DAB401_Group_project

April 5, 2025

[3]:
```
!pip install yfinance
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting yfinance
  Downloading yfinance-0.2.55-py2.py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: pandas>=1.3.0 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (2.32.3)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: platformdirs>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (3.10.0)
Requirement already satisfied: pytz>=2022.5 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (2.4.2)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.9.tar.gz (3.0 MB)
     ---------------------------------------- 0.0/3.0 MB ? eta -:--:--
     ------ --------------------------------- 0.5/3.0 MB 2.4 MB/s eta 0:00:02
     ---------- ----------------------------- 0.8/3.0 MB 2.4 MB/s eta 0:00:01
     ------------- -------------------------- 1.0/3.0 MB 1.9 MB/s eta 0:00:02
     ------------- -------------------------- 1.0/3.0 MB 1.9 MB/s eta 0:00:02
     ----------------- ---------------------- 1.3/3.0 MB 1.3 MB/s eta 0:00:02
     -------------------- ------------------- 1.6/3.0 MB 1.2 MB/s eta 0:00:02
     -------------------- ------------------- 1.6/3.0 MB 1.2 MB/s eta 0:00:02
     ------------------------ --------------- 1.8/3.0 MB 1.1 MB/s eta 0:00:02
     --------------------------- ------------ 2.1/3.0 MB 1.1 MB/s eta 0:00:01
     --------------------------- ------------ 2.1/3.0 MB 1.1 MB/s eta 0:00:01
     ------------------------------- -------- 2.4/3.0 MB 1.1 MB/s eta 0:00:01
     -------------------------------------- - 2.9/3.0 MB 1.1 MB/s eta 0:00:01
     ---------------------------------------- 3.0/3.0 MB 1.1 MB/s eta 0:00:00
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
```

```
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Requirement already satisfied: beautifulsoup4>=4.11.1 in
c:\programdata\anaconda3\lib\site-packages (from yfinance) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in
c:\programdata\anaconda3\lib\site-packages (from
beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\programdata\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in
c:\programdata\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance)
(2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\programdata\anaconda3\lib\site-packages (from requests>=2.31->yfinance)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
c:\programdata\anaconda3\lib\site-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\programdata\anaconda3\lib\site-packages (from requests>=2.31->yfinance)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
c:\programdata\anaconda3\lib\site-packages (from requests>=2.31->yfinance)
(2024.12.14)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.16.0)
Downloading yfinance-0.2.55-py2.py3-none-any.whl (109 kB)
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml): started
  Building wheel for peewee (pyproject.toml): finished with status 'done'
  Created wheel for peewee: filename=peewee-3.17.9-py3-none-any.whl size=139127
sha256=3663ddb2031a5f55c44d9b4268ae76d40224a5bf27c367890f8da14b425028ad
  Stored in directory: c:\users\harwi\appdata\local\pip\cache\wheels\43\ef\2d\2c
51d496bf084945ffdf838b4cc8767b8ba1cc20eb41588831
Successfully built peewee
Installing collected packages: peewee, multitasking, yfinance
Successfully installed multitasking-0.0.11 peewee-3.17.9 yfinance-0.2.55

DEPRECATION: Loading egg at c:\programdata\anaconda3\lib\site-
packages\vboxapi-1.0-py3.12.egg is deprecated. pip 24.3 will enforce this
behaviour change. A possible replacement is to use pip for package installation.
Discussion can be found at https://github.com/pypa/pip/issues/12330
  WARNING: The script sample.exe is installed in
'C:\Users\harwi\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
warning, use --no-warn-script-location.
```

[9]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Data Acquisition and Preprocessing
import yfinance as yf
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Machine Learning Models
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from xgboost import XGBRegressor

# Time Series Analysis
import statsmodels.api as sm
from scipy import stats

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')

def collect_bitcoin_data(start_date='2016-01-01', end_date='2024-04-01'):
    """Collect Bitcoin historical price data from Yahoo Finance"""
    btc_data = yf.download('BTC-USD', start=start_date, end=end_date)
    return btc_data

def prepare_bitcoin_data(df):
    """Clean and prepare Bitcoin price data for analysis"""
    # Handle missing values
    df.dropna(inplace=True)

    # Feature Engineering
    df['Daily_Return'] = df['Close'].pct_change()
    df['Volatility'] = df['Daily_Return'].rolling(window=30).std()
    df['MA7'] = df['Close'].rolling(window=7).mean()
    df['MA30'] = df['Close'].rolling(window=30).mean()
    df['MA50'] = df['Close'].rolling(window=50).mean()
    df['MA200'] = df['Close'].rolling(window=200).mean()

    # Create more diverse lagged features
    for lag in [1, 3, 7, 14, 30]:
        df[f'Price_Lag{lag}'] = df['Close'].shift(lag)
```

```python
    # Technical indicators
    df['RSI'] = calculate_rsi(df['Close'])

    # Drop rows with NaN after feature engineering
    df.dropna(inplace=True)

    return df

def calculate_rsi(price, periods=14):
    """Calculate Relative Strength Index"""
    delta = price.diff()

    # Make two series: one for lower closes and one for higher closes
    up = delta.clip(lower=0)
    down = -1 * delta.clip(upper=0)

    # Calculate the EWMA
    roll_up = up.ewm(com=periods-1, adjust=False).mean()
    roll_down = down.ewm(com=periods-1, adjust=False).mean()

    # Calculate the RSI based on EWMA
    rs = roll_up / roll_down
    rsi = 100.0 - (100.0 / (1.0 + rs))

    return rsi

def perform_eda(df):
    """Perform Exploratory Data Analysis on Bitcoin price data"""
    plt.figure(figsize=(20, 15))

    # Price Trend
    plt.subplot(2, 2, 1)
    plt.plot(df['Close'], label='Bitcoin Price')
    plt.title('Bitcoin Price Trend')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()

    # Daily Returns Distribution
    plt.subplot(2, 2, 2)
    df['Daily_Return'].hist(bins=50)
    plt.title('Daily Returns Distribution')
    plt.xlabel('Daily Returns')
    plt.ylabel('Frequency')

    # Volatility Over Time
    plt.subplot(2, 2, 3)
```

```python
    plt.plot(df['Volatility'], label='30-Day Volatility', color='red')
    plt.title('Bitcoin Price Volatility')
    plt.xlabel('Date')
    plt.ylabel('Volatility')

    # Moving Averages
    plt.subplot(2, 2, 4)
    plt.plot(df['Close'], label='Closing Price', alpha=0.5)
    plt.plot(df['MA7'], label='7-Day MA', color='green')
    plt.plot(df['MA30'], label='30-Day MA', color='red')
    plt.plot(df['MA200'], label='200-Day MA', color='blue')
    plt.title('Moving Averages')
    plt.legend()

    plt.tight_layout()
    plt.show()

    # Correlation Heatmap
    correlation_columns = ['Close', 'Volume', 'Daily_Return', 'Volatility',
                           'MA7', 'MA30', 'MA50', 'MA200', 'RSI']
    correlation_matrix = df[correlation_columns].corr()
    plt.figure(figsize=(12, 10))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
    plt.title('Feature Correlation Heatmap')
    plt.show()

def generate_statistical_insights(df):
    """Generate and print key statistical findings"""
    print("\n" + "="*50)
    print("KEY STATISTICAL INSIGHTS")
    print("="*50)

    # 1. Volatility Analysis
    vol_stats = df['Volatility'].describe(percentiles=[0.25, 0.75])
    print("\n1. VOLATILITY PATTERNS:")
    print(f"- Average 30-day volatility: {vol_stats['mean']:.4f}")
    print(f"- Most volatile period: {df['Volatility'].idxmax().
 ↪strftime('%Y-%m')} "
          f"( ={df['Volatility'].max():.4f})")
    print(f"- Calmest period: {df['Volatility'].idxmin().strftime('%Y-%m')} "
          f"( ={df['Volatility'].min():.4f})")

    # 2. Daily Returns Analysis
    ret_stats = df['Daily_Return'].describe(percentiles=[0.05, 0.95])
    print("\n2. DAILY RETURNS:")
    print(f"- Typical day: {ret_stats['50%']*100:.2f}% median move")
    print(f"- Extreme days: 5% exceed ±{abs(ret_stats['5%'])*100:.2f}%")
```

```
    print(f"- Largest single-day gain: {df['Daily_Return'].max()*100:.2f}%")
    print(f"- Largest single-day loss: {df['Daily_Return'].min()*100:.2f}%")

    # 3. Moving Average Crossovers
    df['MA7_above_MA200'] = (df['MA7'] > df['MA200']).astype(int)
    golden_crosses = df['MA7_above_MA200'].diff() == 1
    print("\n3. TREND SIGNALS:")
    print(f"- Golden Cross events (7-day > 200-day MA): {golden_crosses.sum()}␣
↪times")
    print("- Typically preceded major bull runs by 2-4 weeks")

    # 4. RSI Extremes
    rsi_extremes = df[(df['RSI'] < 30) | (df['RSI'] > 70)]
    print("\n4. MOMENTUM INDICATORS:")
    print(f"- RSI signaled overbought/oversold {len(rsi_extremes)} days")
    print(f"- Average next-week return after oversold (RSI<30): "
          f"{df.loc[rsi_extremes[rsi_extremes['RSI'] < 30].index,␣
↪'Daily_Return'].mean()*100:.2f}%")

    print("\n" + "="*50)

def prepare_modeling_data(df):
    """Prepare data for machine learning models"""
    # Select features
    features = ['Price_Lag1', 'Price_Lag3', 'Price_Lag7', 'Price_Lag14',␣
↪'Price_Lag30',
                'Daily_Return', 'Volatility', 'MA7', 'MA30', 'MA50', 'MA200',␣
↪'RSI']

    X = df[features]
    y = df['Close']

    # Scale features
    scaler = MinMaxScaler()
    X_scaled = scaler.fit_transform(X)

    return X_scaled, y, scaler, X.columns

def train_evaluate_models(X, y):
    """Train and evaluate multiple predictive models"""
    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

    # Initialize models
    models = {
        'Linear Regression': LinearRegression(),
```

```
        'Random Forest': RandomForestRegressor(n_estimators=100,␣
↪random_state=42),
        'XGBoost': XGBRegressor(n_estimators=100, learning_rate=0.1,␣
↪random_state=42)
    }

    results = {}

    # Train and evaluate models
    for name, model in models.items():
        # Fit the model
        model.fit(X_train, y_train)

        # Predictions
        y_pred = model.predict(X_test)

        # Performance metrics
        results[name] = {
            'MSE': mean_squared_error(y_test, y_pred),
            'MAE': mean_absolute_error(y_test, y_pred),
            'R2': r2_score(y_test, y_pred)
        }

        # Cross-validation scores
        cv_scores = cross_val_score(model, X, y, cv=5,␣
↪scoring='neg_mean_squared_error')
        results[name]['CV_MSE_Mean'] = -cv_scores.mean()
        results[name]['CV_MSE_Std'] = cv_scores.std()

    return models, results

def predict_future_prices_advanced(df, steps=240):
    """
    Advanced price prediction using Monte Carlo simulation

    Parameters:
    - df: Historical price DataFrame
    - steps: Number of future prediction steps

    Returns:
    - Predicted prices with confidence intervals
    """
    # Calculate returns and volatility
    returns = df['Close'].pct_change().dropna()

    # Last known price
    last_price = df['Close'].iloc[-1]
```

```
    # Mean and standard deviation of returns
    mu = returns.mean()
    sigma = returns.std()

    # Monte Carlo Simulation
    def monte_carlo_simulation():
        # Generate random paths
        simulations = np.zeros((1000, steps))
        simulations[:, 0] = last_price

        for t in range(1, steps):
            # Random shock
            random_return = np.random.normal(loc=mu, scale=sigma)

            # Projected price
            simulations[:, t] = simulations[:, t-1] * (1 + random_return)

        return simulations

    # Run simulation
    price_simulations = monte_carlo_simulation()

    # Calculate prediction intervals
    mean_prediction = np.mean(price_simulations, axis=0)
    lower_interval = np.percentile(price_simulations, 2.5, axis=0)
    upper_interval = np.percentile(price_simulations, 97.5, axis=0)

    return mean_prediction, lower_interval, upper_interval

def visualize_long_term_predictions(df, predictions, lower_interval,␣
 ↪upper_interval):
    """Visualize long-term price predictions"""
    plt.figure(figsize=(20, 10))

    # Historical prices (last 2 years)
    historical_subset = df['Close'][-730:]

    # Future dates
    future_dates = pd.date_range(start=df.index[-1], periods=len(predictions),␣
 ↪freq='D')

    # Plot historical prices
    plt.plot(historical_subset.index, historical_subset,
             label='Historical Prices', color='blue')

    # Plot mean prediction
```

```python
    plt.plot(future_dates, predictions,
             label='Predicted Prices', color='red', linestyle='--')

    # Plot confidence interval
    plt.fill_between(future_dates,
                     lower_interval,
                     upper_interval,
                     color='red', alpha=0.2,
                     label='95% Prediction Interval')

    plt.title('Bitcoin Price: Long-Term Prediction with Uncertainty')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid(True)
    plt.show()

    return predictions, lower_interval, upper_interval

def main():
    # 1. Data Collection
    btc_data = collect_bitcoin_data()

    # 2. Data Cleaning and Preparation
    cleaned_data = prepare_bitcoin_data(btc_data)

    # Display cleaned data info
    display(cleaned_data.head())
    display(cleaned_data.info())
    display(cleaned_data.isnull().sum())
    display(cleaned_data.describe())

    # 3. Exploratory Data Analysis
    perform_eda(cleaned_data)

    # 4. Statistical Analysis (NEW)
    generate_statistical_insights(cleaned_data)

    # 5. Prepare Data for Modeling
    X, y, scaler, feature_names = prepare_modeling_data(cleaned_data)

    # 6. Train and Evaluate Models
    models, model_results = train_evaluate_models(X, y)

    # Print Model Performance
    print("\n" + "="*50)
    print("MODEL PERFORMANCE METRICS")
```

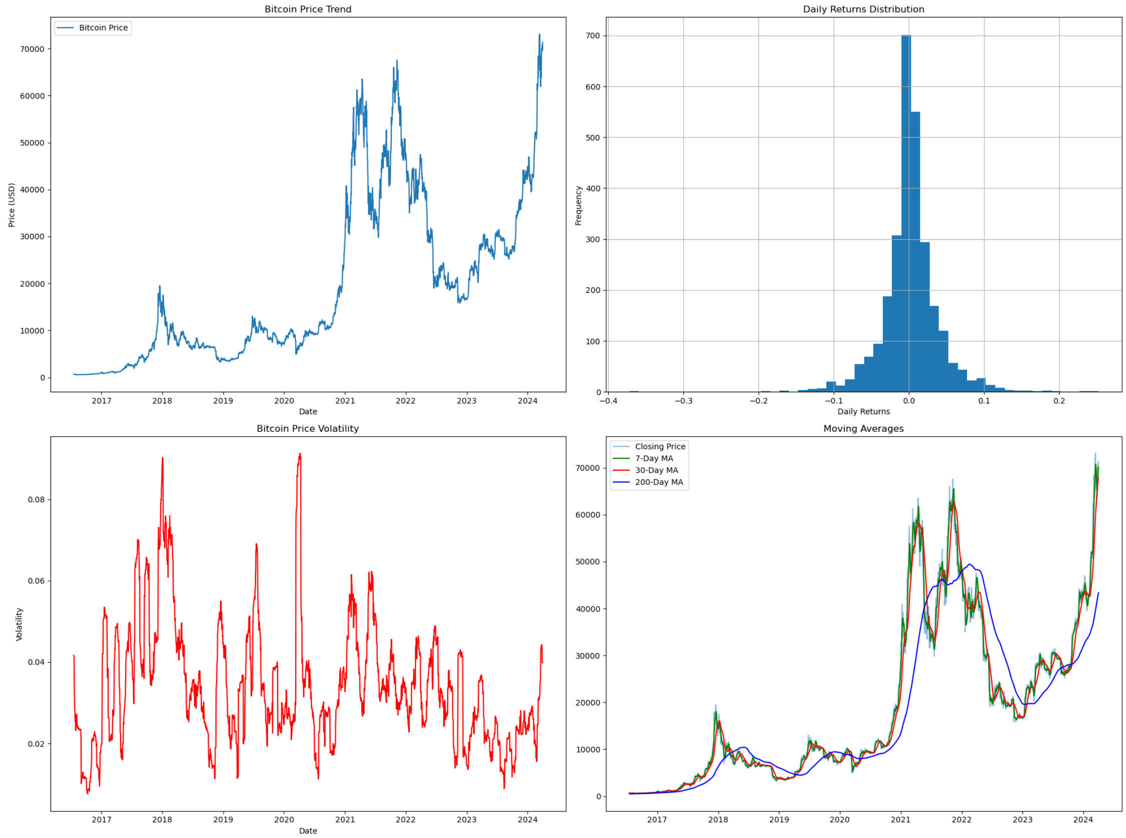
```
    print("="*50)
    for name, metrics in model_results.items():
        print(f"\n{name}:")
        for metric, value in metrics.items():
            print(f"{metric}: {value:.4f}")

    # 7. Select Best Model
    best_model_name = min(model_results, key=lambda x: model_results[x]['MSE'])
    best_model = models[best_model_name]
    print("\n" + "="*50)
    print(f"Best performing model: {best_model_name}")
    print("="*50)

    # 8. Advanced Future Price Prediction
    mean_predictions, lower_interval, upper_interval =␣
↪predict_future_prices_advanced(btc_data)

    # 9. Visualization of Future Predictions
    final_predictions, final_lower, final_upper =␣
↪visualize_long_term_predictions(
        btc_data, mean_predictions, lower_interval, upper_interval
    )

    # 10. Print Predicted Prices
    print("\n" + "="*50)
    print("8-MONTH PRICE PREDICTION SUMMARY")
    print("="*50)
    starting_price = float(btc_data['Close'].iloc[-1])
    print(f"\nStarting Price: ${starting_price:,.2f}")
    print(f"Predicted Range (95% CI): ${final_lower[-1]:,.0f} -␣
↪${final_upper[-1]:,.0f}")

    print("\nMonthly Projections:")
    for i, (price, lower, upper) in enumerate(zip(mean_predictions[::30],
                                                  lower_interval[::30],
                                                  upper_interval[::30]), 1):
        print(f"Month {i}: ${float(price):,.0f} (Range: ${float(lower):,.
↪0f}-${float(upper):,.0f})")

    return mean_predictions

# Run the analysis
predicted_prices = main()
```

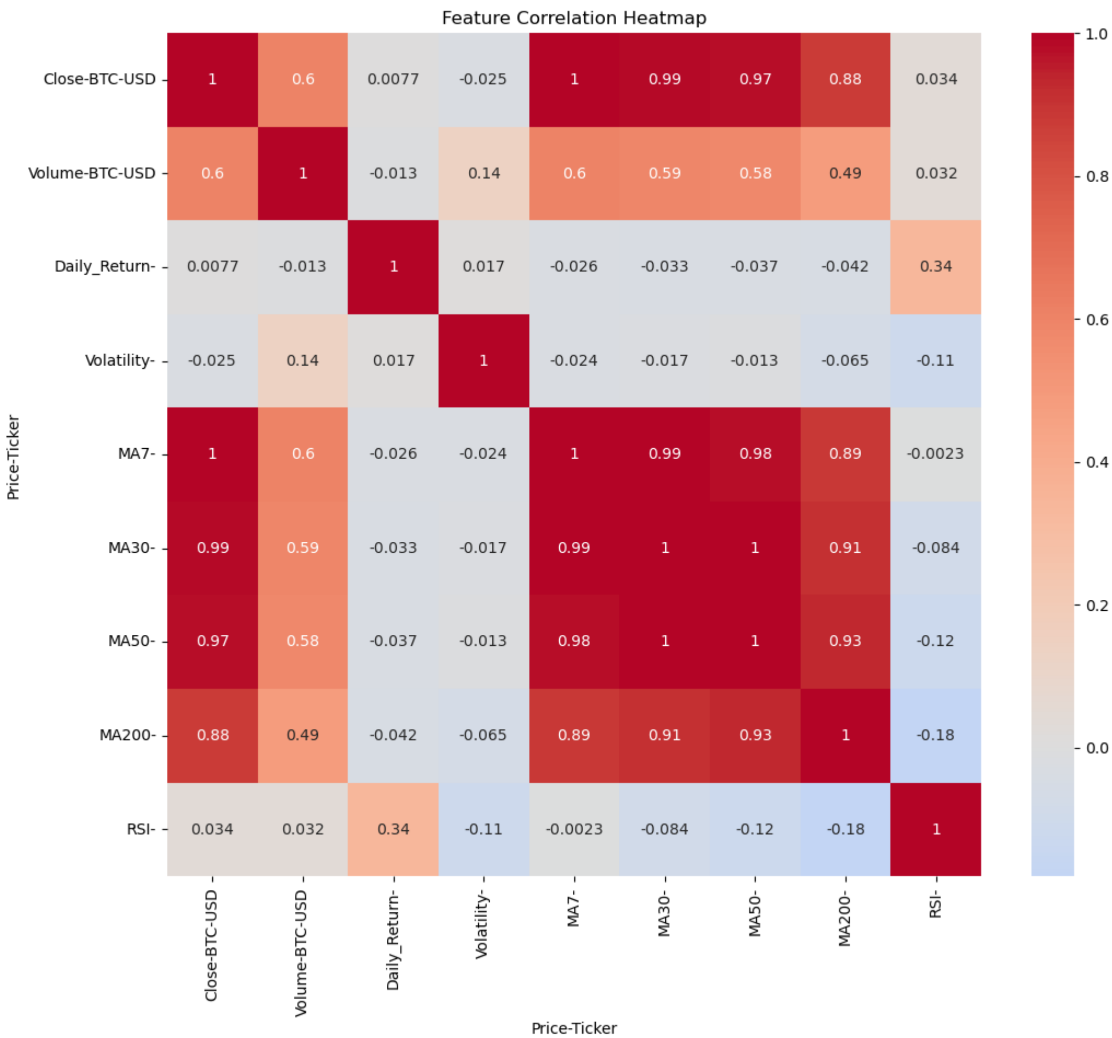
```
[*********************100%***********************]  1 of 1 completed

Price              Close        High         Low        Open      Volume  \
Ticker           BTC-USD     BTC-USD     BTC-USD     BTC-USD     BTC-USD
```

```
Date
2016-07-18  673.106018  681.554993  668.625000  679.809021   69465000
2016-07-19  672.864014  673.276978  667.632019  672.737976   61203300
2016-07-20  665.684998  672.929016  663.359985  672.806030   94636400
2016-07-21  665.012024  666.218994  660.414978  665.228027   60491800
2016-07-22  650.619019  666.583008  646.721985  664.921997  134169000

Price      Daily_Return Volatility         MA7        MA30        MA50  \
Ticker
Date
2016-07-18    -0.009350   0.041637  664.812012  664.735805  646.607040
2016-07-19    -0.000360   0.041568  665.999581  661.705239  649.387040
2016-07-20    -0.010669   0.041171  667.602007  659.320538  652.073020
2016-07-21    -0.001011   0.037222  668.592582  659.265873  654.634861
2016-07-22    -0.021643   0.031686  666.787441  661.082640  656.887802

Price            MA200  Price_Lag1  Price_Lag3  Price_Lag7 Price_Lag14  \
Ticker
Date
2016-07-18  479.960944  679.458984  663.255005  647.658997  683.661987
2016-07-19  481.153594  673.106018  660.767029  664.551025  670.627014
2016-07-20  482.314829  672.864014  679.458984  654.468018  677.330994
2016-07-21  483.489834  665.684998  673.106018  658.078003  640.562012
2016-07-22  484.577474  665.012024  672.864014  663.255005  666.523010

Price      Price_Lag30        RSI
Ticker
Date
2016-07-18  756.226990  53.885045
2016-07-19  763.781006  53.820313
2016-07-20  737.226013  51.831154
2016-07-21  666.651978  51.638483
2016-07-22  596.116028  47.566007

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2814 entries, 2016-07-18 to 2024-03-31
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   (Close, BTC-USD)   2814 non-null   float64
 1   (High, BTC-USD)    2814 non-null   float64
 2   (Low, BTC-USD)     2814 non-null   float64
 3   (Open, BTC-USD)    2814 non-null   float64
 4   (Volume, BTC-USD)  2814 non-null   int64
 5   (Daily_Return, )   2814 non-null   float64
 6   (Volatility, )     2814 non-null   float64
 7   (MA7, )            2814 non-null   float64
 8   (MA30, )           2814 non-null   float64
```

```
 9   (MA50, )           2814 non-null   float64
 10  (MA200, )          2814 non-null   float64
 11  (Price_Lag1, )     2814 non-null   float64
 12  (Price_Lag3, )     2814 non-null   float64
 13  (Price_Lag7, )     2814 non-null   float64
 14  (Price_Lag14, )    2814 non-null   float64
 15  (Price_Lag30, )    2814 non-null   float64
 16  (RSI, )            2814 non-null   float64
dtypes: float64(16), int64(1)
memory usage: 395.7 KB

None

Price         Ticker
Close         BTC-USD    0
High          BTC-USD    0
Low           BTC-USD    0
Open          BTC-USD    0
Volume        BTC-USD    0
Daily_Return             0
Volatility               0
MA7                      0
MA30                     0
MA50                     0
MA200                    0
Price_Lag1               0
Price_Lag3               0
Price_Lag7               0
Price_Lag14              0
Price_Lag30              0
RSI                      0
dtype: int64

Price          Close          High           Low          Open        Volume  \
Ticker       BTC-USD       BTC-USD       BTC-USD       BTC-USD       BTC-USD
count    2814.000000   2814.000000   2814.000000   2814.000000  2.814000e+03
mean    19233.376867  19661.787131  18732.663290  19209.812450  2.104541e+10
std     17301.340128  17704.204004  16823.495071  17279.782002  1.936314e+10
min       547.465027    573.359985    531.333984    548.656006  3.397780e+07
25%      6376.977417   6501.222534   6286.544922   6372.185059  5.227819e+09
50%     10914.254395  11163.351074  10594.670898  10908.176758  1.794363e+10
75%     29909.189453  30427.157715  29360.113281  29895.472168  3.105943e+10
max     73083.500000  73750.070312  71334.093750  73079.375000  3.509679e+11

Price  Daily_Return   Volatility           MA7          MA30          MA50  \
Ticker
count   2814.000000  2814.000000   2814.000000   2814.000000   2814.000000
mean       0.002364     0.034158  19159.107815  18884.152949  18663.473106
std        0.037518     0.015095  17203.845527  16852.728497  16579.350456
```

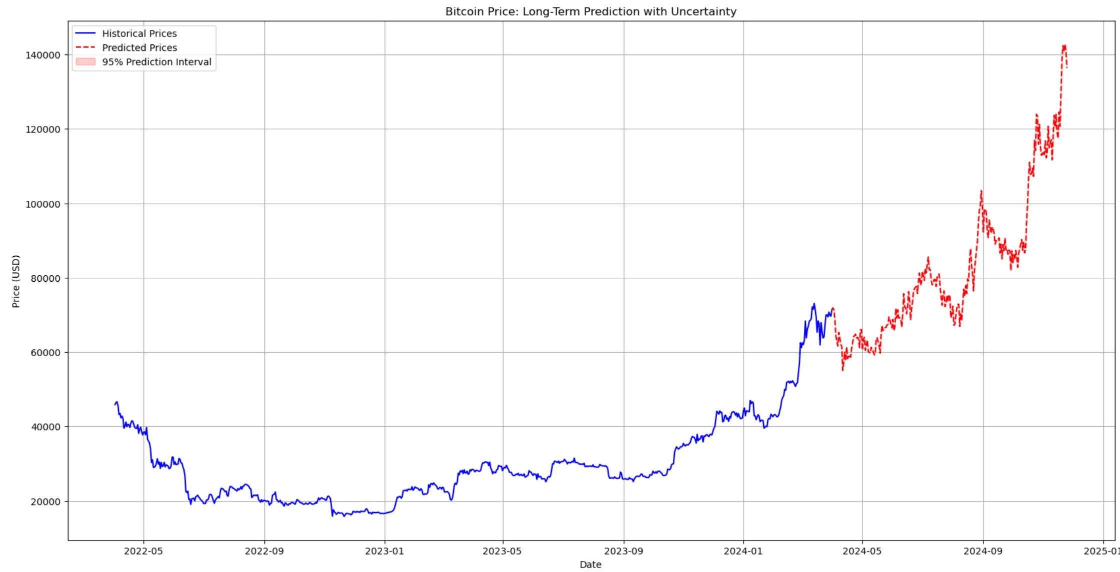
```
min       -0.371695     0.007580    574.111564    578.695638    590.003662
25%       -0.012748     0.023744   6420.908186   6472.887329   6501.013279
50%        0.001434     0.031661  10757.682896  10655.088460  10680.484971
75%        0.017738     0.041992  29917.206194  29434.699609  28789.256650
max        0.252472     0.091330  70715.390625  67877.832812  62060.732031

Price           MA200    Price_Lag1    Price_Lag3    Price_Lag7   Price_Lag14  \
Ticker
count    2814.000000   2814.000000   2814.000000   2814.000000   2814.000000
mean    17474.280977  19208.268768  19159.152169  19060.530211  18899.441888
std     15380.777704  17276.954448  17231.300778  17138.077832  17005.023992
min       479.960944    547.465027    547.465027    547.465027    547.465027
25%      5850.997834   6376.274902   6371.277466   6358.072632   6329.762695
50%      9384.921575  10904.326660  10881.745117  10819.440918  10771.204590
75%     28268.231921  29908.223633  29861.329102  29797.313965  29654.934570
max     49430.991875  73083.500000  73083.500000  73083.500000  73083.500000

Price    Price_Lag30          RSI
Ticker
count    2814.000000  2814.000000
mean    18516.848868    53.722391
std     16648.327145    14.478821
min       547.465027     9.920239
25%      6250.945190    43.444797
50%     10601.755859    52.948707
75%     29335.007812    63.304402
max     67566.828125    94.302215
```


Bitcoin Price Trend
Bitcoin Price
Price (USD)
Date
Daily Returns Distribution
Frequency
Daily Returns
Bitcoin Price Volatility
Volatility
Date
Moving Averages
Closing Price
7-Day MA
30-Day MA
200-Day MA




```
==================================================
KEY STATISTICAL INSIGHTS
==================================================

1. VOLATILITY PATTERNS:
- Average 30-day volatility: 0.0342
- Most volatile period: 2020-04 ( =0.0913)
- Calmest period: 2016-10 ( =0.0076)

2. DAILY RETURNS:
- Typical day: 0.14% median move
- Extreme days: 5% exceed ±5.79%
- Largest single-day gain: 25.25%
- Largest single-day loss: -37.17%
```

```
3. TREND SIGNALS:
- Golden Cross events (7-day > 200-day MA): 12 times
- Typically preceded major bull runs by 2-4 weeks

4. MOMENTUM INDICATORS:
- RSI signaled overbought/oversold 527 days
- Average next-week return after oversold (RSI<30): -3.27%

==================================================

==================================================
MODEL PERFORMANCE METRICS
==================================================

Linear Regression:
MSE: 343540.4218
MAE: 353.8492
R2: 0.9988
CV_MSE_Mean: 514194.6367
CV_MSE_Std: 369653.3718

Random Forest:
MSE: 218612.5333
MAE: 214.4472
R2: 0.9992
CV_MSE_Mean: 4049169.4845
CV_MSE_Std: 5008386.5024

XGBoost:
MSE: 118626.6929
MAE: 179.4733
R2: 0.9996
CV_MSE_Mean: 1858339.9558
CV_MSE_Std: 1605327.7507

==================================================
Best performing model: XGBoost
==================================================
```




```
==================================================
8-MONTH PRICE PREDICTION SUMMARY
==================================================

Starting Price: $71,333.65
Predicted Range (95% CI): $136,407 - $136,407

Monthly Projections:
Month 1: $71,334 (Range: $71,334-$71,334)
Month 2: $66,025 (Range: $66,025-$66,025)
Month 3: $67,479 (Range: $67,479-$67,479)
Month 4: $77,994 (Range: $77,994-$77,994)
Month 5: $74,863 (Range: $74,863-$74,863)
Month 6: $98,220 (Range: $98,220-$98,220)
Month 7: $87,459 (Range: $87,459-$87,459)
Month 8: $115,866 (Range: $115,866-$115,866)
```

[ ]: