Bitcoin Price Prediction Project Documentation

# Introduction

This project aims to predict the price of Bitcoin using historical data. The dataset used for this project is sourced from Kaggle, and includes minute-by-minute price data from January 2012 to March 2021.

# Dataset Description

The dataset contains the following columns:  
1. \*\*Timestamp\*\*: Unix timestamp representing the date and time of the observation.  
2. \*\*Open\*\*: Opening price at the start of the minute.  
3. \*\*High\*\*: Highest price during the minute.  
4. \*\*Low\*\*: Lowest price during the minute.  
5. \*\*Close\*\*: Closing price at the end of the minute.  
6. \*\*Volume\_(BTC)\*\*: Volume of Bitcoin traded during the minute.  
7. \*\*Volume\_(Currency)\*\*: Volume of the traded currency during the minute.  
8. \*\*Weighted\_Price\*\*: The weighted price of Bitcoin during the minute.

# Code Explanation

## 1. Import Libraries

```python  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from statsmodels.tsa.seasonal import seasonal\_decompose  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from datetime import datetime  
```

## 2. Load the Dataset

```python  
url = r"C:\Users\Asus\Downloads\bitstampUSD\_1-min\_data\_2012-01-01\_to\_2021-03-31.csv"  
df = pd.read\_csv(url)  
```

## 3. Convert Timestamp to Datetime

```python  
df['Timestamp'] = pd.to\_datetime(df['Timestamp'], unit='s')  
df.set\_index('Timestamp', inplace=True)  
```

## 4. Handling Missing Values

```python  
df.fillna(method='ffill', inplace=True)  
```

## 5. Resample Data to Daily Frequency

```python  
daily\_df = df.resample('D').agg({  
 'Open': 'first',  
 'High': 'max',  
 'Low': 'min',  
 'Close': 'last',  
 'Volume\_(BTC)': 'sum',  
 'Volume\_(Currency)': 'sum',  
 'Weighted\_Price': 'mean'  
})  
```

## 6. Plotting the Close Price

```python  
plt.figure(figsize=(12, 6))  
plt.plot(daily\_df['Close'])  
plt.title('Bitcoin Close Price')  
plt.xlabel('Date')  
plt.ylabel('Close Price')  
plt.show()  
```

## 7. Seasonal Decomposition

```python  
decomposition = seasonal\_decompose(daily\_df['Close'], model='additive', period=30)  
plt.figure(figsize=(12, 8))  
plt.subplot(411)  
plt.plot(decomposition.observed)  
plt.subplot(412)  
plt.plot(decomposition.trend)  
plt.subplot(413)  
plt.plot(decomposition.seasonal)  
plt.subplot(414)  
plt.plot(decomposition.resid)  
plt.show()  
```

## 8. Feature Selection

```python  
features = ['Open', 'High', 'Low', 'Volume\_(BTC)', 'Volume\_(Currency)']  
X = daily\_df[features]  
y = daily\_df['Close']  
```

## 9. Train-Test Split

```python  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
```

## 10. Linear Regression Model

```python  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
y\_pred = model.predict(X\_test)  
```

## 11. Model Evaluation

```python  
mse = mean\_squared\_error(y\_test, y\_pred)  
rmse = np.sqrt(mse)  
print(f"RMSE: {rmse}")  
```

## 12. Plotting Predictions

```python  
plt.figure(figsize=(12, 6))  
plt.plot(y\_test.index, y\_test, label='Actual')  
plt.plot(y\_test.index, y\_pred, label='Predicted')  
plt.legend()  
plt.show()  
```