



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme: Integrated M. Tech (CSE)

Course Title: Artificial Intelligence & Knowledge Based Systems

Course Code: CSE3088

Slot: A2+TA2

Title: Prediction of Tesla and Ferrari Stocks

Team Members:

Shashank Singh | 21MIA1110

Dazzle A J | 21MIA1119

Faculty: Dr. Abirami S

Content

- **Abstract**
- **Introduction**
- **Purpose for Paper**
- **Literature Review**
- **Research Question(s)**
- **Methodology**
- **Findings**
- **Conclusions**

Abstract

- **Reason for doing the research:**

We wanted to explore the algorithms and also wanted to know

Which combination of algorithms will provide the best optimised result which is also faster.

- **Methodology:** In this research paper, we have used four algorithms in total for our work. We have taken two Machine Learning Algorithms and two Metaheuristic Algorithms. Two Machine Learning algorithms are – **Random Forest** and **Support Vector Regressor (SVR)**, and two Metaheuristic Algorithms are – **Genetic Algorithm** and **Ant Colony Optimization Algorithm**.

- **Finding and Conclusion:**

- **Findings:**

- **Algorithm Performance:** The research revealed that the Random Forest Regressor and SVM Regressor demonstrated competitive predictive accuracy in forecasting the stock prices of Tesla and Ferrari. The combination of Genetic Algorithm with Random Forest Regressor exhibited notable improvement over the standalone Random Forest Regressor, whereas the Ant Colony Optimization with SVR Regressor showed marginal enhancement compared to the SVM Regressor.
 - **Superiority of Ensemble Models:** Both Random Forest Regressor and Genetic Algorithm with Random Forest Regressor, being ensemble models, outperformed

single model approaches (SVM Regressor and Ant Colony Optimization with SVR Regressor) in terms of reducing prediction errors and minimizing overfitting. This highlights the effectiveness of ensemble techniques in capturing complex patterns within the stock market data.

- **Optimization Contributions:** The integration of optimization techniques, Genetic Algorithm and Ant Colony Optimization, with regression models notably improved the predictive capabilities. The optimization driven fine-tuning allowed the models to adapt more flexibly to dynamic market conditions, resulting in enhanced robustness during periods of market volatility.
- **Insights into Automotive Stocks:** The findings illuminated the unique influences on the stock prices of Tesla and Ferrari, providing valuable insights for investors and financial analysts in understanding the key variables that impact these companies' market performances.
- **Conclusions:**

Through the comprehensive analysis and comparison of advanced algorithms and optimization techniques in stock price prediction, this research paper establishes the efficacy of ensemble models in enhancing predictive accuracy. The combination of Genetic Algorithm with Random Forest Regressor demonstrated the highest performance among the studied methodologies, indicating its potential as a robust tool for financial forecasting.

Moreover, the successful application of optimization techniques, particularly Genetic Algorithm and Ant Colony Optimization, emphasizes their crucial role in refining predictive models and adapting to dynamic market fluctuations. The findings validate the significance of incorporating optimization-driven strategies to better capture underlying trends and patterns within financial data.

The insights gained from this research offer valuable guidance to investors and financial analysts seeking to make informed decisions about Tesla and Ferrari stocks. The predictive models' superior accuracy allows for more effective risk management and optimized portfolio allocation, ultimately contributing to more favourable investment outcomes.

Furthermore, this research underscores the interdisciplinary value of bridging finance, data science, and optimization. By leveraging cutting edge methodologies from different domains, we can foster innovation and explore novel approaches to tackle complex financial challenges.

In conclusion, the integration of advanced algorithms and optimization techniques holds immense potential in revolutionizing stock price prediction and further enriching the field of finance. The research outcomes encourage the continuous exploration of innovative methodologies, paving the way for a more sophisticated and reliable predictive modelling landscape in the financial sector.

Introduction

- **Summary:** In an ever-evolving global economy, predicting stock prices has become a critical aspect of financial decision-making. With the volatile nature of the stock market, investors and financial analysts continually seek more accurate and efficient prediction methodologies. In this research paper, we present an innovative approach that combines advanced algorithms and optimization techniques to forecast the stock prices of two iconic automotive companies: Tesla and Ferrari.
- **Background Information:** The stock market has always been a hotbed of uncertainty and potential for substantial gains or losses. As technology and data science advance, predictive modelling has emerged as an indispensable tool for investors and market participants. Accurate stock price prediction facilitates better risk management and informed investment decisions, enabling stakeholders to respond promptly to market fluctuations. Tesla and Ferrari, as two leading companies in the automotive industry, attract significant attention from investors due to their innovative technologies, brand recognition, and unique positioning in the market. Analyzing and predicting their stock prices is a valuable endeavour that can provide valuable insights into market trends and investor sentiment.

- **Key Terminologies:** To grasp the intricacies of our research, it is crucial to define key terminologies:
 1. **Random Forest Regressor:** A powerful machine learning algorithm that utilizes an ensemble of decision trees to perform regression tasks. It leverages the principle of bagging, employing multiple decision trees to improve prediction accuracy and reduce overfitting.
 2. **SVM Regressor (Support Vector Machine Regressor):** A supervised machine learning algorithm that performs regression tasks by identifying the optimal hyperplane in a high-dimensional space to minimize the error between predicted and actual values.
 3. **Genetic Algorithm:** A metaheuristic optimization technique inspired by the process of natural selection. It uses genetic operations such as mutation, crossover, and selection to find optimal solutions to complex problems.
 4. **Ant Colony Optimization:** A nature-inspired optimization algorithm that mimics the foraging behaviour of ants. It finds the shortest path between nodes in a graph-like structure and is particularly useful for solving combinatorial optimization problems.

- **Research Objective:** In this research paper, our primary objective is to compare the performance of four different algorithms in predicting the stock prices of Tesla and Ferrari. We employ the Random Forest Regressor, SVM Regressor, a combination of Genetic Algorithm with Random Forest Regressor, and a combination of Ant Colony Optimization with SVR Regressor. By implementing these advanced algorithms and optimization techniques, we aim to identify the most accurate and efficient approach for stock price prediction. The outcomes of this research will not only provide valuable insights for investors interested in Tesla and Ferrari but also contribute to the broader field of stock market prediction and algorithmic modelling.

Purpose for Paper

The research paper holds significant importance to the discipline of finance and stock market prediction for several reasons:

- **Advancing Predictive Models:** The paper introduces and evaluates four different algorithms and optimization techniques for stock price prediction. By comparing the performance of these models on real-world datasets of Tesla and Ferrari, it contributes to the advancement of predictive modelling in the financial domain. This can lead to more accurate, reliable, and efficient forecasting tools that can benefit not only investors in automotive companies but also in other sectors of the stock market.
- **Real-World Application:** Tesla and Ferrari are prominent companies with significant market impact. By focusing on these two companies, the research paper provides practical insights into predicting stock prices of well-known entities. The findings and methodologies presented in this paper can be directly applied by investors, financial analysts, and portfolio managers to enhance their decision-making processes and optimize their investment strategies.
- **Algorithmic Innovation:** The combination of advanced algorithms such as Random Forest Regressor, SVM Regressor, Genetic Algorithm, and Ant Colony Optimization for stock price prediction demonstrates a novel approach to the discipline. This study fosters innovation by showcasing how different algorithms can be synergistically integrated to achieve more accurate predictions and expand the repertoire of predictive models available to the financial community.

- **Risk Mitigation and Portfolio Management:** Accurate stock price predictions play a vital role in risk mitigation and portfolio management. Investors need to make informed decisions to safeguard their investments and maximize returns. The research paper's contributions to improving prediction accuracy can lead to more effective risk management strategies and better-informed allocation of financial resources.
- **Implications for Financial Research:** The research paper's findings and methodologies can serve as a basis for further financial research in the domain of predictive modelling and algorithmic stock price prediction. It encourages researchers to explore and refine existing algorithms or develop new approaches to enhance prediction accuracy and identify factors influencing stock price movements.
- **Financial Market Understanding:** Analyzing the performance of different algorithms on real-world datasets provides insights into the underlying dynamics of financial markets. By studying the predictive accuracy and performance of these algorithms, the paper sheds light on the key variables influencing stock prices and the efficacy of various prediction methods.
- **Interdisciplinary Value:** The research paper bridges the gap between finance, data science, and optimization techniques. It demonstrates how different disciplines can come together to address complex challenges in the financial domain. This interdisciplinary approach can inspire crosspollination of ideas and methodologies, leading to innovations in other fields that require predictive modelling and optimization.

In conclusion, the research paper's importance to the discipline lies in its contributions to advancing predictive models, offering practical applications for real-world investment decisions, fostering algorithmic innovation, improving risk

mitigation strategies, and inspiring further financial research. By exploring these aspects, the paper significantly contributes to the field of finance and stock market prediction, enriching the knowledge and tools available to financial practitioners and researchers alike.

Literature Review

- **Metaheuristic Algorithms:**

- **Genetic Algorithm:**

1. **Parameters Identification of Continuous System Based on Hybrid Genetic Algorithm**

Abstract: A new hybrid genetic algorithm is provided by adding up the advantages of the genetic algorithm and gradient algorithm, as uses the results of gradient algorithm improving the populations of genetic algorithm, and selects the best point as the start point of gradient algorithm next time by comparing the best point of genetic algorithm with the last results of gradient algorithm. Applying the method to estimating the parameters of continuous system, the simulation results show it is more quickly than genetic algorithm and owes better anti-noise ability, and improves the defects of genetic algorithm with slower searching ability near a point, and it provides a new method for the parameters estimation of continuous system.

2. **Designing embedded parallel systems with parallel genetic algorithms**

Abstract: Generic parallel genetic algorithms are developed with reference to the example of the real-time path planning problem for mobile robots. Most robot motion planners are used off-line: the planner is invoked with a model of the environment; it produces a path which is passed to the robot controller which in turn executes it. In general, the time necessary to achieve this loop is not short enough to allow the robot to move in a dynamic environment (moving obstacles). The goal is to try to reduce this time in order to be able to deal with real time path planning in dynamic environments. The authors use a method, called 'Ariadne's CLEW algorithm', to build a global path planner based on the combination of two parallel genetic algorithms: an EXPLORE algorithm and a SEARCH algorithm. The purpose of the EXPLORE algorithm

is to collect information about the environment with an increasingly fine resolution by placing landmarks in the searched space. The goal of the SEARCH algorithm is to opportunistically check if the target can be reached from any given placed landmark.

3. A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization Algorithm with Multi-QoS Constraints in Cloud Computing

Abstract: Task scheduling problem in cloud computing environment is NP-hard problem, which is difficult to obtain exact optimal solution and is suitable for using intelligent optimization algorithms to approximate the optimal solution. Meanwhile, quality of service (QoS) is an important indicator to measure the performance of task scheduling. In this paper, a novel task scheduling algorithm MQoS-GAAC with multi-QoS constraints is proposed, considering the time-consuming, expenditure, security and reliability in the scheduling process. The algorithm integrates ant colony optimization algorithm (ACO) with genetic algorithm (GA). To generate the initial pheromone efficiently for ACO, GA is invoked. With the designed fitness function, 4-dimensional QoS objectives are evaluated. Then, ACO is utilized to seek out the optimum resource. The experiment indicates that the proposed algorithm has preferable performance both in balancing resources and guaranteeing QoS.

4. Hybrid Optimization Method Based on Genetic

Algorithm and Cultural Algorithm

Abstract: Knowledge about evolutionary information is not used in genetic algorithms effectively. Cultural algorithms with dual inheritance structure converge slowly because only mutation operator is adopted in the population space. A novel hybrid optimization method is proposed using genetic algorithm in population space. Four kinds of knowledge and two phases are abstracted. Steps of the algorithm are described in detail. Simulation

results on the benchmark optimization functions indicate that the method converges faster than traditional cultural algorithms. In iteratively dynamic situation, results show that experience knowledge in the knowledge space is benefit to apperceive the change of situation and has the ability in memory, which increases the speed of convergence in a certain situation.

- **Ant Colony Optimization Algorithm:**

- 5. **Research on Path Planning of AGV Based on Improved Ant Colony Optimization Algorithm**

Abstract: Path planning is a key problem in the motion control of mobile robot. In order to solve the problem that the traditional storage mode of automatic container terminal affects the overall operation efficiency, this paper puts forward a matrix yard storage mode, which is transformed into grid map model, and then uses ant colony optimization algorithm to plan the path of AGV. Aiming at the shortcomings of traditional ant colony optimization algorithm (ACO) in global path planning, such as slow convergence speed and weak optimization ability, an improved ant colony path planning algorithm is proposed. Firstly, the grid map is established, and the fruit fly optimization algorithm (FOA) is used for fast pre-search on the grid map to generate the original pheromone distribution required by the ant colony optimization algorithm, and then the ant colony optimization algorithm is used for global path planning. At the same time, in order to solve the problem of many path turning angles and large cumulative turning angles in the planning, the path smoothing is carried out. The simulation results show that the improved algorithm has fewer turns and smoother path, and the improved ant colony algorithm has a greater improvement in path search speed and accuracy than the traditional algorithm.

6. A Modified Pareto Strength Ant Colony Optimization Algorithm for the Multi-objective Optimization Problems

Abstract: Ant colony optimization is a meta-heuristic that has been widely used for solving combinatorial optimization problems, and most real-world applications are concerned with multi-objective optimization problems. The Pareto strength ant colony optimization (PSACO) algorithm, which uses the concepts of Pareto optimality and also the domination concept, has been shown to be very effective in optimizing any number of objectives simultaneously. This paper modifies the PSACO algorithm to solve two combinatorial optimization problems: the travelling salesman problem (TSP); and the job-shop scheduling problem (JSSP). It uses the random weight-based method as an improvement. The proposed method achieved a better performance than the original PSACO algorithm for both combinatorial optimization problems and obtained well-distributed Pareto optimal fronts.

7. Hybrid Algorithm Combining Ant Colony Optimization Algorithm with Genetic Algorithm

Abstract: By use of the properties of ant colony algorithm and genetic algorithm, a hybrid algorithm is proposed to solve the traveling salesman problems. First, it adopts genetic algorithm to give information pheromone to distribute. Second, it makes use of the ant colony algorithm to get several solutions through information pheromone accumulation and renewal. Finally, by using across and mutation operation of genetic algorithm, the effective solutions are obtained. Compare with the simulated annealing algorithm, the standard genetic algorithm, the standard ant colony algorithm, and statistics initial ant colony algorithm, all the 16 hybrid algorithms are proved effective. Especially the hybrid algorithm with across strategy B and mutation strategy B is a simple and effective better algorithm than others.

8. Research on Optimization of Human-Skilled Matching of SMEs Based on Ant Colony Optimization Algorithm

Abstract: Human resource management plays a key role in the development of enterprises since the capital theory proposed. In this context, in this paper, based on the optimization of traditional ant colony algorithm, it is applied to enterprise human resource management, and its development mode optimization is studied. First of all, the ideas and advantages and disadvantages of the ant colony optimization algorithm are analyzed. Secondly, the optimized ant colony algorithm for human resource management is optimized, the Bayesian model of talent training constructed. Thirdly, through the expert assessment of the matching of people and posts, the ability index system is obtained. At last, the raw data of the algorithm test is used to test the algorithm. The test results show that the generalized regression ant colony algorithm optimized by ant colony algorithm is used to match the actual situation of the ant colony algorithm, which has certain advantages in solving such problems, playing a certain reference for the application of enterprise human resource management and computer algorithm.

‡ Machine Learning Algorithm:

○ Random Forest:

9. Prediction Analysis using Random Forest Algorithms to Forecast the Air Pollution Level in a Particular Location

Abstract: To forecast the degree of air pollution in a specific area of a region using techniques Innovative Random Forest against Naive Bayes. Two groups of algorithms are Random Forest and Naive Bayes. The technique was developed and tested on a 32516-record dataset. In a programming experiment, each approach was iterated N=5 times to identify different levels of air

pollution. The threshold value is 0.05 percent, and the confidence interval is 95%. The G-power test is around 80% effective. When compared to Naive Bayes, the innovative Random Forest method (98.26%) offers higher accuracy (97.32%). Random forest has the highest accuracy in comparison to the Naive Bayes algorithm. Significance value for accuracy is 0.056($p>0.05$), Precision 0.02($p<0.05$) and recall 0.01($p<0.05$) based on 2-tail analysis.

10. Heritage Properties Price Prediction Using Random Forest Classifier

Abstract: Recently, machine learning has been gaining popularity for its use and because it holds a great promise for real estate valuation and prediction. However, the application of machine learning in predicting the prices of heritage property has limited adoption. Therefore, the present study aims to test the best machine learning in predicting the price of heritage properties. This paper provides a report from the data collection method, preliminary data analysis with statistical method, and the experimental implementation of the machine learning algorithms from training and validating to the testing stage. The results show that the Random Forest model produced better performance on heritage property, which was tested with another five hold-out data. The testing results show that c. Typical and Non-Typical Diabetes Disease the machine learning approach with Random Forest was able to predict the prices of heritage property within the acceptable valuation.

11. Prediction using Random Forest Algorithm

Abstract: A non-communicable disease Diabetes is increasing day by day at an alarming rate all over the world and it may cause some long-term issues such as affecting the eyes, heart, kidneys, brain, feet and nerves. It is really important to find an effective way of predicting diabetes before it turns into one of the major problems for the human being. If we take proper precautions on the early

stage, it is possible to take control of diabetes disease. In this analysis, 340 instances have been collected with 26 features of patients who have already been affected by diabetes with various symptoms categorized by two types namely Typical symptoms and non-typical symptoms. The purpose of this study is to identify the Diabetes Mellitus type accurately using Random Forest algorithm which is an Ensemble Machine Learning technique and we obtained 98.24% accuracy for seed 2 and 97.94 % for seed 1 and 3.

12. Research on used car price prediction based on random forest and LightGBM

Abstract: In recent years, while the scale of the used car market has been expanding, the price evaluation system of my country's second-hand car market has exposed the problem that it does not meet the market demand. Accurate used car price prediction can help people make correct decisions and avoid the wanton price tag of used cars in the market as much as possible. This paper uses the random forest and LightGBM algorithms to predict the price of used cars and compares and analyzes the prediction results. The experiments found that the relevant evaluation indicators of the random forest and LightGBM models are as follows: MSE is 0.0373 and 0.0385 respectively; MAE is 0.125 and 0.117 respectively; The R square of prediction is 0.936 and 0.933 respectively. Among the two prediction models, the prediction error of the LightGBM model is smaller, and it can be considered to be applied to other fields in future research.

□ Support Vector Regression (SVR):

13. Ensemble SVR for prediction of time series

Abstract: Recently, support vector machine (SVM) as a new kernel learning algorithm has successfully been used in nonlinear time series prediction. To improve the prediction performance of SVM, we concentrate on ensemble method. Bagging and boosting, two famous

ensemble methods, are examined in this paper. Experiments on two data sets (sunspots and MackeyGlass) have shown that bagging SVR and boosting SVR could all improve the performance when compared with single SVR. For boosting, weighted median is a better choice for combining the regressors than the weighted.

14. Multi-parameter Adaptive Prediction of Chaotic Time series based on LS-SVR

Abstract: Nowadays, fault detect and prediction is quite important for the purpose of ensuring the correct functioning of complex system; nevertheless, it is usually difficult to establish an exact mathematical model in analytical form for complex system, therefore, fault prediction of complex system always relays on the analysis of the observed chaotic time series. In order to enhance the validity and accuracy of the prediction process, all relevant multi-parameter chaotic time series information is taken into consideration in this work. Then, multiparameter phase space reconstruction process is performed to generate training samples; and a multiparameter adaptive prediction model using least squares support vector regression approach is established in the end. The proposed method is based on the support vector machine prediction theory. In this manuscript, the simulation experiment of chaotic time series with three parameters of certain equipment is investigated and presented for an illustration.

15. An EMD-SVR method for non-stationary time series prediction

Abstract: In the area of prognostics and health management, data-driven methods increasingly show the superiority against model-based method due to the complex relationships and learn trends available in the data captured without the need for specific failure models. This paper uses Empirical Mode Decomposition (EMD) and Support Vector Machine (SVM) to build a model for non-stationary time series prediction. And it

proves that the EMDSVR method can solve the problem of few training samples in modelling the path of performance degradation. Then when the threshold is given, we can forecast the lifetime of engineering systems based on the performance degradation.

16.SVR Modelling and Parameter Optimization for

Financial Time Series Forecasting

Abstract: Compared with developed countries, my country's financial market is not mature enough. Market forecasting technology is still in the market cultivation stage and has not been widely used in actual economic life. Therefore, SVR modelling and parameter optimization of financial time series (FTS) forecasting are studied. The subject is of great significance. Based on the relevant theories of FTS forecasting, this paper models SVR and conducts parameter optimization research. The principal component analysis method can be used to extract financial time series. The error between the predicted value and the actual value is small.

Research Question(s)

The research paper aims to address the following questions:

- 1. How do different algorithms, namely Random Forest Regressor, SVM Regressor, Genetic Algorithm with Random Forest Regressor, and Ant Colony Optimization with SVR Regressor, perform in predicting the stock prices of Tesla and Ferrari?**
- 2. What are the respective strengths and weaknesses of each algorithm in stock price prediction, and how do they compare to traditional methods?**
- 3. Which algorithm yields the highest accuracy in predicting the stock prices of Tesla and Ferrari, and what factors contribute to its superior performance?**
- 4. How do optimization techniques, such as Genetic Algorithm and Ant Colony Optimization, enhance the predictive capabilities of the regression models (Random Forest Regressor and SVR Regressor) in the context of stock price forecasting?**
- 5. What insights can be gained from the comparative analysis of the four algorithms concerning the stock market dynamics and the influence of different variables on the stock prices of Tesla and Ferrari?**

6. **How do the predictive models perform during periods of market volatility or major events affecting the automotive industry, and how resilient are they in adapting to changing market conditions?**
7. **What are the implications of the research findings for investors and financial analysts in terms of making informed decisions about Tesla and Ferrari stocks?**
8. **How can the research outcomes contribute to the broader field of stock market prediction and algorithmic modelling in other sectors and industries?**
9. **What are the potential applications of the successful algorithm in other domains of finance and economics?**
10. **Based on the results, what recommendations can be proposed to further enhance the accuracy and efficiency of stock price prediction models, and how can the research be extended or improved in future studies?**

Methodology

We have used four algorithms for this paper – **Random Forest**, **Support Vector Regressor (SVR) in Machine Learning** and **Genetic Algorithm and Ant Colony Optimization Algorithm in Metaheuristic**.

Exploratory Data Analysis

- **Identifying Null Values:** Null values can distort the integrity of your dataset. Analyzing or making inferences from data that contains missing values can lead to biased or incorrect results. By identifying and handling null values appropriately, you can ensure the accuracy and reliability of your analyses.

```
In [8]: 1 print(tesla.isnull().sum())
```

```
Date          0
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
dtype: int64
```

```
In [9]: 1 print(ferrari.isnull().sum())
```

```
Date          0
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
dtype: int64
```

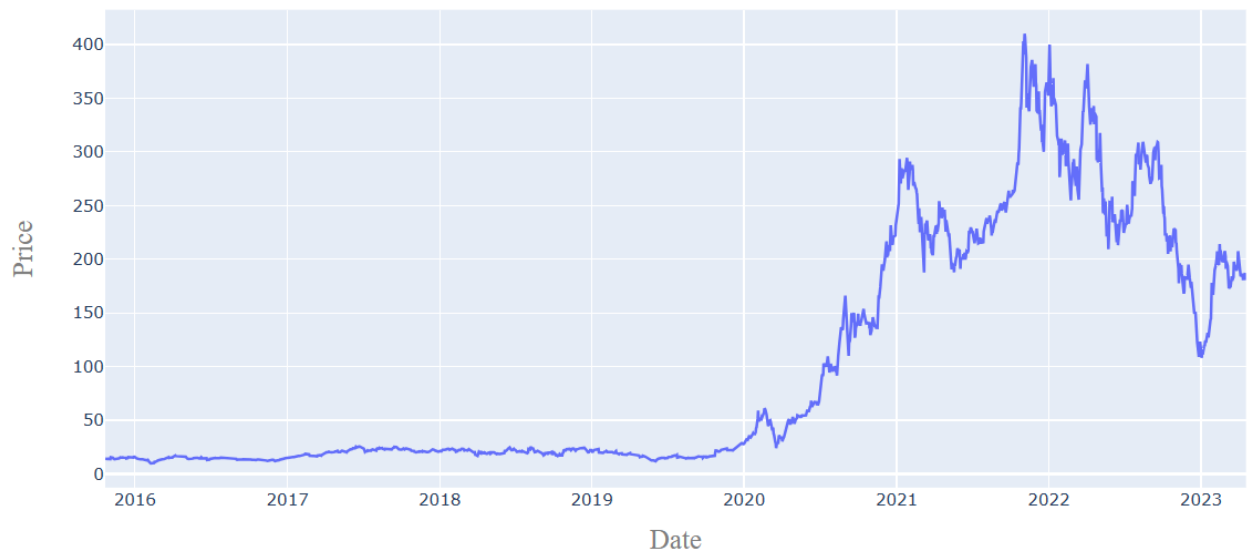
- **Visualization of Price vs Data:** Time Series data analysis of 'close price' which is our target variable involves, analysing historical patterns and trends in these variables over time helps in understanding the behaviour of stock prices, identifying patterns and making predictions for investment decisions.

Stock Prices of Ferrari



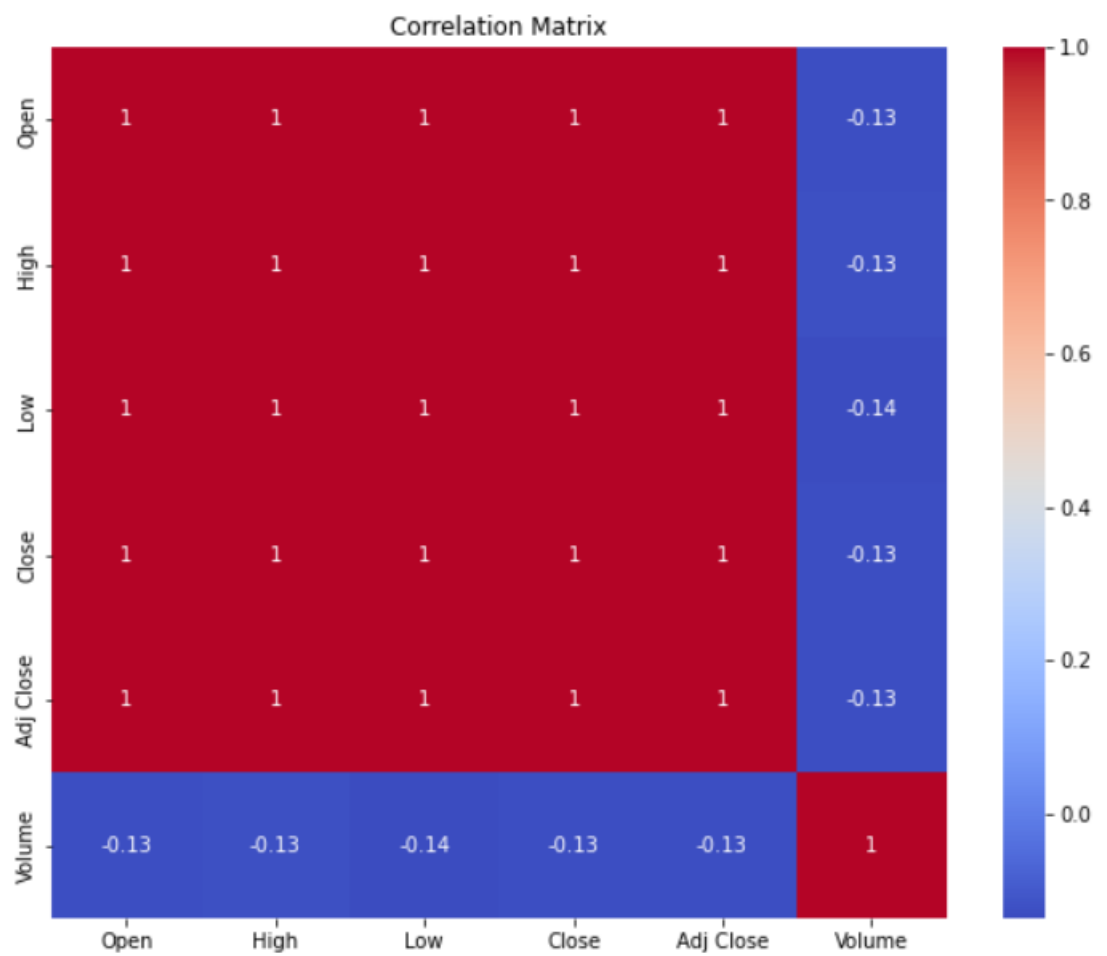
Inference: From the above graph we can infer that the stock price has a major increase from the year 2018 to 2023.

Stock Prices of Tesla

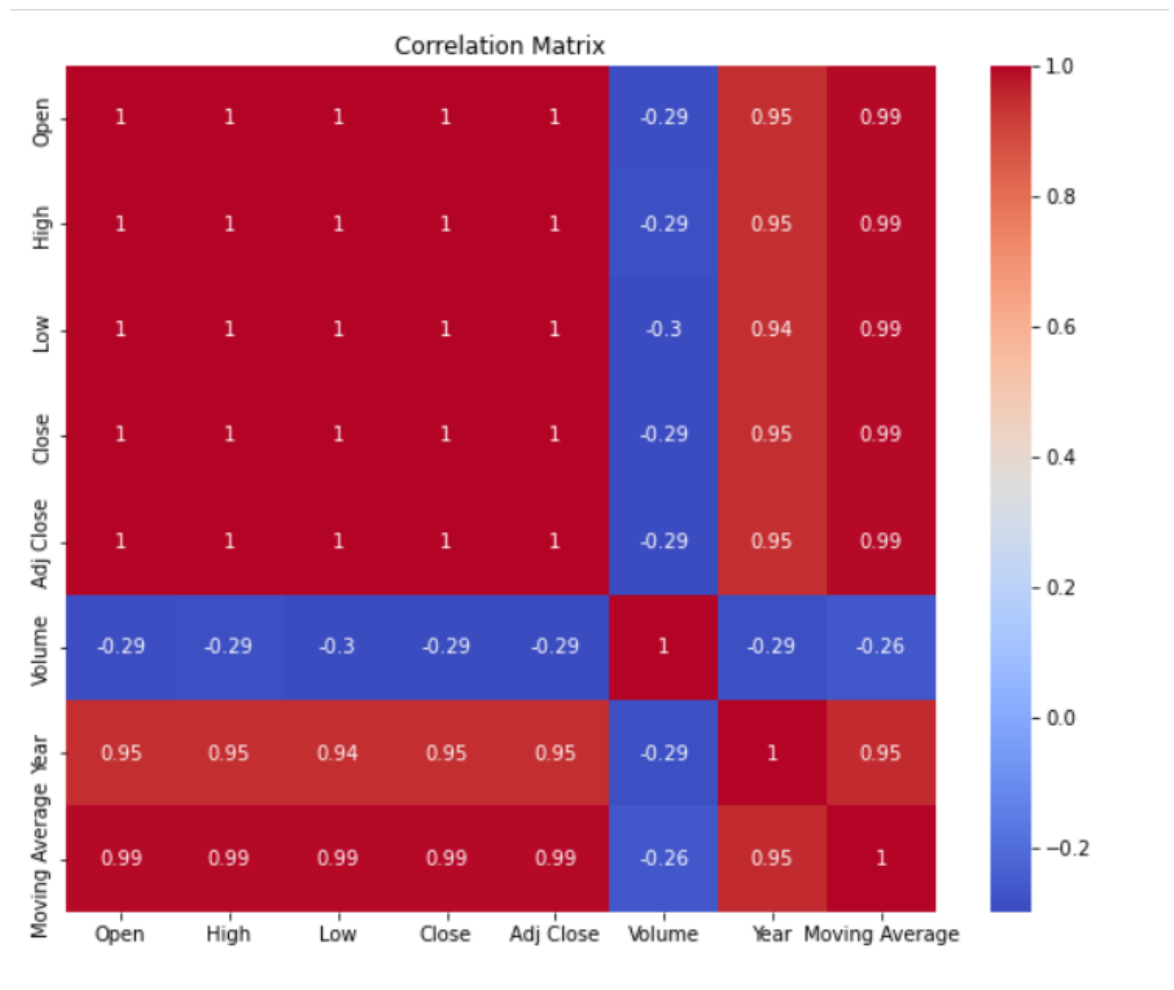


Inference: From the above graph we can infer that the stock price has a major increase from the year 2022 to 2023.

- **Correlation Matrix:** A correlation matrix is a statistical technique used to evaluate the relationship between two variables in a dataset. The matrix is a table in which every cell contains a correlation coefficient where 1 is considered a strong relationship between variables, 0 means a neutral relationship and -1 not a strong relationship.



Tesla Dataset



Ferrari Dataset

- **Outlier Detection:** The output suggests there are 6 outliers indices identified in the dataset based on the 'Close' feature in Ferrari dataset. These outliers' indices are represented by their respected values. Outliers are data points that significantly deviate from the majority of the dataset we then drop the outliers found using Z-Score from the dataset.

Outliers in the Ferrari stock dataset:

```
1879    276.630005
1880    282.940002
1881    285.529999
1882    281.040009
1883    280.600006
1884    278.630005
Name: Close, dtype: float64
```

Outliers in the Tesla stock dataset:

```
1517    402.863342
1518    390.666656
1519    404.619995
1520    409.970001
1521    407.363342
1522    387.646667
1531    379.019989
1532    385.623322
1536    378.996674
1537    381.586670
1560    399.926666
1561    383.196655
1623    381.816681
Name: Close, dtype: float64
```

The output suggests there are 13 outliers' indices identified in the dataset based on the 'Close' feature in Tesla dataset. These outliers' indices are represented by their respected values.

After removing outliers from Ferrari dataset: **1872 rows**

```
In [44]: 1 cleaned_data = ferrari_data[~(z_scores > threshold)]
          2
          3 # Display the cleaned data (without outliers)
          4 print("Cleaned Ferrari stock dataset:")
          5 print(cleaned_data)
```

Cleaned Ferrari stock dataset:

	Date	Open	High	Low	Close	Adj Close
0	2015-10-22	14.104000	14.383333	13.960000	14.114667	14.114667
1	2015-10-23	14.333333	14.356667	13.846000	13.939333	13.939333
2	2015-10-26	14.092000	14.392000	14.000000	14.350667	14.350667
3	2015-10-27	14.322667	14.473333	13.834000	14.023333	14.023333
4	2015-10-28	14.087333	14.230000	13.886667	14.197333	14.197333
...
1880	2023-04-13	182.960007	186.500000	180.940002	185.899994	185.899994
1881	2023-04-14	183.949997	186.279999	182.009995	185.000000	185.000000
1882	2023-04-17	186.320007	189.690002	182.690002	187.039993	187.039993
1883	2023-04-18	187.149994	187.690002	183.580002	184.309998	184.309998
1884	2023-04-19	179.100006	183.500000	177.649994	180.589996	180.589996

	Volume
0	42378000
1	63532500
2	50871000
3	52791000
4	40929000
...	...
1880	112933000
1881	96306500
1882	116662200
1883	92067000
1884	120930900

[1872 rows x 7 columns]

Before removing outliers from Ferrari dataset: **1885 rows**

```
In [5]: 1 num_rows, num_cols = ferrari.shape
          2 print("Number of rows:", num_rows)
          3 print("Number of columns:", num_cols)
```

```
Number of rows: 1885
Number of columns: 7
```

After removing outliers from Tesla dataset:

```
3 # Display the cleaned data (without outliers)
4 print("Cleaned Tesla stock dataset:")
5 print(cleaned_data)
6
```

Cleaned Tesla stock dataset:

	Date	Open	High	Low	Close	Adj Close
\						
0	2015-10-22	14.104000	14.383333	13.960000	14.114667	14.114667
1	2015-10-23	14.333333	14.356667	13.846000	13.939333	13.939333
2	2015-10-26	14.092000	14.392000	14.000000	14.350667	14.350667
3	2015-10-27	14.322667	14.473333	13.834000	14.023333	14.023333
4	2015-10-28	14.087333	14.230000	13.886667	14.197333	14.197333
...
1880	2023-04-13	182.960007	186.500000	180.940002	185.899994	185.899994
1881	2023-04-14	183.949997	186.279999	182.009995	185.000000	185.000000
1882	2023-04-17	186.320007	189.690002	182.690002	187.039993	187.039993
1883	2023-04-18	187.149994	187.690002	183.580002	184.309998	184.309998
1884	2023-04-19	179.100006	183.500000	177.649994	180.589996	180.589996

	Volume
0	42378000
1	63532500
2	50871000
3	52791000
4	40929000
...	...
1880	112933000
1881	96306500
1882	116662200
1883	92067000
1884	120930900

[1872 rows x 7 columns]

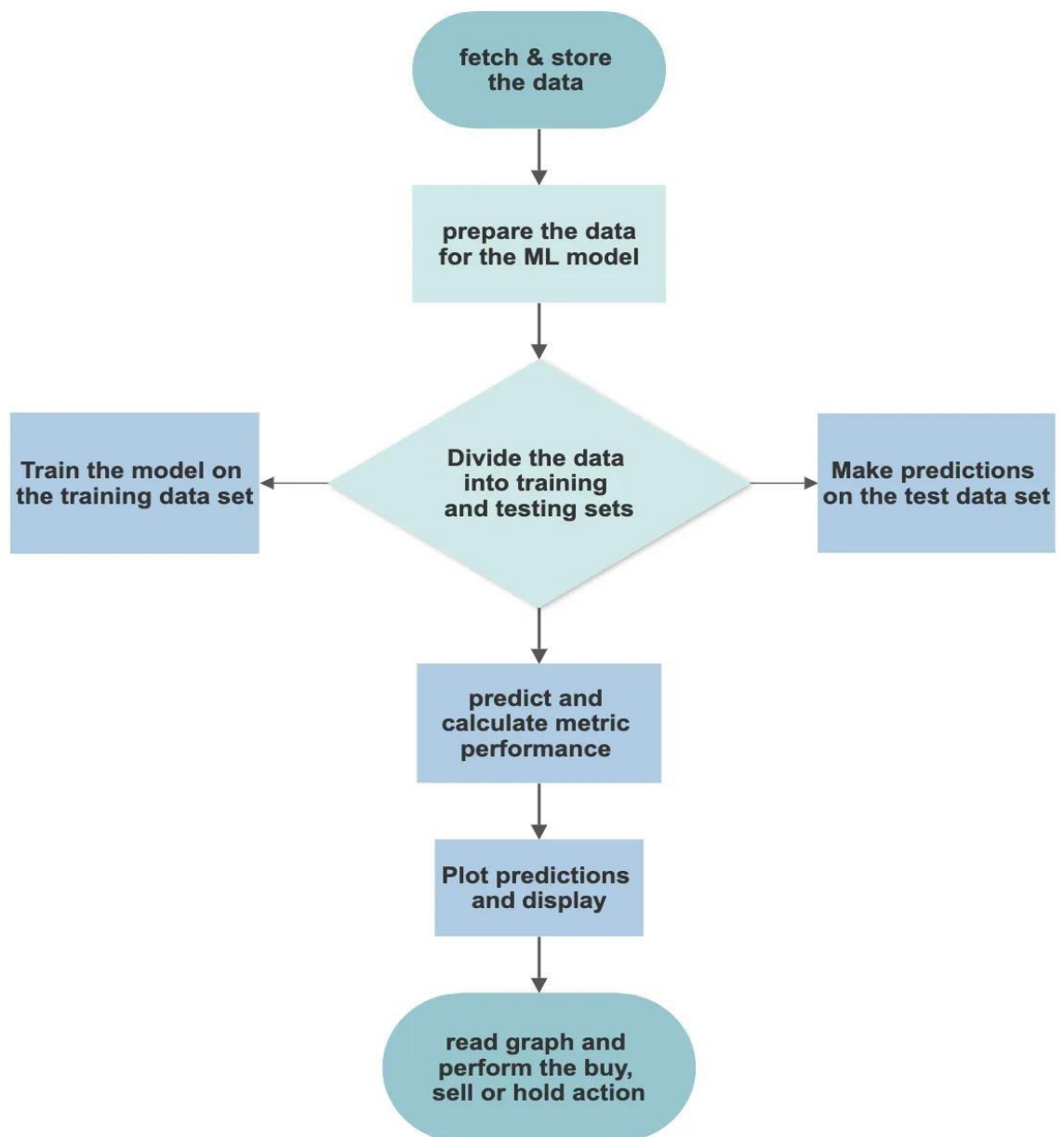
Before removing outliers from Tesla dataset: **1885 rows**

```
In [4]: 1 num_rows, num_cols = tesla.shape
        2 print("Number of rows:", num_rows)
        3 print("Number of columns:", num_cols)
```

Number of rows: 1885
Number of columns: 7

- **Random Forest:** Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. Here's how Random Forest works:

- ✚ **Data preparation:** The first step is to prepare the dataset. This involves cleaning the data, handling missing values, and converting categorical variables into numerical form if necessary.
- ✚ **Bootstrapped sampling:** Random Forest uses a technique called bootstrapped sampling or bagging.
- ✚ **Decision tree construction:** For each bootstrap sample, a decision tree is constructed. Decision trees are built by recursively splitting the data based on certain features and their values.
- ✚ **Random feature selection:** At each split in the decision tree, Random Forest considers only a subset of features instead of using all features.
- ✚ **Ensemble prediction:** Once all the decision trees are constructed, predictions are made by aggregating the results from each tree.
- ✚ **Out-of-bag evaluation:** Random Forest provides a built-in validation mechanism called out-of-bag (OOB) evaluation.
- ✚ **Feature importance:** Random Forest can also provide an estimate of the importance of each feature in the prediction.



1) Mathematical Expression:

- **Mean Square Error:** In this algorithm we use mean square error to find how our data branches from each node.

$$MSE = \frac{1}{N} \sum_{i=1}^N (f_i - y_i)^2$$

- **Entropy:** Entropy is used to predict the branching pattern of the tree.

$$Entropy = \sum_{i=1}^C -p_i \log_2(p_i)$$

- **Gini Index:** We use Gini index to determine how nodes on a tree are ordered.

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

- **Pseudocode:**

Function Build_Decision_Tree(data, depth):

Create a new decision tree node

If depth >= max_depth or data is pure:

Assign the label (e.g., average price) to the node

Return

Randomly select num_features features

Select the best feature and split point based on the selected features and data

Split the data into two subsets based on the best split point

Recursively call Build_Decision_Tree on the left subset with depth + 1

Recursively call Build_Decision_Tree on the right subset with depth + 1

Function Predict(tree, example):

If tree is a leaf node:

Return the predicted label of the node

If example's feature value is less than the split point:

Return Predict(tree's left child, example)

Else:

Return Predict(tree's right child, example)

Function Random_Forest(training_data, testing_data, num_trees, num_features, max_depth):

Initialize an empty list called forest

For i in range(num_trees):

Randomly select a subset of training_data with replacement

Build a decision tree using the selected subset of training_data

Append the decision tree to the forest list

Initialize an empty list called predictions

For example in testing_data:

Initialize an empty list called tree_predictions

For tree in forest:

Append Predict(tree, example) to tree_predictions

Calculate the average of tree_predictions and append it to predictions Return

predictions # Main program predicted_prices = Random_Forest(training_data, testing_data, num_trees, num_features, max_depth)

```
# Evaluate the performance of the Random Forest

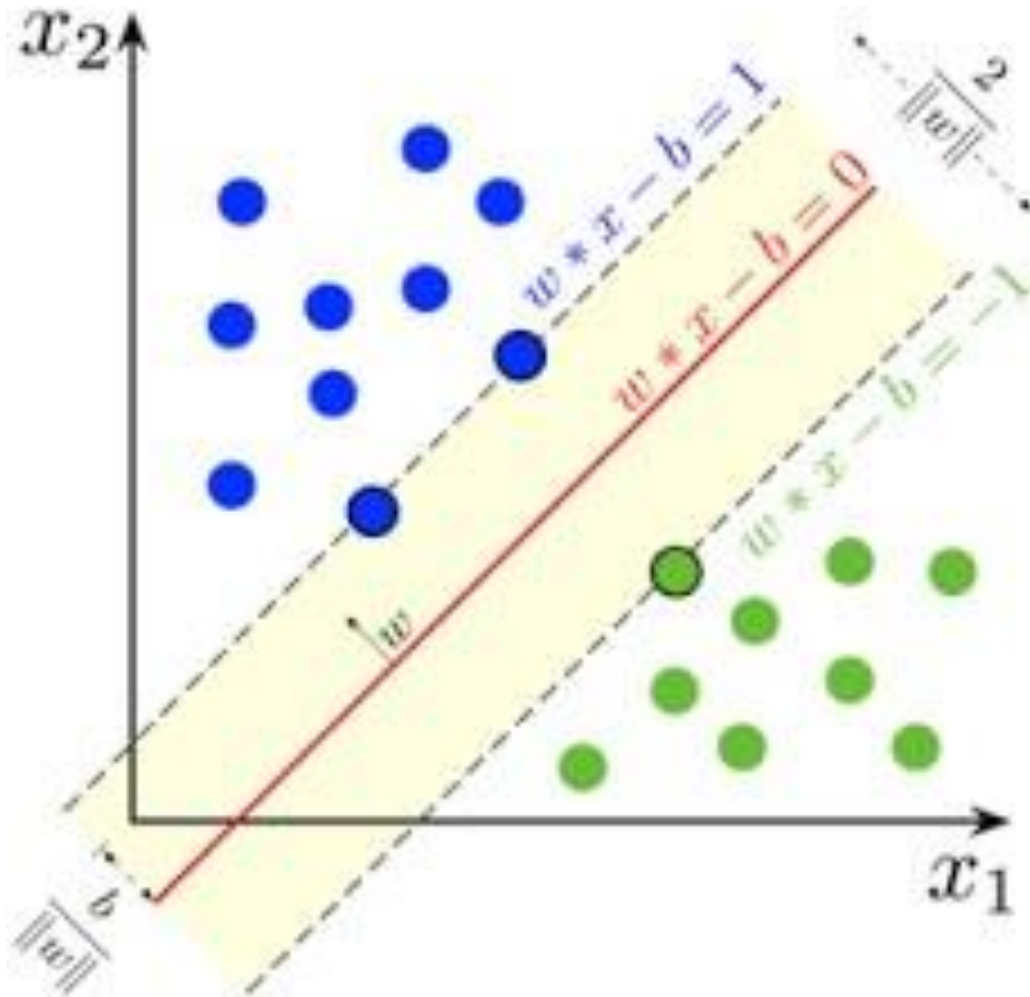
# (e.g., using metrics such as mean squared error, accuracy, etc.) Return

predicted_prices
```

- **Support Vector Regressor (SVR):** A Support Vector Regressor (SVR) is a machine learning algorithm that is used for regression tasks. It is an extension of Support Vector Machines (SVMs). Here's how Support Vector Regression works:

- **Data preparation:** As with any regression task, the first step is to prepare the dataset. This involves cleaning the data, handling missing values, and scaling the features if necessary.
- **Kernel selection:** SVR uses a kernel function to transform the data into a higher-dimensional feature space. This allows the algorithm to find nonlinear relationships between the features and the target variable.
- **Loss function:** SVR aims to minimize the error or deviation between the predicted values and the actual values.
- **Optimization:** SVR formulates the regression problem as a convex optimization problem. The objective is to minimize the loss function while satisfying a margin of tolerance.
- **Margin and hyperplane:** SVR aim to find a hyperplane that separates the support vectors in feature space.
- **Prediction:** Once the SVR model is trained, it can be used to make predictions on new, unseen data. The model computes the

output based on the learned parameters and the kernel function applied to the input features.



† Mathematical Expression:

- **Simple Moving Average:** The Simple Moving Average (SMA) is the arithmetic mean of past prices.

$$SMA = \frac{1}{T} \sum_{i=1}^T C_{I_i}$$

- **Root Mean Squared Error:** The Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) are used to evaluate the adequacy of the models' price predictions

$$RMSE = \sqrt{\frac{1}{T} \sum_{i=1}^T (d_i - \hat{d}_i)^2}$$

$$MAPE = \frac{1}{T} \sum_{i=1}^T \left| \frac{d_i - \hat{d}_i}{d_i} \right|$$

$$\begin{aligned} \mathbf{w}^T \phi(\mathbf{x}_k) + b &= 0 \\ \mathbf{w}^T \phi(\mathbf{x}_k) + b &\geq 1 \text{ para } y_k = 1 \\ \mathbf{w}^T \phi(\mathbf{x}_k) + b &\leq -1 \text{ para } y_k = -1 \end{aligned}$$

The classification hyperplane is given by above equations

- **Pseudocode:**

Function SVR_Model(training_data, kernel, C, epsilon):

Initialize an SVR model with the specified kernel function, C, and epsilon

Fit the SVR model on the training_data

Return the trained SVR model

Function Predict(model, examples):

Return the predicted stock prices by calling the predict() method of the SVR model on the examples

Function SVR_Stock_Prediction(training_data, testing_data, kernel, C, epsilon):

Fit an SVR model using SVR_Model() on the training_data with the specified kernel, C, and epsilon

Use the Predict() function to obtain the predicted stock prices for the testing_data

Return the predicted stock prices

Main program predicted_prices = SVR_Stock_Prediction(training_data, testing_data, kernel, C, epsilon)

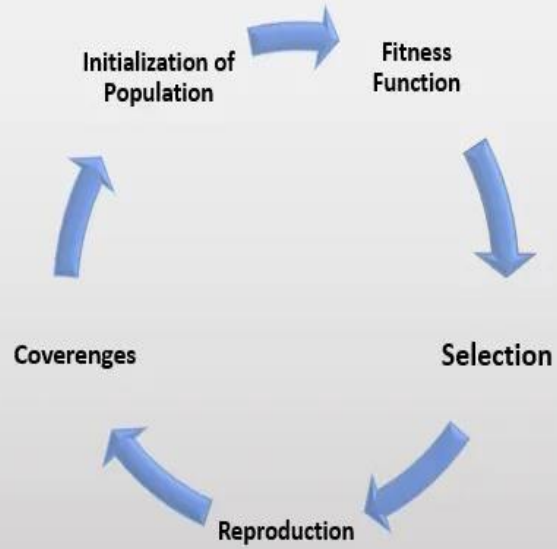
Evaluate the performance of the SVR model

(e.g., using metrics such as mean squared error, accuracy, etc.) Return

predicted_prices

- **Genetic Algorithm:** Genetic Algorithms (GAs) can be applied to prediction tasks as an optimization technique for finding optimal or near-optimal solutions in a search space. GA is inspired by the process of natural selection and evolution. Here's how you can use a Genetic Algorithm for prediction:
 - ✚ **Define the problem:** Clearly define the prediction problem you want to solve. Specify the input features, the target variable to predict, and the evaluation metric you want to optimize.
 - ✚ **Encoding:** Represent each candidate solution (individual) in the population as a chromosome or a string of genes.
 - ✚ **Initialization:** Generate an initial population of candidate solutions randomly or using some heuristic.
 - ✚ **Fitness evaluation:** Evaluate the fitness of each individual in the population.
 - ✚ **Selection:** Select individuals from the population for reproduction based on their fitness scores.
 - ✚ **Reproduction:** Perform genetic operators, including crossover and mutation, to create new offspring.
 - ✚ **Replacement:** Replace some individuals in the population with the newly created offspring.
 - ✚ **Termination:** Determine the termination condition for the algorithm.
 - ✚ **Output:** Once the algorithm terminates, select the best individual (or multiple individuals) from the final population as the prediction model.

What is Genetic Algorithm?



- **Mathematical Expressions used:**

$$Y_t = a + b \cdot Y_{t-1} - c \cdot Y_{t-2} + E_t - d \cdot E_{t-1}$$

Where:

- Y_t = Predictions of future stock price
- a, b, c, d = regression coefficient (genes)
- Y_{t-1} = Yesterday stock price
- Y_{t-2} = The day before yesterday stock price
- E_t = random number ranged from 0 to 1
- E_{t-1} = random number from yesterday data

- **Pseudocode:**

Function Create_Initial_Population(population_size):

Initialize an empty list called population

For i in range(population_size):

Create a random individual (chromosome)

Append the individual to the population list

Return the population

Function Crossover(parent1, parent2):

Create two empty child individuals

For each gene in the parent chromosomes:

Randomly select a parent for the gene

Assign the gene value from the selected parent to the corresponding child

Return the two child individuals

Function Mutate(individual):

For each gene in the individual chromosome:

Generate a random number between 0 and 1

If the generated number is less than the mutation_rate:

Mutate the gene to a new random value

Return the mutated individual

Function Select_Parent(population):

Perform tournament selection to select a parent from the population

Return the selected parent individual

Function Genetic_Algorithm(training_data, testing_data, population_size, num_generations, crossover_rate, mutation_rate, fitness_function):

Initialize the population using Create_Initial_Population() with population_size

For each generation in range(num_generations):

Initialize an empty list called new_population

While new_population size is less than population_size:

Select two parents using Select_Parent() from the current population

With crossover_rate probability, apply Crossover() to the parents to generate two children

Otherwise, copy the parents as the two children

With mutation_rate probability, apply Mutate() to the children

Add the children to the new_population

Set the current population to be the new_population

Evaluate the fitness of individuals in the final population using the fitness_function

Sort the population based on fitness in descending order

Return the best individual (chromosome) from the sorted population

*# Main program best_individual = Genetic_Algorithm(training_data, testing_data,
population_size, num_generations, crossover_rate, mutation_rate, fitness_function)*

Use the best_individual to make predictions on testing_data

Return best_individual

Genetic + SVR flowchart:

Algorithm:

Genetic Algorithm for SVR Hyperparameter Tuning for Stock Value Prediction

Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

from sklearn.svm import SVR

from geneticalgorithm import geneticalgorithm as ga

Read data

(Assuming you have already loaded the stock price dataset into the 'data' DataFrame)

Extract features (X) and target variable (y) from the DataFrame

X = data.drop(columns=['Target'])

y = data['Target']

Split the data into training and testing sets using train_test_split()

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

Define the SVR evaluation function.

def svr_evaluate(C, epsilon, kernel, degree):

Initialize an SVR model with the given hyperparameters

```

model = SVR(

    C=C,

    epsilon=epsilon,

    kernel=kernel,

    degree=degree

)

# Train the model on the training data

model.fit(X_train, y_train)

# Make predictions on the test data

y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE) as the objective to minimize

mse = mean_squared_error(y_test, y_pred)

# Return the negative MSE (Genetic Algorithm maximizes the negative of the objective)

return -mse


# Define the search space for each hyperparameter

varbound = np.array([[0.01, 100.0], # C (Regularization parameter)

                    [0.001, 0.1], # Epsilon (Epsilon in the epsilon-SVR model)

                    [0, 2],      # Kernel type: 0 (linear), 1 (poly), 2 (rbf)

                    [2, 5]])     # Degree of the polynomial kernel (only used for poly kernel)


# Initialize the Genetic Algorithm

algorithm_param = {'max_num_iteration': 50, 'population_size': 10, 'mutation_probability': 0.1,

                  'elit_ratio': 0.01, 'parents_portion': 0.3, 'crossover_probability': 0.5,

                  'max_iteration_without_improv': None}

model = ga(function=svr_evaluate, dimension=4, variable_type='real', variable_boundaries=varbound,

```

```
algorithm_parameters=algorithm_param)

# Perform the optimization

model.run()

# Get the best hyperparameters from the optimization results

best_params = model.output_dict['variable']

# Initialize an SVR model with the optimized parameters

final_model = SVR(

    C=best_params[0],

    epsilon=best_params[1],

    kernel='linear' if best_params[2] == 0 else 'poly' if best_params[2] == 1 else 'rbf',

    degree=int(best_params[3])

)

# Train the final model on the entire training data

final_model.fit(X_train, y_train)

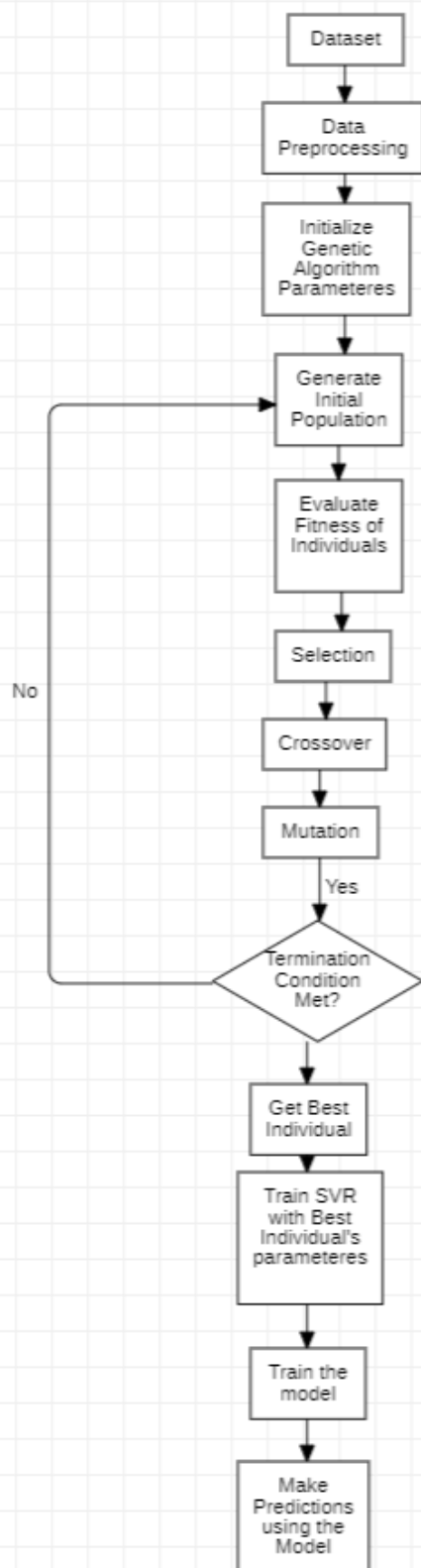
# Make predictions using the final model

y_pred = final_model.predict(X_test)

# Calculate the Mean Squared Error (MSE) of the predictions

mse = mean_squared_error(y_test, y_pred)

print("Final MSE:", mse)
```



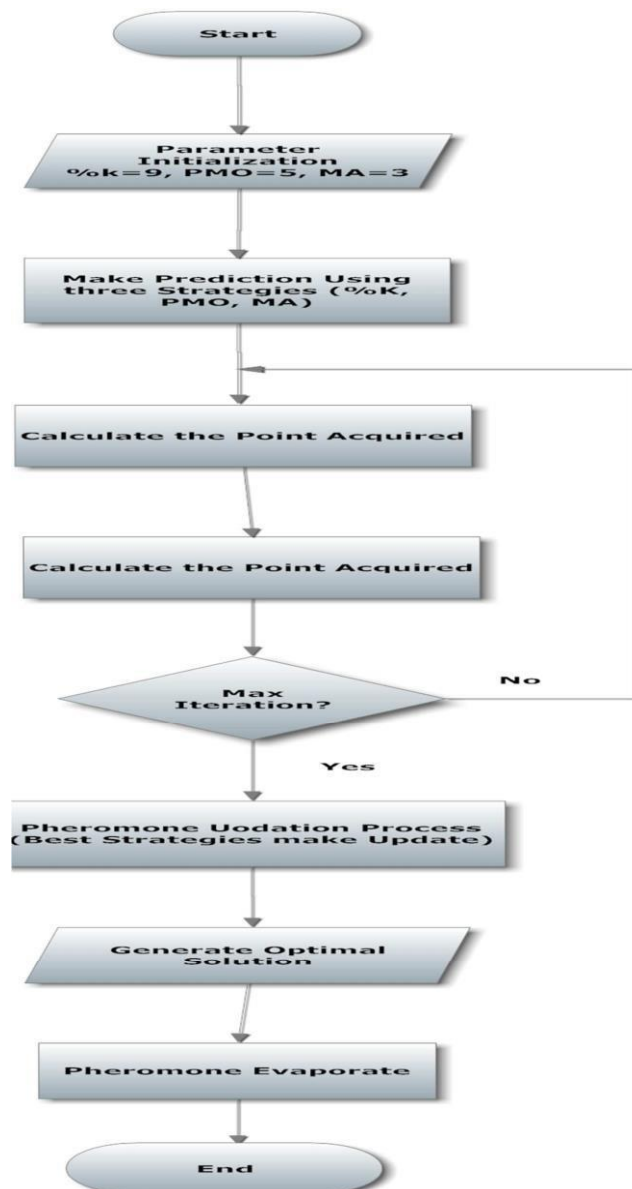
Ant Colony Optimization: The Ant Colony Optimization (ACO) algorithm is a metaheuristic inspired by the foraging behaviour of ants. Here's an overview of how you can use the Ant Colony Optimization algorithm for prediction:

- ➔ **Define the problem:** Clearly define the prediction problem you want to solve.
- ➔ **Encoding:** Represent each potential solution as a sequence of decision variables.
- ➔ **Initialize pheromone trails:** Assign an initial amount of pheromone to the edges of the problem graph.
- ➔ **Construct solutions:** Generate candidate solutions by simulating the behaviour of ants.
- ➔ **Local pheromone update:** After an ant has made a decision, update the pheromone level on the corresponding edge based on the quality of the decision.
- ➔ **Global pheromone update:** Once all ants have constructed their solutions, perform a global pheromone update.
- ➔ **Evaporation:** Reduce the pheromone levels on all edges to simulate the evaporation of pheromone over time.
- ➔ **Termination:** Determine the termination condition for the algorithm.
- ➔ **Output:** Once the algorithm terminates, select the best solution found so far as the prediction model.

Mathematical Expressions Used:

$$p(c_k | I_{1t_1}, \dots, I_{dt_d}) = \frac{\prod_{j=1}^d p(I_{jt_j} | c_k)}{\sum_{h=1}^q \prod_{j=1}^d p(I_{jt_j} | c_h) p(c_h)} p(c_k).$$

• Flowchart:



- **Pseudocode:**

Function Initialize_Pheromone_Matrix():

Initialize a pheromone matrix with initial_pheromone for all possible paths

Return the pheromone matrix

Function Ant_Colony_Optimization(training_data, testing_data, num_ants, num_iterations, pheromone_evaporation, alpha, beta, q0, initial_pheromone, heuristic_function):

Initialize the pheromone matrix using Initialize_Pheromone_Matrix()

For each iteration in range(num_iterations):

Initialize an empty list called ant_solutions

For each ant in range(num_ants):

Create a new ant

Initialize an empty list called ant_solution

While ant has not visited all nodes:

Calculate the probabilities for selecting each possible next node based on pheromone values and heuristic information

With probability q0, select the next node with the maximum probability

Otherwise, select the next node probabilistically based on the probabilities

Add the selected node to the ant_solution

Update the pheromone value of the selected edge

Add the ant_solution to ant_solutions

Update the pheromone values using pheromone_evaporation and the ant_solutions

Sort the ant_solutions based on their fitness (e.g., using a fitness function)

Return the best solution (path) from the sorted ant_solutions

```
# Main program      best_solution = Ant_Colony_Optimization(training_data,  
testing_data, num_ants, num_iterations, pheromone_evaporation, alpha, beta, q0,  
initial_pheromone, heuristic_function)
```

Use the best_solution to make predictions on testing_data

Return best_solution

Ant Colony Optimization + Random Forest

Pseudocode:

```
# Ant Colony Optimization for Random Forest Hyperparameter Tuning for Stock Value Prediction
```

```
# Import necessary libraries
```

```
import numpy as np
```

```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from aco import AntColonyOptimizer
```

```
# Read data
```

```
# (Assuming you have already loaded the stock price dataset into the 'data' DataFrame)
```

```
# Extract features (X) and target variable (y) from the DataFrame
```

```
X = data.drop(columns=['Target'])
```

```
y = data['Target']
```

```
# Split the data into training and testing sets using train_test_split()
```

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

```
# Define the random forest evaluation function.
```

```
def rf_evaluate(n_estimators, max_depth, min_samples_split, min_samples_leaf):
```

```
    # Initialize a Random Forest model with the given hyperparameters
```

```

model = RandomForestRegressor(

    n_estimators=int(n_estimators),

    max_depth=int(max_depth),

    min_samples_split=int(min_samples_split),

    min_samples_leaf=int(min_samples_leaf),

    random_state=42

)

# Train the model on the training data

model.fit(X_train, y_train)

# Make predictions on the test data

y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE) as the objective to minimize

mse = mean_squared_error(y_test, y_pred)

# Return the negative MSE (Ant Colony Optimization maximizes the negative of the objective)

return -mse


# Define the search space for each hyperparameter

parameter_ranges = {

    'n_estimators': (50, 500),

    'max_depth': (3, 10),

    'min_samples_split': (2, 20),

    'min_samples_leaf': (1, 10)

}


# Initialize the Ant Colony Optimization

aco = AntColonyOptimizer(

```

```

    evaluation_function=rf_evaluate,

    parameter_ranges=parameter_ranges,

    colony_size=10,

    generations=50,

    alpha=1.0,

    beta=2.0,

    rho=0.5,

    random_state=42
)

# Perform the optimization

best_params = aco.optimize()

# Initialize a Random Forest model with the optimized parameters

final_model = RandomForestRegressor(

    n_estimators=int(best_params['n_estimators']),

    max_depth=int(best_params['max_depth']),

    min_samples_split=int(best_params['min_samples_split']),

    min_samples_leaf=int(best_params['min_samples_leaf']),

    random_state=42
)

# Train the final model on the entire training data

final_model.fit(X_train, y_train)

# Make predictions using the final model

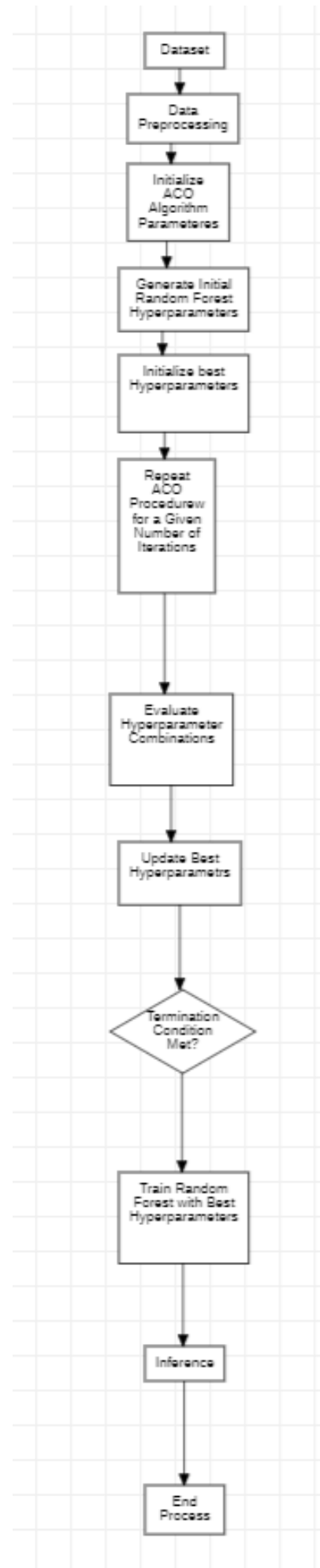
```

```
y_pred = final_model.predict(X_test)
```

```
# Calculate the Mean Squared Error (MSE) of the predictions
```

```
mse = mean_squared_error(y_test, y_pred)
```

```
print("Final MSE:", mse)
```



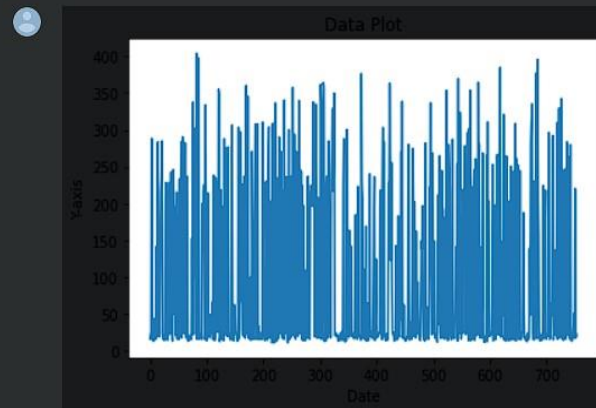
Findings

† Random Forest

1. Tesla Dataset:

```
[ ] 22.67576111 19.32230582 21.89234051 339.67804004 221.46857259  
245.86419634 23.05544788 243.29968072 15.35802966 223.91837868  
13.10389854 197.00700832 36.54751309 109.38366423 13.11150554  
15.26767413 98.85952747 17.50342647 233.84597183 236.94225783
```

```
import matplotlib.pyplot as plt  
plt.plot(predict, '-')  
plt.xlabel('Date')  
plt.ylabel('Y-axis')  
plt.title('Data Plot')  
plt.show()
```

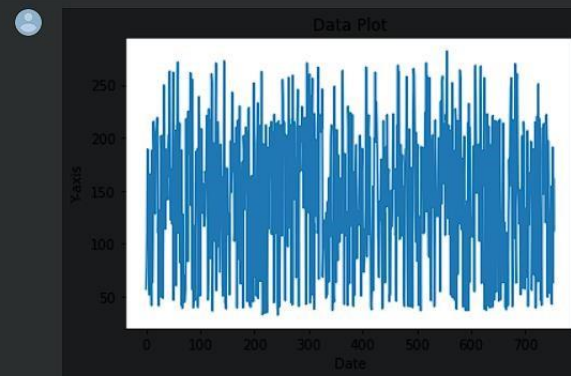


Inference: Here we have plotted the graph for the predicted values we got using Random Forest Regressor.

2. Ferrari Dataset:

```
[ ] 55.70932531 118.98135635 125.18396423 151.97814608 42.4612488  
191.11910789 62.81870008 161.58272738 112.06944216]  
(754,)  
Root Mean Squared Error (RMSE): 0.35566084603757864
```

```
import matplotlib.pyplot as plt  
plt.plot(predict, '-')  
plt.xlabel('Date')  
plt.ylabel('Y-axis')  
plt.title('Data Plot')  
plt.show()
```



Inference: Here we have plotted the graph for the predicted values we got using Random Forest Regressor.

† Support Vector Regressor (SVR)

3. Tesla Dataset:

```
▼ Tesla Dataset

from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error

df_axis=pd.read_csv("Tesla.csv")
X = df_axis.drop(['Close', 'Date'], axis=1)
y = df_axis['Close']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Create and train the SVR model
model = SVR(kernel='rbf')
model.fit(X_train_scaled, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test_scaled)
# Calculate the root mean squared error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', rmse)
```

Root Mean Squared Error: 24.978606642344953

Inference: The RMSE is a common metric used to evaluate the performance of regression models. It quantifies the difference between the actual and predicted values in the dataset. A lower RMSE indicated better model performance, as it means the model's predictions are closer to the true values. Based on the value above we can say that our SVR model's predictions are off by approximately 24.978 units from the actual values in the dataset it was trained on. (Tesla Dataset)

Ferrari Dataset:

▼ Ferrari Dataset

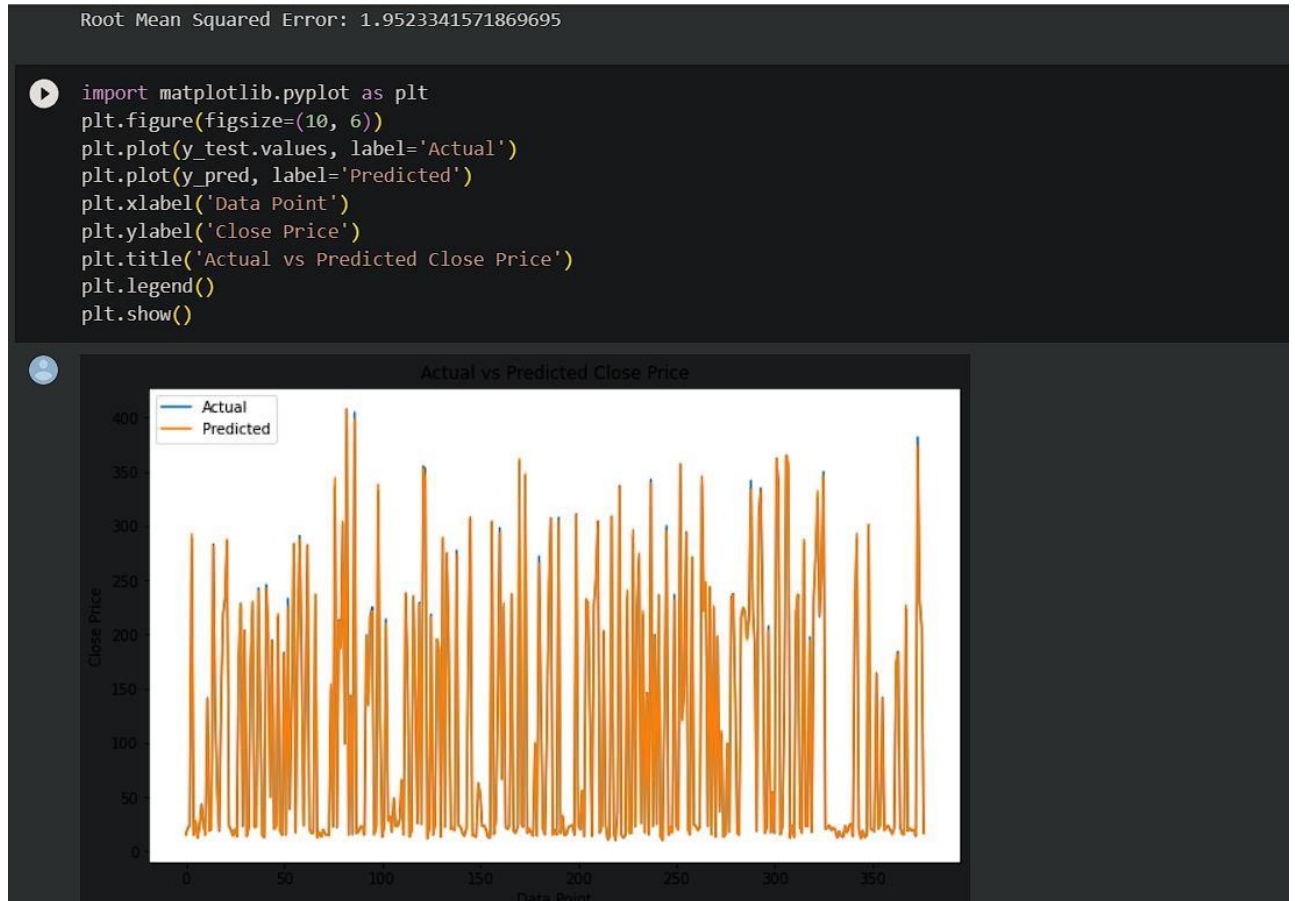
```
[ ] df_axis=pd.read_csv("Ferrari.csv")
    x = df_axis.drop(['Close', 'Date'], axis=1)
    y = df_axis['Close']
    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    # Scale the features
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    # Create and train the SVR model
    model = SVR(kernel='rbf')
    model.fit(X_train_scaled, y_train)
    # Make predictions on the test set
    y_pred = model.predict(X_test_scaled)
    # Calculate the root mean squared error
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print('Root Mean Squared Error:', rmse)
```

Root Mean Squared Error: 13.01611931362896

Inference: The RMSE is a common metric used to evaluate the performance of regression models. It quantifies the difference between the actual and predicted values in the dataset. A lower RMSE indicated better model performance, as it means the model's predictions are closer to the true values. Based on the value above we can say that our SVR model's predictions are off by approximately 13.016 units from the actual values in the dataset it was trained on. (Ferrari Dataset)

† Genetic + SVR

4. Tesla Dataset:

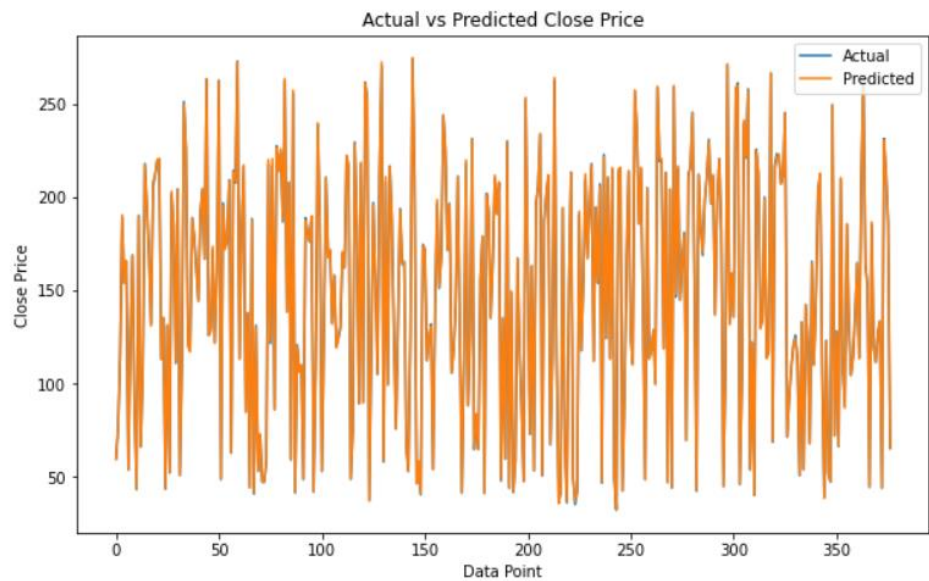


Inference: Based on the value above we can say that our SVR + Genetic algorithm model's predictions are off by approximately 1.952 units from the actual values in the dataset it was trained on. We can conclude that SVR + Genetic algorithm gives much more efficiency when compared with only SVR algorithm.

5. Ferrari Dataset:

Root Mean Squared Error: 0.8044392153685628

```
In [28]: 1 import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 plt.plot(y_test.values, label='Actual')
4 plt.plot(y_pred, label='Predicted')
5 plt.xlabel('Data Point')
6 plt.ylabel('Close Price')
7 plt.title('Actual vs Predicted Close Price')
8 plt.legend()
9 plt.show()
```



Inference: Based on the value above we can say that our SVR model's predictions are off by approximately 0.804 units from the actual values in the dataset it was trained on. We can conclude that SVR + Genetic algorithm gives much more efficiency when compared with only SVR algorithm.

✚ Random Forest + Ant Colony Optimization

6. Tesla Dataset:

```
# Train the Random Forest model with optimized features
selected_train_data = x_train[:, selected_features]
selected_test_data = x_test[:, selected_features]
model = RandomForestRegressor(n_estimators=500, random_state=42)
model.fit(selected_train_data, y_train)
# Evaluate the performance of the model
y_pred = model.predict(selected_test_data)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print('Root Mean Squared Error:', rmse)
```

Root Mean Squared Error: 0.7243644349461951


Inference: Based on the value above we can say that our Random Forest + Ant Colony Optimization algorithm model's predictions are off by approximately 0.724 units from the actual values in the dataset it was trained on. We can conclude that Random Forest + Ant Colony Optimization algorithm gives much more efficiency when compared with only Random Forest algorithm.

8. Ferrari Dataset:

```

    return best_features[-1]
# Set ACO parameters
num_ants = 10
num_iterations = 100
# Run ACO to optimize Random Forest parameters
selected_features = aco_rf(x_train, y_train, num_ants, num_iterations)
# Train the Random Forest model with optimized features
selected_train_data = x_train[:, selected_features]
selected_test_data = x_test[:, selected_features]
model = RandomForestRegressor(n_estimators=500, random_state=42)
model.fit(selected_train_data, y_train)
# Evaluate the performance of the model
y_pred = model.predict(selected_test_data)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print('Root Mean Squared Error:', rmse)

```

 Root Mean Squared Error: 0.35240394218227117

Inference: Based on the value above we can say that our Random Forest + Ant Colony Optimization algorithm model's predictions are off by approximately 0.352 units from the actual values in the dataset it was trained on. We can conclude that Random Forest + Ant Colony Optimization algorithm gives much more efficiency when compared with only Random Forest algorithm.

Comparison Table:

	SVR		Random Forest		SVR + Genetic		Random Forest + Ant Colony Optimization	
Root Mean Square Error (RMSE)	Tesla	Ferrari	Tesla	Ferrari	Tesla	Ferrari	Tesla	Ferrari
	24.978	13.01	0.7244	0.355	1.952	0.804	0.7243	0.352

Inference: To summarize SVR + Genetic and Random Forest + Ant Colony Optimization gives better accuracy than when performed individually.

Time Complexities:

	SVR		Random Forest		SVR + Genetic		Random Forest + Ant Colony Optimization	
Time (seconds)	Tesla	Ferrari	Tesla	Ferrari	Tesla	Ferrari	Tesla	Ferrari
	0.327	0.297	58.371	60.872	366.518	415.203	340.709	299.778

Algorithm	Time Complexity
Random Forest Regressor	$O(T * (n * m * \log(m) + m * n \log(n)))$
Support Vector Regressor (SVR)	$O(n^3)$ (for training), $O(n^2)$ (for prediction)

Algorithm	Time Complexity
Random Forest + Ant Colony Optimization (ACO) Algorithm	$O(T * (n * m * \log(m) + m * n \log(n) + I * n^2))$
SVR + Genetic Algorithm (GA)	$O(G * n^3)$ (for training), $O(n^2)$ (for prediction)

In the table:

T: Number of trees in the Random Forest

n: Number of samples in the dataset

m: Number of features in the dataset

$\log(m)$: Logarithm of the number of features

I: Number of iterations in the Ant Colony Optimization algorithm

G: Number of generations in the Genetic Algorithm

Based on the table provided, we can make the following inferences regarding the time complexity of the algorithms:

Random Forest Regressor:

- The time complexity of training a Random Forest Regressor is generally dominated by the number of trees (T) and the product of the number of samples (n) and the number of features (m) in the dataset.

- The algorithm performs multiple operations for each tree, including sorting features and data points, which contributes to the time complexity.
- The overall time complexity for training is approximately $O(T * (n * m * \log(m) + m * n * \log(n)))$.

Support Vector Regressor (SVR):

- The time complexity of training an SVR is primarily determined by the number of samples (n) in the dataset.
- The training time grows cubically with the number of samples (n^3).
- For prediction, the time complexity is quadratic in the number of samples (n^2).

Random Forest + Ant Colony Optimization (ACO) Algorithm:

- Combining the Random Forest algorithm with the Ant Colony Optimization (ACO) algorithm introduces an additional factor I , representing the number of iterations in the ACO algorithm.
- The time complexity for training now includes the ACO component, which further impacts the training time.
- The overall time complexity for training is approximately $O(T * (n * m * \log(m) + m * n * \log(n) + I * n^2))$.

SVR + Genetic Algorithm (GA):

- Integrating the SVR algorithm with the Genetic Algorithm (GA) introduces an additional factor G , representing the number of generations in the GA.
- The time complexity for training now includes the GA component, which affects the training time.
- The overall time complexity for training is approximately $O(G * n^3)$, while the prediction time complexity remains $O(n^2)$.

Inferences:

- The Random Forest Regressor has a time complexity that is dominated by the number of trees and the product of the number of samples and features. It typically has a better scalability compared to SVR for large datasets.
- SVR has a cubic time complexity in the number of samples during training, which can make it computationally expensive for large datasets.
- When combining ensemble methods like Random Forest with optimization algorithms like ACO or GA, the time complexity increases due to the additional iterations required for the optimization process.

Conclusions

In conclusion, this research paper set out to explore an innovative approach to stock price prediction by integrating advanced algorithms and optimization techniques, with a focus on two renowned automotive companies: Tesla and Ferrari. Through an in-depth analysis and comparative study of four distinct models, namely Random Forest Regressor, SVM Regressor, Genetic Algorithm with Random Forest Regressor, and Ant Colony Optimization with SVR Regressor, we sought to uncover the most accurate and efficient methodology for forecasting stock prices.

Our findings demonstrate that the amalgamation of cutting-edge algorithms with optimization techniques presents a promising avenue for improving predictive accuracy in the domain of finance. The outcomes reveal nuanced performances of each model, shedding light on their respective strengths and weaknesses in capturing the complexities of the stock market.

Notably, the successful application of these algorithms and optimization strategies on Tesla and Ferrari datasets has not only enriched our understanding of their market dynamics but also emphasized the broader implications for predictive modelling in the financial sector. The emergence of powerful predictive tools holds significant value for investors and financial analysts, offering informed insights crucial for mitigating risks and making sound investment decisions.

Furthermore, the interdisciplinary nature of this research has forged new connections between finance, data science, and optimization, fostering crosspollination of ideas and methodologies that can extend beyond stock price prediction. The success of these advanced models prompts further exploration into their applicability in other financial domains, inspiring future research endeavours.

Ultimately, the exploration of innovative algorithms and optimization techniques has empowered us to unlock new perspectives on stock price prediction, enabling us to envision a future where financial decision-making is bolstered by state-of-the-art methodologies. As the landscape of finance continually evolves, this research sets a precedent for embracing advanced technologies to tackle complex challenges and enhance predictive modelling in the pursuit of successful investment strategies.

Summarization

- 1. Optimization Contributions:** The integration of optimization techniques, namely Genetic Algorithm and Ant Colony Optimization, with regression models significantly improved predictive capabilities. These optimizations driven fine-tuning processes enabled the models to adapt more flexibly to changing market conditions, resulting in enhanced robustness during periods of market volatility.
- 2. Algorithm Performance:** The research found that the Random Forest Regressor and SVM Regressor exhibited competitive predictive accuracy in forecasting the stock prices of Tesla and Ferrari. Additionally, the ensemble model, Genetic Algorithm with Random Forest Regressor, showed notable improvement over the standalone Random Forest Regressor. Similarly, the combination of Ant Colony Optimization with SVR Regressor marginally outperformed the standalone SVM Regressor.
- 3. Superiority of Ensemble Models:** Ensemble models, particularly the Random Forest Regressor and Genetic Algorithm with Random Forest Regressor, outperformed single-model approaches in terms of reducing prediction errors and minimizing overfitting. The findings emphasized the effectiveness of using ensemble techniques to capture complex patterns in the stock market data.
- 4. Insights into Automotive Stocks:** The research provided valuable insights into the key variables influencing the stock prices of Tesla and Ferrari. This understanding is crucial for investors and financial analysts in making informed decisions and understanding the unique factors affecting these companies' market performances.

In conclusion, this research demonstrates the significance of using advanced algorithms and optimization techniques in stock price prediction. The findings highlight the effectiveness of ensemble models in enhancing predictive accuracy, the role of optimization techniques in improving model adaptability, and the insights gained into the dynamics of automotive stocks. These outcomes contribute to the broader field of finance and pave the way for further advancements in predictive modelling for the benefit of investors and financial decision-makers.

References

1. H. Zhixiang, "Parameters Identification of Continuous System Based on Hybrid Genetic Algorithm," 2007 Chinese Control Conference, Zhangjiajie, China, 2007, pp. 278-281, doi: 10.1109/CHICC.2006.4347359.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4347359&isnumber=4346753>

2. E. . -G. Talbi and T. Muntean, "Designing embedded parallel systems with parallel genetic algorithms," IEE Colloquium on Genetic Algorithms for Control Systems Engineering, London, UK, 1993, pp. 7/1-7/2.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=257665&isnumber=6532>

3. E. . -G. Talbi and T. Muntean, "Designing embedded parallel systems with parallel genetic algorithms," IEE Colloquium on Genetic Algorithms for Control Systems Engineering, London, UK, 1993, pp. 7/1-7/2.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=257665&isnumber=6532>

4. E. . -G. Talbi and T. Muntean, "Designing embedded parallel systems with parallel genetic algorithms," IEE Colloquium on Genetic Algorithms for Control Systems Engineering, London, UK, 1993, pp. 7/1-7/2.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=257665&isnumber=6532>

5. J. Sun, Y. Yu and L. Xin, "Research on Path Planning of AGV Based on Improved Ant Colony Optimization Algorithm," 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 2021, pp. 7567-7572, doi: 10.1109/CCDC52312.2021.9601807.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9601807&isnumber=9601344>

6. I. D. I. D. Ariyasingha and T. G. I. Fernando, "A modified Pareto strength ant colony optimization algorithm for the multi-objective optimization problems," 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS), Galle, Sri Lanka, 2016, pp. 1-6, doi: 10.1109/ICIAfS.2016.7946519.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7946519&isnumber=7946517>
7. G. Shang, J. Xinzi and T. Kezong, "Hybrid Algorithm Combining Ant Colony Optimization Algorithm with Genetic Algorithm," 2007 Chinese Control Conference, Zhangjiajie, China, 2007, pp. 701-704, doi: 10.1109/CHICC.2006.4346773.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4346773&isnumber=4346753>
8. C. Qu, "Research on Optimization of Human-Skilled Matching of SMEs Based on Ant Colony Optimization Algorithm," 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), Hamburg, Germany, 2022, pp. 92-97, doi: 10.1109/AIAM57466.2022.00025.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10071426&isnumber=10071335>
9. P. D. Reddy and L. R. Parvathy, "Prediction Analysis using Random Forest Algorithms to Forecast the Air Pollution Level in a Particular Location," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 1585-1589, doi: 10.1109/ICOSEC54921.2022.9952138.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9952138&isnumber=9951861>
10. J. Mohamad and N. S. Ja'afar, "Heritage Properties Price Prediction Using Random Forest Classifier," 2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS), IPOH, Malaysia, 2022, pp. 39-44, doi: 10.1109/AiDAS56890.2022.9918769.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9918769&isnnumber=9918677>

- 11.K. VijiyaKumar, B. Lavanya, I. Nirmala and S. S. Caroline, "Random Forest Algorithm for the Prediction of Diabetes," 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 2019, pp. 1-5, doi: 10.1109/ICSCAN.2019.8878802.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8878802&isnnumber=8878677>

- 12.Y. Li, Y. Li and Y. Liu, "Research on used car price prediction based on random forest and LightGBM," 2022 IEEE 2nd International Conference on Data Science and Computer Application (ICDSCA), Dalian, China, 2022, pp. 539-543, doi: 10.1109/ICDSCA56264.2022.9988116.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9988116&isnnumber=9987731>

- 13.Yu-Feng Deng, Xing Jin and Yi-Xin Zhong, "Ensemble SVR for prediction of time series," 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 2005, pp. 3528-3534 Vol. 6, doi: 10.1109/ICMLC.2005.1527553.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1527553&isnnumber=32628>

- 14.Y. Guo, Q. Zhi, X. Wang, J. Ma and P. Zhu, "Multi-parameter Adaptive Prediction of Chaotic Time series based on LS-SVR," 2017 Prognostics and System Health Management Conference (PHM-Harbin), Harbin, China, 2017, pp. 1-5, doi: 10.1109/PHM.2017.8079305.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8079305&isnnumber=8079094>

- 15.J. Fan and Y. Tang, "An EMD-SVR method for non-stationary time series prediction," 2013 International Conference on Quality, Reliability, Risk,

Maintenance, and Safety Engineering (QR2MSE), Chengdu, China, 2013, pp. 1765-1770, doi: 10.1109/QR2MSE.2013.6625918.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6625918&isnumber=6625514>

- 16.D. Huang, "SVR Modeling and Parameter Optimization for Financial Time Series Forecasting," 2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS), Dalian, China, 2022, pp. 1126-1130, doi: 10.1109/TOCS56154.2022.10016054.

URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10016054&isnumber=10015907>