Machine Learning

Machine Learning for Bitcoin price prediction

Bitcoin price prediction

A comprehensive analysis using Linear Regression, Random Forest, and Gradient Boosting models

Natasha Lertsansiri 1650900234
Thanathon Satthayaphan 1650900234

Bitcoin price prediction

Overview :

- Introduction
- Data Overview
- Data Preprocessing
- Model Selection
- Model Training
- Testing and Evaluation
- Results
- Conclusion

Introduction

- We delve into the exciting realm of Bitcoin prediction through machine learning. Bitcoin, the pioneering cryptocurrency, has captivated the world with its meteoric rise and volatility. Understanding its price dynamics is not only a matter of financial interest but also a significant challenge due to its complex and often unpredictable nature.

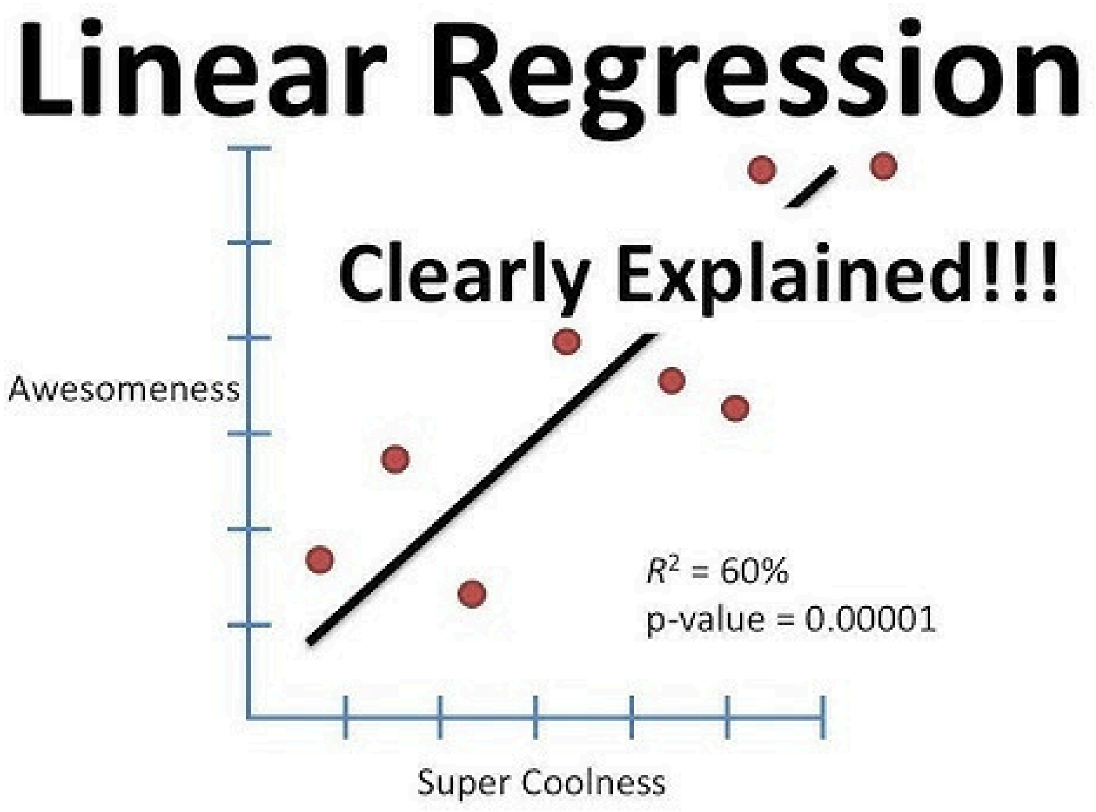
- In this project, we harness the power of machine learning algorithms to forecast Bitcoin prices. By analyzing historical data, identifying patterns, and leveraging advanced predictive models, we aim to provide valuable insights into the future movements of this digital asset.

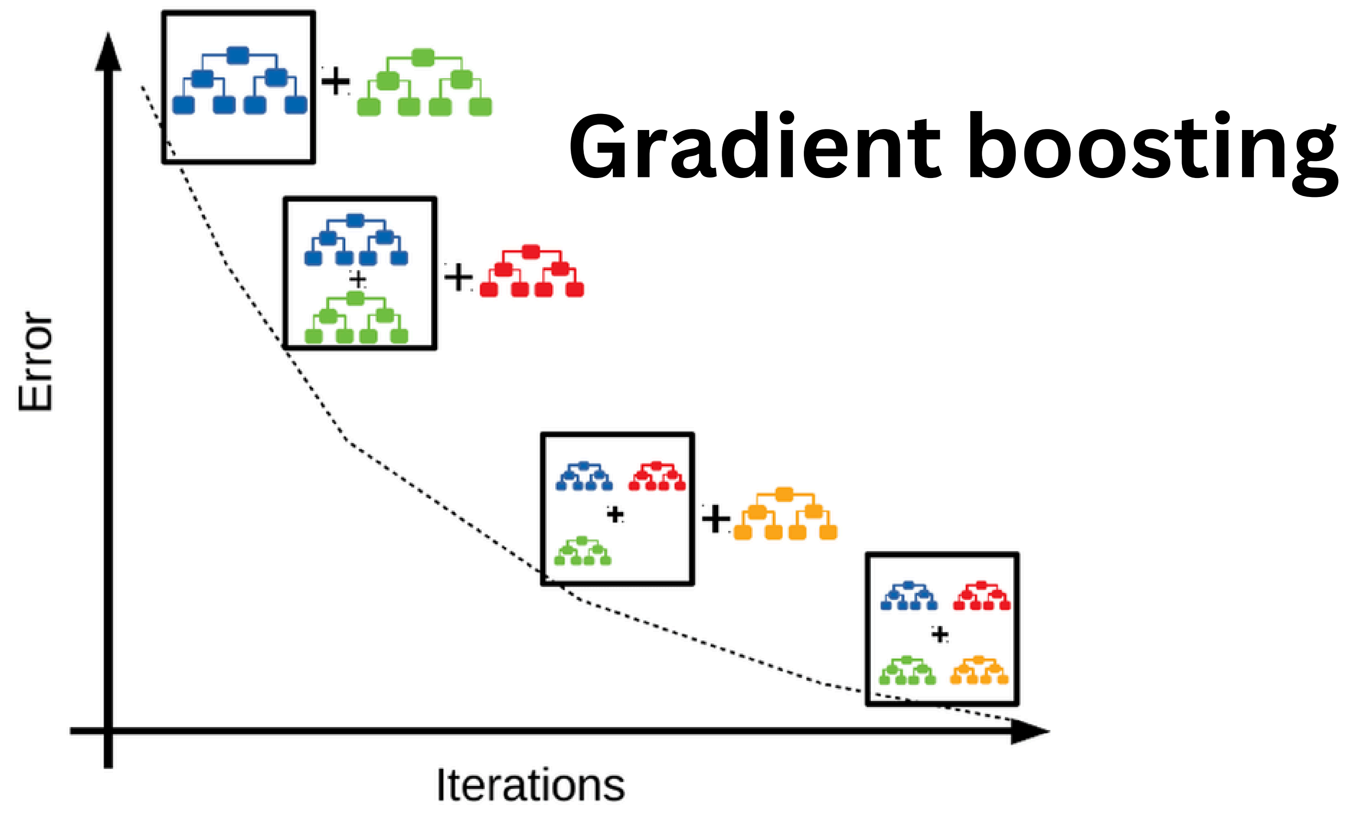
Data Overview

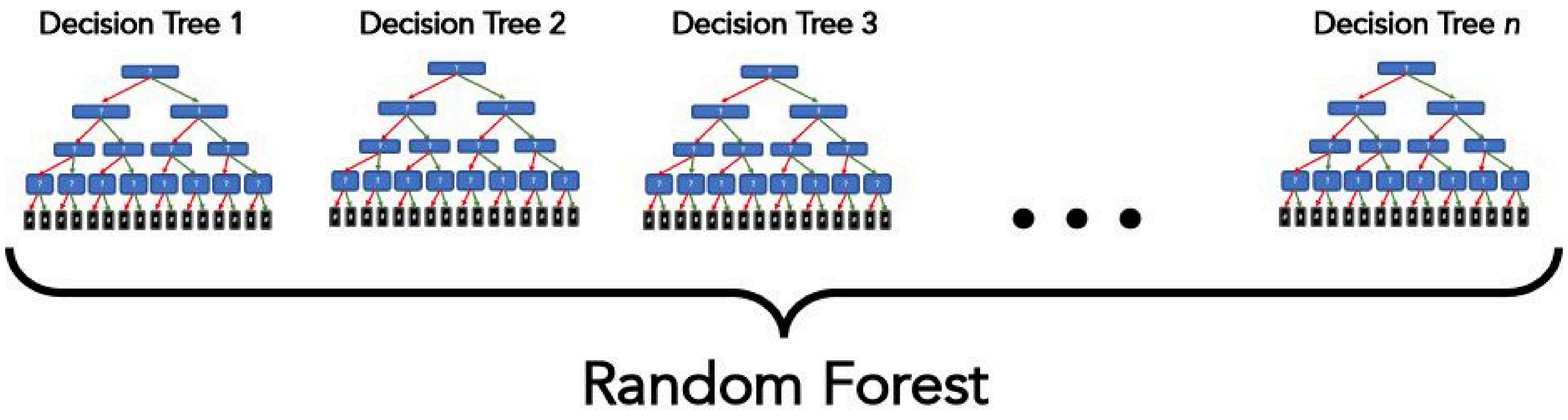
- **Data Source: https://coincodex.com/crypto/bitcoin/historical-data/**
- **Key Features: List the main features used (Open, High, Low, Volume, Market Cap).**
- **Date Range: Mention the time period the data covers.**

Start	End	Open	High	Low	Close	Volume	Market Cap
1/1/2011	1/2/2011	0.3	0.3	0.3	0.3	0	0
1/2/2011	1/3/2011	0.3	0.3	0.3	0.3	0	0
1/3/2011	1/4/2011	0.295	0.295	0.295	0.295	0	0
1/4/2011	1/5/2011	0.299	0.299	0.299	0.299	0	0
1/5/2011	1/6/2011	0.299	0.299	0.299	0.299	0	0
1/6/2011	1/7/2011	0.298	0.298	0.298	0.298	0	0
1/7/2011	1/8/2011	0.32	0.32	0.32	0.32	0	0
1/8/2011	1/9/2011	0.3229	0.3229	0.3229	0.3229	0	0
1/9/2011	1/10/2011	0.323	0.323	0.323	0.323	0	0
1/10/2011	1/11/2011	0.3266	0.3266	0.3266	0.3266	0	0
1/11/2011	1/12/2011	0.3266	0.3266	0.3266	0.3266	0	0
1/12/2011	1/13/2011	0.3188	0.3188	0.3188	0.3188	0	0
1/13/2011	1/14/2011	0.3176	0.3176	0.3176	0.3176	0	0
1/14/2011	1/15/2011	0.4	0.4	0.4	0.4	0	0
1/15/2011	1/16/2011	0.386	0.386	0.386	0.386	0	0
1/16/2011	1/17/2011	0.3868	0.3868	0.3868	0.3868	0	0
1/17/2011	1/18/2011	0.3495	0.3495	0.3495	0.3495	0	0
1/18/2011	1/19/2011	0.313	0.313	0.313	0.313	0	0
1/19/2011	1/20/2011	0.313	0.313	0.313	0.313	0	0
1/20/2011	1/21/2011	0.39	0.39	0.39	0.39	0	0
1/21/2011	1/22/2011	0.4199	0.4199	0.4199	0.4199	0	0
1/22/2011	1/23/2011	0.4443	0.4443	0.4443	0.4443	0	0
1/23/2011	1/24/2011	0.4424	0.4424	0.4424	0.4424	0	0
1/24/2011	1/25/2011	0.4199	0.4199	0.4199	0.4199	0	0
1/25/2011	1/26/2011	0.41	0.41	0.41	0.41	0	0
1/26/2011	1/27/2011	0.417	0.417	0.417	0.417	0	0
1/27/2011	1/28/2011	0.4212	0.4212	0.4212	0.4212	0	0
1/28/2011	1/29/2011	0.446	0.446	0.446	0.446	0	0
1/29/2011	1/30/2011	0.439	0.439	0.439	0.439	0	0

Model Selection







Libraries and Tools used

- Pandas
- sklearn
- numpy
- matplotlib
- Visual Studio Code - IDE

Data **Preprocessing**

Missing data and Convert to datetime

```python
import pandas as pd

# โหลดข้อมูล
data = pd.read_csv('bitcoin_dialy.csv')
print(data.head())

# ตรวจสอบmissing data
missing_data = data.isnull().sum()

# แปลงคอลัมน์ 'Start' และ 'End' ให้เป็น datetime
data['Start'] = pd.to_datetime(data['Start'])
data['End'] = pd.to_datetime(data['End'])

print("Missing data:\n", missing_data)
print("\nData types after conversion:\n", data.dtypes)

from sklearn.model_selection import train_test_split
# ตัวแปรอิสระ (features)
X = data[['Open', 'High', 'Low', 'Volume', 'Market Cap']]
# ตัวแปรตาม (target)
y = data['Close']
# แบ่งข้อมูลtrain/tast 80/20
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# ขนาดของชุดข้อมูลที่แบ่งได้
print("Training set:", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
```

```
        Start         End   Open   High    Low  Close  Volume  Market Cap
0  2011-01-01  2011-01-02  0.300  0.300  0.300  0.300     0.0         0.0
1  2011-01-02  2011-01-03  0.300  0.300  0.300  0.300     0.0         0.0
2  2011-01-03  2011-01-04  0.295  0.295  0.295  0.295     0.0         0.0
3  2011-01-04  2011-01-05  0.299  0.299  0.299  0.299     0.0         0.0
4  2011-01-05  2011-01-06  0.299  0.299  0.299  0.299     0.0         0.0
Missing data:
 Start         0
End           0
Open          0
High          0
Low           0
Close         0
Volume        0
Market Cap    0
dtype: int64

Data types after conversion:
 Start         datetime64[ns]
End           datetime64[ns]
Open                 float64
High                 float64
Low                  float64
Close                float64
Volume               float64
Market Cap           float64
dtype: object
Training set: (3799, 5) (3799,)
Test set: (950, 5) (950,)
```

Model Training and Evaluation

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

# สร้างและ train Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Predict test data
y_pred_linear = linear_model.predict(X_test)

# ประเมินLinear Regression model
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

# สร้างและ train Random Forest Regression model
random_forest_model = RandomForestRegressor(random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred_rf = random_forest_model.predict(X_test)

# ประเมิน Random Forest Regression model
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# # สร้างและ train Gradient Boosting Regression model
gb_model = GradientBoostingRegressor(random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_test)

# ประเมิน
mse_gb = mean_squared_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

model_evaluation_results = {
    "Linear Regression": {"MSE": mse_linear, "R2": r2_linear},
    "Random Forest Regression": {"MSE": mse_rf, "R2": r2_rf},
    "Gradient Boosting Regression": {"MSE": mse_gb, "R2": r2_gb}
}
model_evaluation_results

print("Linear Regression - MSE:", mse_linear, "\nR2 Score:", r2_linear)
print("Random Forest Regression - MSE:", mse_rf, "\nR2 Score:", r2_rf)
print("Gradient Boosting Regression - MSE:", mse_gb, "\nR2 Score:", r2_gb)
```

```
Linear Regression - MSE: 66069.4376948983 R2 Score: 0.9997035066528899
Random Forest Regression - MSE: 114626.86487298366 R2 Score: 0.999485600240888
Gradient Boosting Regression - MSE: 159301.79658384464 R2 Score: 0.999285116923684
```

Linear Regression - MSE: 66069.4376948983
R2 Score: 0.9997035066528899

Random Forest Regression - MSE: 114626.86487298366
R2 Score: 0.999485600240888

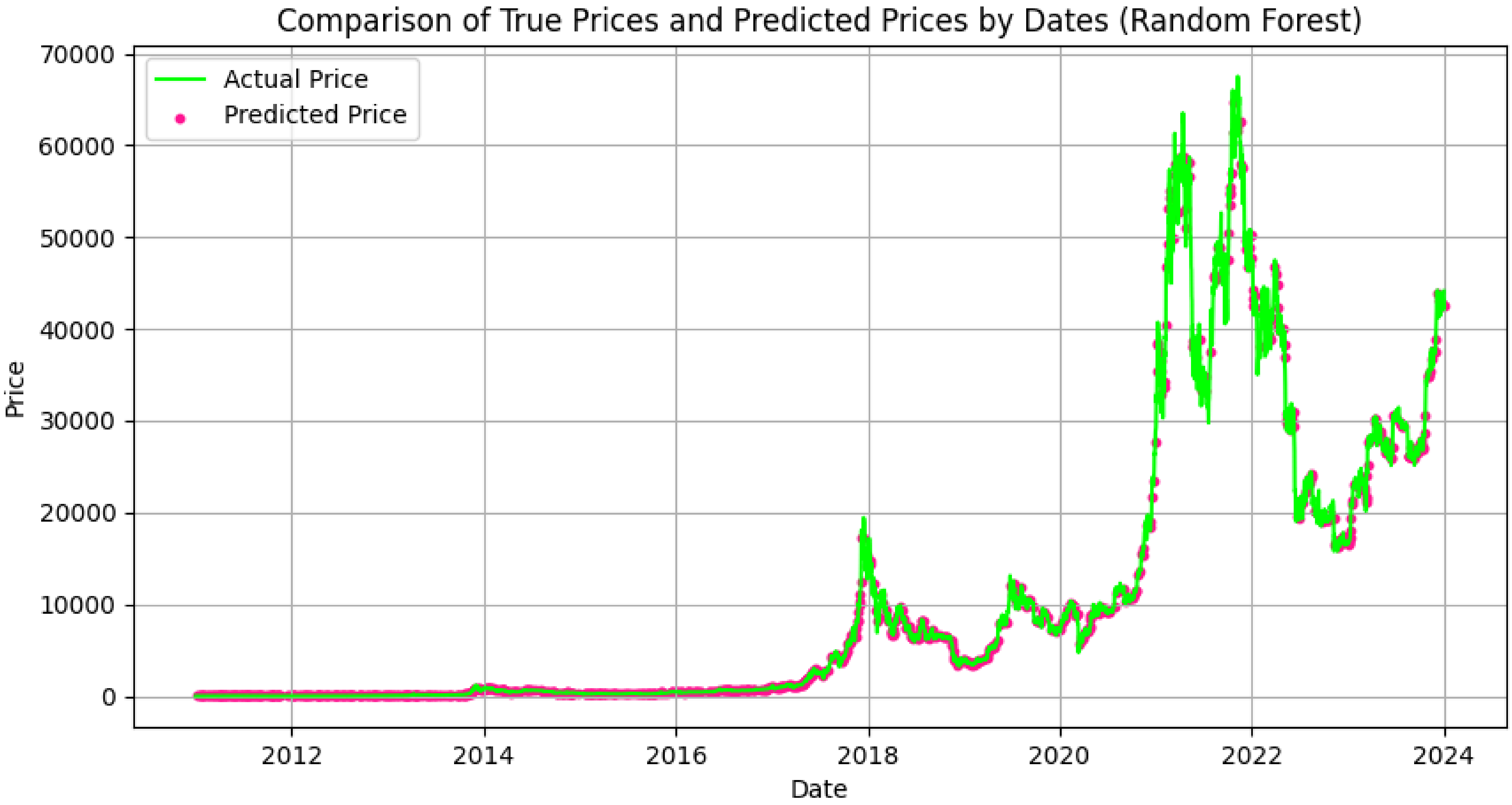
Gradient Boosting Regression - MSE: 159301.79658384464
R2 Score: 0.999285116923684

Model Prediction and Evaluation

```python
data['Start'] = pd.to_datetime(data['Start'])
data.set_index('Start', inplace=True)

data['Time_Index'] = (data.index - data.index.min()) / pd.Timedelta(days=1)
X = data[['Time_Index']]
y = data['Close']

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

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
print('\n\n',y_pred_rf)


plt.figure(figsize=(10, 5))
plt.plot(data.index, y, label='Actual Price', color='lime')
plt.scatter(X_test.index, y_pred_rf, color='deeppink', label='Predicted Price', s=10)
plt.title('Comparison of True Prices and Predicted Prices by Dates (Random Forest)')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
4.30974910e+02 2.16539433e+04 1.65266350e+01 1.00392109e+04
6.64310867e+03 3.69073926e+04 9.02079700e+02 6.00140020e+02
2.30943483e+04 6.46728200e+02 2.41997206e+04 1.44235930e+03
1.10359310e+01 4.82246425e+03 2.97545019e+04 8.20734742e+03
3.88274606e+03 5.70189965e+04 2.01788699e+04 7.09975422e+03
7.44670000e-01 8.27310458e+03 4.24771442e+04 6.45015800e+00
2.69529310e+01 3.31814971e+04 1.05646519e+04 1.22475830e+03
1.23997400e+01 2.78136586e+04 4.85557295e+04 3.21745200e+00
6.26126400e+00 8.29160910e+03 4.56914016e+04 3.48964084e+04
8.20044171e+03 2.68071680e+02 7.49726220e+01 1.14610510e+02
1.70317986e+04 2.15056940e+01 1.13101840e+03 8.70363116e+03
4.02104180e+02 6.93310633e+03 1.24279770e+03 6.13981610e+02
3.87367700e+00 7.52004600e+00 6.41697000e+02 6.00925540e+02
4.57924000e+02 6.87017656e+03 9.19689838e+03 6.11632642e+03
3.91671800e+00 1.34815710e+03 7.39462592e+03 2.87601200e+00
5.03520891e+04 2.75781250e+03 7.28632908e+03 2.29509410e+02
8.55242600e+00 5.56261600e+00 3.86434680e+02 6.03853300e+00
1.36093880e+01 6.40164000e+00 6.42031714e+03 6.89498171e+03
2.08493362e+04 6.40493640e+02 1.67843015e+04 1.70176585e+04
3.54321247e+03 6.77225300e+03 1.35492335e+04 4.36578105e+04
3.91166250e+02 3.03332725e+04 8.22946020e+02 1.17507130e+01
9.28434482e+03 4.53737370e+01 2.35130410e+04 2.68139466e+04
4.30810000e-01 1.72164999e+04 4.23052058e+04 2.75633076e+04
4.14068320e+02 2.69019710e+04 9.73524785e+03 7.61485791e+03
2.30762150e+02 1.01531626e+04 4.23671930e+02 1.02235931e+04
9.40549866e+03 7.83147821e+03 8.76204741e+03 4.48864240e+03
4.45574260e+02 3.87758413e+04 6.30801080e+02 1.17130237e+04
1.19035550e+01 4.04687470e+02 7.32909137e+03 2.12647198e+04
3.71017770e+01 1.40900800e+02 6.63271500e+00 5.37035583e+03
8.88241157e+03 1.07406270e+03 3.68353636e+03 9.75675370e+01
```

Result



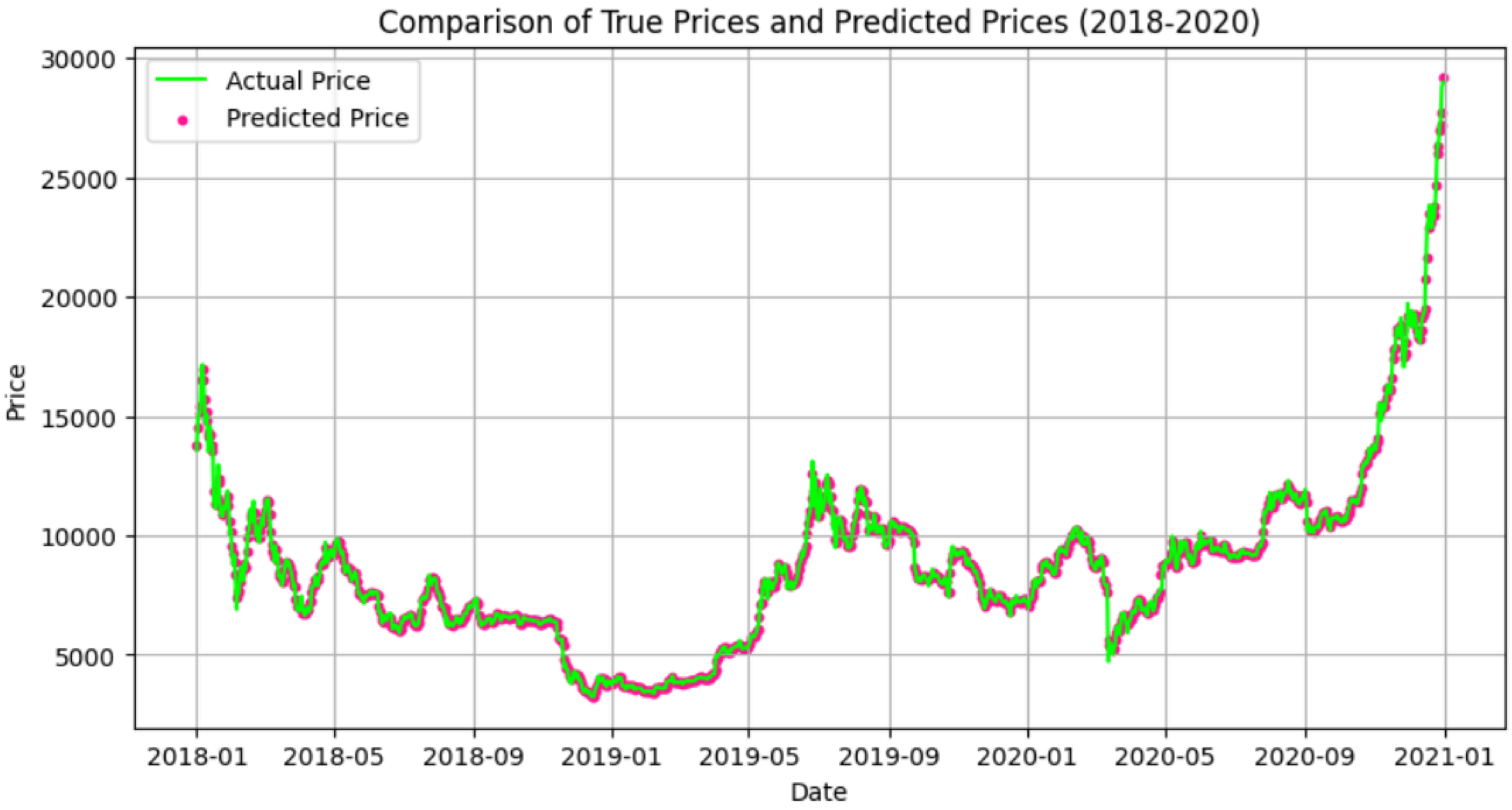
Comparison of True Prices and Predicted Prices by Dates (Random Forest)
Actual Price
Predicted Price
70000
60000
50000
40000
30000
20000
10000
0
Price
2012
2014
2016
2018
2020
2022
2024
Date

Model Prediction and Evaluation

```
start_date = '2018-01-01'
end_date = '2020-12-31'

data_filtered = data.loc[start_date:end_date]
X_filtered = X.loc[start_date:end_date]
y_filtered = y.loc[start_date:end_date]

y_pred_filtered = rf_model.predict(X_filtered)
print('\n\n',y_pred_filtered)

plt.figure(figsize=(10, 5))
plt.plot(data_filtered.index, y_filtered, label='Actual Price', color='lime')
plt.scatter(data_filtered.index, y_pred_filtered, color='deeppink', label='Predicted Price', s=10)
plt.title('Comparison of True Prices and Predicted Prices (2018-2020)')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()
```

```
[13765.01040549 14524.86439227 15116.96377777 ... 27231.21023167
 27754.31058647 29188.28721168]
```

Result



Comparison of True Prices and Predicted Prices (2018-2020)
Actual Price
Predicted Price
Price
30000
25000
20000
15000
10000
5000
Date
2018-01
2018-05
2018-09
2019-01
2019-05
2019-09
2020-01
2020-05
2020-09
2021-01

Thank you!

```
most_recent_data = data[['Open', 'High', 'Low', 'Volume', 'Market Cap']].iloc[-1].values.reshape(1, -1)

next_day_prediction = random_forest_model.predict(most_recent_data)
print("Predicted closing price for tomorrow:", next_day_prediction[0])


# ==========================================================================================
num_days = 7
input_features = np.array(data[['Open', 'High', 'Low', 'Volume', 'Market Cap']].iloc[-1]).reshape(1, -1)

predictions = []

for i in range(num_days):
    next_day_prediction = random_forest_model.predict(input_features)
    predictions.append(next_day_prediction[0])
    input_features = np.array([[next_day_prediction[0], next_day_prediction[0], next_day_prediction[0],
                                input_features[0, 3], input_features[0, 4]]])


predictions
```

```
[43218.324227665114,
 42708.869423206765,
 42499.351815459064,
 42152.798996691636,
 41996.97053298736,
 41930.95580938235,
 41929.71168243981]
```

```
Predicted closing price for tomorrow: 43218.324227665114
```