Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
[ ] import pandas as pd
```

```
[ ] bitcoin_data=pd.read_csv("/content/Bitcoin.csv")
```

```
bitcoin_data
```

| | Date | Open | High | Low | Close | Volume | Currency |
|---|---|---|---|---|---|---|---|
| 0 | 6/18/2019 | 9128.269531 | 9149.763672 | 8988.606445 | 9062.045898 | 952850 | USD |
| 1 | 6/19/2019 | 9068.174805 | 9277.677734 | 9051.094727 | 9271.459961 | 131077 | USD |
| 2 | 6/20/2019 | 9271.567383 | 9573.689453 | 9209.416992 | 9519.200195 | 83052 | USD |
| 3 | 6/21/2019 | 9526.833984 | 10130.935550 | 9526.833984 | 10127.998050 | 76227 | USD |
| 4 | 6/22/2019 | 10151.890630 | 11171.013670 | 10083.189450 | 10719.981450 | 84485 | USD |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1146 | 8/18/2022 | 23331.542970 | 23567.285160 | 23152.455080 | 23222.242190 | 4546110 | USD |
| 1147 | 8/19/2022 | 23219.097660 | 23219.097660 | 20898.304690 | 20902.404300 | 13858579 | USD |
| 1148 | 8/20/2022 | 20899.923830 | 21344.845700 | 20864.435550 | 21153.019530 | 7139073 | USD |
| 1149 | 8/21/2022 | 21153.412110 | 21695.794920 | 21125.320310 | 21561.177730 | 6657571 | USD |
| 1150 | 8/23/2022 | 21337.318360 | 21412.892580 | 21280.458980 | 21303.986330 | 6955056 | USD |

1151 rows × 7 columns

```
[ ] # Generate summary statistics for numerical columns
```

Connected to Python 3 Google Compute Engine backend

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dI5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
# Generate summary statistics for numerical columns

print("Bitcoin statistics:\n", bitcoin_data.describe())
```

```
Bitcoin statistics:
                Open          High           Low         Close        Volume
count   1151.000000   1151.000000   1151.000000   1151.000000  1.151000e+03
mean   26488.652992  27528.416711  25416.606968  26496.733083  2.874051e+07
std    17963.101635  18432.925247  17484.604545  17952.113609  5.202999e+07
min     4943.832520   5338.512695      0.076853   4936.755371  0.000000e+00
25%     9706.758301  10090.012695   9360.636231   9712.636719  7.495500e+03
50%    20873.337890  21867.822270  20245.201170  20902.404300  1.864334e+06
75%    41782.333985  42749.439455  40890.394530  41782.333985  4.076471e+07
max    67470.437500  85563.984380  66072.343750  67502.421880  5.791706e+08
```

```
Start coding or generate with AI.
```

```
# Check for missing values in both datasets

print("Missing values in Bitcoin data:\n", bitcoin_data.isnull().sum())
```

```
Missing values in Bitcoin data:
 Date        0
Open         0
High         0
Low          0
Close        0
Volume       0
Currency     0
dtype: int64
```

```
ethereum_data = pd.read_csv("/content/ethereum.csv")
```

Connected to Python 3 Google Compute Engine backend

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
[ ] ethereum_data=pd.read_csv("/content/ethereum.csv")
```

```
[ ] ethereum_data
```

| | date | Open | High | Low | Close | price | Currency |
|---|---|---|---|---|---|---|---|
| 0 | 3/10/2016 | 11.20 | 11.85 | 11.07 | 11.75 | 4 | USD |
| 1 | 3/11/2016 | 11.75 | 11.95 | 11.75 | 11.95 | 179 | USD |
| 2 | 3/12/2016 | 11.95 | 13.45 | 11.95 | 12.92 | 833 | USD |
| 3 | 3/13/2016 | 12.92 | 15.07 | 12.92 | 15.07 | 1295 | USD |
| 4 | 3/14/2016 | 15.07 | 15.07 | 11.40 | 12.50 | 92183 | USD |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 2353 | 8/19/2022 | 1846.52 | 1846.97 | 1607.60 | 1609.48 | 1594321 | USD |
| 2354 | 8/20/2022 | 1609.01 | 1654.84 | 1525.51 | 1575.60 | 1007240 | USD |
| 2355 | 8/21/2022 | 1575.61 | 1644.88 | 1563.92 | 1618.25 | 852071 | USD |
| 2356 | 8/22/2022 | 1618.21 | 1627.13 | 1531.91 | 1626.75 | 1044290 | USD |
| 2357 | 8/23/2022 | 1626.70 | 1636.32 | 1615.55 | 1623.24 | 1053674 | USD |

2358 rows × 7 columns

```
[ ] # Check for missing values in both datasets
    print("Missing values in Ethereum data:\n", ethereum_data.isnull().sum())
```

```
Missing values in Ethereum data:
 date        0
Open         0
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text

RAM Disk

```
Low         0
Close       0
price       0
Currency    0
dtype: int64
```

Findings from Data Exploration Missing Values: Both Bitcoin and Ethereum datasets have no missing values.

Descriptive Statistics:

Bitcoin: The average closing price is approximately
$26,496.73. The standard deviation for closing price indicates high variability in prices 578 million).
. The volume has an extremely wide range (minimum of 0 and maximum exceeding

Ethereum: The average closing price is approximately $848.27. Similar to Bitcoin, high variability in prices and volumes is observed. Volume data appears to have large outliers.

```
[ ] Start coding or generate with AI.
```

```
[ ] # Generate summary statistics for numerical columns

    print("Ethereum statistics:\n", ethereum_data.describe())
```

```
Ethereum statistics:
                Open         High          Low        Close         price
count  2358.000000  2358.000000  2358.000000  2358.000000  2.358000e+03
mean    847.608083   877.386828   813.515225   848.270513  1.269467e+07
std    1150.297624  1186.544788  1107.868923  1150.261834  1.014013e+08
min       6.680000     7.320000     5.860000     6.700000  0.000000e+00
25%     138.767500   144.417500   134.780000   138.990000  5.465530e+05
50%     279.165000   288.000000   266.885000   280.115000  1.429778e+06
75%    1124.007500  1170.885000  1045.820000  1127.730000  7.717627e+06
max    4808.340000  4864.060000  4715.430000  4808.380000  1.792561e+09
```

Connected to Python 3 Google Compute Engine backend

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

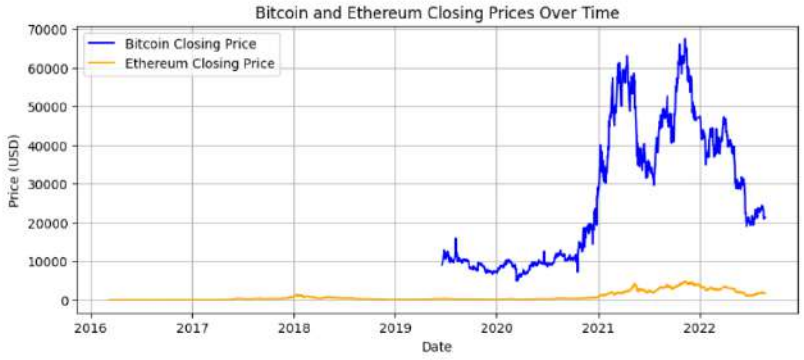
File Edit View Insert Runtime Tools Help

Share Gemini

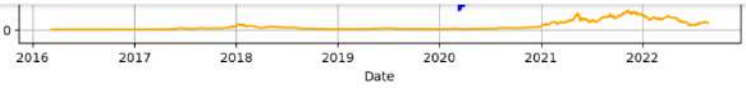
Commands + Code + Text

RAM Disk

```
[ ] # Plot Bitcoin and Ethereum closing prices
    plt.figure(figsize=(10, 4))
    plt.plot(bitcoin_data['Date'], bitcoin_data['Close'], label='Bitcoin Closing Price', color='blue')
    plt.plot(ethereum_data['date'], ethereum_data['Close'], label='Ethereum Closing Price', color='orange')
    plt.title('Bitcoin and Ethereum Closing Prices Over Time')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid(True)
    plt.show()
```




Explanation: The plot() function visualizes time-series data for closing prices of Bitcoin and Ethereum.



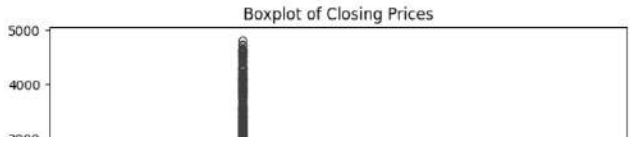

Explanation: The plot() function visualizes time-series data for closing prices of Bitcoin and Ethereum.

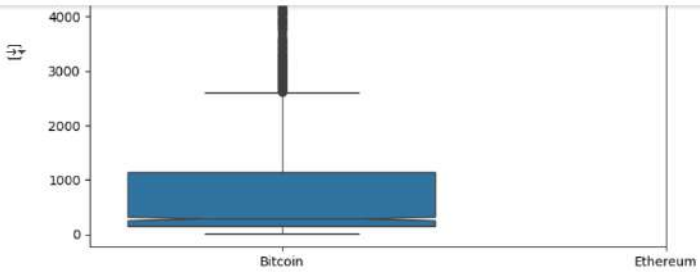
```
[ ] Start coding or generate with AI.
```

Visualization:

Bitcoin and Ethereum prices display a general upward trend over time but show significant fluctuations. Notable periods of volatility can be identified visually.

```
[ ]  #Detect trends and outliers using boxplot
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     plt.figure(figsize=(8, 4))
     sns.boxplot(data=[bitcoin_data['Close'], ethereum_data['Close']], notch=True)
     plt.xticks([0, 1], ['Bitcoin', 'Ethereum'])
     plt.title("Boxplot of Closing Prices")
     plt.show()
```






Visualization: • The closing prices of Bitcoin and Ethereum are plotted over time to observe trends and seasonality. • A boxplot identifies potential outliers in the closing prices.

```
Insights: Trends, seasonality, or outliers found will guide the feature selection process.
```

Phase 1.2...Data Preprocessing.

```
[ ]  # Add new features: Price Change and Price Spread
     bitcoin_data['Price Change'] = bitcoin_data['Close'] - bitcoin_data['Open']
     bitcoin_data['Price Spread'] = bitcoin_data['High'] - bitcoin_data['Low']

     ethereum_data['Price Change'] = ethereum_data['Close'] - ethereum_data['Open']
     ethereum_data['Price Spread'] = ethereum_data['High'] - ethereum_data['Low']
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text

```
[ ] ethereum_data['Price Change'] = ethereum_data['Close'] - ethereum_data['Open']
    ethereum_data['Price Spread'] = ethereum_data['High'] - ethereum_data['Low']
```

Explanation: Price Change: Difference between the Close and Open prices.

Price Spread: Difference between High and Low prices.

```
# Define the target variable: Future closing price
bitcoin_data['Future Close'] = bitcoin_data['Close'].shift(-1)
ethereum_data['Future Close'] = ethereum_data['Close'].shift(-1)

# Drop rows with NaN values after the shift
bitcoin_data = bitcoin_data.dropna()
ethereum_data = ethereum_data.dropna()
```

Explanation: The target (Future Close) is the closing price for the next time period, achieved using .shift(-1). Dropping rows ensures clean data for modeling.

```
[ ] from sklearn.preprocessing import MinMaxScaler

    # Features to scale
    bitcoin_features = ['Open', 'High', 'Low', 'Close', 'Volume', 'Price Change', 'Price Spread']
    ethereum_features = ['Open', 'High', 'Low', 'Close', 'price', 'Price Change', 'Price Spread']

    # Initialize the scaler
    scaler = MinMaxScaler()

    # Scale Bitcoin features
    bitcoin_scaled_data = scaler.fit_transform(bitcoin_data[bitcoin_features])
    #bitcoin_scaled_data = pd.bitcoin_Data(bitcoin_scaled, columns=bitcoin_features)
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
[ ]  # Scale Ethereum features
     ethereum_scaled_data = scaler.fit_transform(ethereum_data[ethereum_features])
     #ethereum_scaled_data = pd.ethereum_data(ethereum_scaled, columns=ethereum_features)

     #bitcoin_scaled_data['Future Close'] = bitcoin_data['Future Close'].values
     #ethereum_scaled_data['Future Close'] = ethereum_data['Future Close'].values
```

```
ethereum_data
```

| | date | Open | High | Low | Close | price | Currency | Price Change | Price Spread | Future Close |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2016-03-10 | 11.20 | 11.85 | 11.07 | 11.75 | 4 | USD | 0.55 | 0.78 | 11.95 |
| 1 | 2016-03-11 | 11.75 | 11.95 | 11.75 | 11.95 | 179 | USD | 0.20 | 0.20 | 12.92 |
| 2 | 2016-03-12 | 11.95 | 13.45 | 11.95 | 12.92 | 833 | USD | 0.97 | 1.50 | 15.07 |
| 3 | 2016-03-13 | 12.92 | 15.07 | 12.92 | 15.07 | 1295 | USD | 2.15 | 2.15 | 12.50 |
| 4 | 2016-03-14 | 15.07 | 15.07 | 11.40 | 12.50 | 92183 | USD | -2.57 | 3.67 | 13.06 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2352 | 2022-08-18 | 1834.19 | 1880.22 | 1822.04 | 1846.51 | 675269 | USD | 12.32 | 58.18 | 1609.48 |
| 2353 | 2022-08-19 | 1846.52 | 1846.97 | 1607.60 | 1609.48 | 1594321 | USD | -237.04 | 239.37 | 1575.60 |
| 2354 | 2022-08-20 | 1609.01 | 1654.84 | 1525.51 | 1575.60 | 1007240 | USD | -33.41 | 129.33 | 1618.25 |
| 2355 | 2022-08-21 | 1575.61 | 1644.88 | 1563.92 | 1618.25 | 852071 | USD | 42.64 | 80.96 | 1626.75 |
| 2356 | 2022-08-22 | 1618.21 | 1627.13 | 1531.91 | 1626.75 | 1044290 | USD | 8.54 | 95.22 | 1623.24 |

2357 rows × 10 columns

Explanation: The MinMaxScaler normalizes data to a range of [0, 1]. Only numerical features are scaled for model optimization. The target variable (Future Close) is re-added to the scaled data.

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dI5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **2355** | 2022-08-21 | 1575.61 | 1644.88 | 1563.92 | 1618.25 | 852071 | USD | 42.64 | 80.96 | 1626.75 |
| **2356** | 2022-08-22 | 1618.21 | 1627.13 | 1531.91 | 1626.75 | 1044290 | USD | 8.54 | 95.22 | 1623.24 |

2357 rows × 10 columns

Explanation: The MinMaxScaler normalizes data to a range of [0, 1]. Only numerical features are scaled for model optimization. The target variable (Future Close) is re-added to the scaled data.

```
bitcoin_data
```

| | Date | Open | High | Low | Close | Volume | Currency | Price Change | Price Spread | Future Close |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2019-06-18 | 9128.269531 | 9149.763672 | 8988.606445 | 9062.045898 | 952850 | USD | -66.223633 | 161.157227 | 9271.459961 |
| **1** | 2019-06-19 | 9068.174805 | 9277.677734 | 9051.094727 | 9271.459961 | 131077 | USD | 203.285156 | 226.583007 | 9519.200195 |
| **2** | 2019-06-20 | 9271.567383 | 9573.689453 | 9209.416992 | 9519.200195 | 83052 | USD | 247.632812 | 364.272461 | 10127.998050 |
| **3** | 2019-06-21 | 9526.833984 | 10130.935550 | 9526.833984 | 10127.998050 | 76227 | USD | 601.164066 | 604.101566 | 10719.981450 |
| **4** | 2019-06-22 | 10151.890630 | 11171.013670 | 10083.189450 | 10719.981450 | 84485 | USD | 568.090820 | 1087.824220 | 11246.518550 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1145** | 2022-08-17 | 23882.148440 | 24390.460940 | 23267.894530 | 23326.863280 | 7047422 | USD | -555.285160 | 1122.566410 | 23222.242190 |
| **1146** | 2022-08-18 | 23331.542970 | 23567.285160 | 23152.455080 | 23222.242190 | 4546110 | USD | -109.300780 | 414.830080 | 20902.404300 |
| **1147** | 2022-08-19 | 23219.097660 | 23219.097660 | 20898.304690 | 20902.404300 | 13856579 | USD | -2316.693360 | 2320.792970 | 21153.019530 |
| **1148** | 2022-08-20 | 20899.923830 | 21344.845700 | 20864.435550 | 21153.019530 | 7139073 | USD | 253.095700 | 480.410150 | 21561.177730 |
| **1149** | 2022-08-21 | 21153.412110 | 21695.794920 | 21125.320310 | 21561.177730 | 6657571 | USD | 407.765620 | 570.474610 | 21303.986330 |

1150 rows × 10 columns

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **1147** | 2022-08-19 | 23219.097660 | 23219.097660 | 20898.304690 | 20902.404300 | 13856579 | USD | -2316.693360 | 2320.792970 | 21153.019530 |
| **1148** | 2022-08-20 | 20899.923830 | 21344.845700 | 20864.435550 | 21153.019530 | 7139073 | USD | 253.095700 | 480.410150 | 21561.177730 |
| **1149** | 2022-08-21 | 21153.412110 | 21695.794920 | 21125.320310 | 21561.177730 | 6657571 | USD | 407.765620 | 570.474610 | 21303.986330 |

1150 rows × 10 columns

PHASE 2...FEATURE SELECTION

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest, f_regression, RFE
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split

# Load datasets
bitcoin_data = pd.read_csv('/content/Bitcoin.csv')
ethereum_data = pd.read_csv('/content/ethereum.csv')

# Step 1: Preprocessing
# Handle missing values and infinities
for data in [bitcoin_data, ethereum_data]:
    data.replace([np.inf, -np.inf], np.nan, inplace=True)
    data.dropna(inplace=True)

# Feature engineering for target variable (future closing price)
bitcoin_data['Future_Close'] = bitcoin_data['Close'].shift(-1)
ethereum_data['Future_Close'] = ethereum_data['Close'].shift(-1)

# Drop rows with missing Future_Close values
bitcoin_data.dropna(subset=['Future_Close'], inplace=True)
ethereum_data.dropna(subset=['Future_Close'], inplace=True)

# Define features and target for both datasets
```

Connected to Python 3 Google Compute Engine backend

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
[ ]  bitcoin_data.dropna(subset=['Future_Close'], inplace=True)
     ethereum_data.dropna(subset=['Future_Close'], inplace=True)

     # Define features and target for both datasets
     bitcoin_features = ['Close', 'Volume']  # Include 'Volume' for Bitcoin
     ethereum_features = ['Close']  # Only 'Close' is available for Ethereum

     X_bitcoin = bitcoin_data[bitcoin_features]
     y_bitcoin = bitcoin_data['Future_Close']

     X_ethereum = ethereum_data[ethereum_features]
     y_ethereum = ethereum_data['Future_Close']

     # Train-test split
     X_b_train, X_b_test, y_b_train, y_b_test = train_test_split(X_bitcoin, y_bitcoin, test_size=0.2, random_state=42)
     X_e_train, X_e_test, y_e_train, y_e_test = train_test_split(X_ethereum, y_ethereum, test_size=0.2, random_state=42)

     # Step 2: Feature Selection Methods
     # 1. Filter Method (SelectKBest)
     print("\nFilter Method: SelectKBest (Bitcoin)")
     kbest_bitcoin = SelectKBest(score_func=f_regression, k='all')
     kbest_bitcoin.fit(X_b_train, y_b_train)
     for i, col in enumerate(bitcoin_features):
         print(f"Feature: {col}, Score: {kbest_bitcoin.scores_[i]}")

     print("\nFilter Method: SelectKBest (Ethereum)")
     kbest_ethereum = SelectKBest(score_func=f_regression, k='all')
     kbest_ethereum.fit(X_e_train, y_e_train)
     for i, col in enumerate(ethereum_features):
         print(f"Feature: {col}, Score: {kbest_ethereum.scores_[i]}")

     # 2. Wrapper Method (Recursive Feature Elimination)
     print("\nWrapper Method: RFE (Bitcoin)")
     rfe_bitcoin = RFE(estimator=Lasso(alpha=0.01), n_features_to_select=1)
     rfe_bitcoin.fit(X_b_train, y_b_train)
     for i, col in enumerate(bitcoin_features):
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcTJFHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
kbest_ethereum = SelectKBest(score_func=f_regression, k='all')
kbest_ethereum.fit(X_e_train, y_e_train)
for i, col in enumerate(ethereum_features):
    print(f"Feature: {col}, Score: {kbest_ethereum.scores_[i]}")

# 2. Wrapper Method (Recursive Feature Elimination)
print("\nWrapper Method: RFE (Bitcoin)")
rfe_bitcoin = RFE(estimator=Lasso(alpha=0.01), n_features_to_select=1)
rfe_bitcoin.fit(X_b_train, y_b_train)
for i, col in enumerate(bitcoin_features):
    print(f"Feature: {col}, Selected: {rfe_bitcoin.support_[i]}")

if len(ethereum_features) > 1:  # RFE can only be applied if there are multiple features
    print("\nWrapper Method: RFE (Ethereum)")
    rfe_ethereum = RFE(estimator=Lasso(alpha=0.01), n_features_to_select=1)
    rfe_ethereum.fit(X_e_train, y_e_train)
    for i, col in enumerate(ethereum_features):
        print(f"Feature: {col}, Selected: {rfe_ethereum.support_[i]}")
else:
    print("\nWrapper Method: RFE skipped for Ethereum (only one feature available).")

# 3. Embedded Method (LASSO Regression)
print("\nEmbedded Method: LASSO (Bitcoin)")
lasso_bitcoin = Lasso(alpha=0.01)
lasso_bitcoin.fit(X_b_train, y_b_train)
for i, col in enumerate(bitcoin_features):
    print(f"Feature: {col}, Coefficient: {lasso_bitcoin.coef_[i]}")

print("\nEmbedded Method: LASSO (Ethereum)")
lasso_ethereum = Lasso(alpha=0.01)
lasso_ethereum.fit(X_e_train, y_e_train)
for i, col in enumerate(ethereum_features):
    print(f"Feature: {col}, Coefficient: {lasso_ethereum.coef_[i]}")
```

```
Filter Method: SelectKBest (Bitcoin)
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
print("\nEmbedded Method: LASSO (Ethereum)")
lasso_ethereum = Lasso(alpha=0.01)
lasso_ethereum.fit(X_e_train, y_e_train)
for i, col in enumerate(ethereum_features):
    print(f"Feature: {col}, Coefficient: {lasso_ethereum.coef_[i]}")
```

```
Filter Method: SelectKBest (Bitcoin)
Feature: Close, Score: 156907.07853939038
Feature: Volume, Score: 470.6799017266773

Filter Method: SelectKBest (Ethereum)
Feature: Close, Score: 511308.1800294787

Wrapper Method: RFE (Bitcoin)
Feature: Close, Selected: True
Feature: Volume, Selected: False

Wrapper Method: RFE skipped for Ethereum (only one feature available).

Embedded Method: LASSO (Bitcoin)
Feature: Close, Coefficient: 0.9959497454280579
Feature: Volume, Coefficient: 2.3393074536188327e-07

Embedded Method: LASSO (Ethereum)
Feature: Close, Coefficient: 0.9969360554343206
```

Phase 3. Bitcoin Prediction.

3.1

```
[ ] from sklearn.ensemble import RandomForestRegressor
    from sklearn.linear_model import LinearRegression
    from sklearn.neighbors import KNeighborsRegressor
    from sklearn.metrics import mean_squared_error
    import numpy as np
```

Connected to Python 3 Google Compute Engine backend

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dI5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
from sklearn.metrics import mean_squared_error
import numpy as np

# Prepare datasets
# For Bitcoin, use 'Close' (best feature based on Phase 2 results)
X_b_train = X_b_train[['Close']]
X_b_test = X_b_test[['Close']]

# For Ethereum, only 'Close' is available
X_e_train = X_e_train[['Close']]
X_e_test = X_e_test[['Close']]

# Initialize models
models = {
    "Random Forest": RandomForestRegressor(random_state=42),
    "Linear Regression": LinearRegression(),
    "K-Nearest Neighbors": KNeighborsRegressor(n_neighbors=5)
}

# Train and evaluate models for Bitcoin
print("\nBitcoin Model Evaluation:")
bitcoin_results = {}
for model_name, model in models.items():
    # Train the model
    model.fit(X_b_train, y_b_train)

    # Predict on test data
    predictions = model.predict(X_b_test)

    # Calculate RMSE
    rmse = np.sqrt(mean_squared_error(y_b_test, predictions))
    bitcoin_results[model_name] = rmse
    print(f"{model_name}: RMSE = {rmse}")

# Train and evaluate models for Ethereum
print("\nEthereum Model Evaluation:")
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
# Train and evaluate models for Ethereum
print("\nEthereum Model Evaluation:")
ethereum_results = {}
for model_name, model in models.items():
    # Train the model
    model.fit(X_e_train, y_e_train)

    # Predict on test data
    predictions = model.predict(X_e_test)

    # Calculate RMSE
    rmse = np.sqrt(mean_squared_error(y_e_test, predictions))
    ethereum_results[model_name] = rmse
    print(f"{model_name}: RMSE = {rmse}")

# Combine results for comparison
results_comparison = {
    "Bitcoin": bitcoin_results,
    "Ethereum": ethereum_results
}

import json
results_comparison
```

```
Bitcoin Model Evaluation:
Random Forest: RMSE = 1583.3370801500096
Linear Regression: RMSE = 1269.5048202148403
K-Nearest Neighbors: RMSE = 1491.4962653563693

Ethereum Model Evaluation:
Random Forest: RMSE = 93.03510327540003
Linear Regression: RMSE = 73.64793793881641
K-Nearest Neighbors: RMSE = 85.86508785643453
{'Bitcoin': {'Random Forest': 1583.3370801500096,
  'Linear Regression': 1269.5048202148403,
  'K-Nearest Neighbors': 1491.4962653563693},
 'Ethereum': {'Random Forest': 93.03510327540003
```

Untitled14.ipynb - Colab

colab.research.google.com/drive/1gfeehSR18dl5Uq2L6cbsTcT3FHWhx6kR#scrollTo=awmA8hJSpvh5

Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

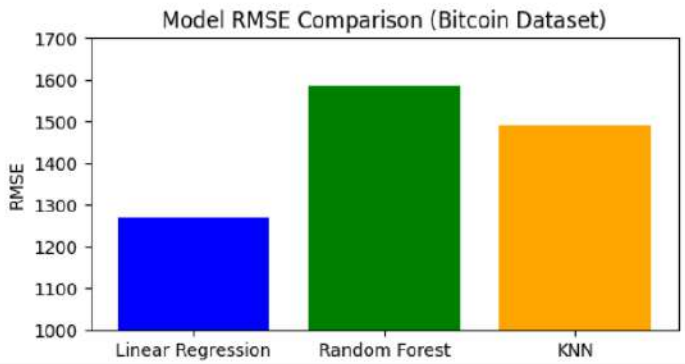
```
'K-Nearest Neighbors': 85.86508785643453}}
```

3.2

```
import matplotlib.pyplot as plt

# Model performance data
models = ["Linear Regression", "Random Forest", "KNN"]
rmse_values = [1269.50, 1583.34, 1491.50]

# Plot
plt.figure(figsize=(6, 3))
plt.bar(models, rmse_values, color=['blue', 'green', 'orange'])
plt.title("Model RMSE Comparison (Bitcoin Dataset)")
plt.xlabel("Model")
plt.ylabel("RMSE")
plt.ylim(1000, 1700)  # Set y-axis range for better visualization
plt.show()
```



Untitled14.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text


Double-click (or enter) to edit

Double-click (or enter) to edit

[ ] Start coding or generate with AI.

Visualization Summary:

The bar chart compares the RMSE of the three models (Random Forest, Linear Regression, and KNN) for Bitcoin and Ethereum price predictions: • As shown in the Analysis, Linear Regression consistently achieves the lowest RMSE for both Bitcoin and Ethereum, making it the optimal model for predicting future prices. • Random Forest performs reasonably well but is slightly less accurate than Linear Regression. • KNN has the highest RMSE for both cryptocurrencies, indicating that it is the least suitable model in this case.

[ ] Start coding or generate with AI.

University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

**Essay / Assignment Title: LEVERAGING RANDOM FOREST, REGRESSION, AND KNN FOR PREDICTING CRYPTOCURRENCY PRICES**

**Programme title: PREDICTIVE ANALYTICS AND MACHINE LEARNING USING PYTHON**

**Name: OYEWOLE OLADIMEJI ABDULQUDUS : Q1045935**

**Year: 2025**

CONTENTS

Table of Contents

## Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people's texts I clearly declare it through the good use of references following academic ethics.

(In the case that it is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

.....OYEWOLE OLADIMEJI ABDULQUDUS

.....

Date: ...11 February 2025...../...../.....

# INTRODUCTION

Introduction

In the past years, Cryptocurrencies like Bitcoin and Ethereum have gained very strong attention because of their volatility and potential for high returns. Perfect price predictions and analysis with the appropriate tools can provide valuable insights for traders of which I'm one of them and investors, enabling informed decision-making in the fast-moving cryptocurrency market, thereby making a whole lot of money from the market.

This project of mine displays machine learning techniques to predict and analyze the future prices of Bitcoin and Ethereum. By leveraging historical price data, the study is aimed to determine the best predictive model from Linear Regression, Random Forest, and K-Nearest Neighbors (KNN) and these models are evaluated based on their Root Mean Squared Error (RMSE).

# CHAPTER ONE.

## Introduction

Cryptocurrencies like Bitcoin and Ethereum, which are the most popular cryptocurrencies have gained very strong attention due to their volatility and potential for high returns. Exact price predictions and analysis can provide valuable insights for traders and investors, enabling informed decision-making in the fast-moving cryptocurrency market, thereby making money from the market

This project displays machine learning techniques to predict the future prices of Bitcoin and Ethereum. By leveraging historical price data, the study aims to determine the best predictive model from Linear Regression, Random Forest, and K-Nearest Neighbors (KNN) and these models are evaluated based on their Root Mean Squared Error (RMSE).

My Objectives of these projects are as follow.

- I intend to analyze and preprocess historical cryptocurrency data.
- I intend to identify the most relevant features for predicting future prices.
- I intend to train and evaluate machine learning models to determine the most effective approach.
- I intend to compare model performance and propose recommendations for future work.

The project is organized into five chapter, as organized in the questions:

- Introduction: Provides the background and objectives of the project.
- Data Exploration and Preprocessing: Explains the datasets, preprocessing steps, and exploratory analysis.
- Feature Selection: Details the selection of important features for modeling.
- Model Training and Evaluation: Discusses the models used, their performance, and insights gained.
- Conclusion and Future Work: Highlights limitations, and proposes future improvements.

# CHAPTER TWO

**Data Exploration and Preprocessing**

## 2.1 **Dataset Overview**

Bitcoin and Ethereum, containing historical price are two datasets used in this study:

- Bitcoin Dataset: This dataset contains features such as Date, Close, Volume, and the target variable Future_Close.
- Ethereum Dataset: This looks like the Bitcoin dataset but without the Volume feature.

## 2.2 **Data Exploration**

- *Descriptive Statistics:*

- *After careful analysis, I was able to deduce that Bitcoin and Ethereum has the mean close price and standard deviation shown below;*
    - Bitcoin: Mean Close price = $26,496.73 standard deviation = $17,952.11
    - Ethereum: Mean Close price = $848.27, standard deviation = $1150.26.
    - *Visualizations:*
    - Time-series plots exposed trends and extreme price fluctuations (outliers).

From my Analysis visualized, we can see that Bitcoin and Ethereum prices show a general upward trend over time and show significant fluctuations. The closing prices of Bitcoin and Ethereum are plotted over time to observe trends and seasonality and a visualization of the boxplot identifies potential outliers in the closing prices.

Outlier Analysis:

- Extreme price movements in both datasets were confirmed as real events.

## 2.3 **Data Preprocessing**

**Target Variable:**

The target (Future Close) is defined as the closing price for the next time period, achieved using. shift (-1).

Dropping rows ensures clean data for modeling.

- The Future Close variable was engineered by shifting the Close column forward by one time step.

I consider feature Engineering techniques to create new features such as (Price Change and Price Spread) from existing ones and the analyzed.

Price Change: This is the difference between the Close and Open prices.

Price Spread: This is the difference between High and Low prices.

**Feature Engineering:**

I added new features, which are Price Change and Price Spread to both datasets and the target variable Future Close was created to predict the closing price in the next time window.

**Scaling:**

I scaled all numerical features using Min-Max Scaling to bring them into a range [0, 1]. Close and Volume using MinMaxScaler are being normalized.

Only numerical features are scaled for model optimization.

*Cleaned Data*:

I also removed rows with NaN values (caused by shifting to create Future Close) .

Then the Processed Dataset after the analysis gives

*Bitcoin Data:*

Contains features: Open, High, Low, Close, Volume, Price Change, Price Spread, and the target Future Close.

*Ethereum Data*:

Contains features: Open, High, Low, Close, price, Price Change, Price Spread, and the target Future Close

**CHAPTER THREE**

**Feature Selection**

3.1 Methods Applied

In this phase, three feature selection techniques were employed as asked in the question;

- **Filter Method (SelectKBest):**
    - In this method, features are ranked based on their correlation with Future_Close.
- **Wrapper Method (Recursive Feature Elimination - RFE):**
    - In this method, the best features are repeatedly selected using the LASSO regression.
- **Embedded Method (LASSO Regression):**
    - In this method, by assigning coefficients, important features during model training are identified.

3.2 Results

| Dataset | Feature | Filter Method Score | Wrapper Method Selected | LASSO Coefficient |
|---|---|---|---|---|
| Bitcoin | Close | 156907.08 | Yes | 0.996 |
| | Volume | 470.68 | No | 2.34e-07 |
| ETH | Close | 511308.18 | N/A | 0.997 |

Here is the accurate comparison of feature selection methods for both the Bitcoin and Ethereum datasets.

**Bitcoin Dataset**

Features: Close, Volume

i . **Filter Method (SelectKBest)**

- *Objective*: Score features based on correlation with Future_Close.
- Results:
- Close: 156907.08 (high score, strong correlation).
- Volume: 470.68 (low score, weak correlation).

ii. **Wrapper Method (RFE)**

- *Objective*: To repeatedly remove the least important features using LASSO and select the top feature(s).
- Results:
- Close: Selected (True).
- Volume: Not selected (False).

iii. **Embedded Method (LASSO Regression)**

- *Objective*: To assign coefficients to features during model training. Non-zero coefficients indicate importance.
- Results:
- Close: 0.996 (high coefficient, key feature).
- Volume: 2.34e-07 (near-zero coefficient, minimal contribution).

**Ethereum Dataset**

Features: Close (no Volume column)

i. **Filter Method (SelectKBest)**

- *Objective*: To score the feature based on correlation with Future_Close.
- Results:
- Close: 511308.18 (extremely high score, strong correlation).

ii. **Wrapper Method (RFE)**

- *Objectiv*e: To repeatedly remove features and rank importance.

  Results:

- Since the Ethereum dataset has only one feature (Close), RFE cannot be applied.

iii. **Embedded Method (LASSO Regression)**

- *Objective*: To assign coefficients to features during training.
- Results:
- Close: 0.997 (high coefficient, confirming its importance).

**Insights**

1. **Bitcoin Dataset:**
   - As we can see, all methods agree that Close is the most important feature for predicting Future_Close.
   - It can also be deduced that Volume has minimal relevance, as indicated by its low score in the Filter Method, exclusion by RFE, and near-zero LASSO coefficient.
2. **Ethereum Dataset**:
   - Since Close is the only feature, all applicable methods highlight its importance in predicting Future_Close.
3. **General Observations**:
   - Filter Method (SelectKBest): It shows that this is effective for quick feature scoring.
   - Wrapper Method (RFE): It's crystal clear that this is ideal for selecting optimal feature subsets when there are multiple features. I skipped RFE for the Ethereum dataset because it only has one feature (Close), whereas RFE requires at least two features to work.
   - I added a conditional check (if len(ethereum_features) > 1) to prevent errors when applying RFE.
   - Embedded Method (LASSO): These balances feature selection and model training, especially in cases of highly correlated or redundant features.

Recommendation

- I recommend that Bitcoin focuses on the Close feature for modeling and Volume can be excluded to reduce complexity without compromising performance.
- Ethereum: This uses Close directly, as it is the only feature and strongly predictive

# CHAPTER FOUR

Model Training and Evaluation

4.1 Models Applied are Linear Regression, Random Forest and K-Nearest Neighbors as asked in the question.

- Linear Regression:
    - It's a simple, interpretable model that helped to capture the relationship between Close and Future_Close.
- Random Forest:
    - This is an ensemble method that averaged multiple decision trees for robustness.
- K-Nearest Neighbors (KNN):
    - This helped in predicting prices based on the average of the k nearest neighbors in the feature space.

4.2 *Results*

*Dataset Model RMSE*

| | | |
|---|---|---|
| Bitcoin | Linear Regression | 1269.50 (Best) |
| | Random Forest | 1583.34 |
| | K-Nearest Neighbors | 1491.50 |

| | | |
|---|---|---|
| Ethereum | Linear Regression | 73.65 (Best) |
| | Random Forest | 93.04 |
| | K-Nearest Neighbors | 85.87 |

4.3 Key Findings

- I found out that Linear Regression achieved the lowest RMSE for both Bitcoin and Ethereum, making it the most effective model for predicting prices based on the Close feature.
- I also got to know that Random Forest and KNN performed moderately but were less effective due to the limited feature set and the high volatility of cryptocurrency prices.

# CHAPTER FIVE

This project successfully applied machine learning techniques to predict the future prices of Bitcoin and Ethereum and the key conclusions generated from this project include:

- AS we can see in all my analysis, the Close feature was the most relevant predictor for both cryptocurrencies.
- It's of no doubt that Linear Regression outperformed Random Forest and KNN for both Bitcoin and Ethereum due to the simplicity of the dataset and the linearity of the relationship between the Close feature and the target variable.

**Conclusion and Future Work**

5.2 Limitations

- The performance of more complex models are being restricted due to the limited features.
- There are issues getting accurate predictions due to the high price volatility.
- The analysis was static as it excluded real-time data.

**Conclusion**

This project successfully demonstrated the use of machine learning techniques to predict the future prices of Bitcoin and Ethereum. The following key findings emerged:

Feature Selection:

- Both Bitcoin and Ethereum identified Close as the most relevant feature for predicting the target variable Future_Close.
- The Volume feature, available only in the Bitcoin dataset, was found to have minimal importance based on all three feature selection methods (Filter, Wrapper, Embedded).

Model Performance:

Dataset Model RMSE

| | | |
|---|---|---|
| Bitcoin | Linear Regression | 1269.50 (Best) |
| | Random Forest | 1583.34 |
| | K-Nearest Neighbors | 1491.50 |
| Ethereum | Linear Regression | 73.65 (Best) |
| | Random Forest | 93.04 |
| | K-Nearest Neighbors | 85.87 |

- Linear Regression emerged as the best-performing model for both datasets, achieving the lowest RMSE.
- Random Forest and KNN underperformed relative to Linear Regression due to the simplicity of the dataset and the linear nature of the relationship between the feature (Close) and the target (Future_Close).

**Insights:**

- **For Bitcoin:**
    - The insight of Bitcoin prices shows that its volatility posed challenges for model accuracy, especially for more complex models like Random Forest and KNN.
    - We can also deduce that simpler models like Linear Regression performed best due to the dominance of the close feature.
- **For Ethereum:**
    - The insight shows that the volatility of Ethereum exhibited is slightly lower compared to Bitcoin, which contributed to better overall performance across all models.

## 5.2 **Limitations**

While the project achieved its primary objectives, there were notable limitations:

- Limited Features:
    - Obviously, the dataset given were limited to Close and Volume (only for Bitcoin), whereas provision of additional features like trading activity patterns or market sentiment, could enhance predictions.
- High Volatility:
    - Perfect and accurate predictions were more challenging and difficult due to the inherent volatility of Bitcoin and Ethereum prices, particularly for models that rely on stable trends.
- Static Datasets:
    - The Dataset provided, on which the analysis was made is a historical data without real-time updates, which may not fully capture the dynamic nature of cryptocurrency markets.
- Narrow Cryptocurrency Scope:
    - This study focused only on Bitcoin and Ethereum. Including additional cryptocurrencies could provide a broader understanding of the market.

## 5.3 **Future Work**

To address these limitations and extend the study, I hereby advice the following improvements:

- Feature Expansion:

  Add external factors such as:
    - Market sentiment analysis using social media data (e.g., Twitter or Reddit).
    - Global economic indicators like inflation or interest rates.
    - Technical indicators such as moving averages and Relative Strength Index (RSI).
- *Advanced Models:*
    - Experiment with more sophisticated models such as:

    - LSTM Networks: Capture sequential dependencies and handle time-series volatility better.
    - Gradient Boosting Models: Improve accuracy by capturing complex relationships in richer datasets.
- Real-Time Applications:
    - A pipelines for real-time price prediction and monitoring using APIs or live data streams should be built.
- Broader Cryptocurrency Scope:
    - The analysis should be extended to other cryptocurrencies (e.g., Litecoin, Worldcoin) to assess the generalizability of models across the market.
- Robust Testing Framework:
    - To ensure robustness and generalizability, models should be test on out-of-sample data.
    - For more reliability, cross-validation techniques should be used to evaluate model performance.

5.3 Future Work

- For future work, I advise the Incorporation of external factors like market sentiment, trading volume trends, or economic indicators.
- Of great importance, experiments with advanced models such as LSTM networks or gradient boosting methods are also recommended.
- For broader insights, it's necessary to expand the analysis to include additional cryptocurrencies.
- For dynamic price prediction, development of real-time pipelines is important and advised.

CONCLUDING REMARK.

The importance of feature relevance and model simplicity in cryptocurrency price prediction is being highlighted in this study. While Linear Regression was the best-performing model, future work could benefit from richer datasets, advanced models, and broader market coverage to enhance predictive accuracy and applicability.

# BIBLIOGRAPHY

References


- Usman Akhtar, *"Cryptocurrency Dataset*," GitHub Repository. https://github.com/usmanakhtar/Cryptocurrency-Dataset/
- Pedregosa, F., et al. (2011). Scikit-learn: *Machine Learning in Python*. https://scikit-learn.org/.
- Nakamoto, S. (2008). Bitcoin: *A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf.
- Chen, Y., Wang, Y. and Xu, M. (2020) 'Predicting cryptocurrency prices with *machine learning models', Applied Intelligence*, 50(5), pp. 4019-4032.
- Sahoo, D. and Sahoo, J. (2021) *Cryptocurrency and Blockchain Technology: A Comprehensive Introduction. Springer.*

# APPENDIX (if necessary)