```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.preprocessing import LabelEncoder
```

```
data=pd.read_csv("/content/Bitcoin.csv")
```

```
data
```

	Date	Open	High	Low	Close	Volume	Market Cap
0	Jul 31, 2017	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000
1	Jul 30, 2017	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
2	Jul 29, 2017	2807.02	2808.76	2692.8	2726.45	803,746,000	46,246,700,000
3	Jul 28, 2017	2679.73	2897.45	2679.73	2809.01	1,380,100,000	44,144,400,000
4	Jul 27, 2017	2538.71	2693.32	2529.34	2671.78	789,104,000	41,816,500,000
...	...	...	...	...	...	...	...
1994	2017-06-07	2869.379883	2869.379883	2700.560059	2732.159912	2732.159912	1517709952
1995	2017-06-08	2720.489990	2815.300049	2670.949951	2805.620117	2805.620117	1281170048
1996	2017-06-09	2807.439941	2901.709961	2795.620117	2823.810059	2823.810059	1348950016
1997	2017-06-10	2828.139893	2950.989990	2746.550049	2947.709961	2947.709961	2018889984
1998	2017-06-11	2942.409912	2996.600098	2840.530029	2958.110107	2958.110107	1752400000

1999 rows × 7 columns

```
data.isnull().sum()
```

	0
Date	0
Open	0
High	0
Low	0
Close	0
Volume	0
Market Cap	0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1999 entries, 0 to 1998
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        1999 non-null   object
 1   Open        1999 non-null   object
 2   High        1999 non-null   object
 3   Low         1999 non-null   object
 4   Close       1999 non-null   object
 5   Volume      1999 non-null   object
 6   Market Cap  1999 non-null   object
dtypes: object(7)
memory usage: 109.4+ KB
```

```
data.describe()
```

	Date	Open	High	Low	Close	Volume	Market Cap
count	1999	1999	1999	1999	1999	1999	1999
unique	1999	1978	1986	1989	1984	1997	1997
top	Jul 31, 2017	418.42	231.22	429.08	236.15	236.153000	3,329,190,000
freq	1	3	3	3	3	2	2

```
data.drop("Date",axis=1,inplace=True)
```

```
data.head()
```

	Open	High	Low	Close	Volume	Market Cap
0	2763.24	2889.62	2720.61	2875.34	860,575,000	45,535,800,000
1	2724.39	2758.53	2644.85	2757.18	705,943,000	44,890,700,000
2	2807.02	2808.76	2692.8	2726.45	803,746,000	46,246,700,000
3	2679.73	2897.45	2679.73	2809.01	1,380,100,000	44,144,400,000
4	2538.71	2693.32	2529.34	2671.78	789,104,000	41,816,500,000

```
data.abs
```

```
pandas.core.generic.NDFrame.abs
def abs() -> Self

Return a Series/DataFrame with absolute numeric value of each element.

This function only applies to elements that are all numeric.

Returns
-------
```

```
data['Open'].dtype
```

```
dtype('O')
```

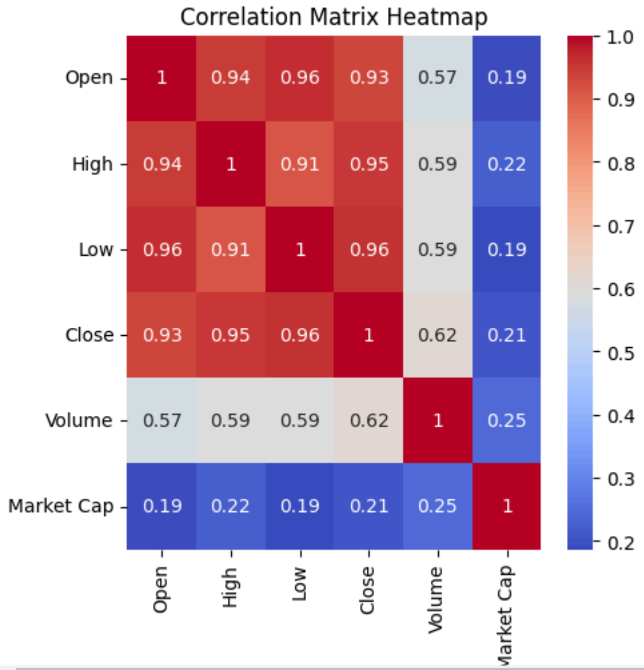
```
Encoder=LabelEncoder()
data['Open']=Encoder.fit_transform(data['Open'])
data['High']=Encoder.fit_transform(data['High'])
data['Low']=Encoder.fit_transform(data['Low'])
data['Close']=Encoder.fit_transform(data['Close'])
data['Volume']=Encoder.fit_transform(data['Volume'])
data['Market Cap']=Encoder.fit_transform(data['Market Cap'])
```

```
data.dtypes
```

	0
Open	int64
High	int64
Low	int64
Close	int64
Volume	int64
Market Cap	int64

```
print(data.dtypes)
# Calculate the correlation matrix
correlation_matrix = data.corr()
# Visualize the correlation matrix with a heatmap
plt.figure(figsize=(5, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
Open          int64
High          int64
Low           int64
Close         int64
Volume        int64
Market Cap    int64
dtype: object
```




```
# split the dataset into dependent and independent data features
x=data.drop("Close",axis=1)
y=data["Close"]
```

```
x.isnull().sum()
y.isnull().sum()
```

```
0
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
```

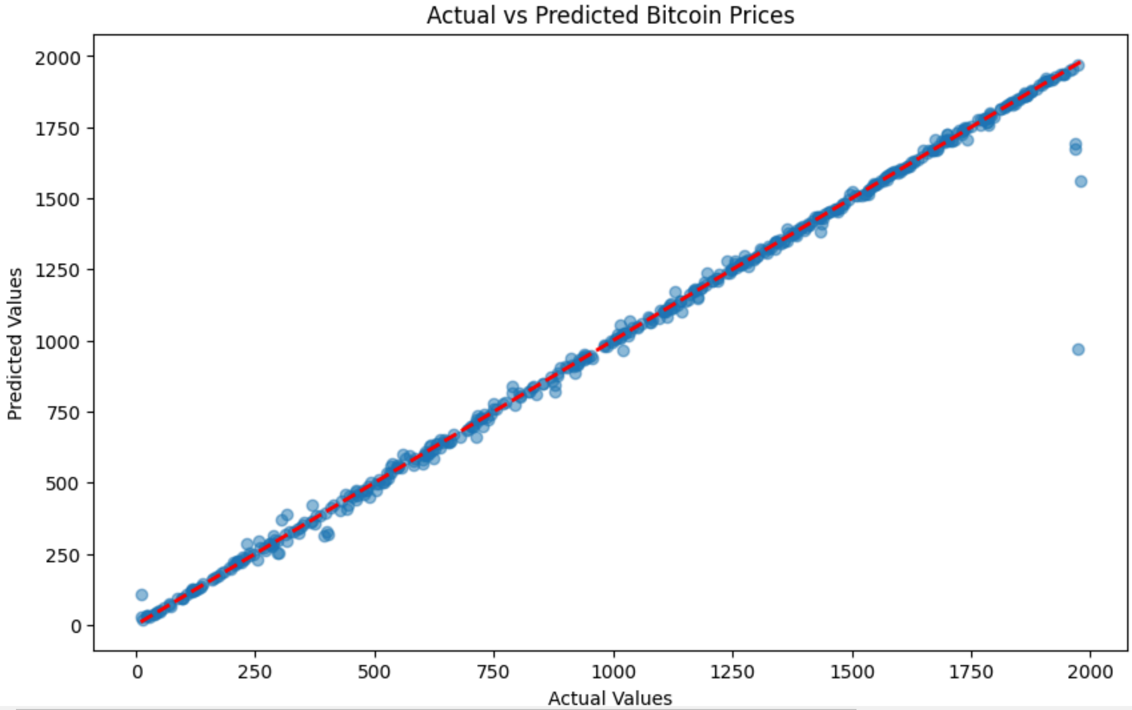
```
model=RandomForestRegressor()
```

```
y_pred=model.fit(x_train,y_train).predict(x_test)
```

```
# Calculate evaluation metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R²) Score: {r2:.2f}")
```

```
Mean Absolute Error (MAE): 13.76
Mean Squared Error (MSE): 2952.37
Root Mean Squared Error (RMSE): 54.34
R-squared (R²) Score: 0.99
```

```
import matplotlib.pyplot as plt

# Plotting actual vs. predicted values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--', lw=2)  # Line of perfect prediction
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Bitcoin Prices')
plt.show()
```




```
# Feature importance
feature_importances = model.feature_importances_
features = x.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances}).sort_values(by='Importance', ascending=False)

print(importance_df)
```

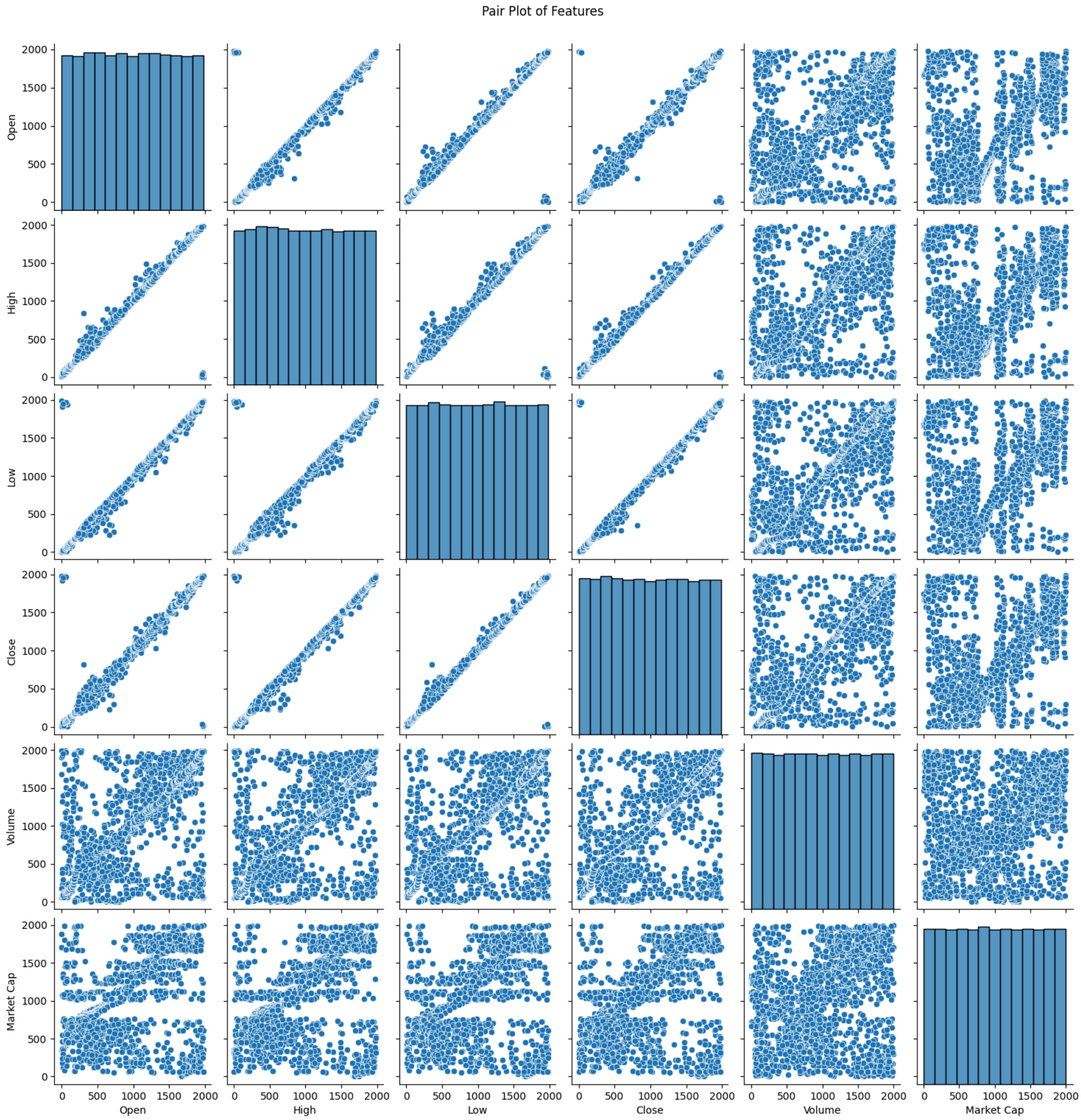
```
     Feature  Importance
1       High    0.509092
2        Low    0.456277
0       Open    0.031168
3     Volume    0.003197
4 Market Cap    0.000266
```

```
from sklearn.model_selection import cross_val_score

# Perform cross-validation
cv_scores = cross_val_score(model, x, y, cv=5, scoring='neg_mean_squared_error')
cv_rmse = np.sqrt(-cv_scores)

print(f"Cross-validated RMSE: {np.mean(cv_rmse):.2f} ± {np.std(cv_rmse):.2f}")
```

```
Cross-validated RMSE: 76.47 ± 64.68
```

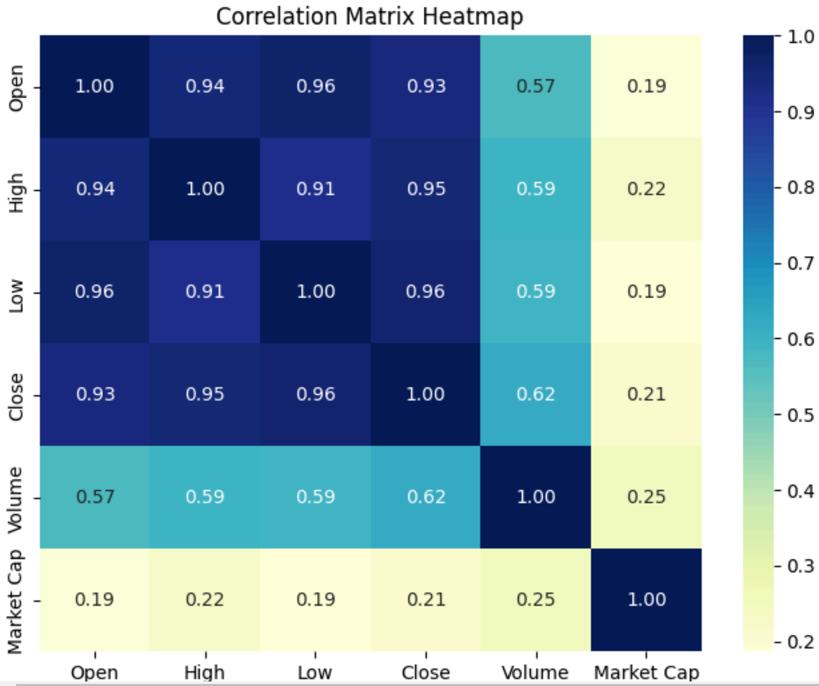
```
# Distribution Plot for 'Close' Price
plt.figure(figsize=(8, 5))
sns.histplot(data['Close'], bins=30, kde=True, color='blue')
plt.title('Distribution of Close Prices')
plt.xlabel('Close Price')
plt.ylabel('Frequency')
plt.show()
```


```
# Pair Plot for all Features
sns.pairplot(data)
plt.suptitle('Pair Plot of Features', y=1.02)
plt.show()
```

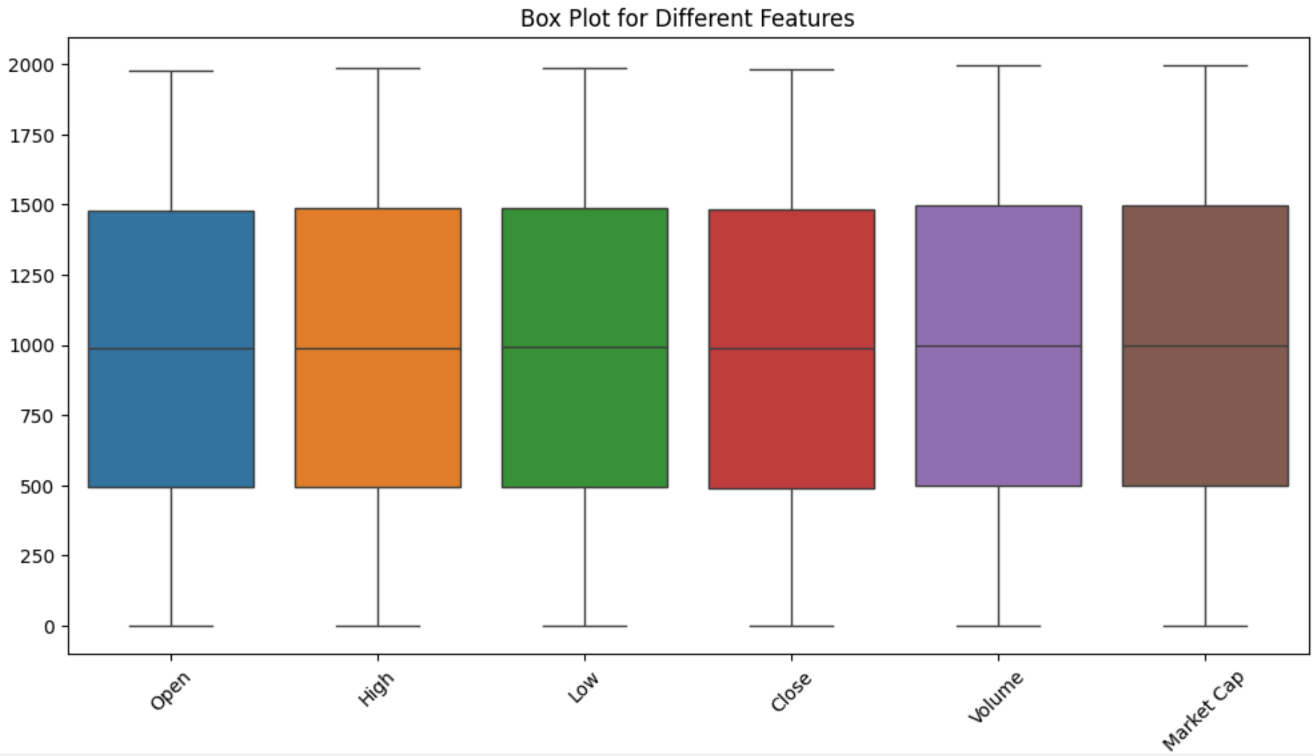



```
# Heatmap of the Correlation Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
plt.show()
```

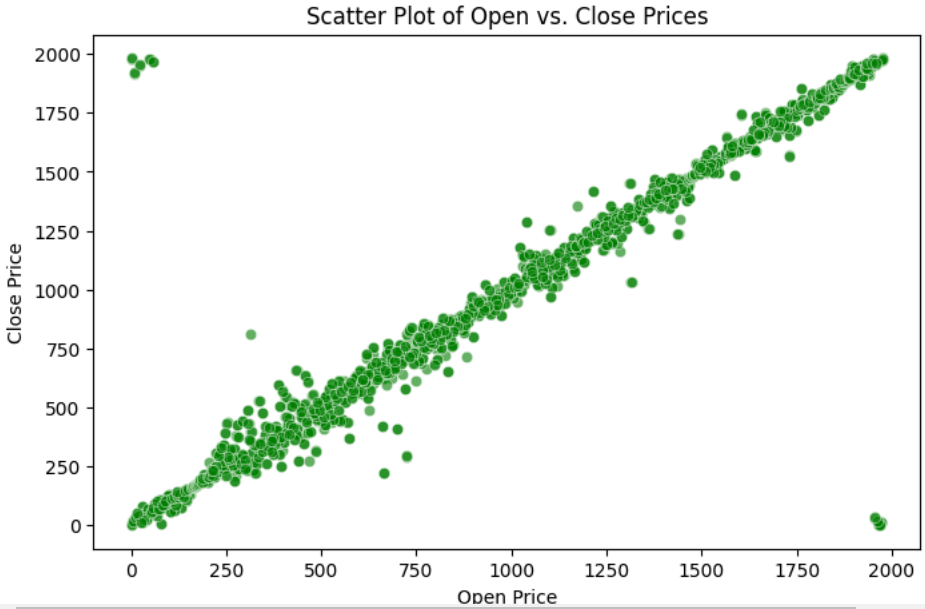
```
plt.show()
```



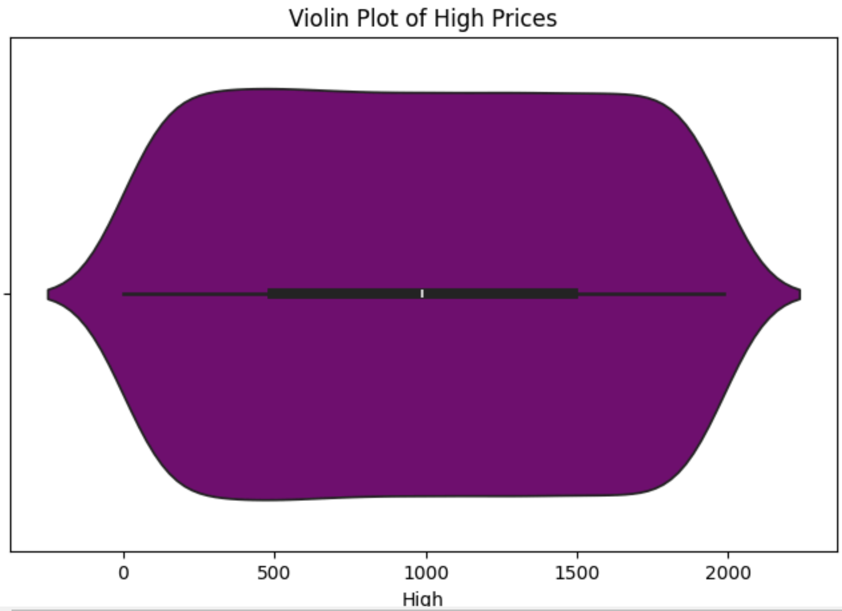

```
# Box Plot for Different Features
plt.figure(figsize=(12, 6))
sns.boxplot(data=data)
plt.title('Box Plot for Different Features')
plt.xticks(rotation=45)
plt.show()
```




```
# Scatter Plot for 'Open' vs. 'Close' Prices
plt.figure(figsize=(8, 5))
sns.scatterplot(x='Open', y='Close', data=data, color='green', alpha=0.6)
plt.title('Scatter Plot of Open vs. Close Prices')
plt.xlabel('Open Price')
plt.ylabel('Close Price')
plt.show()
```

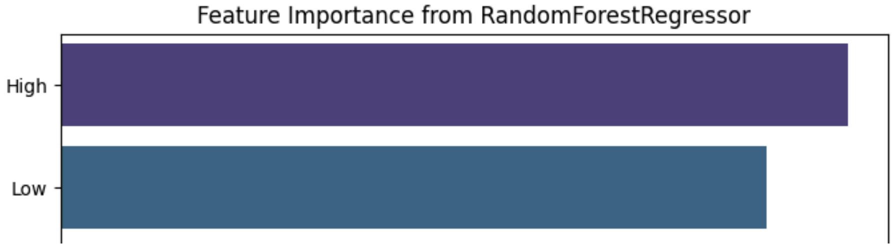



```
# Violin Plot for the Distribution of 'High' Prices
plt.figure(figsize=(8, 5))
sns.violinplot(x='High', data=data, color='purple')
plt.title('Violin Plot of High Prices')
plt.show()
```




```
# Bar Plot for Feature Importance
plt.figure(figsize=(8, 5))
sns.barplot(x='Importance', y='Feature', data=importance_df, palette='viridis')
plt.title('Feature Importance from RandomForestRegressor')
plt.show()
```

```
<ipython-input-60-661aa1a6f7c5>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x='Importance', y='Feature', data=importance_df, palette='viridis')
```




```
# Joint Plot for 'High' vs. 'Low' Prices
sns.jointplot(x='High', y='Low', data=data, kind='scatter', color='red')
plt.suptitle('Joint Plot of High vs. Low Prices', y=1.02)
plt.show()
```