In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.preprocessing import StandardScaler
from IPython.display import display, HTML
import re
```

In [2]:

```python
data = pd.read_csv('isedataset.csv')

df = data.copy()
```

In [3]:

```python
def getting_primary_info(df):
    print("------------------------------------------------------")
    print("Veri setinin şekli", df.shape)
    print("------------------------------------------------------")
    print("Veri seti değişken tipleri:\n", df.dtypes)
    print("------------------------------------------------------")
    print("Veri setinin ilk 5 satırı")
    display(HTML(df.head().to_html()))
    print("------------------------------------------------------")
    print("Veri setinin istatistiki verileri")
    description = df.describe()
    display(HTML(description.to_html()))
    print("------------------------------------------------------")
getting_primary_info(df)
```

```
------------------------------------------------------
Veri setinin şekli (115986, 9)
------------------------------------------------------
Veri seti değişken tipleri:
 Unnamed: 0     object
Open          float64
High          float64
Low           float64
Close         float64
Volume          int64
Symbol         object
Predict       float64
Unnamed: 8    float64
dtype: object
------------------------------------------------------
Veri setinin ilk 5 satırı
```

	Unnamed: 0	Open	High	Low	Close	Volume	Symbol	Predict	Unnamed: 8
0	2023-06-15 00:00:00+03:00	27.500000	27.500000	27.500000	27.500000	262214	A1CAP.IS	30.240000	NaN
1	2023-06-16 00:00:00+03:00	30.240000	30.240000	30.240000	30.240000	1169499	A1CAP.IS	29.940001	NaN
2	2023-06-20 00:00:00+03:00	31.000000	31.100000	29.940001	29.940001	8064437	A1CAP.IS	26.959999	NaN
3	2023-06-21 00:00:00+03:00	26.959999	26.959999	26.959999	26.959999	2147415	A1CAP.IS	25.940001	NaN
4	2023-06-22 00:00:00+03:00	25.620001	27.620001	25.500000	25.940001	71898180	A1CAP.IS	25.900000	NaN

```
------------------------------------------------------
Veri setinin istatistiki verileri
```

	Open	High	Low	Close	Volume	Predict	Unnamed: 8
count	1.159440e+05	1.159440e+05	1.159440e+05	1.159440e+05	1.159860e+05	1.159810e+05	0.0
mean	3.373242e+03	3.389169e+03	3.358713e+03	3.374240e+03	1.240340e+07	3.446984e+03	NaN
std	7.847550e+04	7.853709e+04	7.842754e+04	7.848479e+04	3.648941e+07	8.162920e+04	NaN
min	6.800000e-01	7.100000e-01	6.500000e-01	6.800000e-01	0.000000e+00	6.800000e-01	NaN
25%	8.989951e+00	9.170000e+00	8.740000e+00	8.980000e+00	4.789150e+05	9.000000e+00	NaN
50%	2.370000e+01	2.438000e+01	2.306632e+01	2.370000e+01	2.009944e+06	2.378000e+01	NaN
75%	6.200000e+01	6.371250e+01	6.025125e+01	6.195000e+01	7.715315e+06	6.210000e+01	NaN
max	8.930900e+06	8.930900e+06	8.930900e+06	8.930900e+06	9.786029e+08	8.930900e+06	NaN

```
------------------------------------------------------
```

In [4]:

```python
df = df.drop(columns=['Unnamed: 8'])
```

In [5]:

```python
df['MA5'] = df.groupby('Symbol')['Close'].transform(lambda x: x.rolling(5).mean())
```

In [6]:

```python
df['MA5'] = df['MA5'].fillna(0)
```

In [7]:

```python
def calculate_rsi(data, window=14):
    delta = data.diff()

    gain = (delta.where(delta > 0, 0)).rolling(window=window).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=window).mean()

    rs = gain / loss
    rsi = 100 - (100 / (1 + rs))

    return rsi

df['RSI'] = calculate_rsi(df['Close'])
```

In [8]:

```python
df["RSI"] = df["RSI"].fillna(0)
```

In [9]:

```python
df.head()
```

Out[9]:

	Unnamed: 0	Open	High	Low	Close	Volume	Symbol	Predict	MA5	RSI
0	2023-06-15 00:00:00+03:00	27.500000	27.500000	27.500000	27.500000	262214	A1CAP.IS	30.240000	0.000	0.0
1	2023-06-16 00:00:00+03:00	30.240000	30.240000	30.240000	30.240000	1169499	A1CAP.IS	29.940001	0.000	0.0
2	2023-06-20 00:00:00+03:00	31.000000	31.100000	29.940001	29.940001	8064437	A1CAP.IS	26.959999	0.000	0.0
3	2023-06-21 00:00:00+03:00	26.959999	26.959999	26.959999	26.959999	2147415	A1CAP.IS	25.940001	0.000	0.0
4	2023-06-22 00:00:00+03:00	25.620001	27.620001	25.500000	25.940001	71898180	A1CAP.IS	25.900000	28.116	0.0

In [10]:

```python
df.head()
```

Out[10]:

	Unnamed: 0	Open	High	Low	Close	Volume	Symbol	Predict	MA5	RSI
0	2023-06-15 00:00:00+03:00	27.500000	27.500000	27.500000	27.500000	262214	A1CAP.IS	30.240000	0.000	0.0
1	2023-06-16 00:00:00+03:00	30.240000	30.240000	30.240000	30.240000	1169499	A1CAP.IS	29.940001	0.000	0.0
2	2023-06-20 00:00:00+03:00	31.000000	31.100000	29.940001	29.940001	8064437	A1CAP.IS	26.959999	0.000	0.0
3	2023-06-21 00:00:00+03:00	26.959999	26.959999	26.959999	26.959999	2147415	A1CAP.IS	25.940001	0.000	0.0
4	2023-06-22 00:00:00+03:00	25.620001	27.620001	25.500000	25.940001	71898180	A1CAP.IS	25.900000	28.116	0.0

In [11]:

```python
df.to_csv("deneme_Bitcoin.csv", index = False)
```

In [12]:

```python
import pandas as pd
data = pd.read_csv("deneme_Bitcoin.csv")
df = data.copy()
```

In [15]:

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

df = df.dropna()

X, y = df.drop(columns=["Predict", "Symbol", "Unnamed: 0"]), df["Predict"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model_linear = LinearRegression()
model_linear.fit(X_train, y_train)

y_pred_linear_train = model_linear.predict(X_train)
r2_linear_train = r2_score(y_train, y_pred_linear_train)
print("Linear Regression Train R^2 Score:", r2_linear_train)

y_pred_linear_test = model_linear.predict(X_test)
r2_linear_test = r2_score(y_test, y_pred_linear_test)
print("Linear Regression Test R^2 Score:", r2_linear_test)
print("-----------------------------")
model_random_forest = RandomForestRegressor(n_estimators=100, min_samples_split=2)
model_random_forest.fit(X_train, y_train)

y_pred_rf_train = model_random_forest.predict(X_train)
r2_rf_train = r2_score(y_train, y_pred_rf_train)
print("Random Forest Train R^2 Score:", r2_rf_train)

y_pred_rf_test = model_random_forest.predict(X_test)
r2_rf_test = r2_score(y_test, y_pred_rf_test)
print("Random Forest Test R^2 Score:", r2_rf_test)
print("-----------------------------")

model_gradient_boosting = GradientBoostingRegressor(n_estimators=80, min_samples_split=2)
model_gradient_boosting.fit(X_train, y_train)

y_pred_gb_train = model_gradient_boosting.predict(X_train)
r2_gb_train = r2_score(y_train, y_pred_gb_train)
print("Gradient Boosting Train R^2 Score:", r2_gb_train)

y_pred_gb_test = model_gradient_boosting.predict(X_test)
r2_gb_test = r2_score(y_test, y_pred_gb_test)
print("Gradient Boosting Test R^2 Score:", r2_gb_test)
```

```
Linear Regression Train R^2 Score: 0.9945002944855964
Linear Regression Test R^2 Score: 0.9971141751800278
-----------------------------
Random Forest Train R^2 Score: 0.9992564591569465
Random Forest Test R^2 Score: 0.9974944177071311
-----------------------------
Gradient Boosting Train R^2 Score: 0.9997687761351137
Gradient Boosting Test R^2 Score: 0.9976799630331733
```

In [ ]: