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: GUDIMETLLA BHARAT NAGENDRA REDDY (Q1076626)

Year: 2024-2025

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 texts I clearly declare it through the good use of references following academic ethics.

(In the case that 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):

GUDIMETLLA BHARAT NAGENDRA REDDY

Date: 2025/02/02

INTRODUCTION

The advent of the concept of Cryptocurrency in the world of Finance was pioneered with the White Paper titled “Bitcoin: A Peer-to-Peer Electronic Cash System” published in October 2008 by Satoshi Nakamoto. Bitcoin transactions (as they were known) were secured through Cryptography, validated, and stored in a decentralized network known as the 'blockchain'. The core idea was that Bitcoin enabled direct online payment transaction between two parties without relying on a trusted third party like say a financial institution.

By March 2017, Bitcoin had become the largest, most popular Cryptocurrency by market capitalization, capturing 72 percent of the market-share of Cryptocurrency. During January-February 2017, Bitcoin transactions numbered 286,419 surpassing all other Cryptocurrencies.

The price of Bitcoin increases the same time significantly from USD 1000 in 2007 to USD 16000 in December 2017. This made Bitcoin's price highly volatile and difficult to predict (phaladisailoed & Numnonda 2018). Cryptocurrencies, especially Bitcoin, garnering wider attention due to their volatile nature and potential for higher returns in recent years.

Bitcoin is now used worldwide for investments and making payments. Transactions made by Bitcoin are easy and are not overseen or monitored by Governments. Predicting prices is a challenging task due to the markets' inherent volatility and the influence of numerous external factors, (Velankar, Valecha & Maji, 2008).

This assessment report follows a structured approach, including data exploration, pre-processing, feature selection, model training and evaluation. The Dataset used for this analysis has been sourced from the Cryptocurrency Dataset.

The goal is to predict the future price of Bitcoin, using historical data and evaluate the performance of three Machine Learning models, namely, Random Forest, K-Nearest Neighbors (KNN) and Linear Regression.

CHAPTER ONE: Data Exploration and Preparation

1.1 Data Exploration

The Dataset has been sourced from the Cryptocurrency-Dataset repository downloaded and loaded into Jupyter notebook. Pandas was used to extract Bitcoin.csv file that was used in this assignment.

The Dataset contains Bitcoin price information from 2019-2022, comprising 1,151 observations across 7 features: Date, Open, High, Low, Close, Volume and Currency. The Time Series data reveals significant price volatility and several notable trends.

The closing price of Bitcoin exhibits dramatic fluctuations with values ranging from approximately USD 4,936 to 67,502. The mean closing price is USD 26,496 with a substantial standard deviation of USD 17,952 indicative of high price volatility. The price data shows a clear upward trend starting in late 2020 and reaching 2 major peaks: one in early 2021 around USD 60,000, and a second in late 2021 nearing USD 70,000. This is followed by a significant decline through 2022. The fluctuation cab attributed to Covid-19 Pandemic which is an external factor that can exert significant influence on the finance sector.

The Dataset is notably clean with no missing values across any features, eliminating the need for imputation techniques. It replaces the missing values with the most frequently occurring values or mean. Another Methodist is dropping the values. The daily trading volume varies significantly, with a mean of 28740512 and values ranging from 0 to 579170589, suggesting periods of both high and low market activity.

Key features identified for price prediction include:

- Historical price data (Open, High, Low, Close)
- Trading volume
- Price volatility metrics
- Day-to-day price movements

The data's completeness and comprehensive price information provide a solid foundation for predictive modeling.

In [2]:
```python
# Loading all the necessary libraries

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

from sklearn.preprocessing import StandardScaler

from sklearn.feature_selection import SelectKBest, f_regression, RFE

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, Lasso

from sklearn.metrics import mean_squared_error

import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```python
# Load data from CSV
df = pd.read_csv("Bitcoin.csv")
df.head()
```

Out[6]:

	Date	Open	High	Low	Close	Volume	Currency
0	2019-06-18	9128.269531	9149.763672	8988.606445	9062.045898	952850.0	USD
1	2019-06-19	9068.174805	9277.677734	9051.094727	9271.459961	131077.0	USD
2	2019-06-20	9271.567383	9573.689453	9209.416992	9519.200195	83052.0	USD
3	2019-06-21	9526.833984	10130.935547	9526.833984	10127.998047	76227.0	USD
4	2019-06-22	10151.890625	11171.013672	10083.189453	10719.981445	84485.0	USD

In [8]:
```python
df.describe()
```

Out[8]:

	Open	High	Low	Close	Volume
count	1151.000000	1151.000000	1151.000000	1151.000000	1.151000e+03
mean	26488.652992	27528.416710	25416.606967	26496.733082	2.874051e+07
std	17963.101635	18432.925246	17484.604545	17952.113609	5.202999e+07
min	4943.832520	5338.512695	0.076853	4936.755371	0.000000e+00
25%	9706.758301	10090.012695	9360.636230	9712.636719	7.495500e+03
50%	20873.337891	21867.822266	20245.201172	20902.404297	1.864334e+06
75%	41782.333984	42749.439453	40890.394531	41782.333984	4.076471e+07
max	67470.437500	85563.984375	66072.343750	67502.421875	5.791706e+08

In [10]:
```
# Converting Date column data-type to datetime

df['Date'] = pd.to_datetime(df['Date'])
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1151 entries, 0 to 1150
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      1151 non-null   datetime64[ns]
 1   Open      1151 non-null   float64
 2   High      1151 non-null   float64
 3   Low       1151 non-null   float64
 4   Close     1151 non-null   float64
 5   Volume    1151 non-null   float64
 6   Currency  1151 non-null   object
dtypes: datetime64[ns](1), float64(5), object(1)
memory usage: 63.1+ KB
```

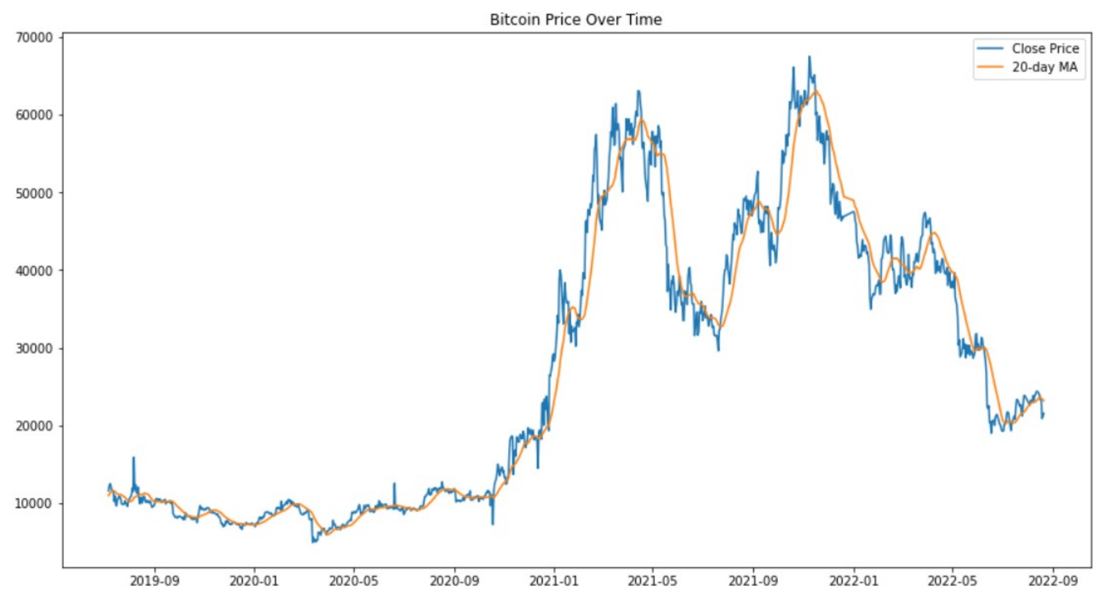
In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Date        0
Open        0
High        0
Low         0
Close       0
Volume      0
Currency    0
dtype: int64
```

In [47]:
```
# Descriptive statistics
print("\nDescriptive Statistics:")
print(df.describe())
```

```
Descriptive Statistics:
                                Date          Open          High  \
count                           1131   1131.000000   1131.000000
mean   2021-01-24 06:23:14.164456192  26755.760753  27804.994442
min              2019-07-07 00:00:00   4943.832520   5338.512695
25%              2020-04-14 12:00:00   9699.631348  10052.406250
50%              2021-01-22 00:00:00  21429.888672  22952.214844
75%              2021-10-31 12:00:00  42151.925781  42878.111328
max              2022-08-21 00:00:00  67470.437500  85563.984375
std                              NaN  18004.549553  18473.557468
```

```
             Low         Close        Volume   Price_Range  Daily_Return  \
count   1131.000000   1131.000000  1.131000e+03   1131.000000   1131.000000
mean   25672.291067  26762.192523  2.924074e+07   2132.703375      0.002037
min        0.076853   4936.755371  0.000000e+00     69.416992     -0.383808
25%     9345.433594   9692.692383  7.255500e+03    554.634277     -0.019873
50%    20792.781250  21561.177734  2.220637e+06   1218.628906      0.000923
75%    41003.773438  42151.925781  4.136760e+07   2542.314453      0.021910
max    66072.343750  67502.421875  5.791706e+08  83961.292725      0.572960
std    17528.461119  17994.940832  5.235054e+07   3765.238710      0.054685

       Volume_Price_Ratio          MA_5         MA_20   Volatility  \
count         1131.000000   1131.000000   1131.000000  1131.000000
mean           687.397866  26744.064978  26663.149541     0.042855
min              0.000000   5208.214941   5962.642139     0.003579
25%              0.741517   9743.102051   9653.592212     0.023367
50%             64.903993  21671.387891  21482.742871     0.033315
75%            994.465729  42100.064453  41556.704395     0.050306
max          11665.062805  65624.415625  63098.463281     0.344515
std           1181.711468  17968.506174  17896.424754     0.037301

             Target
count   1131.000000
mean   26770.778978
min     4936.755371
25%     9692.692383
50%    21561.177734
75%    42151.925781
max    67502.421875
std    17990.012403
```



1.2 Data Pre-Processing

In order to Pre-Process Data, I applied certain tested methods. This required the implementation of feature engineering techniques specifically used in data pre-processing, to enhance the predictive capability and accuracy of the model for Bitcoin price forecasting.

The implementation focused on creating meaningful technical indicators, that would help in ensuring the proper scaling of data.

These included the following:

1. Price Range: Calculated as the difference between High and Low prices, capturing their daily price volatility.
2. Daily Return: This was Implemented using pct_change() on Close prices to measure and ascertain the daily price fluctuations.
3. Volume Price Ratio: This was generated by dividing Volume by the Closing price to capture daily trading intensity
4. Moving Averages: Two moving averages are calculated:
   - MA_5: 5-day moving average for short-term price trends.
   - MA_20: 20-day moving average for medium-term price trends
5. Volatility: This is computed as the rolling standard deviation of Daily Returns over a 5-day window.
6. Target Variable Definition: The TVD (target variable definition) is used to ascertain and forecast the next day's Closing price. It employs a shift(-1) operation on the Closing Price column. This framework creates the 1-day ahead prediction.

Feature Scaling: Standardization (StandardScaler) is applied to normalize all features to a common scale, where, Mean is Zero (0) and Standard Deviation is One (1). Scaling features is used to avoid giving preference or dominance of one feature over the other during model training and testing. This, in return, helps to improve the interpretability, robustness, and performance of Machine Learning algorithms (Ozsahin et al., 2022). The scaled features include:

- Price – Open, High, Low, Close
- Volume
- Price Range
- Daily Return
- Volume Price Ratio

- Both moving averages (MA_5, MA_20)
- Volatility

The preprocessing pipeline handles missing values through a dropna() method. Thus, removing any rows with NaN values that may have been created, during the calculation of technical indicators, especially at the beginning of the rolling windows. Moving averages are stock indicators common in technical analysis. They reduce fluctuations in price plus reduce noise and give a better idea on market trends (Arumugam and Uma). In this case, both the 5 day and 20 day moving average are used. This ensures a clean, scaled data output that is ready for model training.

In [14]:
```
# Feature Engineering

df['Price_Range'] = df['High'] - df['Low']

df['Daily_Return'] = df['Close'].pct_change()
# The pct_change() function in pandas is used to calculate the percentage change between the current and a prior element.

df['Volume_Price_Ratio'] = df['Volume'] / df['Close']

df['MA_5'] = df['Close'].rolling(window=5).mean()

df['MA_20'] = df['Close'].rolling(window=20).mean()

df['Volatility'] = df['Daily_Return'].rolling(window=5).std()
```

In [15]:
```
# Create target variable (next day's closing price)
df['Target'] = df['Close'].shift(-1)
```

In [17]:
```
# df.isnull().sum()

df.isnull().sum()/len(df)*100

# checking the null values in percentage %
```

Out[17]:
```
Date                  0.000000
Open                  0.000000
High                  0.000000
Low                   0.000000
Close                 0.000000
Volume                0.000000
Currency              0.000000
Price_Range           0.000000
Daily_Return          0.086881
Volume_Price_Ratio    0.000000
MA_5                  0.347524
MA_20                 1.650738
Volatility            0.434405
Target                0.086881
dtype: float64
```

In [18]:
```
# Drop rows with NaN values

df = df.dropna()
```

In [20]:
```
df.isnull().sum()/len(df)*100
```

Out[20]:
```
Date                  0.0
Open                  0.0
High                  0.0
Low                   0.0
Close                 0.0
Volume                0.0
Currency              0.0
Price_Range           0.0
Daily_Return          0.0
Volume_Price_Ratio    0.0
MA_5                  0.0
MA_20                 0.0
Volatility            0.0
Target                0.0
dtype: float64
```

In [18]:
```
# Prepare features and target
features = ['Open', 'High', 'Low', 'Close', 'Volume', 'Price_Range',
            'Daily_Return', 'Volume_Price_Ratio', 'MA_5', 'MA_20', 'Volatility']
X = df[features]
y = df['Target']
```

In [19]:
```
# Standardize features

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

CHAPTER TWO: Feature Selection

2.1 Feature Selection Techniques

Feature Selection is a process that filters out features that are not really relevant to a Dataset. This allows the Models to produce better results when one is making predictions on target variables. (Chen et al., 2020). Any limitations, of a single feature selection are sifted through the use of ensemble methods, which combine different features that yield better selection results. The three methods are Filter, Wrapper, and Embedded. For this assignment, we have applied only Filter and Wrapper methods to help achieve high efficiency and enhanced classification (Stańczyk, 2015). The feature selection process employs multiple complementary methods, so it can identify the most significant predictors for Bitcoin price movements. The analysis utilized four distinct approaches: Correlation, Analysis, F-regression, and Recursive Feature Elimination (RFE).

Correlation Analysis Results

The correlation heatmap depicts strong relationships between several features and the target variables. The selected features, through correlation analysis, were 'Close', 'Low', 'MA_20', 'MA_5' and 'Open'. Notably, price-related features (Open, High, Low, Close) show very high correlations with each other.

Comparison of Feature Selection Methods:

1. Filter Methods:

- Correlation-based selection identified five key features, focusing on price trends and volume indicators.
- F-regression produced identical results to the correlation method, confirming the importance of these features with the replacement of ‘MA_20’ with ‘High’.
- Both methods emphasized the significance of moving averages (MA_5, MA_20) and volume-related metrics.

2. Wrapper Method (RFE):

- Recursive Feature Elimination (RFE), with Random Forest as the base estimator, also selected a similar feature-set like Correlation and F-regression. It excluded the remaining features.
- It maintained the importance of both Moving Averages and Core Price/Volume Metrics.

3. Final Feature Selection:

   Based on the consensus across methods, six features were ultimately selected:

- Open price
- Close price
- High
- Low
- MA_5 (5-day moving average)
- MA_20 (20-day moving average)

The Feature Selection frequency demonstrates that Open, Close, and MA_5 were consistently chosen across all methods. This affirms their strong predictive power. On the other hand, High, Low and MA_20 showed lower selection frequency, suggesting they may be less crucial for prediction.

4. The final selection balances multiple factors:

- Technical indicators (moving averages)
- Market activity (volume and volume-price ratio)
- Price dynamics (volatility)
- Historical price information (close price)

This comprehensive Feature-set captures both Trend and Momentum aspects of Bitcoin price movements. It thus provides a firm and robust foundation for predictive modeling. Thereby, the selection process, efficiently and effectively reduced dimensionality, while retaining the most informative features for price prediction.

Prepare features and target

In [18]:
```
# Prepare features and target
features = ['Open', 'High', 'Low', 'Close', 'Volume', 'Price_Range',
            'Daily_Return', 'Volume_Price_Ratio', 'MA_5', 'MA_20', 'Volatility']
X = df[features]
y = df['Target']
```

In [19]:
```
# Standardize features

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Feature selection using Filter Methods and RFE

In [21]:
```
correlations = abs(pd.DataFrame(X_scaled, columns=features).corrwith(y))
correlation_selected = set(correlations.nlargest(5).index)
```

In [22]:
```
correlation_selected
```

Out[22]:
```
{'Close', 'Low', 'MA_20', 'MA_5', 'Open'}
```

In [ ]:

In [23]:
```
f_selector = SelectKBest(score_func=f_regression, k=5)
f_selector.fit(X_scaled, y)
f_selected = set(pd.Series(f_selector.scores_, index=features).nlargest(5).index)
```

In [24]:
```
f_selected
```

Out[24]:
```
{'Close', 'High', 'Low', 'MA_5', 'Open'}
```

In [ ]:

In [25]:
```
rfe = RFE(RandomForestRegressor(n_estimators=100, random_state=42), n_features_to_select=5)
rfe.fit(X_scaled, y)
rfe_selected = set(pd.Series(features)[rfe.support_])
```

In [26]:
```
rfe_selected
```

Out[26]:
```
{'Close', 'High', 'MA_20', 'MA_5', 'Open'}
```

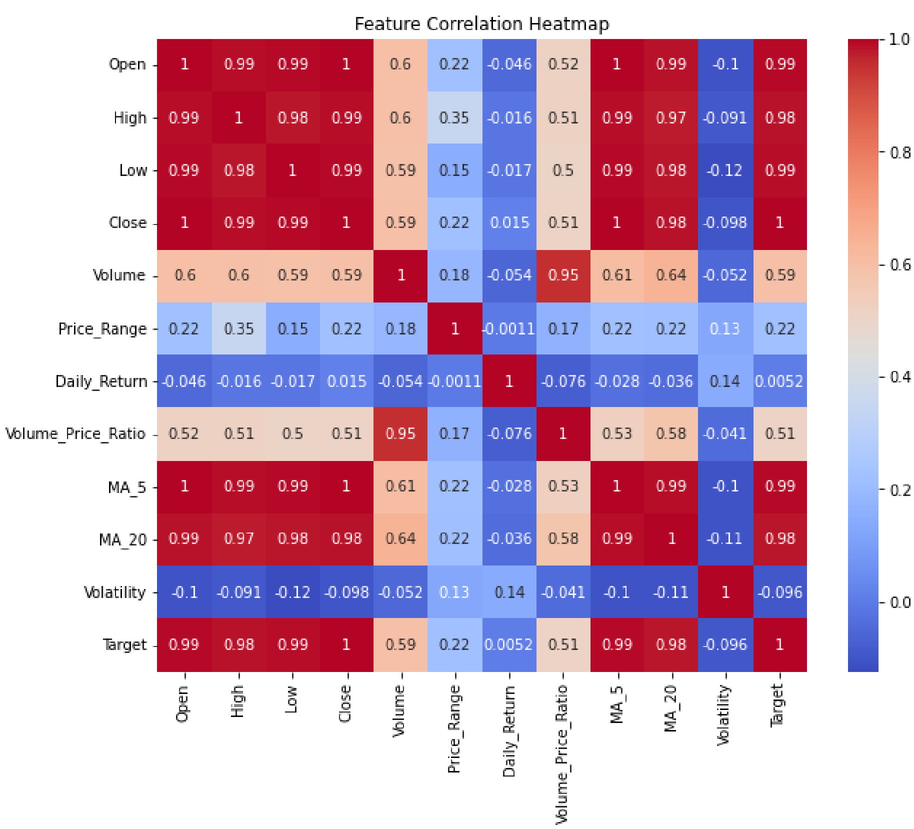
```
In [27]: # Combine selected features
         final_features = list(correlation_selected | f_selected | rfe_selected)
         X_final = pd.DataFrame(X_scaled, columns=features)[final_features]
```

```
In [28]: X_final
```

Out[28]:

	Open	High	Low	MA_5	Close	MA_20
0	-0.862685	-0.852771	-0.828158	-0.854887	-0.843359	-0.875630
1	-0.842507	-0.839179	-0.816414	-0.851375	-0.806476	-0.866702
2	-0.805514	-0.815599	-0.766918	-0.836647	-0.791715	-0.857617
3	-0.791065	-0.795740	-0.796664	-0.823866	-0.813297	-0.850310
4	-0.812451	-0.848646	-0.826327	-0.819085	-0.839602	-0.846028
...	...	...	...	...	...	...
1126	-0.159675	-0.184915	-0.137232	-0.152194	-0.190990	-0.172913
1127	-0.190270	-0.229495	-0.143820	-0.165431	-0.196806	-0.174627
1128	-0.196519	-0.248351	-0.272477	-0.203318	-0.325779	-0.182329
1129	-0.325386	-0.349852	-0.274410	-0.236440	-0.311846	-0.188608
1130	-0.311301	-0.330846	-0.259520	-0.262293	-0.289154	-0.193462

1131 rows × 6 columns



CHAPTER THREE: Model Training and Evaluation

3.1 Model Selection and Training

I have chosen Three Machine Learning Models and implemented for this assignment:

1. **Random Forest Regressor**: This is an ensemble technique which combines multiple decision tree.
2. **K-Nearest Neighbors (KNN) Regressor**: This is a non-parametric method that predicts based on the average of the nearest neighbors.
3. **Linear Regression**: This is a baseline model that assumes a linear relationship between the independent features and the target variable (Itoo, Meenakshi and Singh, 2021).

These Models were used to create a Validation Dataset. A further splitting of this Dataset was done to achieve an 80:20 split of which 20% was used for evaluation of the model. Each of them was trained on standardized features: Close Price, Volume, Volume-Price Ratio, Moving Averages (MA_5, MA_20), and Volatility.

While configuring the Random Forest model, two parameters were tuned: the number of estimators = 100 and the maximum depth = 10. For KNN model, the number of neighbors was 5 by applying the distance-weighted predictions. On other hand, the Linear Regression model was applied to establish a starting point to determine the default parameters.

3.2 Model Evaluation

The purpose of Model Evaluation is to analyze and compare the performance of three different Machine Learning models, namely, Linear Regression, Random Forest, and K-Nearest Neighbors (KNN). Their individual and combined performance helps in predicting Bitcoin prices using their selected features.

Comparison of Model Performance

1. Linear Regression: RMSE = 1407.10
2. Random Forest: RMSE = 1516.05
3. K-Nearest Neighbors: RMSE = 1602.23

The Linear Regression model registering the lowest RMSE of 1407.10 suggests that it correctly unravels price trends of the shares better than the other models. The model which was nearest to, the Random Forest model was the one with an RMSE of 1516.05. The highest error was displayed by the KNN model with a RMSE of 1602.23.

Accuracy Analysis of Predictions

In this study, it was found that all the three models have high linear correlation coefficients, between the actual and predicted values, as can be seen from the scatter plots.

Key observations:

1. Linear Regression:

- It shows consistent performance across the price range
- It demonstrates the tightest clustering around the ideal prediction line
- It performs particularly well in the mid-price range (30,000-50,000)

2. Random Forest:

- It exhibits strong feature importance hierarchy with Close Price dominating (72%),
- Thereby delivering good prediction accuracy with slight scatter at higher prices
- Both MA_5 (6%) and MA_20 (3.5%) contribute significantly to predictions

3. KNN:

- It shows wider scatter, particularly at higher price points
- It demonstrates increased variance in predictions
- Its performance degrades slightly at extreme price values

Model Selection Implications

The Linear Regression model's superior performance suggests that Bitcoin price movements, despite their complexity, maintain relatively linear relationships with the selected features. The minimal difference between Linear Regression and Random Forest RMSE values indicates that the added complexity of the Random Forest model may not justify its use over the simpler linear model.

Recommendation

Based on the evaluation metrics and visualization analysis, the Linear Regression model is recommended as the primary prediction model in light of its following benefits:

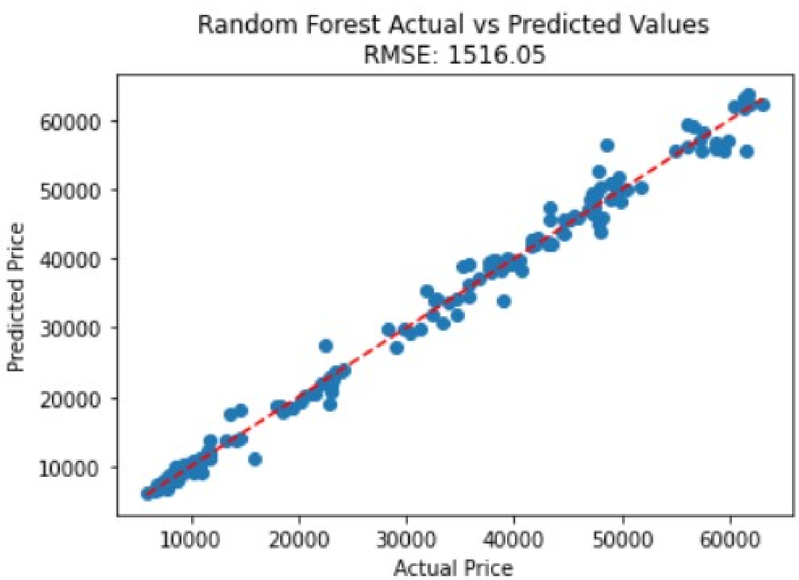
- Lower RMSE
- Computational efficiency
- Consistent performance across price ranges
- Simpler interpretation and implementation

Note, however, for production deployment, given their similar performance parameters, ensemble methods combining Linear Regression and Random Forest predictions could potentially provide more robust results.

```
In [30]: # Spliting the data into train set and training set

         X_train, X_test, y_train, y_test = train_test_split(X_final, y, test_size=0.2, random_state=42)

In [ ]:

In [31]: # Initializing models with optimised parameters

         models = {
             'Random Forest': RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42),
             'KNN': KNeighborsRegressor(n_neighbors=5, weights='distance'),
             'Linear Regression': LinearRegression()
         }
```

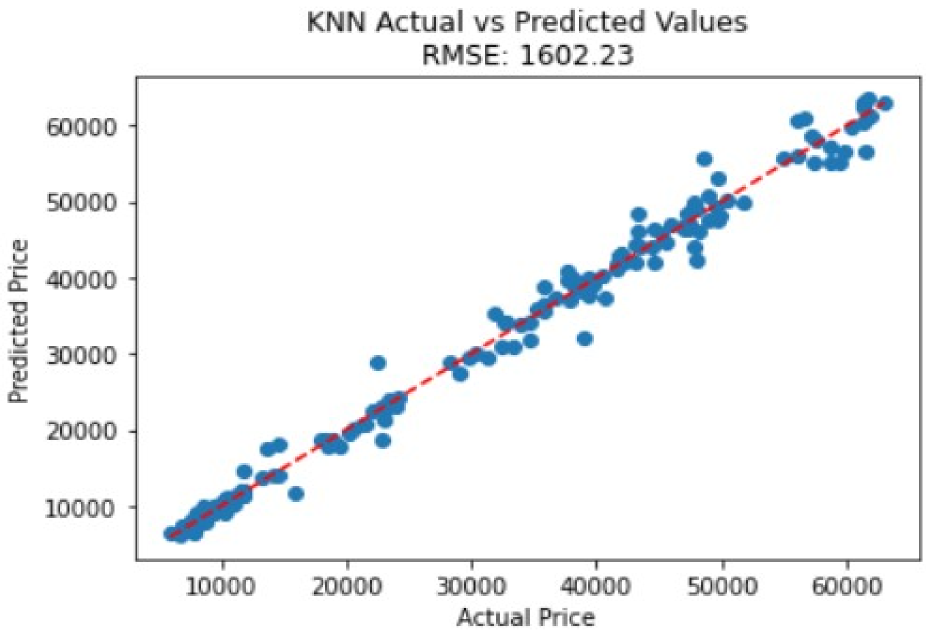
In [35]:
```python
# Training and evaluating each model

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print(f"{name} RMSE: {rmse:.2f}")


    plt.scatter(y_test, y_pred)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
    plt.xlabel('Actual Price')
    plt.ylabel('Predicted Price')
    plt.title(f'{name} Actual vs Predicted Values\nRMSE: {rmse:.2f}')
    plt.show()

    # Check for feature importances in the model and print them
    if hasattr(model, 'feature_importances_'):
        feature_imp = pd.Series(model.feature_importances_, index=final_features)
        print("\nFeature Importances:")
        print(feature_imp.sort_values(ascending=False))
        print('\n')
```

```
Random Forest RMSE: 1516.05
```

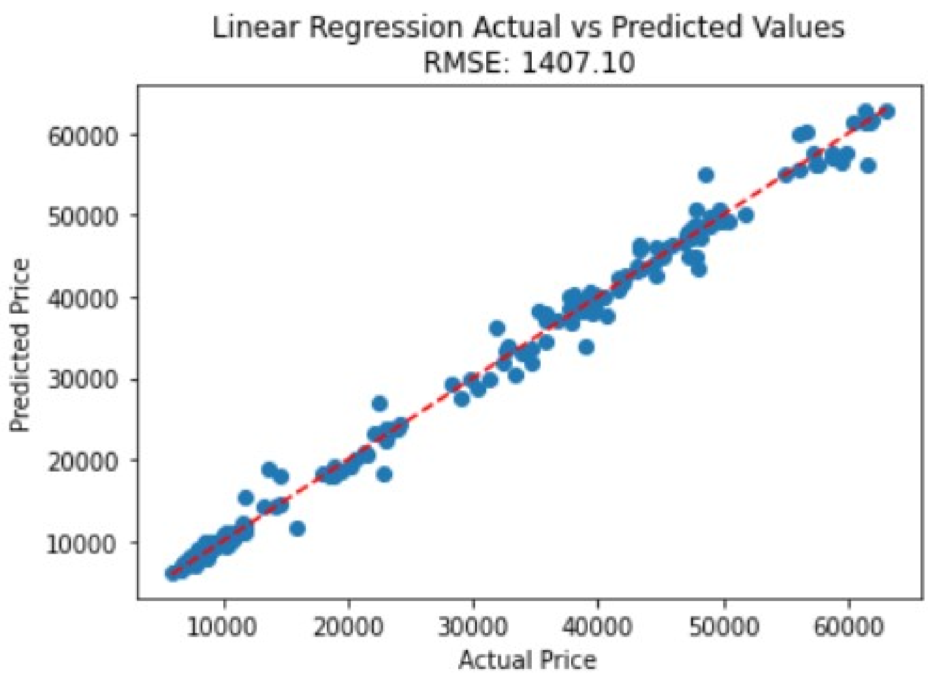



```
Feature Importances:
Close     0.724805
Open      0.168364
MA_5      0.060471
MA_20     0.035104
High      0.007519
Low       0.003735
dtype: float64
```

KNN RMSE: 1602.23



Linear Regression RMSE: 1407.10



CONCLUDING and Future Work

Summary of Findings

Encapsulated below are the main findings of the Study on Cryptocurrency:

1. Cryptocurrency is the hot topic in financial and digital trading circuits. The main reason being that it enables transactions that do not require intermediary agents or agencies.

2. The process in itself is completely transparent offered by Blockchain technology.

3. However, it is highly volatile, therefore necessitating accurate predictions that are used to support investment decisions.

4. Research points out that advanced models like LSTM are better at predicting Bitcoin prices. (Istaltofa, Sarwido and Sucipto, 2024).

5. The analysis of three Machine Learning Models for Bitcoin price predictions revealed interesting patterns in predictive accuracy.

6. The Linear Regression model ranked as the top performer with an RMSE of 1407.10. Close on its heels was Random Forest (RMSE: 1516.05). The KNN model displayed relatively lower accuracy (RMSE: 1602.23).

7. The Features selection process significantly impacted model performance with the Close Price. Yet, Moving Averages (MA_5 and MA_20) emerged as the most influential Predictors.

8. The Random Forest model featured an important analysis that confirmed this, showing Close Price contributing 72% to predictions, followed by Open and MA_5 at 16% and 6% respectively.

9. The Linear Regression model's superior performance suggests that Bitcoin price movements, despite their complexity, predominantly maintain linear relationships with the selected technical indicators.

Limitations and Challenges

1. Several limitations must need to be acknowledged in this Study. The inherent volatility of Cryptocurrency markets poses a significant challenge to predictive modeling, as historical patterns may not reliably indicate future price movements.

2. The Models' reliance on technical indicators alone could miss important external factors that influence Bitcoin prices. Additionally, the Training Data's time-frame (2019-2022) includes several unusual market events, including the COVID-19 pandemic, besides major market corrections, which may affect the Models' generalization pertaining to varying market conditions.

3. The linear nature of our best-performing Model might also limit its ability to capture complex, non-linear market dynamics that often characterize Cryptocurrency markets.

Future Research Directions

1. To address these limitations and enhance prediction accuracy, several other promising research directions emerge.

2. Deep Learning approaches, particularly LSTM networks, could better capture temporal dependencies in price movements.

3. Engineering of Features could be expanded to include more sophisticated technical indicators, and market microstructure metrics.

4. Incorporation of alternative Data Sources, such as Social Media Sentiment analysis, News headlines, and Blockchain network metrics, could provide additional predictive signals.

5. Furthermore, developing ensemble methods, which combine multiple Models' predictions, might offer more robust results across different market conditions.

6. Future research could, or should, also explore the implementation of adaptive Models that can adjust to changing market parameters and volatility patterns in real-time.

BIBLIOGRAPHY

Mudassir, M., Bennbaia, S., Unal, D. and Hammoudeh, M., 2020. Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach. *Neural computing and applications*, pp.1-15.

Ozsahin, D.U., Mustapha, M.T., Mubarak, A.S., Ameen, Z.S. and Uzun, B., 2022. Impact of feature scaling on machine learning models for the diagnosis of diabetes. In *2022 International Conference on Artificial Intelligence in Everything (AIE)* (pp. 87-94). IEEE.

Stańczyk, U., 2015. Feature evaluation by filter, wrapper, and embedded approaches. *Feature selection for data and pattern recognition*, pp.29-44.

Arumugam, S.K. and Uma, V.R., (2024) Analysing Crypto Trends: Unveiling Ethereum and Bitcoin Price Forecasts Through Analytics-Driven Weighted Moving Averages. *Intelligent Systems Design and Applications: Information and Network Security, Volume 3*, *3*, p.450.

Itoo, F., Meenakshi and Singh, S., 2021. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. *International Journal of Information Technology*, *13*(4), pp.1503-1511.

Chen, C.W., Tsai, Y.H., Chang, F.R. and Lin, W.C., 2020. Ensemble feature selection in medical datasets: Combining filter, wrapper, and embedded feature selection results. *Expert Systems*, *37*(5), p.e12553.

Istaltofa, M., Sarwido, S. and Sucipto, A., 2024. Comparison of Linear Regression and LSTM (Long Short-Term Memory) in Cryptocurrency Prediction. *Journal of Dinda: Data Science, Information Technology, and Data Analytics*, *4*(2), pp.141-148.

Velankar, S., Valecha, S. and Maji, S., 2018, February. Bitcoin price prediction using machine learning. In *2018 International Conference on Advanced Communication Technology (ICACT)* (pp. 144-147). IEEE.

Phaladisailoed, T. and Numnonda, T., 2018, July. Machine learning models comparison for bitcoin price prediction. In *2018 10th international conference on information technology and electrical engineering (ICITEE)*. IEEE.

APPENDIX (if necessary)