Time Series Cryptocurrency (Bitcoin)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = "Bitcoin Historical Data.csv"
df = pd.read_csv(file_path)

# Display basic information and first few rows
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3669 entries, 0 to 3668
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Date      3669 non-null   object
 1   Price     3669 non-null   object
 2   Open      3669 non-null   object
 3   High      3669 non-null   object
 4   Low       3669 non-null   object
 5   Vol.      3669 non-null   object
 6   Change %  3669 non-null   object
dtypes: object(7)
memory usage: 200.8+ KB
```

```
df.head()
```

```
           Date     Price      Open      High       Low     Vol.  Change %
0  Aug 02, 2020  11,105.8  11,802.6  12,061.1  10,730.7  698.62K   -5.91%
1  Aug 01, 2020  11,803.1  11,333.2  11,847.7  11,226.1  611.47K    4.14%
2  Jul 31, 2020  11,333.4  11,096.5  11,434.8  10,964.6  530.95K    2.14%
3  Jul 30, 2020  11,096.2  11,105.8  11,164.4  10,861.6  501.14K   -0.09%
4  Jul 29, 2020  11,105.9  10,908.4  11,336.5  10,771.8  576.83K    1.81%
```

```
df.tail()
```

```
              Date Price Open High  Low   Vol. Change %
3664  Jul 22, 2010   0.1  0.1  0.1  0.1  2.16K    0.00%
3665  Jul 21, 2010   0.1  0.1  0.1  0.1  0.58K    0.00%
```

```
3666  Jul 20, 2010   0.1  0.1  0.1  0.1  0.26K    0.00%
3667  Jul 19, 2010   0.1  0.1  0.1  0.1  0.57K    0.00%
3668  Jul 18, 2010   0.1  0.0  0.1  0.1  0.08K    0.00%
```

```
df.describe()
```

```
                Date Price  Open  High   Low   Vol. Change %
count           3669  3669  3669  3669  3669   3669     3669
unique          3669  2718  2712  2708  2712   3275     1391
top     Jul 18, 2010   0.1   0.1   0.1   0.1  1.05M    0.00%
freq               1   100   100    97   101      6      425
```

```
df.shape
```

```
(3669, 7)
```

Handle Missing Values

```
# Clean and convert data types
def parse_price(x):
    return float(x.replace(',', ''))

def parse_volume(x):
    if x[-1] == 'K':
        return float(x[:-1].replace(',', '')) * 1e3
    elif x[-1] == 'M':
        return float(x[:-1].replace(',', '')) * 1e6
    elif x == '-':
        return np.nan
    else:
        return float(x.replace(',', ''))

def parse_percentage(x):
    return float(x.replace('%', '').replace(',', ''))

# Apply parsing functions
df['Date'] = pd.to_datetime(df['Date'])
df['Price'] = df['Price'].apply(parse_price)
df['Open'] = df['Open'].apply(parse_price)
df['High'] = df['High'].apply(parse_price)
df['Low'] = df['Low'].apply(parse_price)
df['Vol.'] = df['Vol.'].apply(parse_volume)
df['Change %'] = df['Change %'].apply(parse_percentage)

# Sort by date ascending
df.sort_values(by='Date', inplace=True)

# Check for missing values
missing_values = df.isnull().sum()

df.dtypes, missing_values
```

```
(Date          datetime64[ns]
 Price                float64
 Open                 float64
 High                 float64
 Low                  float64
 Vol.                 float64
 Change %             float64
 dtype: object,
 Date        0
 Price       0
 Open        0
 High        0
 Low         0
 Vol.        6
 Change %    0
 dtype: int64)
```

```
# Handle missing values using forward fill
df['Vol.'] = df['Vol.'].fillna(method='ffill')

# Confirm missing values are handled
df.isnull().sum()
```

```
C:\Users\Windows\AppData\Local\Temp\ipykernel_8124\23906020.py:2:
FutureWarning: Series.fillna with 'method' is deprecated and will
raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df['Vol.'] = df['Vol.'].fillna(method='ffill')
```

```
Date        0
Price       0
Open        0
High        0
Low         0
Vol.        0
Change %    0
dtype: int64
```

Visualization

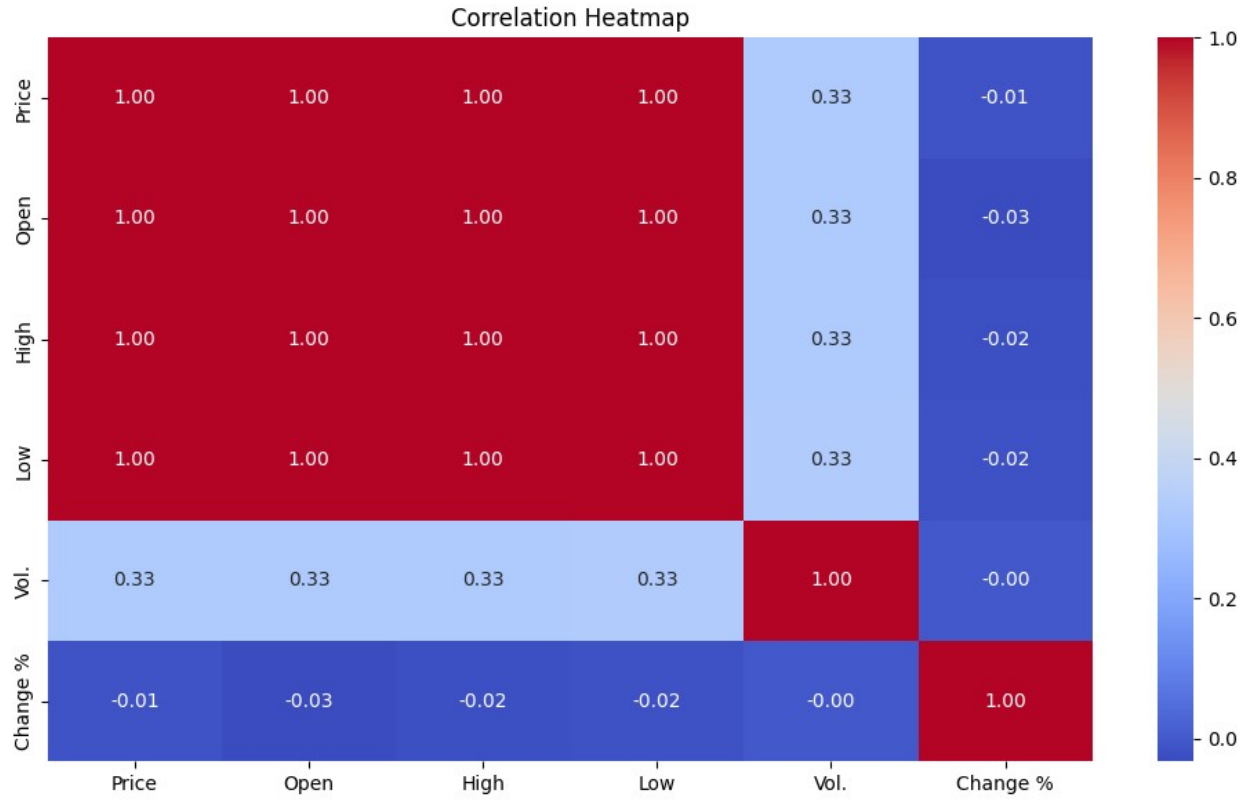
```
# Summary statistics
summary_stats = df.describe()

# Correlation matrix
correlation_matrix = df[['Price', 'Open', 'High', 'Low', 'Vol.',
'Change %']].corr()

# Plot correlation heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title("Correlation Heatmap")
```

```
plt.tight_layout()
plt.show()

summary_stats
```

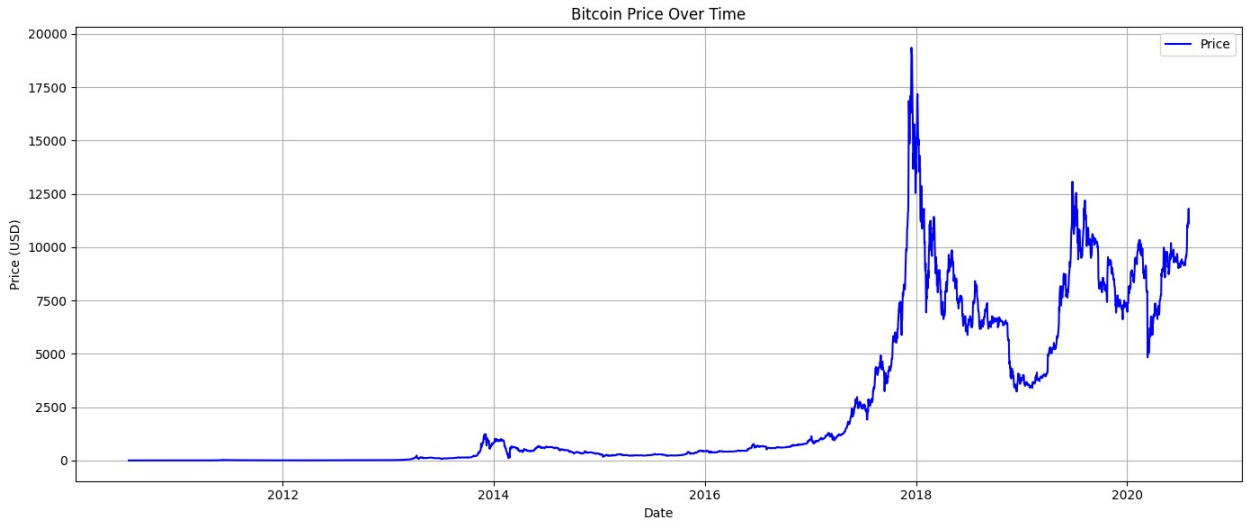


	Date	Price	Open	High \
count	3669	3669.000000	3669.000000	3669.000000
mean	2015-07-26 00:00:00	2544.577651	2541.522131	2614.517989
min	2010-07-18 00:00:00	0.100000	0.000000	0.100000
25%	2013-01-20 00:00:00	18.700000	18.500000	19.800000
50%	2015-07-26 00:00:00	448.100000	447.700000	456.500000
75%	2018-01-28 00:00:00	4356.000000	4352.300000	4440.100000
max	2020-08-02 00:00:00	19345.500000	19346.600000	19870.600000
std	NaN	3689.539283	3687.016195	3800.658515

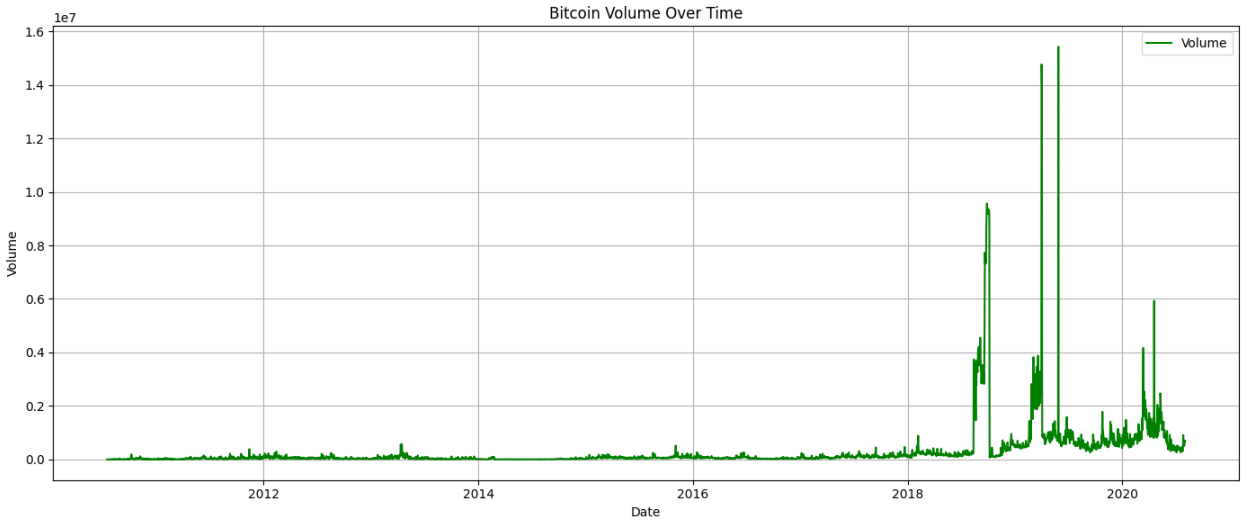
Low	Vol.	Change %

```
count   3669.000000  3.669000e+03  3669.000000
mean    2460.348378  2.861828e+05     0.497114
min        0.000000  8.000000e+01   -57.210000
25%       17.500000  2.461000e+04    -1.120000
50%      439.500000  5.817000e+04     0.000000
75%     4185.300000  1.749000e+05     1.840000
max    18750.900000  1.543000e+07   336.840000
std     3552.725714  8.701963e+05     8.032196
```

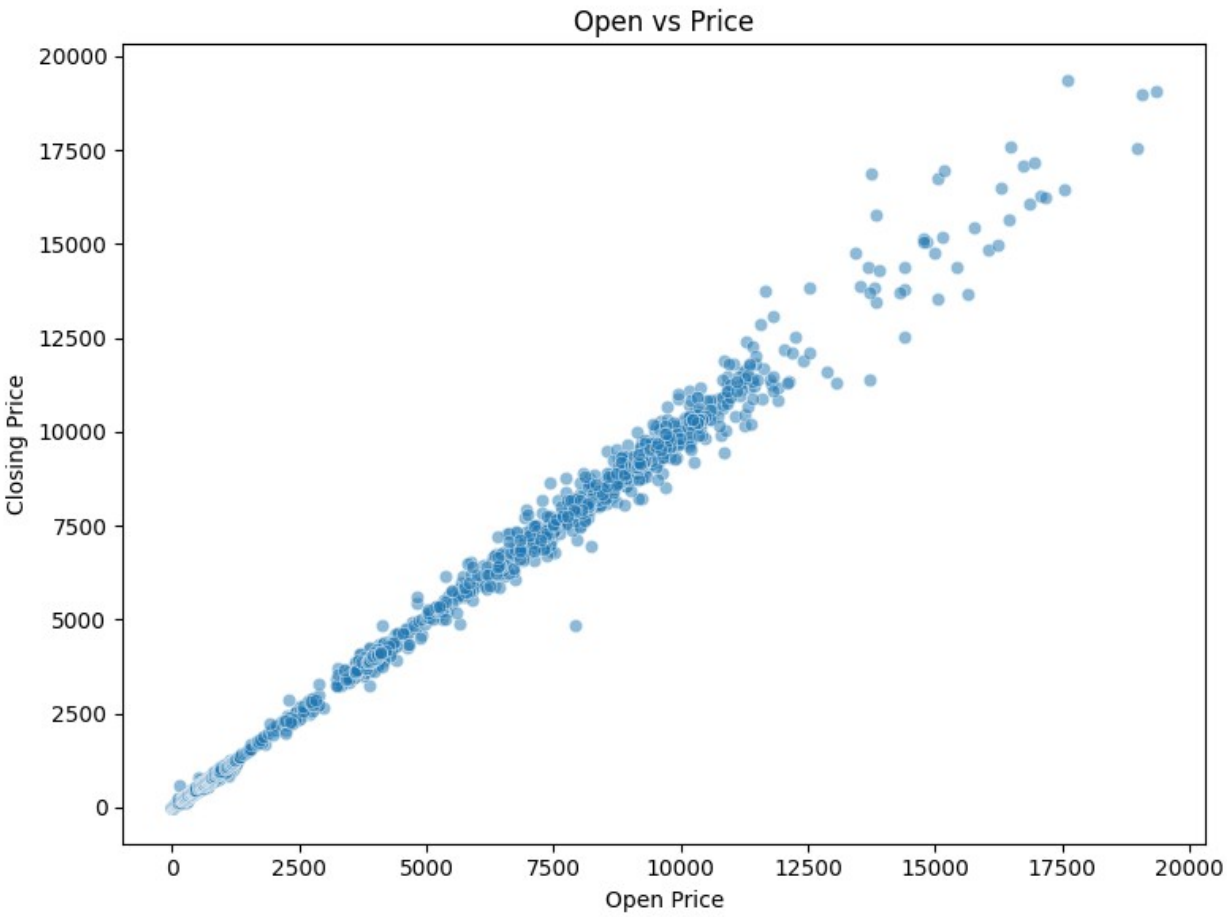
```
# Plotting setup
plt.figure(figsize=(14, 6))
plt.plot(df['Date'], df['Price'], label='Price', color='blue')
plt.title('Bitcoin Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



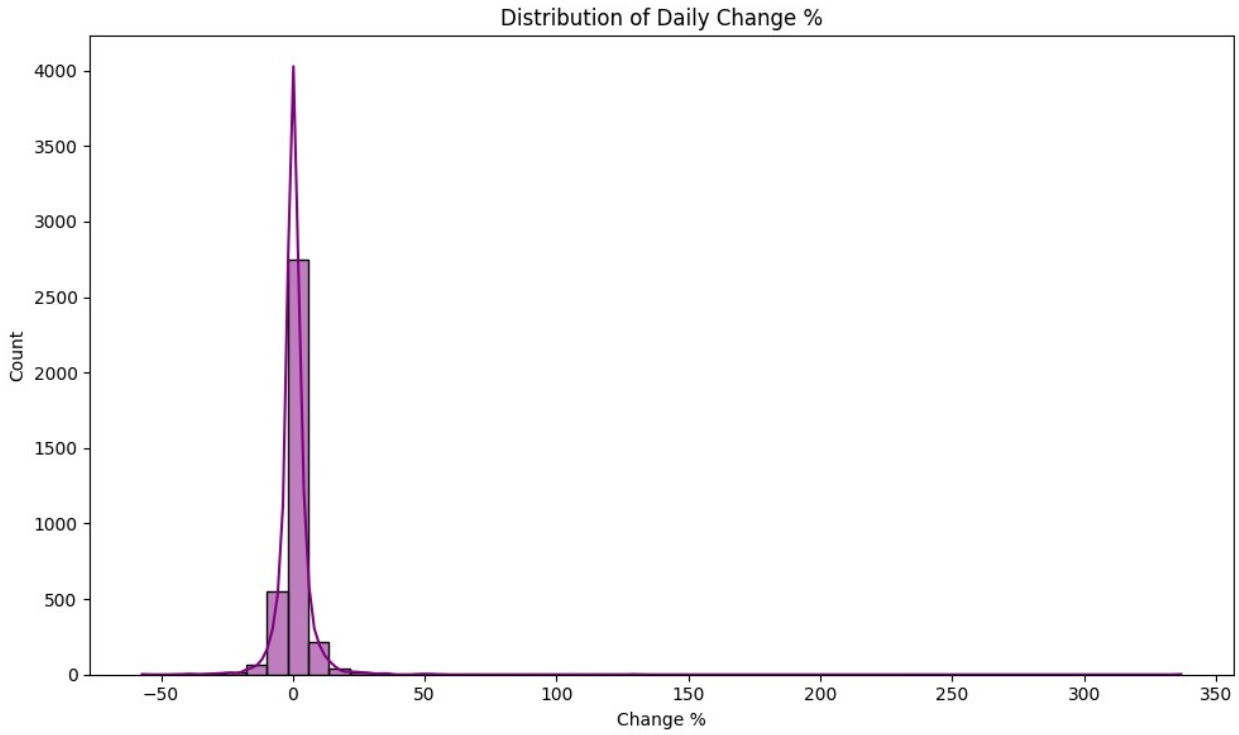

```
# Volume over time
plt.figure(figsize=(14, 6))
plt.plot(df['Date'], df['Vol.'], label='Volume', color='green')
plt.title('Bitcoin Volume Over Time')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```




```
# Open vs Price
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Open', y='Price', data=df, alpha=0.5)
plt.title("Open vs Price")
plt.xlabel("Open Price")
plt.ylabel("Closing Price")
plt.tight_layout()
plt.show()
```




```
# Change % distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Change %'], bins=50, kde=True, color='purple')
plt.title("Distribution of Daily Change %")
plt.xlabel("Change %")
plt.tight_layout()
plt.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Feature selection
features = ['Open', 'High', 'Low', 'Vol.', 'Change %']
target = 'Price'

X = df[features]
y = df[target]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

# Linear Regression Model
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

# Evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

mae, mse, rmse
```

```
(122.3881387196417, 31989.872988398467,
np.float64(178.85713010220886))

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Visualize clusters on Price vs Volume
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Vol.', y='Price', hue='Cluster',
palette='Set2', alpha=0.7)
plt.title("K-Means Clustering: Price vs Volume")
plt.xlabel("Volume")
plt.ylabel("Price")
plt.tight_layout()
plt.show()
```