CS 210 TERM PROJECT

STEP 3

Hüseyin Eren YILDIZ - 31047

INTRODUCTION

My dataset, the Cryptocurrency Interest dataset, is provided by Google Trends. This Google Trends Dataset provides information about weekly cryptocurrency searches around the world, i.e. interest in cryptocurrencies. This dataset will illustrate the relationship that exists between the public's interest in cryptocurrency, as revealed by Google searches, and the variations in Bitcoin values that have been seen throughout the given time frame.

Here is my datasets:


Stock Market Dataset True.csv


Google_Crypto Currency Interest Data

The main aim of this project is to explore the relationship between Bitcoin prices and public interest through statistical and predictive *machine-learning models*. By investigating how changes in online search interest might relate to or impact Bitcoin prices, the analysis seeks to predict future price changes, offering insights for investment and financial planning strategies. Through rigorous data exploration, hypothesis testing, regression analysis and machine-learning models the project aims to uncover patterns and relationships that can provide more effective insight into the cryptocurrency market.

In the final stage of the project, two machine learning models were used. These machine learning models are K-Nearest Neighbors(kNN) and Random Forest. As mentioned above, by investigating how changes in online search interest might relate to or impact Bitcoin prices, the analysis seeks to predict future price changes, offering insights for investment and financial planning strategies.

Machine Learnin Models:

1) *K-Nearest Neighbor (kNN) Model*

K-Nearest Neighbor (kNN) emerges as a straightforward yet effective method in machine learning model, serving purposes in both classification and regression tasks with its intuitive approach. This model is classified as a non-parametric approach, indicating it does not presume any specific data distribution. Rather, it operates by gauging the distance between data points to formulate predictions.

For classification tasks, kNN determines the class of a new data point by selecting the most prevalent class among its k-nearest neighbors. In regression, it forecasts the value of a new data point by averaging the values of its k-nearest neighbors.

Working Algorithm of kNN Model

The kNN machine learning model works as follows:

1) *Choose the number of neighnors (k):* This hyperparameter is pivotal in shaping the algorithm's efficacy. A small value of k can cause overfitting by being too sensitive, while a large value of k can lead to underfitting by smoothing out important details.

2) *Calculate the distance:* Distances between data points are computed using metrics like Euclidean, Manhattan, or Minkowski distance.

3) *Find the k-nearest neighbors:* Find the k data points in the training set that are nearest to the new data point.

4) *Make a prediction:*

Classification: Determine the class that occurs most frequently among the k-nearest neighbors and assign it to the new data point.

Regression: Calculate the average value of the k-nearest neighbors.

Applying kNN Machine-learning Model to My Datasets:

In addition to the steps mentioned above, the steps mentioned below were applied to implement this machine learning model.

1) Data Loading: I started by importing the Bitcoin price along with the cryptocurrency search data.

2) Data Preprocessing:

   - *Handling Missing Values*: Any missing values in the datasets were handled by either filling them with appropriate values.
   - *Feature Scaling*: Since kNN relies on distance metrics, the features were normalized to ensure that each feature contributes equally to the distance computation. This was done using techniques like Min-Max scaling or Standardization.

3) Splitting the Data: The datasets were split into training and testing sets to evaluate the model's performance. Typically, 80% of the data was used for training, and the remaining 20% was used for testing.

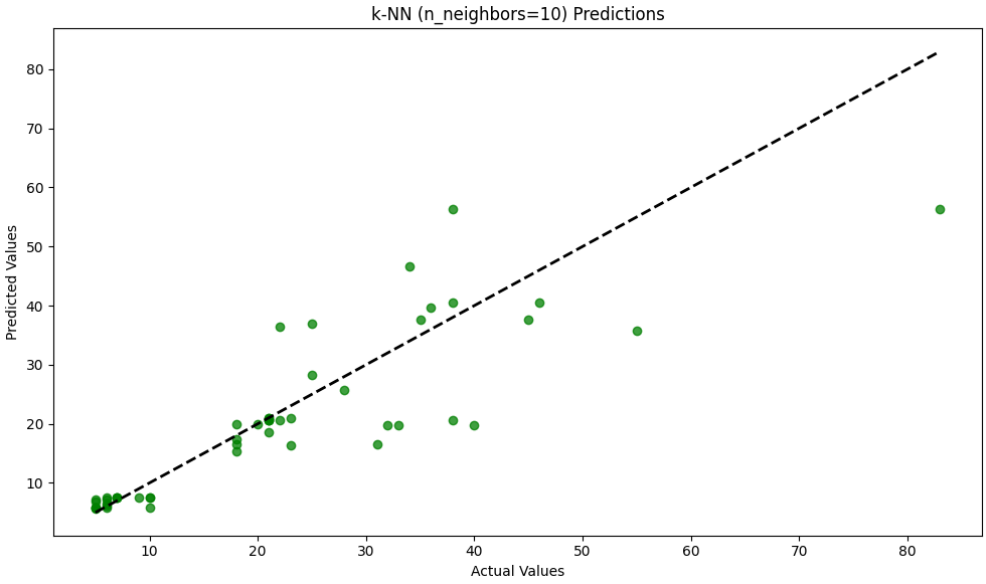
4) Initilize the Value of 'k' : An initial k value was selected and tried to find the optimal value through cross-validation. For regression tasks, the KNeighborsRegressor was initialized with a chosen k value and distance metric and trained on the training dataset.

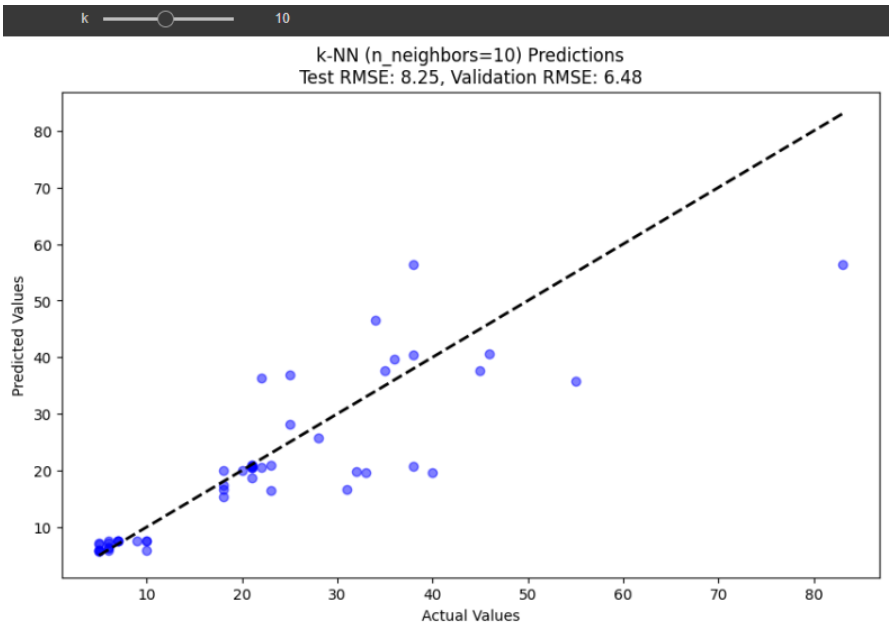
5) Model Predeciton: Applied the trained kNN model to generate predictions for the testing dataset. For regression, the predicted values were calculated.

6) Model Evaulation: The kNN model's performance was assessed using relevant metrics. Regression metrics like the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE) were calculated to evaluate the accuracy of prediction.

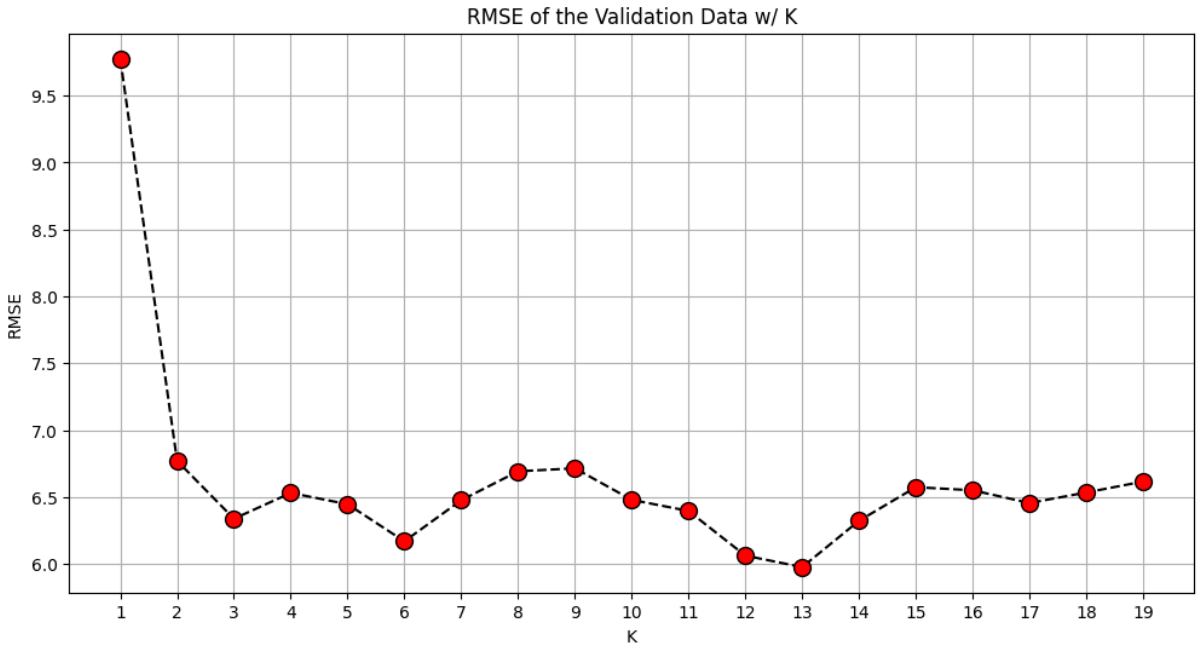
7) Hyperparameter Tuning: To find optimal values of 'k' several values were tested. This process included employing cross-validation to identify the optimal k value that yields the best results on validation sets.

Firstly, I tried some values of k. For example, I initialized the k=1. Then, I found MSE AND RMSE values for k=1. Finally, I visiualized to show relationship between predicted Cryptocurrency_Interest and actual values. (RMSE = 9.22 / MSE=84.96)

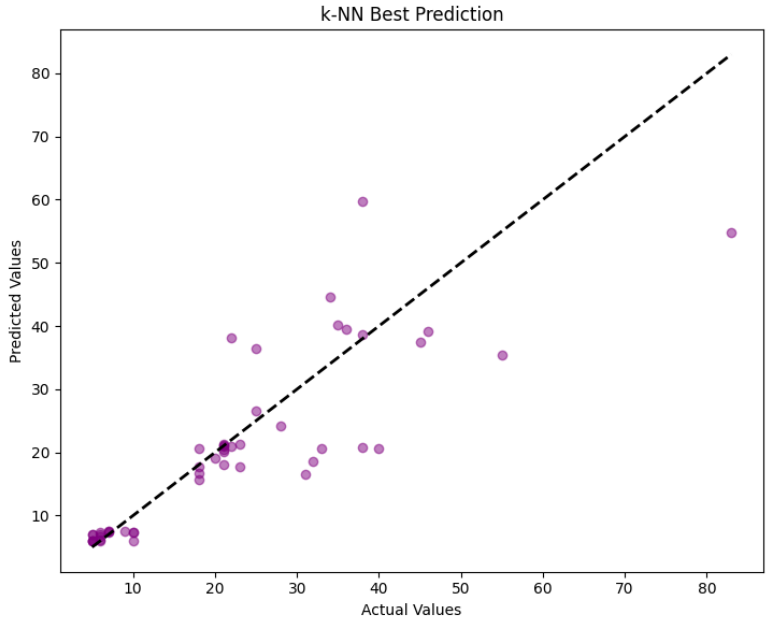




- To determine the optimum ‘k’ value, an interaction graph that illustrates the relationship between predicted Cryptocurrency_Interest and actual values, which vary according to different k values was created



- Then, k value graph was created to show the optimum k value for kNN machine learning model prediction. This graph shows that the smallest RMSE is the best k value for the model which is 13.



- As seen in the k Value graph, the optimum k value is 13. After receiving this information, I visualized the model that gave the best prediction.

RMSE: 8.52

MSE: 72.59

In applying the k-Nearest Neighbors (kNN) machine learning model to my datasets, I followed a systematic approach that ensured robust and accurate predictions. Starting with data loading and preprocessing, I handled missing values and normalized the features to maintain consistency in distance calculations. Splitting the data into training and testing sets allowed me to evaluate the model's performance effectively.

By initially selecting a value for k, I employed cross-validation to find the optimal k value, ensuring that the model balanced bias and variance. Through iterative testing and evaluation, I determined that k=13 provided the best performance, as indicated by the lowest Root Mean Squared Error (RMSE) and Mean Squared Error (MSE) values.

Visualizing the relationship between predicted and actual values for different k values facilitated the identification of the optimal k. The final model, with k=13, demonstrated superior predictive capability, achieving an RMSE of 8.52 and an MSE of 72.59.

In summary, selecting the optimal k value and other parameters significantly reduces prediction errors, providing more accurate and reliable results for Bitcoin price predictions based on cryptocurrency search interest.

2) *Random Forest Model:*

The Random Forest model shines as a versatile and robust ensemble learning technique in machine learning, well-suited for both classification and regression tasks. By leveraging the strengths of multiple decision trees, it enhances performance and efficiency. Renowned for its capacity to handle large datasets and maintain accuracy, even in the presence of significant missing data, this model stands out as a reliable choice for various real-world scenarios.

Working Algorithm of *Random Forest Model*:

The Random Forest machine learning model works as follows:

1) *Build Multiple Decision Trees:*

- Bootstrap Sampling: To construct each decision tree, random subsets of the training data are created with replacement (bootstrap sampling), ensuring diversity among the trees
- Feature Randomness: To enhance the model's performance, a random subset of features is selected at each split in the tree. This introduces diversity and reduces correlation among the trees, contributing to the model's effectiveness.

2) *Grow Each Tree:* Every decision tree is allowed to grow fully without any pruning, which can cause overfitting to the bootstrapped sample it was trained on. Nonetheless, the ensemble nature of Random Forests helps mitigate this overfitting by averaging the predictions across multiple trees.

3) *Make Predictions*:

Regression: When presented with a new data point, each tree generates a prediction. The Random Forest model's final prediction is the average of all these predicted values.

Applying Random Forest Machine-learning Model to My Datasets:

In addition to the steps mentioned above, the steps mentioned below were applied to implement this machine learning model.

1) Data Loading: I started by importing the Bitcoin price along with the cryptocurrency search data.

2) Data Preprocessing:

- *Handling Missing Values*: Any missing values in the datasets were handled by either filling them with appropriate values.
- *Feature Scaling*: Since kNN relies on distance metrics, the features were normalized to ensure that each feature contributes equally to the distance computation. This was done using techniques like Min-Max scaling or Standardization.

3) Splitting the Data: The datasets were split into training and testing sets to evaluate the model's performance. Typically, 80% of the data was used for training, and the remaining 20% was used for testing.

4) Random Forest Model Initialization: Different n_estimators (number of decision trees) was selected and tried to find the optimal value.

5) Train the Random Forest Model: Random Forest ML model was trained using the training set. Training started with the optimal values for n_estimators and max_depth.

   n_estimators: The n_estimators parameter sets the number of trees in the random forest. Testing different values helps find the best trade-off between accuracy and computational load. More trees can improve performance but also increase processing time, so an optimal number ensures efficiency and effectiveness.

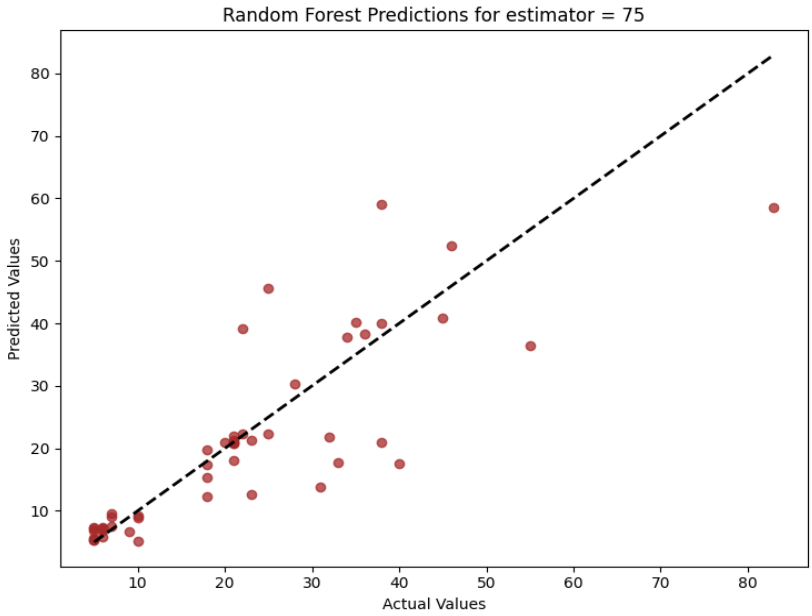
   max_depth: The max_depth parameter defines the maximum depth of each tree. Adjusting this parameter helps prevent overfitting by controlling the complexity of the trees. A suitable max depth ensures the model captures essential patterns without fitting to noise, promoting better generalization to new data.

6) Model Prediction: Use the trained Random Forest model to make predictions on the testing set.

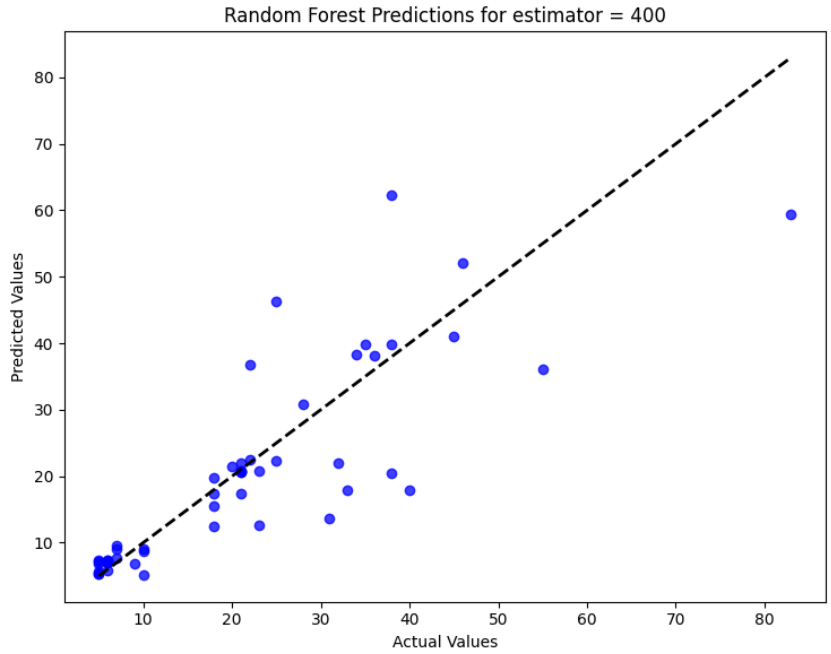
7) Model Evaluation: The Random Forest Model's performance was assessed using relevant metrics. These metrics are the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE).

8) Hyperparameter Tuning: Utilize techniques such as grid search or randomized search to discover the optimal hyperparameters, thereby improving the model's performance.

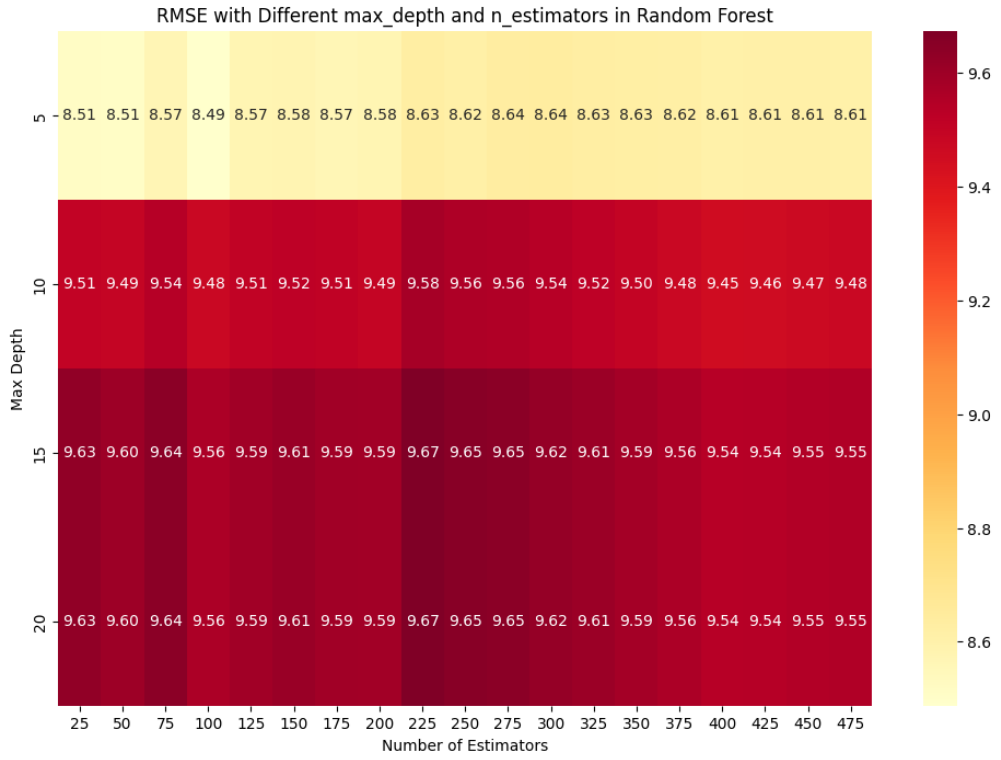
Just like the kNN model, I first tried some n_estimator and max_depth values. For example, I initialized the n_estimator = 75 and max_depth = 15 . Then, I found MSE AND RMSE values for n_estimator = 75 and max_depth = 15. Finally, I visiualized to show relationship between predicted Cryptocurrency_Interest and actual values. (RMSE = 8.8 / MSE= 77.44)



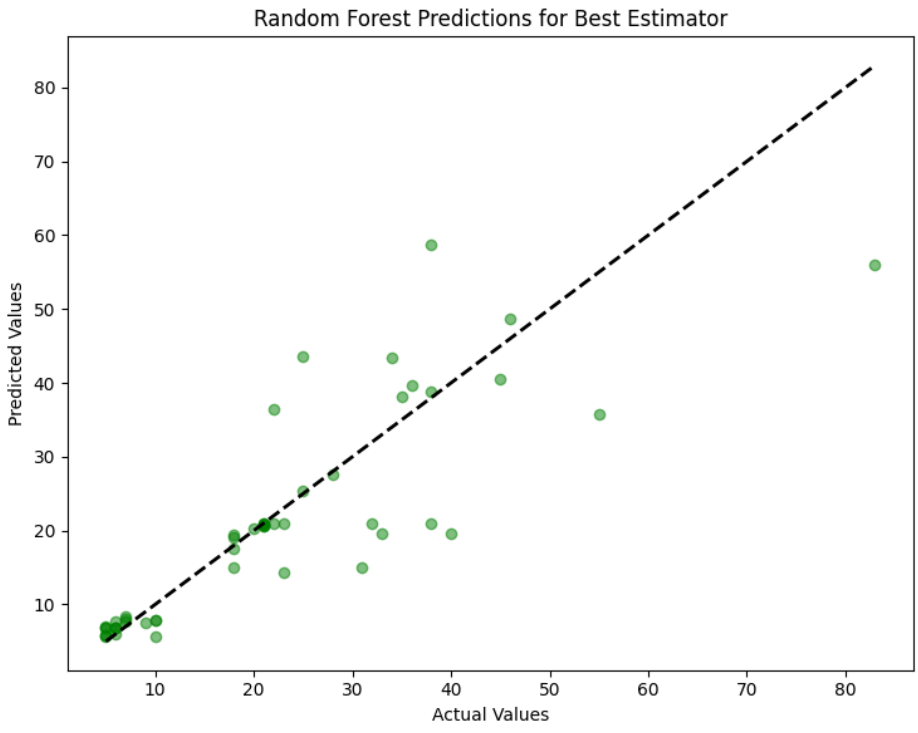
Then, I tried n_estimator = 400 and max_depth = 15. (RMSE = 8.89 / MSE= 79.06)



Finally, I used a colored RMSE chart to find the optimal n_estimator and max_depth values. As can be seen from the below, the n_estimator(100) and max_length(5) values that give the lowest RMSE value (8.49) were easily found with the help of this graph.



After finding the optimum values, I visualized these best values. (RMSE = 8.49 and MSE = 72.52)



Conclusion:

Which one performs better ?

The Random Forest model, leveraging an ensemble of decision trees, proved to be more adept at capturing the dataset's intricate relationships, resulting in higher accuracy and reliability in predictions compared to the kNN model. Scatter plots and performance metrics revealed the Random Forest model's superior ability to minimize prediction errors and effectively manage data complexities. This analysis underscores the necessity of carefully choosing and fine-tuning machine learning algorithms for predicting Bitcoin prices using Google cryptocurrency search interest. The Random Forest model, in particular, demonstrated greater effectiveness and consistency. While both models benefited from hyperparameter tuning, the Random Forest model showed greater effectiveness in managing the intricacies of the data and delivering robust predictions.

Advantages of kNN:

- Easy to understand and implement.
- It simply calculates the distance of a new data point to all other training data points.

Advantages of Random Forest Model:

- Reduction of overfitting: By aggregating predictions from multiple decision trees, random forests reduce overfitting compared to individual decision trees.
-
- Robustness to noise: Random forests are less susceptible to noise in the data due to their ensemble nature.

The Random Forest model outperformed the kNN model, achieving a lower RMSE (8.49 vs. 8.52) and MSE (72.59 vs. 72.52). Its ensemble approach was more effective at capturing the complex relationships within the data, resulting in more accurate and reliable predictions.

This comparison emphasizes the critical role of selecting and optimizing the right machine learning algorithms for predicting Bitcoin prices based on Google cryptocurrency search interest. The Random Forest model stood out, showcasing its strength and delivering better performance.