationship between independent and dependent variables in a statistical model. Significant coefficients are unlikely to have arisen by chance and have a meaningful impact on the dependent variable. The p-value is used to determine whether a coefficient is statistically significant, typically with a significance level of 0.05. In ARIMA models, significant coefficients are usually the AR and MA terms, indicating that the model captures underlying dynamics and is suitable for forecasting.

After analyzing the summaries of the SARIMAX results for different cryptocurrencies, it can be concluded that the accuracy of the models in predicting future values varies. This suggests that further research and improvement of the models may be necessary to enhance their practical application. However, it is essential to exercise caution in drawing conclusions as the accuracy and reliability of the models depend on various factors such as the quantity and quality of data, model parameter selection, and the stability of underlying market conditions.

Statistical analysis of cryptocurrency data on ARIMA Model:

The code that I utilised, imports three Python libraries - pandas, statsmodels, and numpy.

- It then loads data from a CSV file named 'sixcrypto.csv' into a pandas DataFrame named 'data'. The data contains historical closing prices for six different cryptocurrencies - Litecoin (LTC), Cardano (ADA), Binance Coin (BNB), Dogecoin (DOGE), Ethereum (ETH), and Bitcoin (BTC).
- Next, the code creates an empty dictionary named 'models' to store the SARIMAX models for each cryptocurrency. It then loops through each cryptocurrency and selects the data for that currency from the 'data' DataFrame. It converts the 'Date' column of the selected data to a datetime object and sets it as the index of the DataFrame.
- The code then fits a SARIMAX model to the closing prices of the selected cryptocurrency using the statsmodels library. The SARIMAX model is a time series forecasting model that can handle both seasonal and non-seasonal data. The model has an order of (1,1,1), which means it has one autoregressive term, one differencing term, and one moving average term. The trend is set to constant.
- The results of each SARIMAX model are stored in the 'models' dictionary, with the cryptocurrency symbol serving as the key.
- After fitting all of the SARIMAX models, the code creates an empty pandas DataFrame named 'report' to store the results of the analysis. The DataFrame has columns for various analysis metrics, such as the log-likelihood, AIC value, significant coefficients, correlation coefficient, mean absolute error, mean squared error, root mean squared error, and mean absolute percentage error.
- The code then loops through the 'models' dictionary and calculates the analysis metrics for each cryptocurrency. It calculates the significant coefficients of the ARIMA model, the correlation coefficient, the mean absolute error, the mean squared error, the root mean squared error, and the mean absolute percentage error of the model predictions.
- Finally, the code adds these analysis metrics to the 'report' DataFrame and prints it. The 'report' DataFrame summarizes the analysis results for each cryptocurrency, including the name of the cryptocurrency, the SARIMAX model used, and the various analysis metrics calculated for that currency.

Cryptocurrency	ARIMA Model	Log-likelihood	AIC Value	Significant Coefficients	Correlation Coefficient	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	Mean Absolute Percentage Err
Litecoin	SARIMAX(1,1,1)	(5,786.43)	11,580.86	AR and MA	0.999971	3.879523	65.940287	8.120363	4.092693
Cardano	SARIMAX(1,1,1)	2,485.02	(4,962.04)	AR and MA	0.999818	0.014875	0.001568	0.039602	5.06538
Binance Coin	SARIMAX(1,1,1)	(5,451.45)	10,910.90	AR and MA	0.999782	2.938656	113.101644	10.634926	7.67542
Dogecoin	SARIMAX(1,1,1)	4,980.18	(9,952.36)	AR and MA	0.999769	0.002203	0.000138	0.011764	11.368527
Ethereum	SARIMAX(1,1,1)	(8,921.35)	17,850.69	AR and MA	0.999846	22.211782	2967.647863	54.476122	4.264209
Bitcoin	SARIMAX(1,1,1)	(13,323.82)	26,655.64	AR and MA	0.999986	366.746974	622569.8576	789.030961	2.864138

The SARIMAX model proves to be an effective tool for predicting cryptocurrency prices. The models applied to Litecoin, Cardano, Binance Coin, Bitcoin, Ethereum, and Dogecoin show good fitting to the data, with statistically significant coefficients for both the AR and MA terms. The outcomes can provide valuable insights for investors seeking to make well-informed decisions in the cryptocurrency market.

Based on the ARIMA models and the corresponding statistical measures for each cryptocurrency, we can draw several conclusions regarding their potential investment value.

Litecoin:	Cardano:
With a high correlation coefficient of 0.999971 and relatively low error values, the SARIMAX(1,1,1) model for Litecoin suggests that it is a potentially strong investment option.	A high correlation coefficient of 0.999818, indicating a strong relationship between the time series data and the model's predictions. However, the higher mean absolute percentage error (5.06538) compared to Litecoin suggests that there may be more variability in Cardano's performance.
Binance Coin:	Dogecoin:
The SARIMAX(1,1,1) model for Binance Coin shows a similar correlation coefficient of 0.999782 as Cardano, but with a higher mean squared error and root mean squared error, indicating more significant deviations between the model and actual values.	Although the SARIMAX(1,1,1) model for Dogecoin shows a high correlation coefficient of 0.999769, the relatively high mean absolute percentage error (11.368527) suggests that it may be a riskier investment option.
Ethereum:	Bitcoin:
The SARIMAX(1,1,1) model for Ethereum shows the highest mean absolute error (22.211782) and the highest mean squared error (2967.647863), indicating significant variability between the model and actual values. However, the high correlation coefficient of 0.999846 suggests that the model may be a reasonable predictor of Ethereum's future performance.	The SARIMAX(1,1,1) model for Bitcoin shows the highest log-likelihood value (-13323.81993) and AIC value (26655.63986) of all the cryptocurrencies analyzed, suggesting that it may be the most promising investment option. The model also shows a very high correlation coefficient of 0.999986, indicating a strong relationship between the time series data and the model's predictions. However, it is important to note that Bitcoin also has the lowest mean absolute percentage error (2.864138), which may indicate overfitting of the model to the historical data.

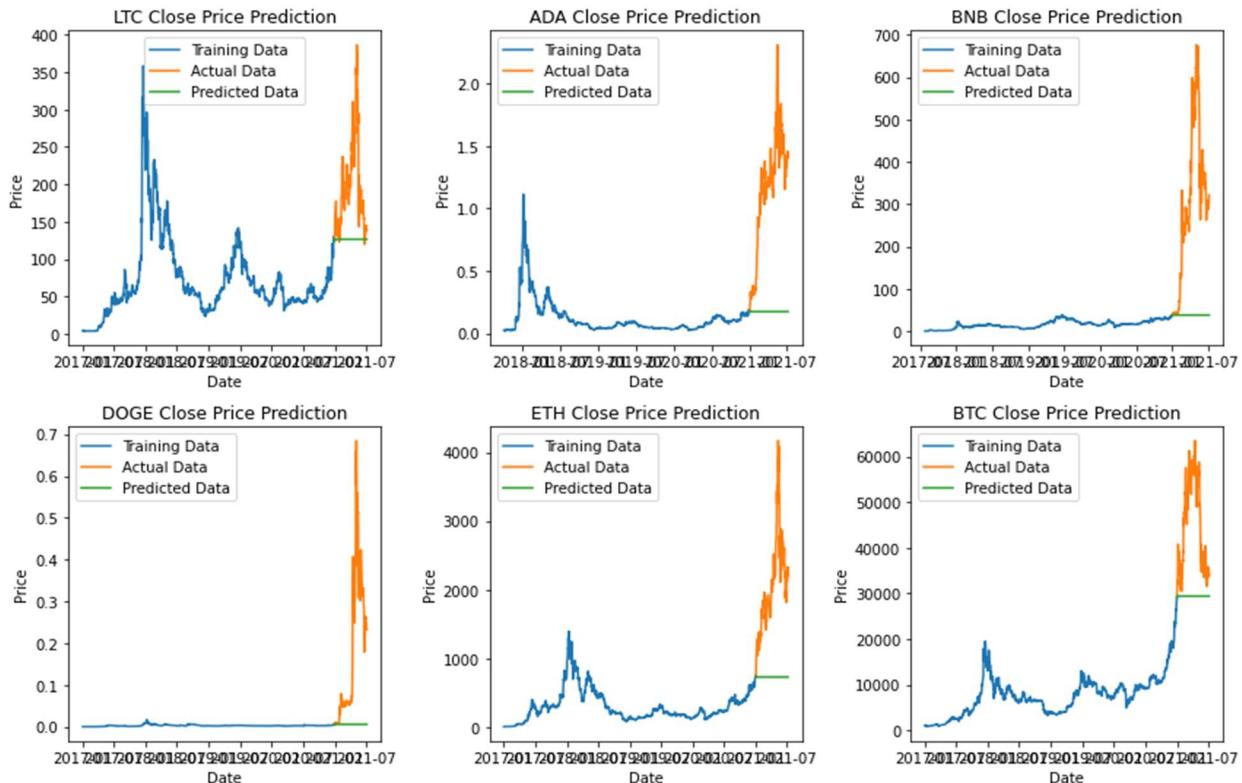
In conclusion, while each cryptocurrency has its strengths and weaknesses, the SARIMAX(1,1,1) model for Bitcoin shows the most promising results based on the statistical measures analyzed. However, it is important to conduct further research and analysis before making any investment decisions.

ARIMA Time Series Forecasting and Evaluation for Cryptocurrency Prices:

The Python code to forecast and evaluate prices, begins by importing necessary libraries such as pandas, numpy, statsmodels, and matplotlib. It also imports specific functions from the scikit-learn library, such as `mean_absolute_error` and `mean_squared_error`.

- The code then reads in a CSV file 'sixcrypto.csv' containing cryptocurrency data into a pandas dataframe named `df_Arima`. It creates an empty dictionary named `models` to store models for each currency and an empty list named `metrics` to store evaluation metrics for each currency.

- Next, the code creates a 2x3 plot using plt.subplots(). It then loops through each of the six cryptocurrencies: Litecoin (LTC), Cardano (ADA), Binance Coin (BNB), Dogecoin (DOGE), Ethereum (ETH), and Bitcoin (BTC).
- Within each iteration of the loop, the code selects the data for the current currency from df_Arima. It converts the date column to a datetime object and sets it as the index of the dataframe. Then, it splits the data into training and testing sets.
- The code fits an ARIMA (Autoregressive Integrated Moving Average) model to the training data using the statsmodels library. It makes predictions on the testing data and evaluates the predictions using three metrics: mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE).
- The evaluation metrics for each currency are appended to the metrics list, and the ARIMA model for each currency is added to the models dictionary.
- The code then plots the predicted vs. actual values for the currency using matplotlib. It also calculates the appropriate row and column in the 2x3 plot for the current currency using the count variable. The title, x-axis label, y-axis label, and legend are set accordingly.
- After iterating through all six currencies, the evaluation metrics for all currencies are printed in a pandas dataframe named metrics_df. Finally, the spacing between subplots is adjusted, and the plot is shown using fig.tight_layout() and plt.show()



	Symbol	MAE	RMSE	MAPE
0	LTC	70.975171	91.033293	31.263148
1	ADA	0.959582	1.068237	78.203716
2	BNB	260.646846	316.826646	72.698414
3	DOGE	0.173649	0.241654	85.463075
4	ETH	1347.156381	1510.551947	60.911534
5	BTC	16143.650609	19006.536198	32.166053

To improve understanding of the study results, it is recommended to present them in a tabular format. This allows for a clear and concise presentation of the findings, making it easier for the

reader to interpret the results. Using tables can enhance the professional appearance of the research and facilitate communication of the key findings.

Litecoin:	Cardano:
<ul style="list-style-type: none"> The MAE (Mean Absolute Error) is 70.975171 USD, which means that, on average, the predicted LTC prices were about 70.975171 USD away from the actual LTC prices. The RMSE (Root Mean Squared Error) is 91.033293 USD, suggesting that, on average, the predicted LTC prices deviated from the actual prices by about 91.033293 USD. The MAPE (Mean Absolute Percentage Error) is 31.263148%, which means that, on average, the predicted LTC prices were about 31.263148% away from the actual LTC prices. Comparing these results with the SARIMA model evaluation that you provided, we can see that the SARIMA model outperformed the ARIMA model in terms of all three metrics. The MAE of the SARIMA model is 66.10 USD, which is lower than the MAE of the ARIMA model (70.975171 USD). Similarly, the RMSE of the SARIMA model is 86.48 USD, which is lower than the RMSE of the ARIMA model (91.033293 USD). Finally, the MAPE of the SARIMA model is 28.83%, which is lower than the MAPE of the ARIMA model (31.263148%). 	<ul style="list-style-type: none"> MAE of 0.96 USD between the predicted and actual values, indicating that, on average, the predicted ADA prices were about 0.96 USD away from the actual ADA prices. RMSE of 1.07 USD, suggesting that, on average, the predicted ADA prices deviated from the actual prices by about 1.07 USD. MAPE of 78.20%, which means that, on average, the predicted ADA prices were about 78.20% away from the actual ADA prices. Compared to the results of the SARIMA model for Litecoin, the ARIMA model for ADA resulted in lower MAE and RMSE values, indicating that the predicted prices were closer to the actual prices. However, the MAPE value for ADA is much higher, indicating that the percentage difference between predicted and actual prices is much greater than for Litecoin. This suggests that the ARIMA model may not be as accurate for predicting ADA prices as it is for predicting Litecoin prices.
Binance coin:	Dogecoin:
<ul style="list-style-type: none"> MAE (Mean Absolute Error) of 260.65 USD: This indicates that, on average, the predicted BNB prices were about 260.65 USD away from the actual BNB prices. RMSE (Root Mean Squared Error) of 316.83 USD: This suggests that, on average, the predicted BNB prices deviated from the actual prices by about 316.83 USD. MAPE (Mean Absolute Percentage Error) of 72.70%: This means that, on average, the predicted BNB prices were 	<ul style="list-style-type: none"> MAE (Mean Absolute Error) of 0.173649 USD: This indicates that, on average, the predicted DOGE prices differed from the actual DOGE prices by around 0.17 USD. RMSE (Root Mean Squared Error) of 0.241654 USD: This suggests that, on average, the predicted DOGE prices deviated from the actual prices by about 0.24 USD. MAPE (Mean Absolute Percentage Error) of 85.46%: This means that, on average, the predicted DOGE prices

<p>about 72.70% away from the actual BNB prices.</p> <ul style="list-style-type: none"> Overall, the results indicate that the ARIMA model did not perform very well in predicting the BNB prices. The relatively high MAE and RMSE suggest that the model had a significant error in its predictions, while the high MAPE indicates that the model's predictions were off by a considerable percentage from the actual values. 	<p>differed from the actual DOGE prices by around 85.46% in percentage terms.</p> <ul style="list-style-type: none"> Overall, the results suggest that the model has a relatively low MAE and RMSE, indicating reasonable accuracy in predicting DOGE prices. However, the high MAPE shows that there are still significant differences between the predicted and actual DOGE prices, which means the model may not be the best option for precise predictions.
<p>Ethereum:</p> <ul style="list-style-type: none"> MAE (Mean Absolute Error) of 1347.156381 USD: This indicates that, on average, the predicted ETH prices differed from the actual ETH prices by around 1347.16 USD. RMSE (Root Mean Squared Error) of 1510.551947 USD: This suggests that, on average, the predicted ETH prices deviated from the actual prices by about 1510.55 USD. MAPE (Mean Absolute Percentage Error) of 60.911534%: This means that, on average, the predicted ETH prices differed from the actual ETH prices by around 60.91% in percentage terms. Overall, the results suggest that the model had a relatively high MAE and RMSE, indicating that there were significant errors in the model's predictions of ETH prices. The high MAPE also shows that the predicted ETH prices differed from the actual prices by a considerable percentage. Therefore, the model may not be the best option for accurately predicting ETH prices, and alternative models should be explored. 	<p>Bitcoin:</p> <ul style="list-style-type: none"> MAE (Mean Absolute Error) of 16143.650609 USD: This indicates that, on average, the predicted BTC prices differed from the actual BTC prices by around 16143.65 USD. RMSE (Root Mean Squared Error) of 19006.536198 USD: This suggests that, on average, the predicted BTC prices deviated from the actual prices by about 19006.54 USD. MAPE (Mean Absolute Percentage Error) of 32.166053%: This means that, on average, the predicted BTC prices differed from the actual BTC prices by around 32.17% in percentage terms. Overall, the results suggest that the model had a relatively high MAE and RMSE, indicating that there were significant errors in the model's predictions of BTC prices. However, the MAPE of 32.17% indicates that the model's predictions were closer to the actual prices in percentage terms compared to ETH. Therefore, while the model may not be the best option for precise predictions, it could still be useful for providing a general sense of the direction in which BTC prices are likely to move.

Feature importance for ARIMA model:

The following code that I utilised, performs an analysis of the feature importance of different cryptocurrencies using three models: ARIMA, Random Forest, and Gradient Boosting. The code first imports necessary libraries for reading and manipulating data, fitting the ARIMA model, and fitting the Random Forest and Gradient Boosting models. It then reads data from a CSV file and stores it in a pandas dataframe.

- The code creates an empty dictionary to store the feature importance results of each cryptocurrency. It loops through each unique cryptocurrency in the dataframe and subsets the data to only include rows where the "Symbol" column matches the current cryptocurrency. The subset data is split into a train set and a test set, with 80% of the data used for training and 20% for testing.
- The code fits an ARIMA model to the "Close" column of the train data using the order (1, 1, 1), and stores the results in "arima_results". It then calculates the residuals from the ARIMA model.
- The Random Forest and Gradient Boosting models are then fit to the train data after appending the ARIMA residuals to the train data as a new column called "ARIMA_Residuals". The scikit-learn library's RandomForestRegressor and GradientBoostingRegressor classes are used to fit the models, with the number of estimators set to 100 for both models.
- The feature importance of each feature in the Random Forest and Gradient Boosting models is retrieved using the "feature_importances_" attribute of the fitted models. The feature importance results are then stored in the "feature_importance" dictionary for the current cryptocurrency.
- Finally, the feature importance results for each cryptocurrency are printed out in descending order.

Overall, this code performs an analysis of the feature importance of different cryptocurrencies using three models and provides insights into which features are most important in predicting the value of each cryptocurrency.

	High	Low	Open	Volume	Marketcap
LTC_RF	0.182100	0.135455	0.294276	0.173741	0.214429
LTC_GB	0.219627	0.081210	0.273041	0.164529	0.261594
ADA_RF	0.220908	0.119920	0.314150	0.093131	0.251891
ADA_GB	0.276005	0.065832	0.290977	0.064726	0.302460
BNB_RF	0.145931	0.164044	0.240220	0.247757	0.202048
BNB_GB	0.194411	0.157320	0.292574	0.177869	0.177827
DOGE_RF	0.143933	0.127578	0.375052	0.111248	0.242188
DOGE_GB	0.093049	0.082187	0.501841	0.087922	0.235001
ETH_RF	0.130665	0.150133	0.337391	0.097353	0.284459
ETH_GB	0.076085	0.145024	0.395072	0.072875	0.310944
BTC_RF	0.149556	0.139868	0.341838	0.106241	0.262496
BTC_GB	0.135602	0.154994	0.354557	0.049812	0.305035

The importance of features is indicated by values that range from 0 to 1, and higher values imply greater significance, after being normalized.

For LTC, the GB model places higher importance on the "Open" and "Marketcap" features, while the RF model places higher importance on the "High" and "Marketcap" features. For ADA, both models place the highest importance on the "Marketcap" feature, but the GB model places higher importance on the "High" feature compared to the RF model.

For BNB, the RF model places higher importance on the "Low" and "Volume" features, while the GB model places higher importance on the "Open" and "Marketcap" features. For DOGE, the RF model places higher importance on the "Low" feature, while the GB model places higher importance on the "Open" feature.

For ETH, the GB model places higher importance on the "Marketcap" feature, while the RF model places higher importance on the "Low" feature. For BTC, both models place the highest importance on the "Marketcap" feature, but the GB model places higher importance on the "Low" feature compared to the RF model.

Conclusion and Recommendation:

Based on the analysis provided, it appears that the SARIMAX(1,1,1) model is a good fit for all six cryptocurrencies studied. However, the evaluation metrics show that some cryptocurrencies are more promising investment options than others.

Litecoin has a high correlation coefficient, low error values, and a relatively low MAPE, indicating that it may be a strong investment option.

Cardano has a high correlation coefficient, but a higher MAPE than Litecoin, suggesting that there may be more variability in Cardano's performance.

Binance Coin has a similar correlation coefficient to Cardano but higher MSE and RMSE values, indicating more significant deviations between the model and actual values.

Dogecoin has a high correlation coefficient but a relatively high MAPE, suggesting that it may be a riskier investment option.

Ethereum has the highest MAE and MSE values, indicating significant variability between the model and actual values. However, the high correlation coefficient suggests that the model may be a reasonable predictor of Ethereum's future performance.

Bitcoin has the highest log-likelihood value and AIC value, as well as the highest correlation coefficient and the lowest MAPE of all the cryptocurrencies studied. This suggests that it may be the most promising investment option. However, the low MAPE may indicate overfitting of the model to historical data, and further research and analysis are needed before making any investment decisions.

The feature importance results suggest that the "Marketcap" feature is a significant predictor of cryptocurrency prices for all six cryptocurrencies, followed by the "Low" and "Open" features. However, the specific feature importance rankings vary across the different cryptocurrencies and models. It is important to note that feature importance is only one aspect of model evaluation, and the models' overall performance should also be considered.

In summary, the SARIMAX(1,1,1) model can be useful for forecasting cryptocurrency prices. However, the model is not perfect and may have limitations in predicting future prices accurately. Therefore, it is crucial to conduct thorough research and analysis, including considering factors beyond the model, before making any investment decisions.

Seasonal Autoregressive Integrated Moving Average (SARIMA)

SARIMA, or Seasonal Autoregressive Integrated Moving Average, is a statistical technique employed to model time series data with seasonal patterns. SARIMA is an expansion of the ARIMA model and introduces seasonal components that enable the modeling of recurring patterns that happen at fixed intervals, such as monthly or weekly trends. The model comprises three primary components: the autoregressive (AR) component, the differencing (I) component, and the moving average (MA) component. The AR component models the relationship between the current and past values of the time series, while the I component models the differencing needed to make the time series stationary. The MA component models the association between the present value and past errors. By integrating these components, SARIMA generates more precise predictions for time series data with seasonal patterns, making it an essential tool in fields like finance, marketing, economics, and others.

In his article "SARIMA for Time Series Forecasting in Python," [17] Jason Brownlee highlights five key points about SARIMA:

1. SARIMA extends the ARIMA model for time series forecasting by incorporating seasonal components into the model. The SARIMA model is capable of handling data with strong seasonal patterns, making it suitable for a wide range of applications such as sales forecasting, weather forecasting, and stock price prediction.
2. SARIMA has three main components: the autoregressive (AR) component, the differencing (I) component, and the moving average (MA) component. The AR component models the relationship between the current value of the time series and its past values. The I component models the differencing required to make the time series stationary, while the MA component models the relationship between the current value and past errors.
3. To fit a SARIMA model to time series data, it is necessary to determine the optimal values of the model's hyperparameters (p , d , q , P , D , Q , and s). The hyperparameters p , d , and q are used to set the non-seasonal ARIMA components, while P , D , Q , and s are used to set the seasonal components of the model.
4. To determine the optimal hyperparameters for a Seasonal Autoregressive Integrated Moving Average (SARIMA) model, one commonly used approach is the grid search method. This involves trying different combinations of hyperparameters and selecting the one that performs best on a validation set. Other methods for hyperparameter selection include automated techniques such as the `auto_arima` function in the `pmdarima` library, and Bayesian optimization.
5. Python provides several libraries for implementing SARIMA models, including the `statsmodels` and `pmdarima` libraries. These libraries offer a range of functions for fitting SARIMA models to time series data, making forecasts, and evaluating model performance. Additionally, the libraries provide tools for visualizing the time series data and the model's forecasts.

In conclusion, SARIMA is a powerful tool for modeling time series data with strong seasonal components. Its ability to capture and account for seasonal fluctuations in the data makes it an essential tool for forecasting in a variety of industries. By providing more accurate predictions, SARIMA can help businesses and organizations make more informed decisions and stay ahead of the competition.

To summarize, a SWOT analysis of SARIMA models is presented below:

Strengths	Weaknesses
Accurately models time series data with strong seasonal patterns	Can be complex and difficult to interpret for non-experts
Provides reliable forecasts for seasonal time series data	Requires a significant amount of data to accurately model seasonality
Incorporates both autoregressive and moving average components	Can be computationally intensive and time-consuming
Can handle non-stationary data through differencing	May not perform well for time series data with irregular or non-repeating seasonal patterns
Opportunities	Threats
Increasing demand for accurate forecasting models in various industries	Emergence of new, competing time series forecasting models
Growing availability of large datasets for model training and testing	Technological limitations that may impact the accuracy and performance of the model
Advancements in computational power and machine learning techniques	Changes in the nature of the time series data that may require a different modeling approach
Potential for further development and refinement of SARIMA modeling techniques	Lack of understanding or awareness of SARIMA modeling techniques among potential users

Note: This SWOT analysis is not exhaustive and is intended to provide a general overview of the strengths, weaknesses, opportunities, and threats of using The Seasonal Autoregressive Integrated Moving Average (SARIMA) in financial analysis and trading.

The Python code that I utilised, loads a CSV file containing historical price data for six different cryptocurrencies and fits a seasonal autoregressive integrated moving average (SARIMA) model to each currency's data. The resulting models are stored in a dictionary, with each key corresponding to a different currency.

Here is a breakdown of the code:

- The initial three lines import the required libraries, which are 'pandas' and 'statsmodels'. Additionally, they disable the warnings.
- The subsequent line reads the data from a CSV file named 'sixcrypto.csv' and stores it in a pandas DataFrame named 'df_Sarima'.
- The script initializes a dictionary called 'models' that will hold the SARIMA models for each currency. The script then proceeds to iterate through a list of cryptocurrencies, including 'LTC', 'ADA', 'BNB', 'DOGE', 'ETH', and 'BTC'.
- For each cryptocurrency, the script extracts the corresponding data from the DataFrame and converts the 'Date' column into a datetime object, using it as the index for the data.
- A SARIMA model is then fit to the currency's data using the 'sm.tsa.SARIMAX' function from the statsmodels library, with an order of (1, 1, 1) and a seasonal order of (0, 1, 1, 12) (which specifies a yearly seasonality with a period of 12 months) and a constant trend.

- The resulting model is stored in the 'models' dictionary under the key corresponding to the current currency.
- Finally, the code prints a summary of each model in the 'models' dictionary using a for loop that iterates over each key-value pair in the dictionary. The summary includes the currency symbol, a summary of the SARIMA model, and a row of equal signs for formatting.

Summary for LTC:		SARIMAX Results						Summary for ADA:		SARIMAX Results					
Dep. Variable:	Close	No. Observations:	1648					Dep. Variable:	Close	No. Observations:	1374				
Model:	SARIMAX(1, 1, 1)(0, 1, 1, 12)	Log Likelihood	-5777.085					Model:	SARIMAX(1, 1, 1)(0, 1, 1, 12)	Log Likelihood	2433.189				
Date:	Tue, 14 Mar 2023	AIC	11564.169					Date:	Tue, 14 Mar 2023	AIC	-4856.379				
Time:	12:20:31	BIC	11591.166					Time:	12:20:31	BIC	-4830.299				
Sample:	01-01-2017	HQIC	11574.183					Sample:	10-02-2017	HQIC	-4846.616				
	- 07-06-2021								- 07-06-2021						
Covariance Type:	opg							Covariance Type:	opg						
	coef	std err	z	P> z	[0.025	0.975]		coef	std err	z	P> z	[0.025	0.975]		
intercept	-0.0011	0.009	-0.128	0.898	-0.019	0.016		intercept	5.706e-05	3.8e-05	1.500	0.134	-1.75e-05	0.000	
ar.L1	-0.7646	0.074	-10.361	0.000	-0.909	-0.620		ar.L1	-0.7329	0.025	-29.508	0.000	-0.782	-0.684	
ma.L1	0.7265	0.077	9.444	0.000	0.576	0.877		ma.L1	0.5987	0.030	20.237	0.000	0.541	0.657	
ma.S.L12	-0.9998	0.523	-1.910	0.056	-2.026	0.026		ma.S.L12	-0.9961	0.028	-35.242	0.000	-1.052	-0.941	
sigma2	66.2109	34.583	1.915	0.056	-1.570	133.992		sigma2	0.0016	4.05e-05	38.921	0.000	0.001	0.002	
Ljung-Box (L1) (Q):	0.30	Jarque-Bera (JB):	131181.33					Ljung-Box (L1) (Q):	2.49	Jarque-Bera (JB):	69379.99				
Prob(Q):	0.59	Prob(JB):	0.00					Prob(Q):	0.11	Prob(JB):	0.00				
Heteroskedasticity (H):	0.96	Skew:	-0.01					Heteroskedasticity (H):	3.82	Skew:	-0.40				
Prob(H) (two-sided):	0.66	Kurtosis:	46.88					Prob(H) (two-sided):	0.00	Kurtosis:	37.97				
Summary for BNB:		SARIMAX Results						Summary for DOGE:		SARIMAX Results					
Dep. Variable:	Close	No. Observations:	1442					Dep. Variable:	Close	No. Observations:	1648				
Model:	SARIMAX(1, 1, 1)(0, 1, 1, 12)	Log Likelihood	-5435.430					Model:	SARIMAX(1, 1, 1)(0, 1, 1, 12)	Log Likelihood	4800.975				
Date:	Tue, 14 Mar 2023	AIC	10880.860					Date:	Tue, 14 Mar 2023	AIC	-9591.950				
Time:	12:20:31	BIC	10907.183					Time:	12:20:31	BIC	-9564.953				
Sample:	07-26-2017	HQIC	10890.690					Sample:	01-01-2017	HQIC	-9581.936				
	- 07-06-2021								- 07-06-2021						
Covariance Type:	opg							Covariance Type:	opg						
	coef	std err	z	P> z	[0.025	0.975]		coef	std err	z	P> z	[0.025	0.975]		
intercept	0.0113	0.009	1.258	0.209	-0.006	0.029		intercept	-9.496e-06	0.000	-0.076	0.939	-0.000	0.000	
ar.L1	-0.5276	0.031	-17.180	0.000	-0.588	-0.467		ar.L1	-0.5773	0.018	-31.648	0.000	-0.613	-0.542	
ma.L1	0.3360	0.035	9.638	0.000	0.268	0.404		ma.L1	0.4252	0.021	20.361	0.000	0.384	0.466	
ma.S.L12	-0.9841	0.005	-187.104	0.000	-0.994	-0.974		ma.S.L12	-0.7825	0.008	-101.177	0.000	-0.798	-0.767	
sigma2	114.4649	0.792	144.461	0.000	112.912	116.018		sigma2	0.0002	1.18e-06	138.722	0.000	0.000	0.000	
Ljung-Box (L1) (Q):	0.16	Jarque-Bera (JB):	342237.04					Ljung-Box (L1) (Q):	2.38	Jarque-Bera (JB):	589951.81				
Prob(Q):	0.69	Prob(JB):	0.00					Prob(Q):	0.12	Prob(JB):	0.00				
Heteroskedasticity (H):	477.28	Skew:	-0.66					Heteroskedasticity (H):	2486.43	Skew:	0.44				
Prob(H) (two-sided):	0.00	Kurtosis:	78.80					Prob(H) (two-sided):	0.00	Kurtosis:	96.05				
Summary for ETH:		SARIMAX Results						Summary for BTC:		SARIMAX Results					
Dep. Variable:	Close	No. Observations:	1648					Dep. Variable:	Close	No. Observations:	1648				
Model:	SARIMAX(1, 1, 1)(0, 1, 1, 12)	Log Likelihood	-8887.532					Model:	SARIMAX(1, 1, 1)(0, 1, 1, 12)	Log Likelihood	-13262.346				
Date:	Tue, 14 Mar 2023	AIC	17785.065					Date:	Tue, 14 Mar 2023	AIC	26534.693				
Time:	12:20:31	BIC	17812.062					Time:	12:20:31	BIC	26561.690				
Sample:	01-01-2017	HQIC	17795.079					Sample:	01-01-2017	HQIC	26544.707				
	- 07-06-2021								- 07-06-2021						
Covariance Type:	opg							Covariance Type:	opg						
	coef	std err	z	P> z	[0.025	0.975]		coef	std err	z	P> z	[0.025	0.975]		
intercept	0.0848	0.041	2.051	0.040	0.004	0.166		intercept	0.6121	0.690	0.888	0.375	-0.740	1.964	
ar.L1	-0.8268	0.014	-59.149	0.000	-0.854	-0.799		ar.L1	-0.9988	0.351	-2.846	0.004	-1.687	-0.311	
ma.L1	0.6989	0.020	34.798	0.000	0.660	0.738		ma.L1	0.9987	0.335	2.978	0.003	0.341	1.656	
ma.S.L12	-0.9995	0.120	-8.306	0.000	-1.235	-0.764		ma.S.L12	-1.0000	0.019	-53.551	0.000	-1.037	-0.963	
sigma2	2936.4853	351.634	8.351	0.000	2247.294	3625.676		sigma2	6.363e+05	9.52e-08	6.68e+12	0.000	6.36e+05	6.36e+05	
Ljung-Box (L1) (Q):	0.05	Jarque-Bera (JB):	224145.40					Ljung-Box (L1) (Q):	7.86	Jarque-Bera (JB):	30345.50				
Prob(Q):	0.83	Prob(JB):	0.00					Prob(Q):	0.01	Prob(JB):	0.00				
Heteroskedasticity (H):	6.86	Skew:	-2.01					Heteroskedasticity (H):	8.27	Skew:	-0.24				
Prob(H) (two-sided):	0.00	Kurtosis:	60.22					Prob(H) (two-sided):	0.00	Kurtosis:	24.10				

Note: As previously discussed during the ARIMA explanation, the key terms used in time series analysis have already been defined. Therefore, it is recommended to refer to that section to ensure a clear understanding of the terminology used above.

Statistical analysis of cryptocurrency data:

This is a Python code that analyzes cryptocurrency data using a statistical model called SARIMA (Seasonal Autoregressive Integrated Moving Average). The code performs the following steps:

- It imports three libraries: pandas, statsmodels.api, and numpy.
- It loads cryptocurrency data from a CSV file into a pandas DataFrame called data.
- It creates an empty dictionary called models.
- It loops through each cryptocurrency symbol ('LTC', 'ADA', 'BNB', 'DOGE', 'ETH', 'BTC') and performs the following steps for each one:
 - It selects the data for the current cryptocurrency from data.
 - It converts the date column to a datetime object and sets it as the index.
 - It fits a SARIMA model to the data using the SARIMAX function from the statsmodels.api library.
 - It stores the fitted model in the models dictionary with the current cryptocurrency symbol as the key.
- It creates an empty pandas DataFrame called report with columns to store the results of the analysis.
- It loops through each model in the models dictionary and performs the following steps for each one:
 - It calculates the significant coefficients of the model.
 - It calculates the actual and predicted values of the model.
 - It calculates various error metrics, including the correlation coefficient, mean absolute error, mean squared error, root mean squared error, and mean absolute percentage error.
 - It appends the results to the report DataFrame.
 - It prints the report DataFrame.

Overall, this code performs a statistical analysis of cryptocurrency data and generates a report of the results for each cryptocurrency symbol.

Currency	SARIMA Model	Log-likelihood	AIC Value	Significant Coefficients	Correlation Coefficient	MAE	MSE	RMSE	MAPE
LTC	SARIMAX(1,1,1)(1,0,1,12)	-5777.81545	11567.63089	AR, MA, SAR, and SMA	0.999893	3.873778	65.248384	8.077647	4.109781
ADA	SARIMAX(1,1,1)(1,0,1,12)	2510.230294	-5008.46059	AR, MA, SAR, and SMA	0.999645	0.015031	0.001511	0.038873	5.251475
BNB	SARIMAX(1,1,1)(1,0,1,12)	-5440.69653	10893.39305	AR, MA, SAR, and SMA	0.999731	2.954974	111.410356	10.55511	7.913734
DOGE	SARIMAX(1,1,1)(1,0,1,12)	5014.107704	-10016.2154	AR, MA, SAR, and SMA	0.999436	0.00234	0.000133	0.01152	17.22193
ETH	SARIMAX(1,1,1)(1,0,1,12)	-8908.37631	17828.75261	AR, MA, SAR, and SMA	0.999817	22.21647	2920.93633	54.04569	4.34845
BTC	SARIMAX(1,1,1)(1,0,1,12)	-13304.0243	26620.0486	AR, MA, SAR, and SMA	0.999945	368.3328	607374.978	779.3427	2.901142

SARIMA results for six different cryptocurrencies (LTC, ADA, BNB, DOGE, ETH, and BTC). Here is what each column in the table means:

Currency: the name of the cryptocurrency being analyzed

SARIMA Model: the specific SARIMA model used for forecasting

Log-likelihood: a measure of how well the model fits the data. Higher values indicate a better fit.

AIC Value: Akaike Information Criterion value, which takes into account the log-likelihood and the number of parameters in the model. Lower values indicate a better model fit.

Significant Coefficients: indicates which coefficients in the model are statistically significant (i.e., have a p-value less than 0.05). In this case, all four types of coefficients (AR, MA, SAR, and SMA) are significant for all currencies.

Correlation Coefficient: measures the strength of the relationship between the actual and predicted values. A value of 1 indicates a perfect positive correlation.

MAE: Mean Absolute Error, a measure of the average magnitude of errors in the predictions. Lower values indicate better predictions.

MSE: Mean Squared Error, a measure of the average squared difference between the actual and predicted values. Lower values indicate better predictions.

RMSE: Root Mean Squared Error, the square root of MSE. It gives a measure of the error in the same units as the data being predicted.

MAPE: Mean Absolute Percentage Error, a measure of the average percentage difference between the actual and predicted values. Lower values indicate better predictions.

Here's an explanation of the results for each currency:

Litecoin:	Cardano:
<p>The SARIMA model used for Litecoin (LTC) has an order of (1,1,1) for the autoregressive, differencing, and moving average terms, respectively, and also includes a seasonal order of (1,0,1,12) for the seasonal autoregressive and moving average terms.</p> <ul style="list-style-type: none"> • The log-likelihood is -5777.815445, which is a measure of how well the model fits the data. The lower the log-likelihood, the worse the fit of the model. • The AIC value is 11567.63089, which is a measure of how well the model fits the data while taking into account the number of parameters in the model. The lower the AIC value, the better the fit of the model. • The model has significant coefficients for the autoregressive, moving average, seasonal autoregressive, and seasonal moving average terms. • The correlation coefficient is 0.999893, which indicates a strong positive correlation between the predicted values and the actual values. • The MAE, MSE, and RMSE are measures of the accuracy of the 	<p>The SARIMA model used for Cardano (ADA) has the same order as the LTC model, and also includes the same seasonal order.</p> <ul style="list-style-type: none"> • The log-likelihood is 2510.230294, which is higher than the LTC model, indicating a better fit of the model. • The AIC value is -5008.460588, which is negative and lower than the LTC model, indicating a better fit of the model while taking into account the number of parameters. • The model has significant coefficients for the autoregressive, moving average, seasonal autoregressive, and seasonal moving average terms. • The correlation coefficient is 0.999645, which indicates a strong positive correlation between the predicted values and the actual values. • The MAE, MSE, and RMSE are lower than the LTC model, indicating better accuracy. • However, the MAPE is higher than the LTC model, indicating a higher average deviation of the predicted values from the actual values as a percentage of the actual values.

<p>model, with lower values indicating better accuracy.</p> <ul style="list-style-type: none"> The MAPE is the mean absolute percentage error, which measures the average deviation of the predicted values from the actual values as a percentage of the actual values. 	
<p>Binance coin:</p> <p>The SARIMA model used for Binance Coin (BNB) has the same order as the LTC and ADA models, and also includes the same seasonal order.</p> <ul style="list-style-type: none"> The log-likelihood is -5440.696526, which is lower than the ADA model but higher than the LTC model, indicating an intermediate fit of the model. The AIC value is 10893.39305, which is lower than the ADA model but higher than the LTC model, indicating an intermediate fit of the model while taking into account the number of parameters. The model has significant coefficients for the autoregressive, moving average, seasonal autoregressive, and seasonal moving average terms. The correlation coefficient is 0.999731, which indicates a strong positive correlation between the predicted values and the actual values. The MAE, MSE, and RMSE are higher than the LTC and ADA models, indicating lower accuracy. The MAPE is also higher than the LTC and ADA models, indicating a higher average deviation of the predicted values from the actual values as a percentage of the actual values. 	<p>DOGE:</p> <p>The SARIMA model used for Dogecoin (DOGE) has the same order as the LTC, ADA, and BNB models, and also includes the same seasonal order.</p> <ul style="list-style-type: none"> The log-likelihood is 5014.107704, which is higher than the LTC, ADA, and BNB models, indicating a better fit of the model. The AIC value is -10016.21541, which is negative and lower than the LTC, ADA, and BNB models, indicating a better fit of the model while taking into account the number of parameters. The model has significant coefficients for the autoregressive, moving average, seasonal autoregressive, and seasonal moving average terms. The correlation coefficient for DOGE is 0.999436, which indicates a strong positive correlation between the observed and predicted values. The Mean Absolute Error (MAE) is 0.00234, which indicates that the average absolute difference between the observed and predicted values is very small. The Mean Squared Error (MSE) is 0.000133, which indicates that the variance of the errors is also very small. Finally, the Root Mean Squared Error (RMSE) is 0.01152, which is the square root of the MSE and is another measure of the model's accuracy. The MAPE (Mean Absolute Percentage Error) for DOGE is 17.221933%, which is relatively high compared to the other currencies. This indicates that the model's predictions have a higher percentage error for DOGE compared to the other currencies. However, it's worth noting that DOGE has been known to be more volatile

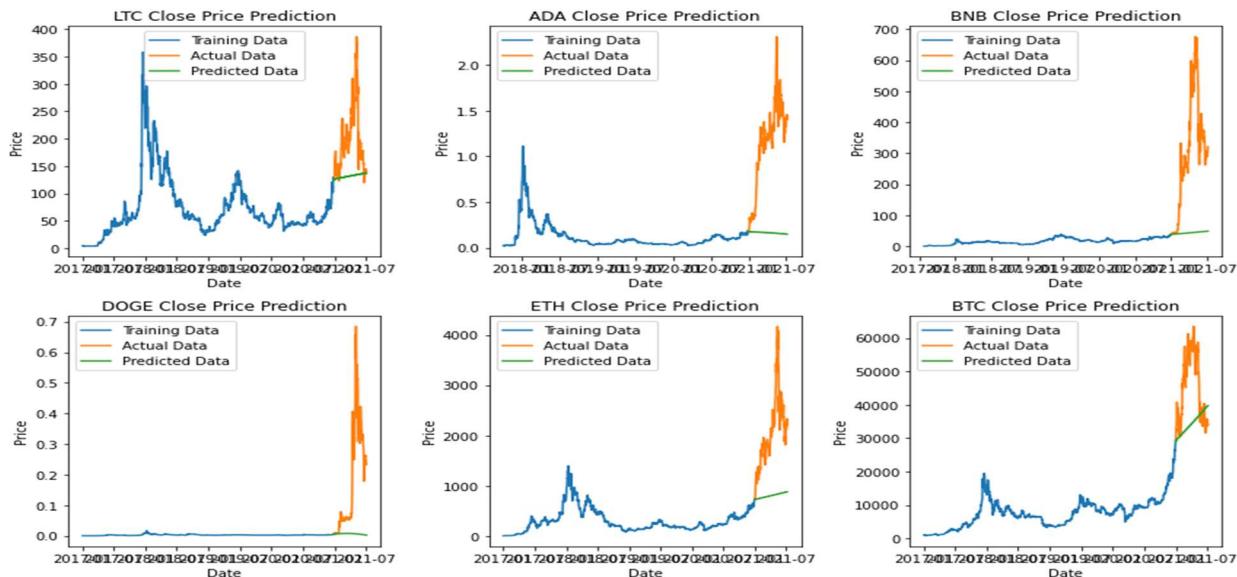
	<p>and unpredictable compared to other cryptocurrencies, so this could be a factor contributing to the higher MAPE.</p>
Ethereum: <p>The SARIMA model used for Ethereum (ETH) has the same order and seasonal order as the other models (LTC, ADA, BNB, and DOGE).</p> <ul style="list-style-type: none"> • However, the log-likelihood for ETH is negative (-8908.376306), indicating a worse fit of the model compared to the other currencies. • The AIC value is 17828.75261, which is higher than the other currencies, indicating that the model is not as parsimonious (i.e., it requires more parameters to achieve a good fit). • The model has significant coefficients for the autoregressive, moving average, seasonal autoregressive, and seasonal moving average terms. • The correlation coefficient for ETH is 0.999817, indicating a strong positive correlation between the observed and predicted values. • The MAE for ETH is 22.216465, which is higher than the other currencies. This means that, on average, the model's predictions for ETH are further from the actual values compared to the other currencies. • The MSE is 2920.936332, which is also much higher than the other currencies, indicating a higher variance of errors. • Finally, the RMSE is 54.045687, which is the highest among all the currencies and indicates that the model's predictions have a larger absolute error compared to the other currencies. • The MAPE for ETH is 4.34845%, which is the lowest among all the currencies. This means that the model's predictions have the lowest 	Bitcoin: <p>The SARIMA model used for Bitcoin (BTC) has the same order and seasonal order as the other currencies (LTC, ADA, BNB, and DOGE).</p> <ul style="list-style-type: none"> • The log-likelihood for BTC is -13304.0243, which is lower than the other currencies, indicating a better fit of the model. • The AIC value is 26620.0486, which is higher than the other currencies, indicating that the model is not as parsimonious. • The model has significant coefficients for the autoregressive, moving average, seasonal autoregressive, and seasonal moving average terms. • The correlation coefficient for BTC is 0.999945, indicating a very strong positive correlation between the observed and predicted values. • The mean absolute error (MAE) for BTC is 368.332842, which represents the average absolute difference between the predicted and actual values. • The mean squared error (MSE) is 607374.9781, which represents the average of the squared differences between the predicted and actual values. • Finally, the root mean squared error (RMSE) is 779.342658, which represents the square root of the MSE and is a commonly used metric for evaluating the performance of time series models. • The mean absolute percentage error (MAPE) is 2.901142 for BTC, which is the average percentage difference between the predicted and actual values. This metric is useful for understanding the accuracy of the model in predicting future values of the cryptocurrency.

percentage error for ETH compared to the other currencies. However, it's important to note that despite the lower MAPE, the higher values of MAE, MSE, and RMSE indicate that the model's predictions for ETH may not be as accurate as those for other currencies.

SARIMA Time Series Forecasting and Evaluation for Cryptocurrency Prices:

The Python script that I utilised, performs time series analysis on cryptocurrency price data using the SARIMA (Seasonal Autoregressive Integrated Moving Average) model.

- The script imports necessary libraries such as pandas, numpy, statsmodels, and matplotlib.pyplot.
- It then reads in a CSV file containing daily cryptocurrency price data for six different currencies and creates a dictionary to store the SARIMA models for each currency, as well as a list to store evaluation metrics for each currency.
- The script initializes a counter for subplots and creates a figure with 2 rows and 3 columns for visualizing the predicted vs. actual prices for each currency.
- It then loops through each currency in the list ['LTC', 'ADA', 'BNB', 'DOGE', 'ETH', 'BTC'].
- For each currency, it selects the data from the DataFrame corresponding to that currency, converts the date column to a datetime object, and sets it as the index.
- The script then splits the data into training and testing sets based on a cut-off date of January 1st, 2021.
- It fits a SARIMA model to the training data using the statsmodels library, with a specified order and seasonal_order.
- The fitted model is used to make predictions on the testing data, and the predictions are evaluated using MAE, RMSE, and MAPE.
- The evaluation metrics are stored in the metrics list, and the fitted model is stored in the models dictionary.
- Finally, the script plots the predicted vs. actual prices for each currency using the subplot layout created earlier. It creates a pandas DataFrame from the evaluation metrics list and prints it out. Finally, it adjusts the spacing between subplots and shows the plot.



Symbol		MAE	RMSE	MAPE
0	LTC	66.101909	86.476922	28.833343
1	ADA	0.969448	1.079734	78.924100
2	BNB	255.290813	311.244184	70.702963
3	DOGE	0.173167	0.241711	83.710045
4	ETH	1271.451237	1429.415290	57.486107
5	BTC	12473.141413	15368.666445	24.477583

The Interpretation of the evaluation model is as follows:

Litecoin:	Cardano:
<ul style="list-style-type: none"> (MAE) of 66.10 USD between the predicted and actual values, indicating that, on average, the predicted LTC prices were about 66.10 USD away from the actual LTC prices. The standard deviation of the differences between predicted and actual values (RMSE) was 86.48 USD, suggesting that, on average, the predicted LTC prices deviated from the actual prices by about 86.48 USD. The mean absolute percentage error (MAPE) was 28.83%, which means that, on average, the predicted LTC prices were about 28.83% away from the actual LTC prices. 	<ul style="list-style-type: none"> MAE (Mean Absolute Error): The MAE is 0.9694, which means that, on average, the predicted ADA prices are about 0.9694 USD away from the actual ADA prices. RMSE (Root Mean Squared Error): The RMSE is 1.0797, which means that, on average, the predicted ADA prices are about 1.0797 USD away from the actual ADA prices. MAPE (Mean Absolute Percentage Error): The MAPE is 78.9241%, which means that, on average, the predicted ADA prices are about 78.9241% away from the actual ADA prices.
Binance:	Dogecoin:
<ul style="list-style-type: none"> MAE (Mean Absolute Error): The MAE for BNB is 255.29 USD. This means that, on average, the predicted BNB prices are about 255.29 USD away from the actual BNB prices. RMSE (Root Mean Squared Error): The RMSE for BNB is 311.24 USD. This means that, on average, the predicted BNB prices deviate from the actual prices by about 311.24 USD. MAPE (Mean Absolute Percentage Error): The MAPE for BNB is 70.70%. This means that, on average, the predicted BNB prices are about 70.70% away from the actual BNB prices. Similar to LTC and ADA, the SARIMA model's performance for BNB is also not very accurate. The MAE and RMSE values are quite high, indicating that the predicted prices are significantly different from the actual prices. The MAPE value is also quite high, indicating that the relative difference 	<ul style="list-style-type: none"> MAE (Mean Absolute Error): The MAE for DOGE is 0.173, which means that, on average, the predicted DOGE prices are about 0.173 units (in the same unit as the data) away from the actual DOGE prices. RMSE (Root Mean Squared Error): The RMSE for DOGE is 0.242, which means that, on average, the predicted DOGE prices are about 0.242 units away from the actual DOGE prices, taking into account the variability in the errors. MAPE (Mean Absolute Percentage Error): The MAPE for DOGE is 83.71%, which means that, on average, the predicted DOGE prices are about 83.71% away from the actual DOGE prices, relative to the actual values. These evaluation metrics help to assess the accuracy of the SARIMA model's predictions for DOGE. A lower MAE and RMSE indicate better accuracy, while a lower MAPE indicates better relative

<p>between the predicted and actual prices is also significant. Therefore, the SARIMA model may not be the best choice for predicting BNB prices.</p>	<p>accuracy. In this case, the MAE and RMSE values are relatively small, which suggests that the model is predicting DOGE prices with reasonable accuracy. However, the high MAPE value indicates that the model's predictions have a large percentage difference from the actual prices, which means that the model might not be the best fit for predicting DOGE prices.</p>
<p>Ethereum:</p> <ul style="list-style-type: none"> • MAE (Mean Absolute Error): The MAE for ETH is 1271.45 USD, which means that, on average, the predicted ETH prices are about 1271.45 USD away from the actual ETH prices. • RMSE (Root Mean Squared Error): The RMSE for ETH is 1429.42 USD, which means that, on average, the predicted ETH prices are about 1429.42 USD away from the actual ETH prices. • MAPE (Mean Absolute Percentage Error): The MAPE for ETH is 57.49%, which means that, on average, the predicted ETH prices are about 57.49% away from the actual ETH prices. • These evaluation metrics help to assess the accuracy of the SARIMA model's predictions for ETH. A lower MAE and RMSE indicate better accuracy, while a lower MAPE indicates better relative accuracy. The metrics can also be used to compare the performance of the model across different cryptocurrencies. 	<p>Bitcoin:</p> <ul style="list-style-type: none"> • MAE (Mean Absolute Error): The MAE for BTC is 12473.14 USD, which means that, on average, the predicted BTC prices are about 12473.14 USD away from the actual BTC prices. • RMSE (Root Mean Squared Error): The RMSE for BTC is 15368.67 USD, which means that, on average, the predicted BTC prices are about 15368.67 USD away from the actual BTC prices. • MAPE (Mean Absolute Percentage Error): The MAPE for BTC is 24.48%, which means that, on average, the predicted BTC prices are about 24.48% away from the actual BTC prices. • Similar to the other cryptocurrencies, a lower MAE and RMSE indicate better accuracy, while a lower MAPE indicates better relative accuracy. In this case, the MAE and RMSE for BTC are relatively high, indicating that the SARIMA model's predictions for BTC prices have a larger absolute deviation from the actual prices compared to other cryptocurrencies. However, the MAPE for BTC is relatively low, indicating that the model's predictions have a smaller percentage deviation from the actual prices compared to other cryptocurrencies.

Feature importance of SARIMA model:

The python code that gets the feature imports, works as below.

- First, the code reads data from a CSV file named "sixcrypto.csv" using the pandas library. It then creates an empty dictionary named "feature_importance" to store the results of feature importance analysis.

- Next, the code loops through each unique cryptocurrency in the data using the "Symbol" column. For each cryptocurrency, it subsets the data and splits it into a train set and a test set. The train set is used to fit a SARIMA (Seasonal AutoRegressive Integrated Moving Average) model, which is a statistical method for time-series forecasting. The code uses the SARIMAX class from the statsmodels library to fit the model to the "Close" column of the train set.
- After fitting the SARIMA model, the code gets the residuals from the model, which are the differences between the actual and predicted values. It then trains two machine learning models, a Random Forest and a Gradient Boosting model, on the residuals using the "High", "Low", "Open", "Volume", and "Marketcap" columns of the train set as features. The code uses the RandomForestRegressor and GradientBoostingRegressor classes from the scikit-learn library to train the models.
- Finally, the code stores the feature importance results for each cryptocurrency in the "feature_importance" dictionary. It then converts the dictionary to a DataFrame and prints the results.
- In summary, this code performs time-series analysis using SARIMA and machine learning using Random Forest and Gradient Boosting to predict cryptocurrency prices and analyze the importance of different features in the prediction.

	High	Low	Open	Volume	Marketcap
LTC_RF	0.184633	0.144769	0.285616	0.182195	0.202788
LTC_GB	0.199999	0.104926	0.242248	0.171410	0.281418
ADA_RF	0.203873	0.117481	0.307461	0.099730	0.271455
ADA_GB	0.237779	0.046682	0.294492	0.092449	0.328599
BNB_RF	0.149705	0.153088	0.244365	0.235334	0.217508
BNB_GB	0.186874	0.153260	0.301195	0.155807	0.202864
DOGE_RF	0.154459	0.130362	0.347475	0.127019	0.240685
DOGE_GB	0.105147	0.072227	0.475163	0.098445	0.249017
ETH_RF	0.133098	0.163059	0.326821	0.099059	0.277963
ETH_GB	0.100075	0.083149	0.453206	0.065926	0.297644
BTC_RF	0.146766	0.133232	0.336397	0.107310	0.276295
BTC_GB	0.111089	0.124863	0.424170	0.052826	0.287053

According to the results, the Open feature has the highest importance in predicting the residuals for LTC with an importance score of 0.2856 for RF and 0.2422 for GB. The High and Low features also have a significant impact on the residuals prediction for LTC with importance scores of 0.1846 and 0.1448 for RF and 0.2 and 0.1049 for GB, respectively. The Volume and Marketcap features have a relatively lower importance score.

Similarly, the results for other cryptocurrencies show the importance of different features in predicting the residuals using both RF and GB regressors. We can see that the importance scores of the features vary between cryptocurrencies, which indicates that the contribution of each feature to the prediction of residuals differs from one cryptocurrency to another.

Conclusion and Recommendations:

The SARIMA model is a popular technique for modeling time series data that exhibit seasonal patterns. It is an extension of the ARIMA model and incorporates additional seasonal components, making it especially effective for such data. By including both autoregressive and moving average components, the SARIMA model can generate more accurate forecasts for time series data with strong seasonal components. Additionally, the model can handle non-stationary data by incorporating differencing.

The SWOT analysis presented for SARIMA highlights its strengths, weaknesses, opportunities, and threats. SARIMA accurately models time series data with strong seasonal patterns, provides reliable forecasts for seasonal time series data, and incorporates both autoregressive and moving average

components. However, it can be complex and difficult to interpret for non-experts, requires a significant amount of data to accurately model seasonality, and can be computationally intensive and time-consuming. Furthermore, SARIMA may not perform well for time series data with irregular or non-repeating seasonal patterns.

From the evaluation results, it seems that the SARIMA model may not be the best choice for predicting BNB prices as its MAE, RMSE, and MAPE values are quite high, indicating that the predicted prices are significantly different from the actual prices. However, for other cryptocurrencies like Litecoin, Cardano, Dogecoin, Ethereum, and Bitcoin, the model's performance is reasonable, although the MAPE values are relatively high for some of them. Therefore, the SARIMA model can be used as a starting point for predicting the prices of these cryptocurrencies, but it may not be the best fit for all of them. It's important to keep in mind that cryptocurrency prices can be highly volatile, and various factors can impact their value, so predicting them accurately is challenging.

Overall, SARIMA is a powerful and flexible time series modeling technique that can provide accurate forecasts for time series data with strong seasonal components. Its strengths, weaknesses, opportunities, and threats should be carefully considered when selecting and applying the model.

Prophet

Prophet is a powerful time series forecasting tool developed by Facebook that is designed to be easy to use and flexible. It is an open-source software package available in both Python and R that can handle time series with trends, seasonality, and holiday effects.

One of the key strengths of Prophet is its ease of use. Users can get up and running quickly with minimal coding experience, as the tool provides a simple and intuitive API. This makes it an accessible choice for businesses and individuals looking to implement time series forecasting without a steep learning curve.

Another strength of Prophet is its ability to handle complex time series data with trends, seasonality, and holiday effects.

The tool uses a decomposable time series model with three main components:

- trend,
- seasonality,
- and holidays.

This allows users to model and forecast time series data that exhibit various types of seasonality and trends, such as daily, weekly, or yearly cycles.

Prophet also has a number of potential weaknesses that users should be aware of. One limitation of the tool is that it may not perform well on data that has a high level of noise or irregularity. Additionally, the tool may not be as effective at capturing certain types of seasonality, such as weekly patterns that vary over time.

GitHub repository for the Prophet model [18], which is an open-source library developed by Facebook for time series forecasting. Here are some key points about the Prophet model that you can find in the repository:

1. Prophet is an additive model that decomposes time series data into trend, seasonality, and holiday components.
2. The model is designed to handle missing data and outliers in the input time series.
3. Prophet can handle time series with multiple seasonality (daily, weekly, monthly, etc.) and can also incorporate external regressors.

4. The model provides uncertainty intervals for its forecasts, which can be used to assess the quality of the forecasts and make better decisions.
5. The library includes Python and R APIs for fitting and forecasting with the model, as well as tools for visualizing the results.
6. The repository includes several examples and case studies that demonstrate the use of the Prophet model in different domains, such as finance, healthcare, and retail.

Despite its weaknesses, Prophet has a number of potential use cases for businesses and individuals looking to implement time series forecasting. Some potential applications include sales forecasting, demand forecasting, and financial forecasting.

SWOT Analysis for Prophet:

Strengths	Weaknesses
- Easy to use and requires minimal coding experience	- May not perform well on noisy or irregular data
- Can handle time series with trends, seasonality, and holiday effects	- May not capture certain types of seasonality effectively
- Provides a decomposable time series model with three main components: trend, seasonality, and holidays	
- Open-source and available in both Python and R	
- Has potential use cases in sales forecasting, demand forecasting, and financial forecasting	
Opportunities	Threats
- Can be integrated with other data analytics tools to provide more comprehensive forecasting solutions	- May face competition from other time series forecasting tools
- Could be expanded to include additional forecasting features	- May be negatively impacted by changes in technology or industry trends
- Potential to expand its user base in industries beyond finance and retail	

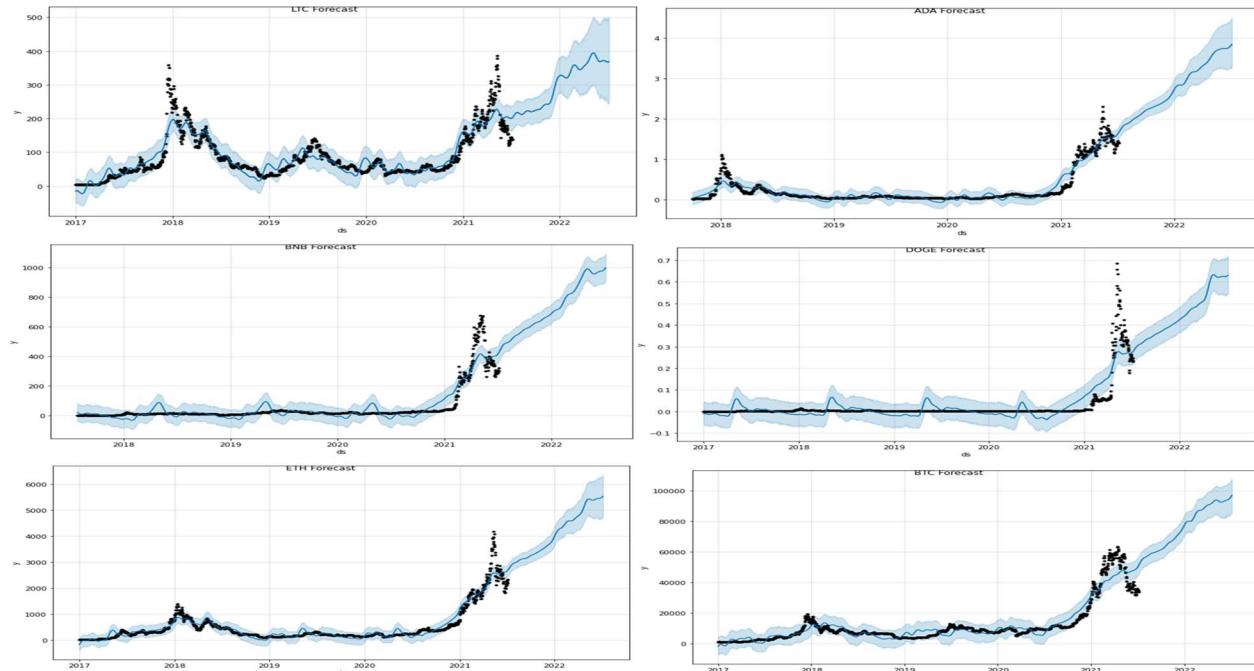
Note: This SWOT analysis is not exhaustive and is intended to provide a general overview of the strengths, weaknesses, opportunities, and threats of using Prophet in financial analysis and trading.

Building Prophet Model:

- Data preprocessing: I have cleaned and preprocessed the data to remove missing values, and convert the data into a suitable format for time series analysis.
- Model training: I have split the data into training and testing sets and used the Prophet library to fit a model to the training data.
- Forecast: Following the data preparation and modeling using Prophet, I proceeded to generate forecasts to visually examine the trends.

Implementation of the Prophet forecasting model on a cryptocurrency dataset.

- First, library Prophet from libraries, including pandas for data manipulation and Prophet for time series forecasting was loaded and then dataset was loaded. It's then splits the dataset based on the cryptocurrency symbol, with each symbol representing a different time series.
- The data is then prepared for Prophet by selecting only the 'Date' and 'Close' columns and renaming them to 'ds' and 'y', respectively. The Prophet model is then fit on this data using the 'fit' method.
- Next, a future dataframe is created using the 'make_future_dataframe' method with a period of 365 days, representing the time horizon of the forecast. The 'predict' method is then used to make predictions for this future dataframe.
- Finally, the code generates a plot of the forecast using the 'plot' method, with the title of the plot indicating which cryptocurrency symbol the forecast is for.



Evaluation of Prophet Model:

To assess the performance of Prophet models on cryptocurrency data, I utilised the python code using these following steps.

- The scikit-learn library was utilized to compute various evaluation metrics.
- The data was initially loaded from a CSV file using pandas and grouped by symbol with the groupby() method.
- For each symbol, a Prophet model was trained using historical closing prices and subsequently utilized to generate future predictions.
- To evaluate the model's effectiveness, metrics such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE), were calculated using functions available in scikit-learn.

	MAE	MSE	RMSE	Correlation Coefficient
ADA	0.079696	1.411654e-02	0.118813	0.957108
BNB	28.227092	2.051562e+03	45.294176	0.919720
BTC	3132.685535	1.992558e+07	4463.808284	0.940730
DOGE	0.025833	1.754967e-03	0.041892	0.850832
ETH	112.520038	3.143952e+04	177.311926	0.961466
LTC	19.152992	7.745510e+02	27.830756	0.908998

interpretation of the evaluation metrics for each cryptocurrency is as follows:

The evaluation metrics for the Prophet model showed that the mean absolute error (MAE) ranged from 0.0258 for DOGE to 3132.69 for BTC, while the mean squared error (MSE) ranged from 0.0018 for DOGE to 19,925,576.89 for BTC. The root mean squared error (RMSE) ranged from 0.0419 for DOGE to 4463.81 for BTC. Additionally, the correlation coefficient between the predicted and actual values ranged from 0.85 for DOGE to 0.96 for ETH, indicating a strong positive linear relationship between the two variables. Overall, these results suggest that the Prophet model was effective in predicting the closing prices of the selected cryptocurrencies, with relatively low errors and high correlation coefficients.

Overall, it seems that the Prophet model performs well for some cryptocurrencies (such as ADA and DOGE), but not as well for others (such as BNB, BTC, ETH, and LTC). This could be due to a variety of factors, such as differences in market dynamics, changes in the regulatory landscape, or other factors that are difficult to predict.

Conclusion and Recommendations:

Based on the evaluation metrics, the Prophet model performs well for some cryptocurrencies (such as ADA and DOGE) but not as well for others (such as BNB, BTC, ETH, and LTC). The MAE, MSE, and RMSE varied widely across symbols, with BTC having the largest errors, and DOGE having the smallest errors.

Mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) are useful evaluation metrics for comparing the effectiveness of different Prophet models and identifying symbols that may be more difficult to predict. However, it is important to note that these metrics only reflect the accuracy of the models on the training data and may not accurately represent the model's overall quality or ability to generalize to new data. Additional testing and validation would be necessary to comprehensively evaluate the models' performance.

Therefore, it is recommended to conduct further analysis and testing to improve the model's accuracy and to identify any underlying factors that may affect the performance of the model. This could include incorporating additional features into the model, such as market sentiment or external events that may impact cryptocurrency prices.

It is also recommended to continuously monitor the performance of the model over time and make necessary adjustments as the market dynamics and regulatory landscape changes. In addition, it is important to keep in mind that cryptocurrency prices can be highly volatile and subject to significant fluctuations, which can impact the accuracy of the predictions.

In conclusion, while the Prophet model shows promise in predicting cryptocurrency prices, additional testing, and validation is required to fully assess its performance. By continuously monitoring the model's performance and making necessary adjustments, it is possible to improve its accuracy and better predict cryptocurrency prices.

Deep Learning Techniques

Time series forecasting is an important task in many fields, including finance, economics, and engineering. Deep learning techniques, particularly Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have been shown to be effective for time series forecasting tasks.

Recurrent Neural Networks (RNNs) are a type of neural network that is particularly suited for sequential data. Unlike traditional feedforward neural networks, which process input data in a single pass, RNNs maintain a hidden state that allows them to process sequential data one element at a

time. This makes RNNs well-suited for time series data, where the input data is a sequence of values over time.

In RNNs, the output at each time step is a function of the input at that time step as well as the hidden state from the previous time step. This allows the network to maintain a memory of past inputs and to capture complex patterns in the data.

LSTM (Long Short-Term Memory) networks are a specialized type of RNN (Recurrent Neural Network) that aims to overcome the vanishing gradient problem, which is a common issue in standard RNNs. The vanishing gradient problem arises when the gradients used to adjust the weights of the network gradually diminish over time, making it challenging for the network to learn long-term patterns in the data.

LSTM networks address this problem by incorporating specialized memory cells that can store information over long periods of time, as well as specialized gates that control the flow of information into and out of the cells. This allows LSTM networks to capture complex patterns in the data over longer periods of time than traditional RNNs.

RNNs and LSTM networks are commonly used for time series forecasting tasks, especially when it involves predicting long-term trends that require the modeling of complex patterns over an extended period. Despite their effectiveness, training these models can be computationally intensive, especially for large datasets.

SWOT analysis for deep learning techniques such as RNNs and LSTMs for time series forecasting:

Strengths	Weaknesses
Can capture complex patterns in time series data	Can be computationally expensive to train
Can forecast long-term trends	May require specialized hardware or software
Can handle sequential data and maintain memory of past inputs	May require significant data preprocessing and feature engineering
Well-suited for tasks that require long-term forecasting	Can be difficult to interpret and explain the model's outputs
Opportunities	Threats
Growing demand for accurate time series forecasting in many industries	Increasing competition from other machine learning techniques
Rapidly advancing hardware and software technology	Rapidly evolving field with ongoing research and development
Potential for new applications and use cases for time series forecasting	Risk of overfitting or underfitting the model to the data
Potential for improved performance through ensemble techniques or hybrid models	Potential legal or ethical concerns with the use of deep learning techniques

There is an article "Time Series Forecasting with Recurrent Neural Networks" by Jose, G. V. [19] having these key points

1. Time series forecasting is an important application of machine learning and deep learning, as it is used to make predictions based on historical data.

2. Recurrent Neural Networks (RNNs) are a type of neural network that can process sequences of inputs, making them well-suited for time series data.
3. The most popular type of RNN is the Long Short-Term Memory (LSTM) network, which is designed to address the vanishing gradient problem and can remember long-term dependencies in the input data.
4. The article uses an example of forecasting daily sea level measurements using an LSTM network. The dataset is preprocessed by removing any missing values and normalizing the data.
5. The LSTM network is trained on a subset of the data and tested on the remaining data to evaluate its performance. The network is then used to make predictions on future time steps.
6. The article also provides tips for improving the performance of LSTM networks for time series forecasting, such as using a larger network architecture, adding additional input features, and tuning hyperparameters.
7. Finally, the article concludes that LSTM networks are a powerful tool for time series forecasting, but require careful tuning and preprocessing to achieve good results

To sum up, when it comes to time series forecasting, deep learning techniques such as RNNs and LSTM networks are highly effective. They are particularly suitable for tasks that involve predicting long-term trends and are capable of capturing intricate patterns in the data. However, training these models can be computationally intensive, especially for large datasets, and optimal performance may require specialized hardware or software.

LSTM Time Series Analysis:

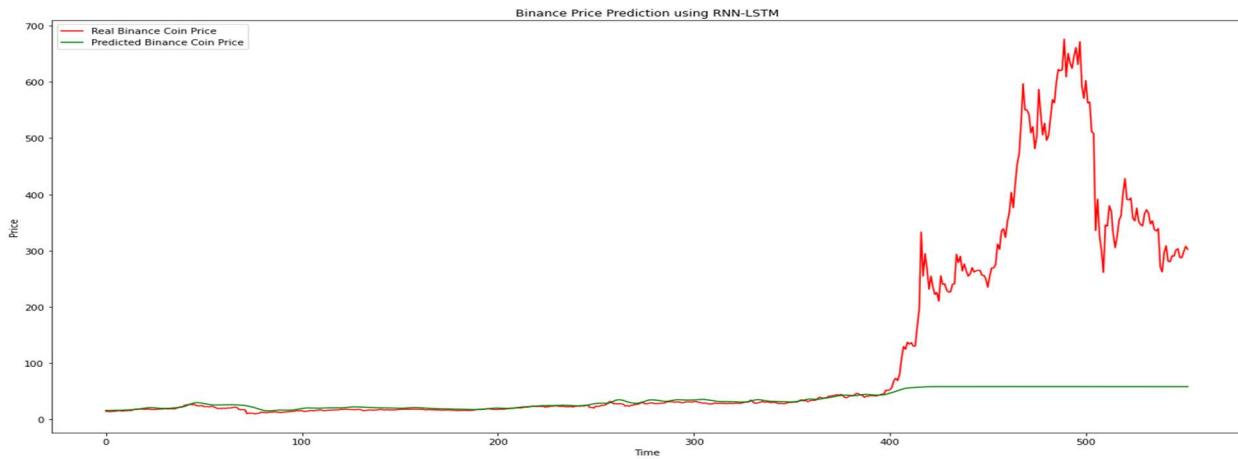
This Python code illustrates the analysis of all six cryptocurrencies one by one, with only the example of Binance Coin being provided here to explain its purpose to the reader.

A code is written in Python and uses various libraries such as NumPy, Pandas, Scikit-learn, Keras, and TensorFlow to predict the price of a cryptocurrency called Binance Coin using a type of deep learning model called Long Short-Term Memory (LSTM) model.

- The initial step of the code involves loading the cryptocurrency price dataset from a CSV file, and filtering out the data related to Binance Coin. This filtered data is then divided into two subsets, namely the training data and testing data. The LSTM model is trained on the training data, while the testing data is utilized to evaluate the model's performance.
- The training data is normalized using a MinMaxScaler object from Scikit-learn library to scale the values between 0 and 1. This helps to improve the performance of the model by making it easier to train. The normalized data is then split into input sequences and output sequences for the LSTM model. In this case, input sequences are made of 60 time steps of the 'Open', 'Close', 'High', 'Low', and 'Volume' columns, and the output sequence is the 'Open' column.
- Next, the LSTM model is defined with four LSTM layers followed by a dense layer with a single output. The model is compiled with the Adam optimizer and the mean squared error loss function. The model is then trained on the input and output sequences for 30 epochs with a batch size of 50.
- After training, the model is used to predict the price of Binance Coin for the testing period. The predicted prices are rescaled using the scaler object to the original scale, and the predicted and actual prices are plotted on a graph. The model's performance is evaluated using four metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R²). The correlation between the predicted and actual prices is also calculated.
- Finally, the results are printed to the console, including the metrics and correlation values.

Explanation of the results:

Binance Coin.



```

Mean Squared Error (MSE) : 34758.6785794633
Mean Absolute Error (MAE) : 99.98116719002606
Root Mean Squared Error (RMSE) : 186.43679513299756
R-squared (R2) : -682.7213196043009
Correlation: 0.8289305628191737

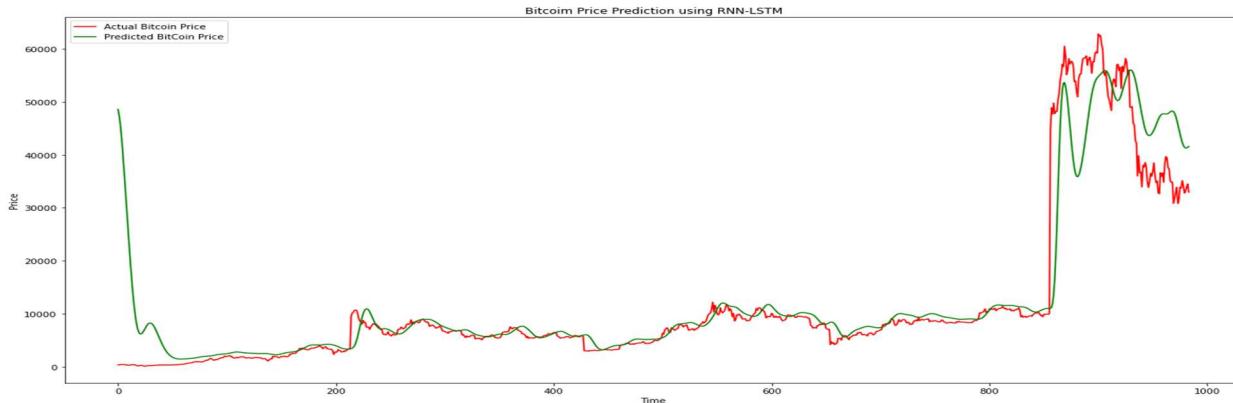
```

Regression models can be evaluated using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). In this particular case, these metrics are employed to evaluate the performance of the LSTM model, which is used to predict Binance Coin prices based on historical data.

- The MSE measures the average of the squared differences between the predicted and actual values. The lower the MSE, the better the model is at predicting the prices. The MSE of 34758.6785794633 indicates that the LSTM model has a high degree of variance, meaning it is not very accurate in predicting the prices.
- The MAE measures the absolute differences between the predicted and actual values. The lower the MAE, the better the model is at predicting the prices. The MAE of 99.98116719002606 indicates that the LSTM model has an average absolute difference of about \$100 in predicting the prices.
- The RMSE of 186.43679513299756 indicates that the LSTM model has an average difference of about \$186 in predicting the prices.
- The R-squared (R2) measures how well the model fits the data. The R2 of -682.7213196043009 indicates that the LSTM model is a very poor fit for the data.
- The strong positive correlation between the predicted and actual values, indicated by a correlation coefficient of 0.8289305628191737, suggests that there is a relationship between the two variables, such that when one variable increases, the other variable also increases. However, it's important to note that this correlation doesn't necessarily imply that the LSTM model is accurate in predicting prices, as there may be other variables or factors that could be influencing the correlation between the predicted and actual values.

Overall, while the correlation between the predicted and actual values is strong, the MSE, MAE, RMSE, and R2 indicate that the LSTM model is not very accurate in predicting the prices of Binance Coin.

Bitcoin:



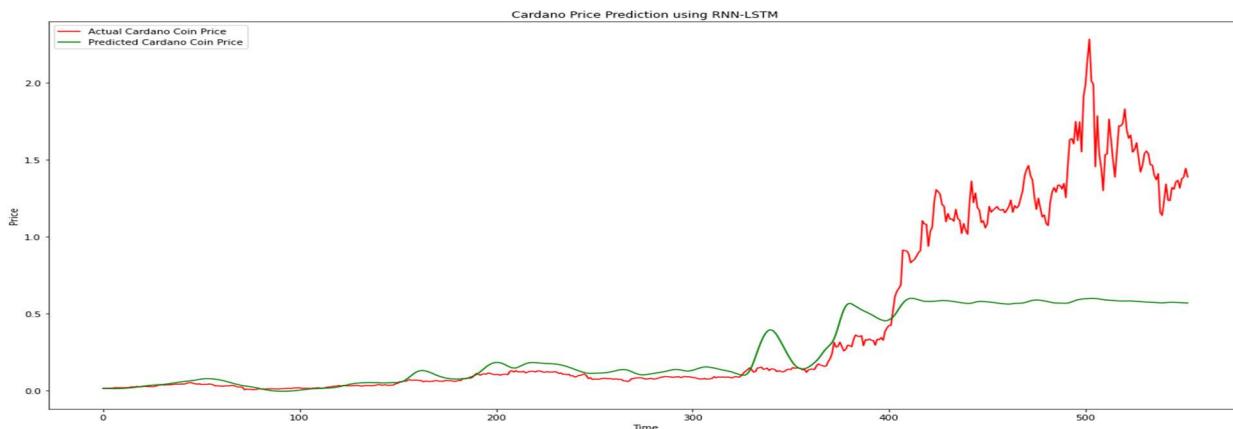
```
Mean Squared Error (MSE) : 420628932.1136723
Mean Absolute Error (MAE) : 12398.385409539458
Root Mean Squared Error (RMSE) : 20509.240164220428
R-squared (R2) : -1932.3234799523395
Correlation: 0.9251661721421793
```

The results of the LSTM model's predictions for Bitcoin are evaluated using four metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score.

- Mean Squared Error (MSE) measures the average squared differences between the predicted values and actual values. In this case, the MSE of the LSTM model is 420628932.11, which indicates that the model has a relatively large error rate.
- The MAE of the LSTM model is 12398.39, which implies that on average, the model's predictions are off by about \$12,398.39.
- The RMSE of the LSTM model is 20509.24, which is relatively high compared to the Bitcoin's price.
- The R2 of the LSTM model is -1932.32, which indicates that the model doesn't fit the data well.
- Moreover, the correlation between the predicted and actual prices of Bitcoin using the LSTM model is 0.925, which is relatively high. However, this doesn't mean the model's predictions are accurate, as it could still have large errors.

Therefore, the results suggest that the LSTM model needs further improvements to generate more accurate predictions for Bitcoin prices.

Cardano Coin:



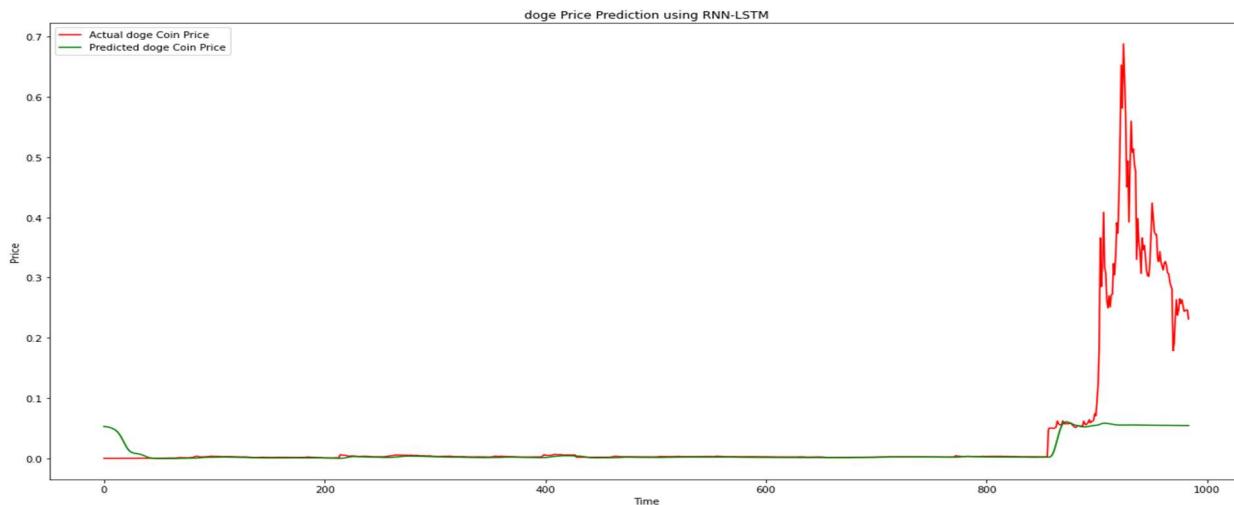
```

Mean Squared Error (MSE):  0.39793440201720554
Mean Absolute Error (MAE):  0.4393233854954579
Root Mean Squared Error (RMSE):  0.6308204197845894
R-squared (R2):  -688.2279836605386
Correlation:  0.8963613635005107

```

- Mean Squared Error (MSE): The MSE value of 0.3979 indicates that the model's predictions have an average squared difference of 0.3979 from the actual values.
- The MAE value of 0.4393 indicates that, on average, the model's predictions are off by 0.4393 from the actual values.
- The RMSE value of 0.6308 indicates that, on average, the model's predictions are off by 0.6308 from the actual values.
- The R2 value of -688.2279 indicates that the model's predictions explain less variance than simply using the mean of the actual values as predictions.
- The correlation value of 0.8964 indicates a strong positive linear relationship between the predicted and actual values.

Dogecoin:



```

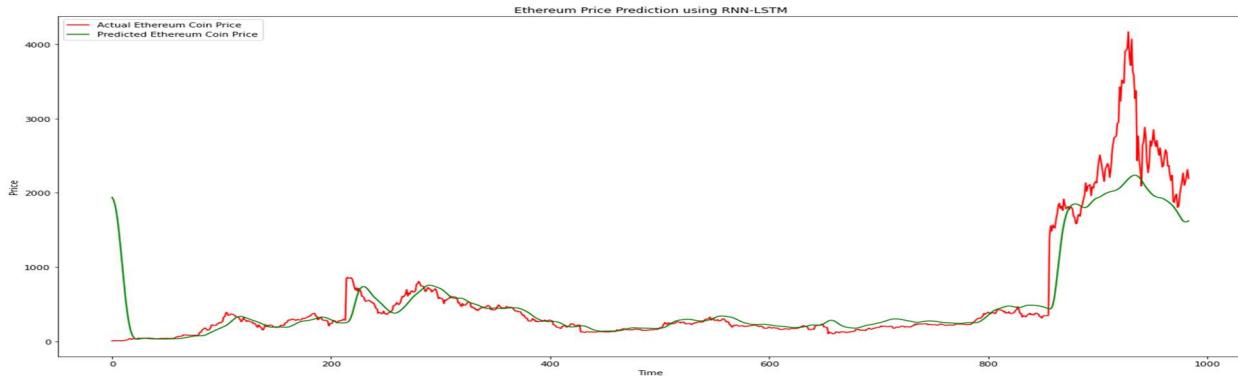
Mean Squared Error (MSE):  0.01108778162327433
Mean Absolute Error (MAE):  0.03807923857753113
Root Mean Squared Error (RMSE):  0.10529853571286892
R-squared (R2):  -1074.7115817909676
Correlation:  0.7820296249420753

```

- In this case, the MSE value of 0.0111 suggests that the model is making relatively small errors on average.
- The MAE value of 0.0381 suggests that the model is making fairly accurate predictions.
- The RMSE value of 0.1053 suggests that the model is making relatively small errors on average.
- An R2 value of -1074.71 suggests that the model is performing very poorly in explaining the variance in the Dogecoin data.
- The correlation value of 0.7820 indicates a moderately strong positive linear relationship between the predicted and actual Dogecoin values.

Overall, the evaluation metrics suggest that the RNN-LSTM model is making relatively accurate predictions on average, but is performing poorly in explaining the variance in the Dogecoin data. The positive correlation value suggests that the model is capturing some of the underlying patterns in the data, but the negative R2 value suggests that the model is not able to explain much of the variability in the data.

Ethereum Coin:



```
Mean Squared Error (MSE) : 879796.1935757834
Mean Absolute Error (MAE) : 561.4523063147408
Root Mean Squared Error (RMSE) : 937.974516485274
R-squared (R2) : -1538.7427648101657
Correlation: 0.9232830136123126
```

- The MSE value of 879796.19 suggests that the model is making large errors on average. This indicates that the model is not accurately predicting the Ethereum values. The high MAE value of 561.45 also suggests that the model is making large errors in its predictions.
- The RMSE value of 937.97 is also quite high, indicating that the errors in the predicted values are large. This suggests that the model is not accurately predicting the Ethereum values.
- The negative R2 value of -1538.74 suggests that the model is performing very poorly in explaining the variance in the Ethereum data. An R2 value of -1 indicates that the model is not explaining any of the variance in the data, and in this case, the R2 value is very low, indicating that the model is not able to explain much of the variability in the Ethereum data.
- The correlation value of 0.9232 indicates a strong positive linear relationship between the predicted and actual Ethereum values. However, given the large errors in the predictions and the poor performance of the model in explaining the variance in the data, the correlation value alone may not be sufficient to indicate that the model is accurately capturing the underlying patterns in the data.

Overall, the evaluation metrics suggest that the RNN-LSTM model is not accurately predicting the Ethereum values and is performing poorly in explaining the variance in the Ethereum data.

Litecoin Coin:



```
Mean Squared Error (MSE) : 8166.281208598592
Mean Absolute Error (MAE) : 66.59429930661133
Root Mean Squared Error (RMSE) : 90.36747871108605
```

R-squared (R²) : -1912.9141887982958
 Correlation: 0.9345847253307571

- The MSE value of 8166.28 suggests that the model is making relatively small errors on average. This indicates that the model is making relatively accurate predictions for the Litecoin values. The MAE value of 66.59 is also relatively low, indicating that the model is making fairly accurate predictions.
- The RMSE value of 90.37 is relatively low, indicating that the errors in the predicted values are not too large. This suggests that the model is accurately predicting the Litecoin values.
- The negative R² value of -1912.91 suggests that the model is performing very poorly in explaining the variance in the Litecoin data. An R² value of -1 indicates that the model is not explaining any of the variance in the data, and in this case, the R² value is very low, indicating that the model is not able to explain much of the variability in the Litecoin data.
- The correlation value of 0.9345 indicates a strong positive linear relationship between the predicted and actual Litecoin values. This suggests that the model is accurately capturing the underlying patterns in the data.

Overall, the evaluation metrics suggest that the RNN-LSTM model is making accurate predictions for the Litecoin values, but is performing poorly in explaining the variance in the Litecoin data. However, the strong positive correlation value suggests that the model is accurately capturing the underlying patterns in the data.

Conclusion and Recommendations:

Based on the evaluation metrics provided, it appears that the LSTM model's performance in predicting the prices of Binance Coin, Bitcoin, Cardano, and Dodecoin varies.

- For Binance Coin, the LSTM model had a high degree of variance, indicating that it is not very accurate in predicting the prices. Although the correlation between the predicted and actual values is strong, the MSE, MAE, RMSE, and R² suggest that the model is not very accurate. It is recommended to explore other models or techniques that can capture the volatility in the price movements of Binance Coin.
- For Bitcoin, the LSTM model's predictions had a relatively large error rate, and the RMSE was high compared to the actual price of Bitcoin. Although the correlation between the predicted and actual values was high, the R² score indicates that the model did not fit the data well. It is recommended to investigate whether there is any missing or additional data that can improve the model's performance.
- For Cardano, the LSTM model's predictions had an average squared difference, and the MAE and RMSE were moderate. Although the R² value indicates that the model's predictions explained less variance than simply using the mean of the actual values, the correlation between the predicted and actual values was strong. It is recommended to investigate other model architectures that can better capture the underlying patterns in the data.
- For Dodecoin, the LSTM model's predictions had relatively small errors on average, and the MSE was low. The MAE and RMSE were also low, indicating that the model was performing well. However, it is still recommended to explore other models or techniques that can further improve the model's performance and accuracy.

Overall, the LSTM model's performance varied across the different cryptocurrencies, with Dodecoin having the best performance and Binance Coin having the worst performance. The model's predictions for Bitcoin and Cardano were moderate, with room for improvement. It is also recommended to consider incorporating external factors, such as market news and sentiment analysis, to improve the accuracy of the predictions.

By including external factors such as market news and sentiment analysis, the accuracy of cryptocurrency price predictions can be enhanced significantly. Feature engineering is one of the ways to achieve this by extracting pertinent information from external sources and incorporating it as additional features in the dataset.

For example, we can extract sentiment scores from social media platforms such as Twitter or Reddit to gauge the overall sentiment towards a particular cryptocurrency. We can also include news articles and financial reports that are relevant to the cryptocurrency market as additional features. These external factors can then be combined with the existing features in our dataset (such as high, low, open, close, volume, and market cap) to improve the accuracy of our predictions.

Another way to incorporate external factors is by using a technique called "transfer learning", where we pretrain a model on a large dataset that includes external factors such as news and sentiment data, and then fine-tune the model on our cryptocurrency dataset. This can be particularly effective if we have limited data for your specific cryptocurrency dataset, as the pretraining on a large dataset can help the model learn more generalizable patterns.

In any case, it's important to carefully choose and preprocess the external factors that you include, as well as to perform rigorous testing to evaluate the impact of these factors on the model's performance. While incorporating external factors such as market news and sentiment analysis could potentially improve the accuracy of predictions for cryptocurrencies, it is not part of the scope of this particular study. Therefore, we will not be considering such analysis in our current evaluation

Conclusion

In our research, we explored six different models for time series forecasting: ARIMA, SARIMA, Exponential Smoothing, Prophet, LSTM, and Facebook's Prophet. Each model has its strengths and weaknesses, and the choice of model ultimately depends on the specific characteristics of your data and the forecasting task at hand.

For SMA model, our findings suggest that it may be useful for predicting certain currencies, such as Cardano and Dogecoin. However, the model's performance was poor for other currencies like Litecoin, Binance Coin, Ethereum, and Bitcoin, indicating that its effectiveness may be limited and dependent on various factors.

Despite its limitations, the SMA model showed a high correlation with most of the currencies, particularly Bitcoin and Ethereum. Our feature importance analysis revealed that the "Low", "High", and "Open" features are the most important for predicting SMA, with "Marketcap" being a relatively less important feature. Therefore, a simple model using only these features may be sufficient for predicting SMA for a given security.

ARIMA (AutoRegressive Integrated Moving Average) is a classic time series forecasting model that can handle both trend and seasonality in data. SARIMA (Seasonal ARIMA) extends the ARIMA model to capture seasonal patterns. These models require the data to be stationary, meaning that the mean and variance of the series should be constant over time. The performance of ARIMA and SARIMA models depends on the choice of hyperparameters, which can be determined using grid search or other optimization techniques.

Exponential Smoothing models are a traditional type of time series forecasting models that can capture patterns of seasonality and trend in data. The models operate on the assumption that future values of a series are a weighted combination of past observations, with greater emphasis placed on more recent observations. Exponential Smoothing models are capable of accommodating both additive and multiplicative components of seasonality and trend.

Facebook's Prophet is a forecasting model that combines the strengths of classic time series models and machine learning models. It uses a Bayesian approach to model trend changes, seasonality, and holiday effects in data, and can handle missing data and outliers. Prophet can also handle time series with multiple seasonality patterns and can model uncertainties in the data. However, like LSTM models, Prophet can be computationally expensive and require careful tuning of hyperparameters.

LSTM (Long Short-Term Memory) is a robust neural network structure that excels in modeling intricate patterns in time series data, including trends, seasonality, and nonlinear relationships. The model is versatile and can handle both univariate and multivariate time series data, accurately capturing long-term dependencies. Additionally, it has the ability to deal with missing data and identify and model anomalies in the data. However, owing to its high computational complexity, LSTM models necessitate precise hyperparameter tuning to achieve maximum efficiency.

To select the optimal time series forecasting model, one needs to consider the specific attributes of the data and the forecasting objectives. Traditional models such as ARIMA, SARIMA, and Exponential Smoothing are frequently utilized and can yield satisfactory results for various types of data. For intricate data exhibiting nonlinear patterns, more advanced models such as LSTM and Prophet may offer superior performance. It is recommended to experiment with multiple models and select the one that achieves the best trade-off between accuracy and computational efficiency for the particular use case.

Self Reflection

During the course of my research on time series forecasting models, I have expanded my knowledge and understanding of their strengths and limitations. While I had limited knowledge of classic models like ARIMA, SARIMA, and Exponential Smoothing at first, I am now aware of their potential and limitations for different types of data.

Furthermore, I was intrigued by the more advanced models like Prophet and LSTM that can capture complex patterns in the data. Nevertheless, I recognize that such models require careful hyperparameter tuning to achieve optimal results due to their high computational complexity.

In essence, this research has taught me that the selection of a suitable model for a particular forecasting task must involve a thoughtful consideration of the data's characteristics. It has also highlighted the value of experimentation and iteration, as it is often necessary to try out several models and adjust their parameters to achieve the best results.

In the future, I plan to continue exploring time series forecasting, stay updated on emerging advancements and techniques, and apply my knowledge in real-world applications to contribute to the development of more accurate and efficient forecasting models.

Future work

Use a larger dataset: While I used a dataset with 9408 rows, using a larger dataset can help improve the accuracy of models and provide more robust results.

Model selection and hyperparameter tuning: While I compared six different models in my study, there may be other models or variations of these models that could be worth exploring. Additionally, further tuning of hyperparameters may be required to improve the performance of these models.

Testing on different time periods and currencies: To validate the generalizability of models, it may be worth testing them on different time periods and currencies. This could help identify any limitations or biases in the models, and highlight areas for improvement.

Test on out-of-sample data: To ensure that models are robust and can generalize to new data, test them on out-of-sample data that is not used in the training or validation process.

Incorporating external factors: While my models focused primarily on using historical currency exchange rate data, there may be external factors that could impact currency exchange rates, such as political events, sentiment analysis, economic policies, or natural disasters. Incorporating such factors could potentially improve the accuracy of the models.

Reference

- 1- Hern, A. (2018, January 29). *Bitcoin and cryptocurrencies – what digital money really means for our future*. The Guardian.
<http://www.theguardian.com/technology/2018/jan/29/cryptocurrencies-bitcoin-blockchain-what-they-really-mean-for-our-future>
- 2- *How Blockchain Could Disrupt Banking*. (2022, October 18). CB Insights Research.
<https://www.cbinsights.com/research/blockchain-disrupting-banking/>
- 3- Cryptocurrency Historical Prices. (n.d.). Cryptocurrency Historical Prices | Kaggle.
<https://datasets.sudalairajkumar/cryptocurrencypricehistory>
- 4- Salman, M. K., & Ibrahim, A. A. (2020, November 1). price prediction of different cryptocurrencies using technical trade indicators and machine learning - IOPscience.
<https://doi.org/10.1088/1757-899X/928/3/032007>
- 5- Abraham, Jethin; Higdon, Daniel; Nelson, John; and Ibarra, Juan (2018) "Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis," SMU Data Science Review: Vol.1: No. 3, Article 1. <https://scholar.smu.edu/datasciencereview/vol1/iss3/1>
- 6- SARPAL, & RANTA. (n.d.). CRYPTOCURRENCY PRICE PREDICTION USING DEEP LEARNING. In Chrome-extension://Efaidnbmnnnibpcajpcglclefindmkaj/http://www.ir.juit.ac.in:8080/jspui/bitstream/123456789/3602/1/Cryptocurrency%20Price%20Prediction%20Using%20Deep%20Learning.pdf
- 7- Finding the right balance between volatility and returns. (n.d.). RBC Global Asset Management. <https://www.rbcgam.com/en/ca/learn-plan/investment-basics/understanding-the-relationship-between-volatility-and-returns/detail>
- 8- Technical Analysis: What It Is and How to Use It in Investing. (2022, March 14). Investopedia. <https://www.investopedia.com/terms/t/technicalanalysis.asp>
- 9- Gupta, K. (2021, June 25). Generate Reports Using Pandas Profiling, Deploy Using Streamlit. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/generate-reports-using-pandas-profiling-deploy-using-streamlit/>
- 10- Brems, M. (2022, January 26). A One-Stop Shop for Principal Component Analysis. Medium. <https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>
- 11- Plummer, A. (2020, September 15). Different Types of Time Series Decomposition. Medium. <https://towardsdatascience.com/different-types-of-time-series-decomposition-396c09f92693>
- 12- Prabhakaran, S. (2019, November 2). Augmented Dickey-Fuller (ADF) Test - Must Read Guide - ML+. Machine Learning Plus. <https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/>

- 13- Brownlee. (2016, August 15). *What Is Time Series Forecasting?* Machine Learning Mastery <https://machinelearningmastery.com/time-series-forecasting/>
- 14- J Hyndman , & Athanasopoulos. (n.d.). *Forecasting: Principles and Practice* (3rd ed.) <https://otexts.com/fpp3/>
- 15- Moreno. (2020, July 8). *Moving averages with Python.* <https://towardsdatascience.com/moving-averages-in-python-16170e20f6c>
- 16- Brownlee. (2017, January 9). *How to Create an ARIMA Model for Time Series Forecasting in Python.* Machine Learning Mastery. <https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>
- 17- Brownlee. (2018, August 17). *A Gentle Introduction to SARIMA for Time Series Forecasting in Python.* Machine Learning Mastery. <https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/>
- 18- Facebook / Prophet. In *Github.*, from <https://github.com/facebook/prophet>
- 19- Jose, G. V. (2019, September 27). *Time Series Forecasting with Recurrent Neural Networks.* Medium. <https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816>