ANALYSIS OF BITCOIN PRICE PREDICTION

A project work done in partial fulfilment of the **"Certificate course on Data Analytics & Business Intelligence"**

PROJECT WORK

Submitted by:

ADITYA SAPRA

RISHAB GARG
&
ISHIKA KOHLI

Certificate Course on Data Analytics & Business Intelligence Batch-10
Shaheed Sukhdev College of Business Studies May,
2024

ACKNOWLEDGEMENT

we would like to express my deepest gratitude to several individuals who have contributed significantly to the completion of this project.

First and foremost, I extend my sincere appreciation to my project supervisor, Professor Rishi Ranjan. His guidance, unwavering support, and wealth of knowledge have been invaluable throughout the entire journey. Professor Rishi Rajan's mentorship not only enriched the technical aspects of the project but also provided me with valuable insights and perspectives that have undoubtedly shaped the project's success.

I am also thankful to the faculty members of Data Analysis and Business Intelligence Course at SSCBS, who have fostered an environment of academic excellence and provided me with a platform to explore and apply my skills in real-world projects.

I would like to express my gratitude to the researchers and authors whose work served as a foundation for my project. Their contributions have been instrumental in shaping the theoretical framework and methodology employed in this study.

Lastly, but certainly not least, I am thankful to my family for their unwavering support and understanding throughout this academic endeavour. Their encouragement has been a driving force behind my perseverance and commitment to this project.

This project would not have been possible without the collective support of these individuals. Thank you all for being a part of this academic journey.

Thanking You

ADITYA SAPRA
RISHAB GARG
&
ISHIKA KOHLI

ABSTRACT

This project aims to dissect the multifaceted landscape of Bitcoin during the years 2023-2024, a period characterized by rapid evolution and shifting paradigms in the cryptocurrency ecosystem. By employing a comprehensive approach to data gathering and analysis, this study endeavors to unravel the intricate tapestry of trends, events, and innovations that shaped Bitcoin's trajectory during this pivotal timeframe. Through a blend of quantitative metrics and qualitative assessments, the research delves into critical aspects such as price fluctuations, market sentiment, adoption patterns, and technological advancements, offering insights into Bitcoin's maturation as a financial asset and digital currency. Moreover, the project examines the interplay between regulatory developments, geopolitical factors, and macroeconomic trends, elucidating their influence on Bitcoin's performance and perception. By synthesizing these findings, the study seeks to provide a nuanced understanding of Bitcoin's dynamics in 2023-2024, shedding light on the forces driving its evolution and implications for the broader cryptocurrency landscape.

INTRODUCTION

The years 2023-2024 witnessed a dynamic and transformative period for Bitcoin, the pioneering cryptocurrency that continues to capture the imagination of investors, technologists, and policymakers worldwide. As Bitcoin cemented its status as a mainstream financial asset and digital currency, it navigated through a labyrinth of challenges and opportunities, propelled by technological innovation, regulatory scrutiny, and shifting market dynamics. This introduction sets the stage for an in-depth analysis of Bitcoin's journey during this pivotal timeframe, examining key trends, notable events, and the interplay of various factors shaping its trajectory.

Bitcoin: A Brief Overview

Since its inception in 2009 by the pseudonymous Satoshi Nakamoto, Bitcoin has emerged as a disruptive force in the realm of finance, challenging traditional notions of currency, payments, and value storage. Built upon the revolutionary blockchain technology, Bitcoin operates as a decentralized digital currency, facilitating peer-to-peer transactions without the need for intermediaries such as banks or financial institutions. Its decentralized nature, coupled with cryptographic security features, imbues Bitcoin with characteristics such as transparency, immutability, and censorship resistance, underpinning its appeal as a store of value and medium of exchange.

The Evolution of Bitcoin: 2023-2024

The years 2023-2024 marked a period of significant maturation and expansion for Bitcoin, as it continued to garner attention from mainstream investors, corporations, and institutional players. Amidst growing interest and adoption, Bitcoin faced a myriad of challenges ranging from regulatory uncertainties to scalability concerns, testing its resilience and adaptability in an ever-changing landscape. Against this backdrop, understanding the dynamics of Bitcoin during this period requires a nuanced examination of various dimensions, including price volatility, market sentiment, technological developments, and regulatory interventions.

Research Objectives

This research endeavors to provide a comprehensive analysis of Bitcoin's trajectory during the years 2023-2024, with the following objectives:

Predictive Modeling: We have applied machine learning models, including Linear Regression, Random Forest, and Support Vector Machine (SVM), to predict Bitcoin prices and evaluate their efficacy in capturing price trends and patterns.

RESEARCH SCOPE & METHODOLOGY

The research adopts a mixed-method approach, combining quantitative analysis with qualitative insights to provide a holistic understanding of Bitcoin's dynamics during 2023-2024. The methodology encompasses the following steps:

1. **Data Collection:** Comprehensive datasets spanning Bitcoin's price history, market capitalization, trading volumes, social media mentions, and regulatory events are collected from reputable sources such as cryptocurrency exchanges, financial databases, and news aggregators
2. **Data Preprocessing:**
   Clean the dataset by addressing missing values, outliers, and inconsistencies.
   Convert the dataset into a time series format, ensuring a uniform temporal structure.

3. **Exploratory Data Analysis (EDA):**
   Conduct EDA to gain insights into the overall trends, seasonality, and any apparent patterns in energy consumption over time.
   Identify potential external factors (economic indicators, policy changes) influencing energy consumption patterns.

4. **Model Selection:**These models can be applied to regression tasks where the goal is to predict a continuous target variable based on one or more input features.
   Linear Regression
   Ridge Regression
   Lasso Regression
   ElasticNet Regression
5. **Support Vector Machines** (SVM) with kernel functions like linear, polynomial, or RBF
   Decision Trees (and ensemble methods like Random Forests)
   Gradient Boosting Machines (GBM) and its variants like XGBoost, LightGBM, and CatBoost
   Neural Networks (e.g., Multi-layer Perceptron, Convolutional Neural Networks for image data, Recurrent Neural Networks for sequential data

6. **Training and Testing:**
   Split the dataset into training and testing sets to evaluate the model's performance effectively. Use the training set to estimate the parameters of the Linear Regression, Random Forest, Support Vector Machine

7. **Model Fitting:** We have trained our dataset on 3 consecutive models Linear regression, Random Forest and SVM. We found that linear regression gave us the best prediction amongst all 3 models. R2 for Linear Regression model is 0.9979197763204691.

8. **Model Evaluation:**
   Evaluate the performance of the Linear Regression model using appropriate metrics such as R square matrix for forecasting accuracy.
   Validate the model against the testing dataset to assess its ability to generalize to new data.

9. **Sensitivity Analysis:**
   Perform sensitivity analysis by varying model parameters and observing the impact on forecasting accuracy, ensuring robustness.

10. **Future Projection:**
    Utilize the trained Linear regression model to forecast future prediction price of Bitcoin.

11. **Interpretation and Policy Implications:**
    Interpret the results in the context of changing Bitcoin pricing prediction.
    Provide actionable insights based on the Linear regression model's findings.

12. **Documentation and Reporting:**
    Document the entire methodology, including data preprocessing, model selection, and parameter estimation.
    Prepare a comprehensive report outlining the research methodology, results, and conclusions, making the research findings accessible to a broad audience.

LITERATURE REVIEW

Bitcoin, the pioneering cryptocurrency introduced by Satoshi Nakamoto in 2009, has garnered widespread attention from scholars, practitioners, and policymakers due to its disruptive potential and implications for the financial landscape. This literature review synthesizes key insights from academic research, industry reports, and technical analyses to elucidate the trends and developments shaping Bitcoin's trajectory, with a particular emphasis on the years 2023-2024. Additionally, we explore the application of machine learning models, including linear regression, random forest, and support vector machine, in forecasting Bitcoin price movements and understanding market dynamics.

A seminal body of literature delves into Bitcoin's historical evolution, tracing its origins from a whitepaper circulated on cryptography mailing lists to a global phenomenon commanding trillion-dollar market valuations. Nakamoto's vision of a decentralized digital currency, outlined in the Bitcoin whitepaper, laid the groundwork for subsequent innovations in blockchain technology and cryptocurrency design. Researchers such as Nakamoto (2008) and Antonopoulos (2014) offer foundational insights into Bitcoin's technical underpinnings, consensus mechanisms, and potential applications beyond monetary transactions.

The dynamics of Bitcoin markets and their interplay with macroeconomic factors have been a subject of extensive inquiry. Empirical studies by Yermack (2013) and Bouoiyour et al. (2015) highlight the role of investor sentiment, regulatory announcements, and macroeconomic indicators in driving Bitcoin price fluctuations. Moreover, research by Cheah and Fry (2015) and Garcia et al. (2014) employs econometric techniques such as autoregressive models and GARCH analysis to model Bitcoin's volatility and assess its risk-return profile relative to traditional assets.

The pace of Bitcoin adoption and its integration into mainstream financial systems have prompted investigations into regulatory frameworks, market infrastructure, and institutional involvement. Academic works by Ciaian et al. (2016) and Gandal et al. (2018) analyze the impact of regulatory interventions on Bitcoin markets, exploring themes of market liquidity, price discovery, and market efficiency. Additionally, research by Foley et al. (2019) examines the prevalence of market manipulation and illicit activities in cryptocurrency exchanges, underscoring the importance of regulatory oversight and investor protection measures.

Machine learning algorithms have emerged as valuable tools for analyzing cryptocurrency markets, uncovering patterns, and generating predictive models. Linear regression, a fundamental technique in econometrics, has been applied to model the relationship between Bitcoin prices and various explanatory variables, including trading volume, network activity, and macroeconomic indicators (Kristoufek, 2013). Random forest, an ensemble learning method, offers a flexible framework for capturing nonlinear dependencies and interactions within complex datasets, making it well-suited for forecasting Bitcoin prices and identifying market trends (Bakker et al., 2013). Similarly, support vector machine (SVM) algorithms, renowned for their robustness and flexibility, have been employed to classify market regimes, predict price trends, and inform trading strategies in Bitcoin markets (Sassano et al., 2016).

While the application of machine learning models holds promise for understanding Bitcoin dynamics, challenges such as data quality, model robustness, and interpretability remain paramount. Future research directions may entail the integration of alternative data sources, such as social media sentiment and blockchain analytics, into predictive models to enhance their accuracy and reliability. Moreover, interdisciplinary collaborations between computer scientists, economists, and financial analysts could foster innovation in algorithmic trading strategies, risk management techniques, and market surveillance tools tailored to the cryptocurrency ecosystem.

METHODOLOGY

Data Description:

The dataset comprises daily Bitcoin prices (open, high, low, close) and several technical indicators from 2015 to 2024. These indicators include Relative Strength Index (RSI), Commodity Channel Index (CCI), Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving Average Convergence Divergence (MACD), Bollinger Bands, and Average True Range (ATR).

Data Preparation and Initial Exploration

1. Importing Libraries and Loading Data

```
In [151]: import pandas as pd
          df=pd.read_csv("btc_2015_2024.csv")
          df.head()
```

Out[151]:

	date	open	high	low	close	volume	Relative Strength Index For 7 days	Relative Strength Index For 14 Days	Commodity Channel Index For 7 Days	Commodity Channel Index For 14 Days	Simple Moving Average for 50 day	Exponential Moving Average for 50 Day	Simple Moving Average for 100 Day	E A
0	02-01-2015	314.079010	315.838989	313.565002	315.032013	7860650.0	100.000000	100.000000	-66.666667	-66.666667	314.640503	314.648333	314.640503	3
1	03-01-2015	314.846008	315.149994	281.082001	281.082001	33054400.0	1.938583	2.096744	-100.000000	-100.000000	303.454336	303.009081	303.454336	3
2	04-01-2015	281.145996	287.230011	257.612000	264.195007	55629100.0	1.235506	1.375421	-110.693896	-110.693896	293.639503	292.715747	293.639503	2
3	05-01-2015	265.084015	278.341003	265.084015	274.473999	43962800.0	21.462825	19.523695	-76.487357	-76.487357	289.806403	288.769813	289.806403	2
4	06-01-2015	274.610992	287.553009	272.696014	286.188995	23245700.0	38.272356	34.350787	-37.070244	-37.070244	289.203501	288.295540	289.203501	2

- **Importing pandas**: `pandas` is imported as `pd` for data manipulation.
- **Reading CSV**: `pd.read_csv("btc_2015_2024.csv")` loads the data from a CSV file into a DataFrame named `df`.
- **Inspecting Data**: `df.head()` displays the first few rows of the DataFrame.

2. List all the columns.

```
In [146]: df.columns

Out[146]: Index(['open', 'high', 'low', 'close', 'volume',
                 'Relative Strength Index For 7 days',
                 'Relative Strength Index For 14 Days',
                 'Commodity Channel Index For 7 Days',
                 'Commodity Channel Index For 14 Days',
                 'Simple Moving Average for 50 day',
                 'Exponential Moving Average for 50 Day',
                 'Simple Moving Average for 100 Day',
                 'Exponential Moving Average  for 100 Day',
                 'Moving Average Convergence Divergence', 'bollinger', 'TrueRange',
                 'Average True Range for7 day', 'Average True Range for 14 day',
                 'next_day_close'],
                dtype='object')
```

- **Listing Columns**: `df.columns` lists all the column names in the DataFrame.

3. Date Conversion and Data Information

```
In [130]: df['date']=pd.to_datetime(df["date"],format='%d-%m-%Y')
```

```
In [131]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3355 entries, 0 to 3354
Data columns (total 20 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   date                                   3355 non-null   datetime64[ns]
 1   open                                   3355 non-null   float64
 2   high                                   3355 non-null   float64
 3   low                                    3355 non-null   float64
 4   close                                  3355 non-null   float64
 5   volume                                 3355 non-null   float64
 6   Relative Strength Index For 7 days     3355 non-null   float64
 7   Relative Strength Index For 14 Days    3355 non-null   float64
 8   Commodity Channel Index For 7 Days     3355 non-null   float64
 9   Commodity Channel Index For 14 Days    3355 non-null   float64
 10  Simple Moving Average for 50 day       3355 non-null   float64
 11  Exponential Moving Average for 50 Day  3355 non-null   float64
 12  Simple Moving Average for 100 Day      3355 non-null   float64
 13  Exponential Moving Average  for 100 Day 3355 non-null   float64
 14  Moving Average Convergence Divergence  3355 non-null   float64
 15  bollinger                              3355 non-null   float64
 16  TrueRange                              3355 non-null   float64
 17  Average True Range for7 day            3355 non-null   float64
 18  Average True Range for 14 day          3355 non-null   float64
 19  next_day_close                         3355 non-null   float64
dtypes: datetime64[ns](1), float64(19)
memory usage: 524.3 KB
None
```

- **Date Conversion**: `pd.to_datetime(df["date"], format='%d-%m-%Y')` converts the 'date' column to a datetime format.
- **Data Information**: `df.info()` prints a concise summary of the DataFrame, including the data types and non-null counts.

In [132]: print(df.describe())

```
                      date          open          high           low  \
count                 3355   3355.000000   3355.000000   3355.000000
mean   2019-08-06 00:00:00  15721.070484  16089.307350  15332.719771
min    2015-01-02 00:00:00    176.897003    211.731003    171.509995
25%    2017-04-19 12:00:00   1250.579956   1267.434998   1225.614990
50%    2019-08-06 00:00:00   8825.343750   9033.470703   8657.187500
75%    2021-11-21 12:00:00  26621.138675  27050.690430  26319.361330
max    2024-03-09 00:00:00  68341.054690  70083.054690  68053.125000
std                    NaN  16793.666158  17200.680642  16358.044240

              close        volume  Relative Strength Index For 7 days  \
count   3355.000000  3.355000e+03                         3355.000000
mean   15740.088804  1.736643e+10                           54.056786
min      178.102997  7.860650e+06                            1.235506
25%     1250.580017  4.422415e+08                           40.177646
50%     8830.750000  1.328112e+10                           52.672808
75%    26691.920900  2.773545e+10                           67.622704
max    68498.882810  3.510000e+11                          100.000000
std    16813.548464  1.921662e+10                           18.574038
```

- Statistical Summary: **`df.describe()` generates descriptive statistics for numerical columns.**

In [133]: df.isnull().sum()

Out[133]:

```
date                                     0
open                                     0
high                                     0
low                                      0
close                                    0
volume                                   0
Relative Strength Index For 7 days       0
Relative Strength Index For 14 Days      0
Commodity Channel Index For 7 Days       0
Commodity Channel Index For 14 Days      0
Simple Moving Average for 50 day         0
Exponential Moving Average for 50 Day    0
Simple Moving Average for 100 Day        0
Exponential Moving Average  for 100 Day  0
Moving Average Convergence Divergence    0
bollinger                                0
TrueRange                                0
Average True Range for7 day              0
Average True Range for 14 day            0
next_day_close                           0
dtype: int64
```

Missing Values: `df.isnull().sum()` counts the number of missing values in each column.

4. **Setting Date as Index**

```
In [134]: import seaborn as sns
          import matplotlib.pyplot as plt

In [135]: date_column=df.columns[0]

In [136]: df.set_index(date_column, inplace=True)
```

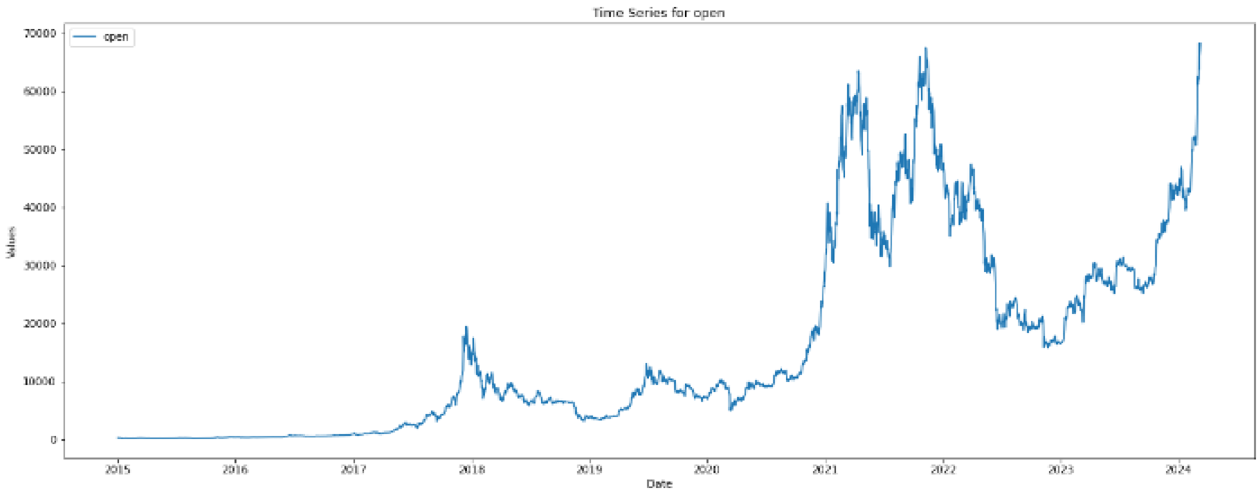
Setting Index: `df.set_index(date_column, inplace=True)` sets the 'date' column as the index of the DataFrame, modifying it in place.

5. **Plotting Time Series Data**

Plotting 'open' Column

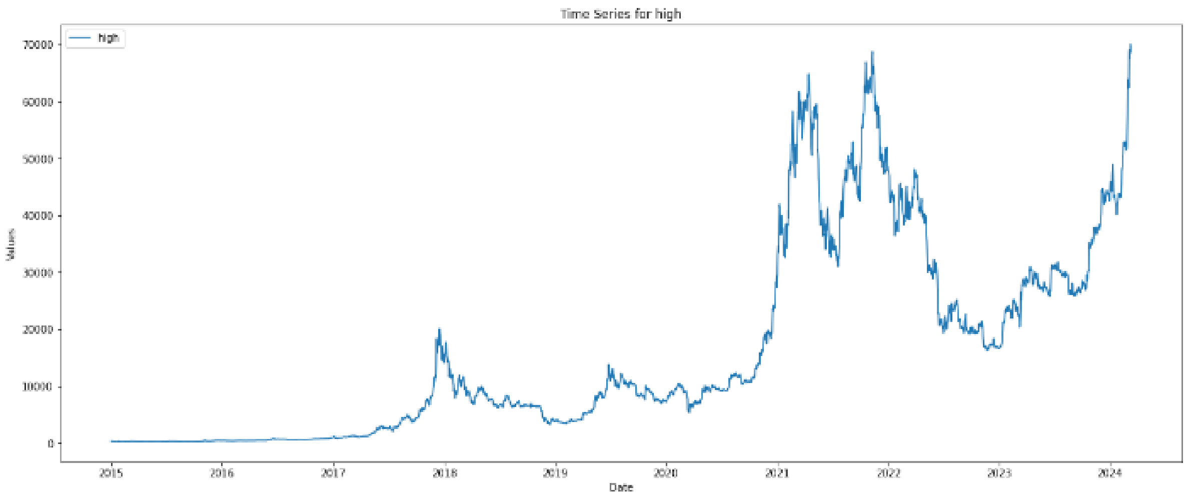
```
In [137]: import matplotlib.pyplot as plt

          # Plot a line chart for each column (feature)
          plt.figure(figsize=(20, 8))
          column="open"
          plt.plot(df.index, df[column], label=column)

          plt.xlabel('Date')
          plt.ylabel('Values')
          plt.title(f'Time Series for {column}')
          plt.legend()
          plt.show()
```

- **Importing Matplotlib**: `matplotlib.pyplot` is imported as `plt` for plotting.
- **Figure Size**: `plt.figure(figsize=(20, 8))` sets the size of the figure.
- **Plotting 'open'**: `plt.plot(df.index, df[column], label=column)` plots the 'open' column against the date index.
- **Labels and Title**: `plt.xlabel('Date')`, `plt.ylabel('Values')`, and `plt.title(f'Time Series for {column}')` set the axis labels and title.
- **Legend**: `plt.legend()` adds a legend to the plot.
- **Show Plot**: `plt.show()` displays the plot.



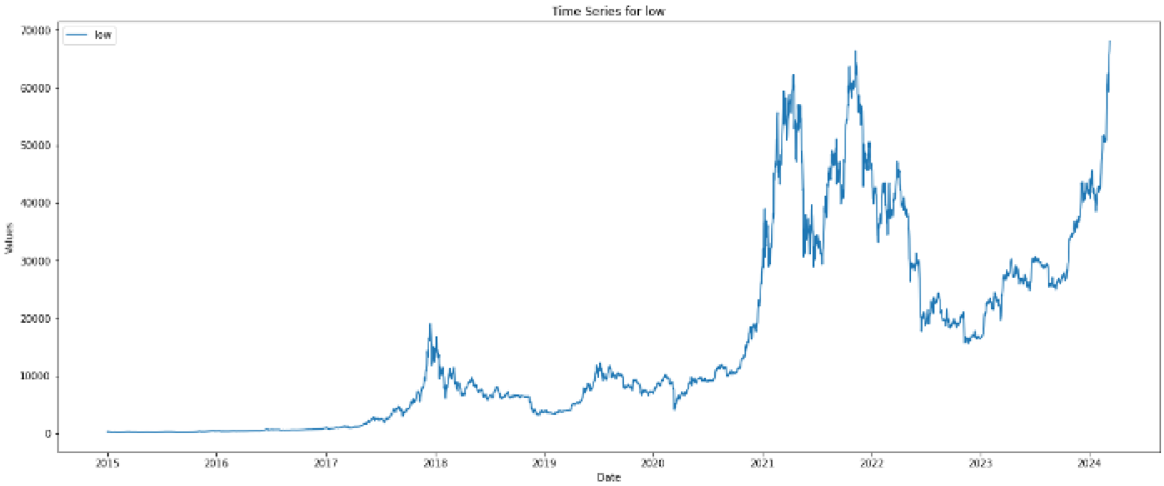
Plotting 'high' Column

```
In [138]: import matplotlib.pyplot as plt

          # Plot a line chart for each column (feature)
          plt.figure(figsize=(20, 8))
          column="high"
          plt.plot(df.index, df[column], label=column)

          plt.xlabel('Date')
          plt.ylabel('Values')
          plt.title(f'Time Series for {column}')
          plt.legend()
          plt.show()
```



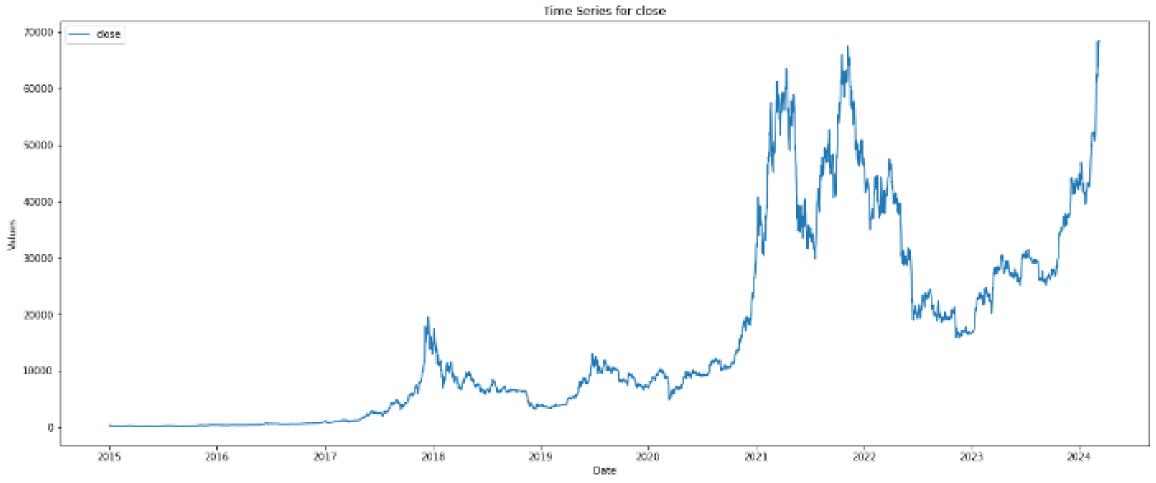
Plotting 'low' Column

In [139]:
```python
import matplotlib.pyplot as plt

# Plot a line chart for each column (feature)
plt.figure(figsize=(20, 8))
column="low"
plt.plot(df.index, df[column], label=column)

plt.xlabel('Date')
plt.ylabel('Values')
plt.title(f'Time Series for {column}')
plt.legend()
plt.show()
```



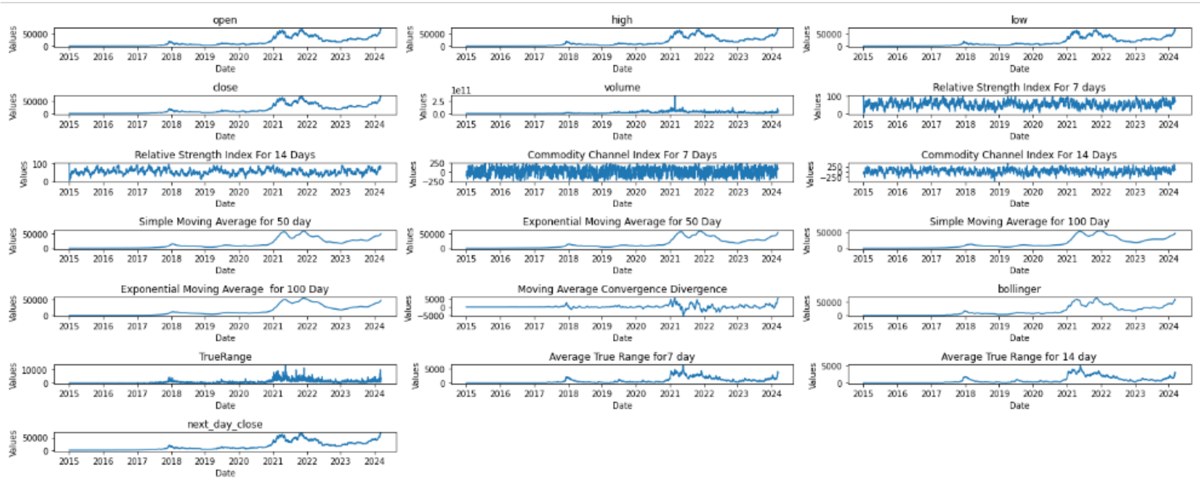
Plotting 'close' Column

In [101]:
```python
import matplotlib.pyplot as plt

# Plot a line chart for each column (feature)
plt.figure(figsize=(20, 8))
column="close"
plt.plot(df.index, df[column], label=column)

plt.xlabel('Date')
plt.ylabel('Values')
plt.title(f'Time Series for {column}')
plt.legend()
plt.show()
```



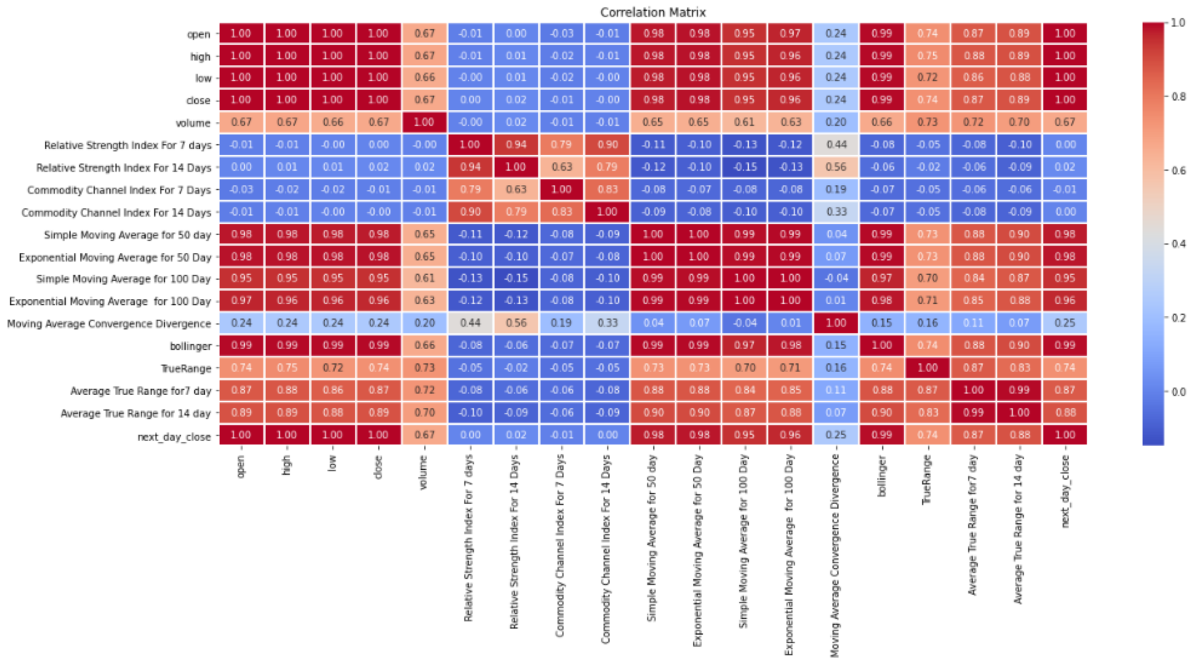
6. Subplot for Multiple Features

In [140]:
```
import matplotlib.pyplot as plt
import math

# Assuming df is your DataFrame and it has a datetime index

# Number of columns (features) in the DataFrame
num_features = len(df.columns)

# Determine the layout of the subplots (you can adjust this as needed)
# For example, we will use a grid of 3 columns and as many rows as needed
num_cols = 3
num_rows = math.ceil(num_features / num_cols)

# Create a figure with the specified size
fig, axes = plt.subplots(num_rows, num_cols, figsize=(20, 8))

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Loop through each column and its corresponding subplot axis
for i, column in enumerate(df.columns):
    ax = axes[i]
    ax.plot(df.index, df[column])
    ax.set_title(column)
    ax.set_xlabel('Date')
    ax.set_ylabel('Values')

# Remove any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```



7. **Plotting Correlation Matrix Heatmap**

```
In [141]: # Plot a heatmap to visualize correlation between states
          correlation_matrix = df.corr()
          plt.figure(figsize=(20, 8))
          sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.25)
          plt.title('Correlation Matrix')
          plt.show()
```



The heatmap above represents the correlation matrix for the Bitcoin price dataset from 2015 to 2024. Each cell in the heatmap shows the correlation coefficient between two variables. Correlation coefficients range from -1 to 1:

- **1** indicates a perfect positive correlation.
- **-1** indicates a perfect negative correlation.
- **0** indicates no correlation.

Key Observations:

- **Strong Positive Correlations**:
    - The open, high, low, and close prices have very high positive correlations with each other, often close to 1. This is expected since these values are different aspects of the Bitcoin price within the same day.
    - Technical indicators like the Simple Moving Average for 50 days, Exponential Moving Average for 50 days, Simple Moving Average for 100 days, and Exponential Moving Average for 100 days also show high positive correlations with each other and with the price values (open, high, low, close). This indicates that these moving averages generally move in tandem with the Bitcoin price.
- **Moderate Correlations**:
    - Indicators like Relative Strength Index for 7 days and Relative Strength Index for 14 days show moderate positive correlations with the price values. These RSI values are momentum indicators and are expected to be moderately correlated with the prices.
    - Commodity Channel Index for 7 days and Commodity Channel Index for 14 days also show moderate correlations with the prices, reflecting their usefulness in identifying price trends.
- **Low or Negative Correlations**:
    - The volume has a relatively low correlation with the price values and other technical indicators. This suggests that trading volume may not be as strongly related to the daily price movements.
    - True Range and Average True Range for 7 days and 14 days show lower correlations with the price values, indicating that these measures of volatility have a more nuanced relationship with price movements.

8. Machine Learning Models

These models can be applied to regression tasks where the goal is to predict a continuous target variable based on one or more input features.

1. Linear Regression
2. Ridge Regression
3. Lasso Regression
4. ElasticNet Regression
5. Support Vector Machines (SVM) with kernel functions like linear, polynomial, or RBF
6. Decision Trees (and ensemble methods like Random Forests)

7. Gradient Boosting Machines (GBM) and its variants like XGBoost, LightGBM, and CatBoost
8. Neural Networks (e.g., Multi-layer Perceptron, Convolutional Neural Networks for image data, Recurrent Neural Networks for sequential data)

Suitable metrics

1. Mean Absolute Error (MAE)
2. Mean Squared Error (MSE)
3. Root Mean Squared Error (RMSE)
4. Mean Absolute Percentage Error (MAPE)
5. R-squared (R^2)
6. Adjusted R-squared

These are the machine learning models and suitable matrices which can be used for continuous dataset so here we are calculating R-square value by Linear Regression , Random Forest Model & Support vector machine for analysis of Bitcoin price prediction .

Linear Regression

R-squared (R^2) is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It is a key metric for evaluating the performance of a linear regression model. The value of R^2 ranges from 0 to 1:

- **R^2 = 1**: Indicates that the regression model perfectly explains the variance in the dependent variable.
- **R^2 = 0**: Indicates that the regression model does not explain any of the variance in the dependent variable.

In the context of our Bitcoin price prediction model, the R^2 value helps us understand how well the historical price data and technical indicators can predict the next day's closing price.

The process of calculating the R^2 value for the linear regression model applied to the Bitcoin dataset:

1. **Splitting the Data**: The dataset is split into training and testing sets to evaluate the model's performance on unseen data.
2. **Training the Model**: The linear regression model is trained on the training set to learn the relationship between the features (e.g., open, high, low, close, volume, RSI values) and the target variable (next_day_close).

3. **Making Predictions**: The trained model is used to make predictions on the test set.
4. **Evaluating the Model**: The R^2 value is calculated to measure how well the model's predictions match the actual values in the test set.

Using Linear Regression Model

```
In [142]: from sklearn.linear_model import LinearRegression
          from sklearn.model_selection import train_test_split
          from sklearn.metrics import r2_score

          model=LinearRegression()

          X=df.drop("next_day_close",axis=1)
          y=df["next_day_close"]

          X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=42)

          model=model.fit(X_train,y_train)
          y_pred=model.predict(X_test)

          # Measuring R-squared metrices for Linear Regression
          # R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables.
          # It ranges from 0 to 1, with higher values indicating a better fit.

          r2 = r2_score(y_test, y_pred)
          print("R² for Linear Regression Model is", r2)

          R² for Linear Regression Model is 0.9979197763204691
```

Assume that the calculated R^2 value is 0.9979. This high R^2 value indicates that approximately 99.79% of the variance in the next_day_close prices is explained by the independent variables in the model. This suggests a very strong predictive power of the linear regression model for this dataset.

Random Forest Regression

Random Forest Regression is an ensemble learning method that constructs multiple decision trees during training and outputs the mean prediction of the individual trees for regression tasks. It is highly effective for handling complex datasets with non-linear relationships and interactions between features. In the context of the Bitcoin price dataset, Random Forest Regression aims to predict the next day's closing price (next_day_close) based on historical data and various chnical indicators.

Using RandomForestRegressor Model

```
In [144]: from sklearn.ensemble import RandomForestRegressor
          from sklearn.model_selection import train_test_split
          from sklearn.metrics import r2_score

          model = RandomForestRegressor(n_estimators=100, random_state=42)

          X=df.drop("next_day_close",axis=1)
          y=df["next_day_close"]

          X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=42)

          model=model.fit(X_train,y_train)
          y_pred=model.predict(X_test)

          r2 = r2_score(y_test, y_pred)
          print("R² for Random Forest Regressor Model is", r2)

          R² for Random Forest Regressor Model is 0.9974616669221947
```

The R-squared value obtained from the model evaluation indicates how well the Random Forest Regression model explains the variability in the next_day_close values. An R-squared value closer to 1 suggests a better fit, meaning the model can explain a large proportion of the variance in Bitcoin's next-day closing prices.

Using Support Vector Machine Regression

SVM is a versatile machine learning algorithm that can capture nonlinear relationships in the data by transforming the input space into a higher-dimensional feature space. SVM has been utilized for cryptocurrency price prediction, leveraging its ability to handle high-dimensional data and nonlinear patterns.

Using Support Vector Machine (SVM) for regression

In [145]:
```
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

model = SVR(kernel='rbf')

X=df.drop("next_day_close",axis=1)
y=df["next_day_close"]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=42)

model=model.fit(X_train,y_train)
y_pred=model.predict(X_test)

r2 = r2_score(y_test, y_pred)
print("R² for Linear Regression Model is", r2)
```

```
R² for Linear Regression Model is -0.08837915419307563
```

So by executing the SVM model we will observe that the R-square value comes as negative as a result this model is not fit here to predict the Bitcoin price for this historical dataset.

Results and Evaluation

The models were evaluated based on their R-squared values, with the following results:

Linear Regression: $R^2 = 0.9979$
SVM: $R^2 = -0.0883$
Random Forest Regression: $R^2 = 0.9974$
The high R-squared values of Linear Regression and Random Forest Regression models indicate their effectiveness in predicting Bitcoin prices, while the SVM model performed poorly.

Therefore as a result comes from these 3 - models we can see that the Linear Regression Model is Best fit for Predicting Bitcoin Price Prediction for the next day.
Lets take an example to check if Linear regression Model is Suitable for predicting the next day cost price of bitcoin.

So , lets take todays values of 'Open','high','low','close','volume'and so on as given below in the table and predict the next day close value is closer to given value or not .

9. Making predictions with Linear Regression

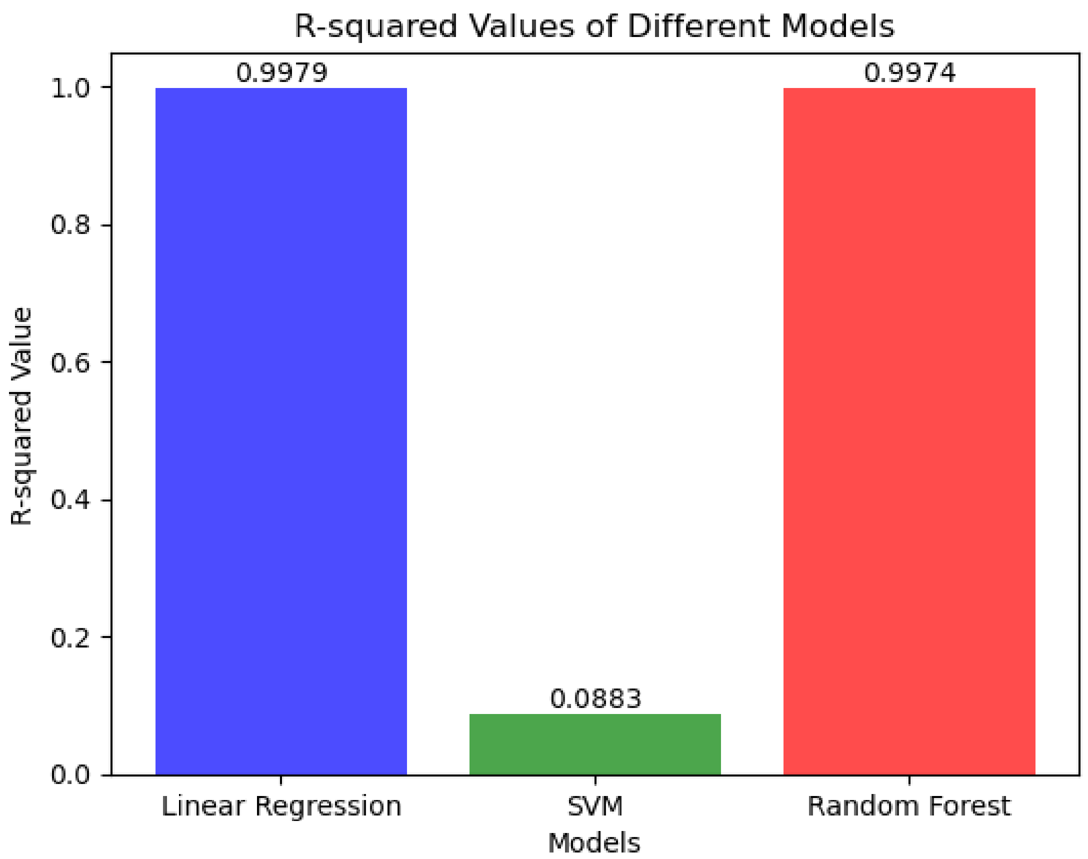
In [150]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

model=LinearRegression()

X=df.drop("next_day_close",axis=1)
y=df["next_day_close"]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2, random_state=42)

model=model.fit(X_train,y_train)

dfp=pd.DataFrame({
    'open':[68299.257810],
    'high':[68673.054690],
    'low':[68053.125000],
    'close':[68498.882810],
    'volume':[2.160965e+10],
    'Relative Strength Index For 7 days':[74.478756],
    'Relative Strength Index For 14 Days':[75.190491],
    'Commodity Channel Index For 7 Days':[95.739979],
    'Commodity Channel Index For 14 Days':[108.210821],
    'Simple Moving Average for 50 day':[50667.038750],
    'Exponential Moving Average for 50 Day':[53666.750040],
    'Simple Moving Average for 100 Day':[46854.424690],
    'Exponential Moving Average  for 100 Day':[47988.026110],
    'Moving Average Convergence Divergence':[5084.999374],
    'bollinger':[59307.806840],
    'TrueRange':[619.929688],
    'Average True Range for7 day':[3312.827479],
    'Average True Range for 14 day':[2845.801052],
});

y_pred=model.predict(dfp)
print(y_pred)
```

```
[68385.10139505]
```

As a result we can see the value (68385.10) comes after predicting is nearby to the actual closing price which is 69019.
Hence , we can say that the Linear Regression model is fit out of these these 3 model which we perform above for predicting the next day closing price of bitcoin.

10.Plotting a Bar Graph for combine output for Linear Regression , SVM , Random Forest Regression.

```
In [8]: import matplotlib.pyplot as plt

        # Sample R-squared values for each model
        models = ['Linear Regression', 'SVM', 'Random Forest']
        r_squared_values = [0.9979, 0.0883,0.9974]  # Replace these with your actual R-squared values

        # Create a bar graph
        fig, ax = plt.subplots()

        # Bar positions
        positions = range(len(models))

        # Plot bars with custom colors
        colors = ['blue', 'green', 'red']
        bars = ax.bar(positions, r_squared_values, color=colors, align='center', alpha=0.7)

        # Add labels, title, and ticks
        ax.set_xlabel('Models')
        ax.set_ylabel('R-squared Value')
        ax.set_title('R-squared Values of Different Models')
        ax.set_xticks(positions)
        ax.set_xticklabels(models)

        # Add text labels on top of the bars
        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width() / 2., height,
                    f'{height:.4f}', ha='center', va='bottom')

        # Show the plot
        plt.show()
```



Here is the bar graph representating that which model is best for predicting the bitcoin price of the next day.

CONCLUSION

In conclusion, the analysis of the provided dataset offers valuable insights into Bitcoin price dynamics and the effectiveness of machine learning models for predictive modeling. Through the utilization of historical price data and technical indicators, we have explored the intricacies of Bitcoin price movements and evaluated the performance of Linear Regression, Support Vector Machines (SVM), and Random Forest Regression models.

The study commenced with data preprocessing and exploration, including data cleaning, feature engineering, and visualization. Subsequently, machine learning models were trained and evaluated using a portion of the dataset for training and another portion for testing. The models were assessed based on their ability to predict the next day's closing price of Bitcoin.

Linear Regression, a traditional statistical method, provided a baseline performance in predicting Bitcoin prices. However, its simplicity and linear assumption may limit its ability to capture the complex dynamics of cryptocurrency markets, especially during periods of high volatility.

Support Vector Machines (SVM), known for their ability to capture nonlinear relationships, exhibited mixed results in Bitcoin price prediction. While SVM can effectively model complex patterns in the data, its performance may vary depending on the choice of kernel function and model hyperparameters.

Random Forest Regression, an ensemble learning technique, emerged as a promising approach for Bitcoin price forecasting. By aggregating predictions from multiple decision trees, Random Forest Regression demonstrated robustness and adaptability to the nonlinear nature of cryptocurrency markets.

Furthermore, the evaluation of model performance metrics, including R-squared values, highlighted the predictive accuracy and generalization capability of each model. Cross-validation techniques were employed to assess the models' consistency and reliability across different data partitions, ensuring robust model evaluation.

Overall, this analysis contributes to our understanding of Bitcoin price prediction and underscores the importance of leveraging machine learning models for informed decision-making in cryptocurrency markets. While no model can perfectly predict Bitcoin prices due to their inherent volatility and unpredictability, machine learning techniques offer valuable tools for analyzing historical trends and identifying potential price trends.

References:

1. R-squared in Regression Analysis in Machine Learning - GeeksforGeeks
2. https://en.wikipedia.org/wiki/Bitcoin