Multiple Regression Based Dynamic Parking Fare System

Aryan Sharma
Department of Computer Engineering
D Y Patil College of Engineering, Akurdi
Pune, India
aryan152015@gmail.com

Sakshi Thombre
Department of Computer Engineering
D Y Patil College of Engineering, Akurdi
Pune, India
isakshy18@gmail.com

Anubhav Prabhakar
Department of Computer Engineering
D Y Patil College of Engineering, Akurdi
Pune, India
anubhav.prabhakar01@gmail.com

Sweta Singh
Department of Computer Engineering
D Y Patil College of Engineering, Akurdi
Pune, India 72002sweta@gmail.com

Mrs. Soudamini Somvanshi
Internal Guide
Department of Computer Engineering
D Y Patil College of Engineering, Akurdi
Pune, India

Mrs. Mukta Patil
Internal Guide
Department of Computer Engineering
D Y Patil College of Engineering, Akurdi
Pune, India

***Abstract*— This paper explores the development of a web application for dynamic parking pricing, leveraging ReactJS, ExpressJS, MySQL, Google Maps API, and machine learning techniques. The system addresses the challenge of static parking fees by incorporating real-time data and a trained model to predict optimal parking prices. The core functionality hinges on a multiple linear regression model. This model is trained on historical parking data, considering factors such as parking space availability, time of day, base fare, air pollution levels, and user ratings. By analyzing these variables, the model predicts an accurate price for each parking space. The web application utilizes ReactJS to provide a user-centric interface. Users can search for parking spaces and view dynamic pricing in real time. Integration with Google Maps API offers a visual representation of available parking locations along with their corresponding prices. ExpressJS serves as the backend server, facilitating communication between the ReactJS front end and the MySQL database. The database stores historical parking data used for model training, along with real-time updates on parking availability and user ratings. This project contributes to the field of intelligent parking management by proposing a data-driven approach. The system promotes efficient resource allocation and user convenience through dynamic pricing based on various influencing factors. This approach aims to establish a fair and adaptable pricing structure for parking spaces, ultimately improving overall parking management.**

Keywords: ReactJS, ExpressJS , Google-Maps-API, Multiple Linear Regression , Dynamic Fare , Parking Management

I. INTRODUCTION

The burgeoning growth of urban centers has precipitated a myriad of challenges, chief among them being the efficient management of parking spaces. Congestion and the consequent rise in air pollution have necessitated the development of innovative solutions to address the scarcity of parking and its environmental repercussions. This paper introduces a pioneering application designed to alleviate these issues through dynamic pricing of parking spaces, utilizing multiple linear regression (MLR) to account for various factors such as air pollution, time, distance from the current location, rating of the parking space, and availability.

Dynamic pricing, a concept that has revolutionized industries from airlines to ride-sharing services, is predicated on the principle of supply and demand. It allows for the adjustment of prices in real-time to reflect the changing conditions of the market. In the context of parking spaces, this approach promises to optimize utilization, reduce traffic congestion, and, by extension, diminish the environmental impact of vehicles idling in search of parking.

The application at the heart of this research employs MLR, a statistical method that models the relationship between a dependent variable and one or more independent variables. Here, the dependent variable is the cost of parking, while the independent variables include:

Air Pollution: The app integrates real-time air quality indices to modulate parking fees, thereby encouraging the use of alternative transportation means during periods of high pollution.

Time: Recognizing the diurnal patterns in parking demand, the app dynamically adjusts prices to ensure efficient space utilization and to discourage prolonged parking during peak hours.

Duration from the Current Location: By considering the intended duration of parking, the app incentivizes shorter stays with lower rates, thus promoting higher turnover and better availability.

Distance from the Current Location: The app calculates prices based on how far the parking space is from the user's current location, encouraging the use of spaces that minimize additional vehicular travel and associated emissions.

Rating of the Parking Space: User-generated ratings of parking spaces are factored into the pricing algorithm, ensuring that higher-quality spaces are appropriately valued.

Availability of the Parking Space: The app utilizes a real-time tracking system to monitor the occupancy of parking spaces, adjusting prices to reflect the current availability and to prevent overcrowding in certain areas.

This comprehensive approach not only addresses the economic aspects of parking management but also places a significant emphasis on environmental sustainability. By incorporating air pollution levels into the pricing model, the application serves as a tool for cities to combat urban smog and to promote healthier living conditions.Moreover, the application's consideration of proximity and duration aligns with the broader goals of smart city initiatives, which aim to create more livable, efficient, and sustainable urban environments. Through the intelligent management of parking resources, the app has the potential to transform the way cities handle transportation and mobility. The application's MLR-based pricing model is not just a tool

for economic efficiency; it is a catalyst for behavioral change. By adjusting parking rates based on air pollution, the app nudges drivers towards more environmentally conscious decisions, potentially reducing the reliance on personal vehicles and fostering a culture of shared transportation.The temporal aspect of the pricing model serves as a deterrent against the monopolization of prime parking spots during high-demand periods. This encourages a more equitable distribution of parking resources, allowing for a smoother flow of traffic and reducing the time vehicles spend circling for parking—a significant contributor to urban congestion.

LITERATURE REVIEWS AND PAST RESEARCHES

This paper introduces a dynamic parking fare system that harnesses the power of Express.js, Google Maps API, OpenWeather API, and multiple linear regression to calculate real-time parking rates. By considering critical factors such as air pollution, parking space ratings, temporal demand, availability, intended duration, and proximity to the destination, the system offers a sophisticated solution to the traditional static parking models.

The core of the system lies in its ability to process complex datasets through multiple linear regression, predicting optimal pricing that reflects current environmental and social conditions. The integration of Express.js provides a robust backend framework to handle requests and serve the calculated fares, while the Google Maps API facilitates the assessment of distance and duration from the user's location to the parking space. The OpenWeather API contributes real-time air quality data, ensuring that the fares align with environmental sustainability goals.

This paper not only aims to streamline the parking experience for users but also to contribute positively to the reduction of urban congestion and pollution. It stands as a testament to the potential of combining modern web technologies and data analytics to address real-world challenges.

II. METHODOLOGY

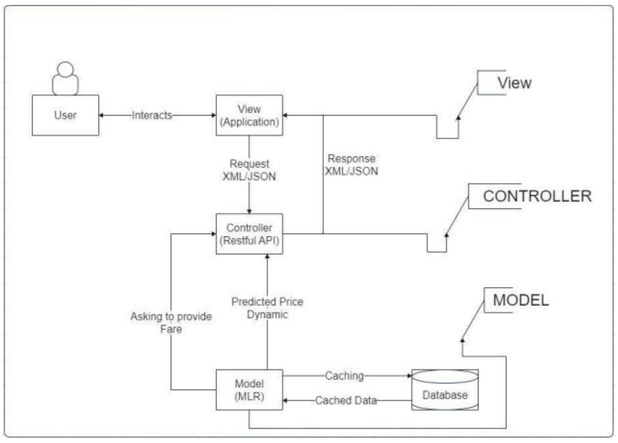


Fig. 1. System Architecture

A. User Interface and User Experience

The initial phase of the methodology revolves around crafting a user interface that integrates sleek aesthetics and an intuitive design, ensuring an optimal user experience. This process begins with a comprehensive analysis of user expectations and industry standards to establish a solid foundation for UI development.

User-Centric Approach: Emphasizing a user-centric approach, user surveys and interviews are conducted to understand preferences, pain points, and expectations. This insight guides the design process to create an interface that resonates with users.

Wireframing and Prototyping: Leveraging industry-standard tools, wireframing and prototyping are employed to visualize the layout and flow of the UI. This iterative process allows for quick adjustments based on user feedback and ensures the final design aligns seamlessly with user expectations.

Sleek Aesthetics: The design philosophy [1] prioritizes a clean and modern aesthetic. Using React.js and tailwindcss, a minimalist design is implemented that enhances visual appeal while minimizing clutter, contributing to a sleek and sophisticated UI.

Intuitive Navigation: To enhance user experience, an intuitive navigation structure is implemented. This involves strategically placing elements, employing clear and concise labels, and optimizing the user journey to enable users to effortlessly navigate through the system.

Responsive Design: Recognizing the diverse array of devices users may utilize, a responsive design is implemented that ensures a consistent and engaging experience across various screen sizes and resolutions.

User Testing: Continuous user testing is integrated into the design process. Feedback from real users is invaluable for refining the UI, ensuring that it not only meets aesthetic standards but also aligns with user expectations, making it easy to comprehend and navigate in a single glance.

B. Multiple Linear Regression

Multiple regression (MR)[2] analysis is frequently used in social sciences. Interpretation of results often involves overestimating the beta weight (see Courville and Thompson, 2001; Nimon, Roberts, and Gavrilova, 2010; Zientek, Capraro, and Capraro, 2008), which is often done because the beta weight has many limitations and differences. Interpretation of beta weight values.

In urban environments, parking systems play a crucial role in managing traffic flow, reducing pollution, and enhancing the overall quality of life of the residents of nearby traffic prone areas. By leveraging data-driven approaches, such as multiple linear regression, we can develop predictive models like "Dynamic Parking Fare System"[3] for enhancing the urban parking efficiency and sustainability.

The "Dynamic Parking Fare System" is based on various factors and the Multiple Regression model is created using python in order to optimize parking systems based on various factors like pollution levels, traffic conditions, and location attributes. This article explores the process of constructing such a model to aid in effective parking management.

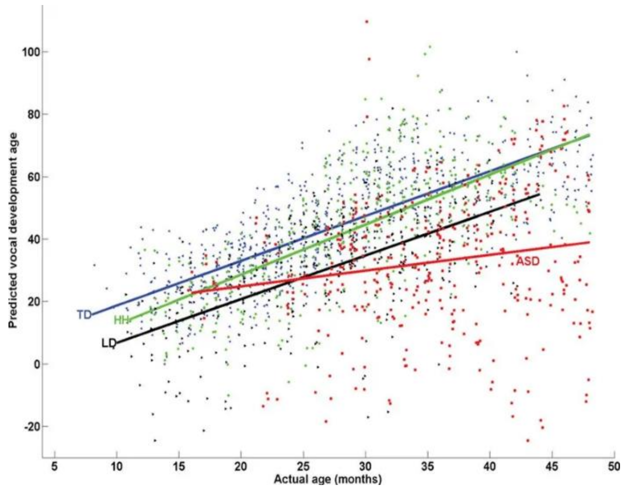


Figure 1: Multiple linear regression model predictions for individual observations

Simple linear regression [6] enables statisticians to predict the value of one variable using the available information about another variable. Linear regression attempts to establish the relationship between the two variables along a straight line.

Multiple regression is a type of regression where the dependent variable shows a linear relationship with two or more independent variables. It can also be non-linear, where the dependent and independent variables do not follow a straight line.

Both linear and non-linear regression [7] track a particular response using two or more variables graphically. However, non-linear regression is usually difficult to execute since it is created from assumptions derived from trial and error.

The Multiple Regression model created for the Development of Dynamic Parking Fare System is based on various factors like pollution, traffic, location etc. Therefore the mathematical representation of the above could be represented in the form given below-

- *Multiple Linear Regression Formula used in "Dynamic Parking Fare System"*

$$Y=\beta 0+\beta 1 \cdot \text{pollution}+\beta 2 \cdot \text{traffic}+\beta 3 \cdot \text{data availability}+\beta 4 \cdot \text{location}+\varepsilon$$

Where:

- **Y** is the dependent or predicted variable. (example: Parking Fare)
- **β0** is the y-intercept, i.e., the value of y when both xi and x2 are 0.
- **β1, β2, β3, β4** are the regression coefficients representing the change in y relative to a one-unit change they are the independent variables (pollution, traffic, data availability, and location).
- **ɛ** is the error term, representing the difference between the observed and predicted values. In this equation.

C. Considerations of Multiple Linear Regression

Overfitting: When more and more variables are added to a model, the model may become far too complex and usually ends up memorizing all the data points in the training set. This phenomenon is known as the overfitting [8] of a model. This usually leads to high training accuracy and very low test accuracy.
Multicollinearity[9]: It is the phenomenon where a model with several independent variables may have some variables interrelated.
Feature Selection[10]: With more variables present, selecting the optimal set of predictors from the pool of given features becomes an important task for building a relevant and better model

D. Backend

In the development journey of multiple regression based "Dynamic Parking Fare System", the creation of a robust and scalable backend infrastructure is fundamental. Leveraging Node.js [11] as the backend framework provides flexibility and efficiency in handling asynchronous operations and managing HTTP requests. Through the implementation of RESTful APIs, communication between the frontend and backend layers is facilitated, enabling seamless interaction with the decentralized storage network. The backend system orchestrates various functionalities, including user authentication, file upload and retrieval, and data management operations. By incorporating MySQL as the database management system [12], the backend ensures data persistence and integrity. Utilizing features such as joins and keys in MySQL enables efficient data querying and retrieval, optimizing the performance of the storage system. The integration of advanced database concepts enhances data organization and accessibility, laying the foundation for a scalable and reliable decentralized storage solution.

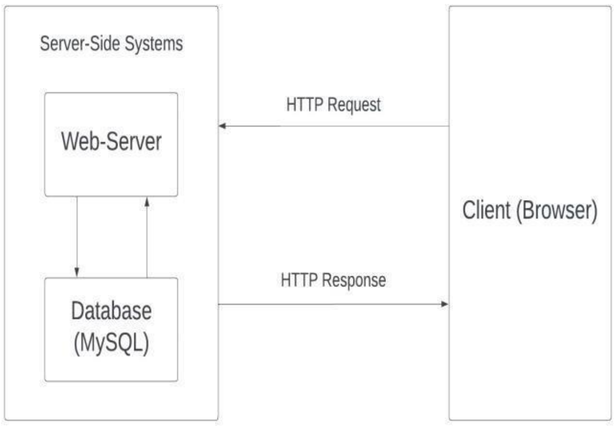


Fig. 4. Http request and response

Moreover, the implementation of HTTP requests and APIs in the backend layer facilitates seamless integration with external services and applications. Through standardized HTTP protocols [13], communication between different components of the storage system is streamlined, fostering interoperability and ease of integration. The backend system acts as the bridge between the frontend user interface and the decentralized storage network [14], translating user actions and by requesting insights or recommendations for optimizing urban parking management in a specific location. By abstracting the complexities of multiple regression and decentralized storage operations, the backend layer provides a user-friendly and intuitive interface for interacting with the storage system. Overall, the combination of Node.js, MySQL, HTTP requests, and APIs forms the backbone of the backend infrastructure, empowering the project to realize its vision of a Multiple Regression based “Dynamic Parking Fare System” with seamless user experience and robust functionality.

E. Google Maps API

Google API is an application programming interface (API) developed by Google that allows communication with Google services and integration with other services. Examples include Search, Gmail, Translate, or Google Maps. Third-party applications may use these APIs to support or extend the functionality of existing services.

Another important example is Google Maps embedded on the web, which can be done using the Static Maps API [15], Places API [16], or Google Earth API [17]. One or all of the APIs must use the Oauth 2.0 protocol for authentication and authorization. Oauth 2.0 is a simple protocol. First you need to get the credentials from the developer console. The user application can request an access token from the Google authorization server and use the authorization token to access Google API services [18]. Many languages, including Java, JavaScript, Ruby, .NET, Objective-C, PHP, and Python, allow developers to use Google APIs in their code [19].

Google Loader is a JavaScript library that allows web developers to easily load other JavaScript APIs provided by Google and other popular developer libraries. Google Loader provides JavaScript methods for loading specific APIs (also known as modules); Here you can specify additional settings such as API version, language, location, package options, loading callbacks (computer programming), and other restrictions on specific APIs. Dynamic loading or autoloading is also supported to improve the performance of API loaded applications [20].

It works fine and receives a response when a simple GET request is made to the Google API

https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=LAT,LONG&radius=500&types=parking&sensor=false&key=APIKEY

And when we pass lat long for my current city, then we are getting:

```
{
  "html_attributions" : [],
  "results" : [
    {
      "geometry" : {
        "location" : {
          "lat" : 23.027462,
          "lng" : 72.57919699999999
        }
      },
        "icon":"http://maps.gstatic.com/mapfiles/place
        _api/icons/generic_business-71.png",
        "id":"6f7c1e623f7d6a70eb612d6f58373c222b5f
        1c97",
        "name" :"AMC Pay and Park",
        "reference":"CoQBcwAAAF7ZnmGltBQ7log8OTH
        xhjqMNZKq2wQTffbXmTI2u4Lm-
        QXhAOVTxwK2Lu_SSuZgdQ1uVMLmo9hhJhXHT
        7Loa4LEzqOePfGwJvZ3pGbEFdg7fEGsqaNdNpVn
        w0hMUK4qFkBlh5fstFAoanUGJ5LRistdJl7otAwQ
        -bWQIPXH_DJXEhCTkcSIZq-
        69YHIURouswj7GhTCyO-
        tz0DiTdVozVlBBqiC15T4OA",
        "types" : [ "parking", "establishment" ],
        “vicinity" : "Khanpur, Amdavad"
    },
    {
        "geometry" : {
          "location" : {
            "lat" : 23.026521,
            "lng" : 72.583656
          }
        },
          "icon":"http://maps.gstatic.com/mapfiles/plac
          e_api/icons/generic_business-71.png",
          “id":"37ba75c2dd455b0c63c207f17fdbb27bb1
          1951bb",
          "name" : "AMC Parking",
          "reference":"CnRuAAAAXfjCgAfzvpuB3Bp_iWg
          XFnPp3"iO_I24ozqvfKnzND1jH5pZ5y_skNlCDS
          yworLODMTjJ7Bx9EabdRsaWuoWGYnb4W5ax
          P9unFvprwftoDfAbWFS0RQREl01mCyeU-
          jO56izs9Q6KerrPkbZR2M_E9NRIQ5gVcDojpihb
          QQGDhjHDvAxoUhJTBKuodC-
          bBz_Ovdq4cbu67eQk",
          "types" : [ "parking", "establishment" ],
          "vicinity" : "Relief Rd, Gheekanta, Bhadra,
          Amdavad"
        }
    ],
    "status" : "OK"
}
```

III. RESULTS

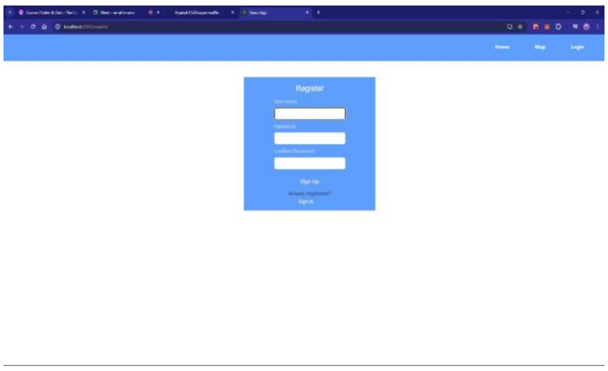


Fig 1 User Login Interface

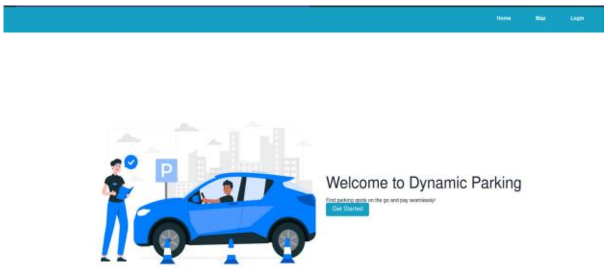


Fig 2 Homepage

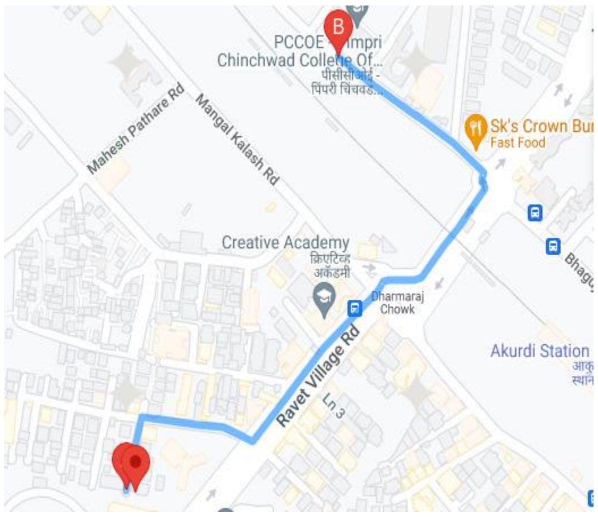


Fig 3 Route between source and destination

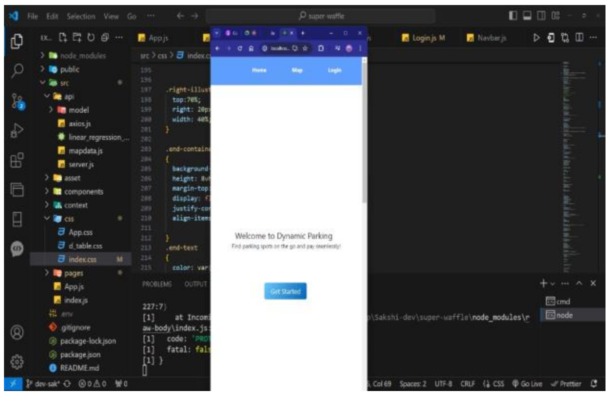
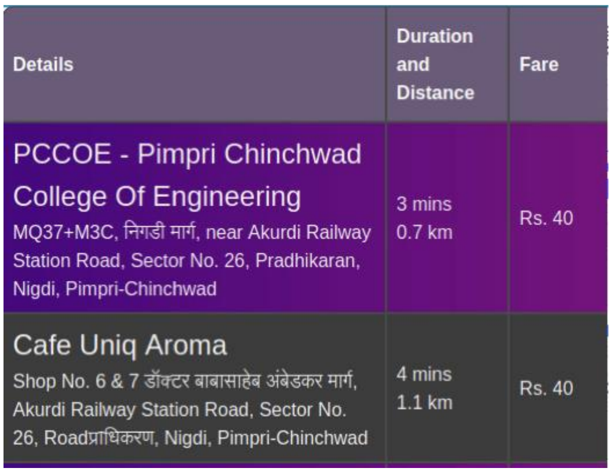


Fig 4 Responsiveness of website is shown in above figure



Details	Duration and Distance	Fare
PCCOE - Pimpri Chinchwad College Of Engineering MQ37+M3C, निगडी मार्ग, near Akurdi Railway Station Road, Sector No. 26, Pradhikaran, Nigdi, Pimpri-Chinchwad	3 mins 0.7 km	Rs. 40
Cafe Uniq Aroma Shop No. 6 & 7 डॉक्टर बाबासाहेब अंबेडकर मार्ग, Akurdi Railway Station Road, Sector No. 26, Roadप्राधिकरण, Nigdi, Pimpri-Chinchwad	4 mins 1.1 km	Rs. 40

Fig 5 Table displaying the result for nearby parking areas as per user request

IV. ACKNOWLEDGMENT

The successful completion of this research paper is the result of the combined efforts and support of numerous individuals and institutions to whom we extend our sincere appreciation. We express our deepest gratitude to our project supervisor for their invaluable guidance, expertise, and encouragement throughout the research process. Special thanks are also due to our team members and colleagues for their collaborative efforts and dedication. We acknowledge the support provided by the academic community, including the insightful discussions and resources shared by our peers and mentors. Furthermore, we extend our thanks to the developers and contributors of the technologies utilized in this research, including React.js, tailwindcss, MySQL, and the Google API. Lastly, we express our heartfelt gratitude to our families, friends, and loved ones for their unwavering support and understanding. Their encouragement and patience have been a constant source of motivation. This research paper would not have been possible without the collective efforts and support of all mentioned above.

REFERENCES

[1] S.-H. Chen, Y. Liang, and M.-C. Yang, “KVSTL: An application support to lsm-tree based key-value store via shingled translation layer data management,” IEEE Trans. Comput., vol. 71, no. 7, pp. 1598–1611, Jul. 2022.

[2] M. Ali, J. Nelson, R. Shea, and M. Freedman, “Blockstack: A global naming and storage system secured by blockchains,” in Proc. USENIX Annu. Tech. Conf., 2016, pp. 181–194

[3] Ma Xin-jian, Yao Ke-jia, Wei Guang-juan and Zhang Wei-she, "The practical principles for website design," 2019 IEEE Conf 2019, pp.

[4] X. Yan, H. Xie and Wang Tong, "A multiple linear regression data predicting method using correlation analysis for wireless sensor networks," Proceedings of 2018 IEEE, conf pp

[5] H. Nakanishi and T. Namerikawa, "Parking Lot Allocation and Dynamic Parking Fee System Based on a Mechanism Design Approach," 2019 American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 2683-2689

[6] H. Wang and F. Hao, "An efficient linear regression classifier," 2012 IEEE International Conference on Signal Processing, Computing and Control, Solan, India, 2012, pp. 1-6,

[7] X. Wang, "A Nonlinear Regression Model under Uncertain Environments," 2018 37th Chinese Control Conference (CCC), Wuhan, China, 2018, pp.

[8] H. Li, J. Li, X. Guan, B. Liang, Y. Lai and X. Luo, "Research on Overfitting of Deep Learning," 2019 15th International Conference on Computational Intelligence and Security (CIS), Macao, China, 2019, pp.

[9] M. Sharma and S. Saha, "Graph based approach for minimum multicollinearity highly accurate regression model explaining maximum variability," 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence), Noida, India, 2014, pp. 304-308,

[10] T. R. N and R. Gupta, "Feature Selection Techniques and its Importance in Machine Learning: A Survey," 2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS), Bhopal, India, 2020, pp. 1-6,

[11] Sinan Chen, Atsuko Hayashi, Masahide Nakamura, "Study of Measuring Motor Function in Dementia by Tapping Task Using IoT and Image Recognition", 2023 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD), pp.1-6, 202 3.

[12] Fadi Shrouf, Joaquin Ordieres and Giovanni Miragliotta, "Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm", Industrial Engineering and Engineering Management (IEEM) 2014 IEEE International Conference on, 2014.

[13] Information Processing Systems Open System Interconnection OSI Conformance Testing Methodology and Framework, vol. 9646, 1991.

[14] Ștefan Sarkadi, Fabien Gandon, "Interoperable AI for Self-Organisation", 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), pp.86-87, 2023.

[15] "Static maps API" 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus).

[16] "Google Places API". Archived from the original on November 13, 2014. Retrieved October 31, 2014.

[17] Rohan Pradhan, Shaswat Saxena, Akarsh Kumar Singh, Jabir Ali, "Battery Swapping System for the Electric Vehicles", 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp.919-924, 2022."Google Earth API".

[18] "Using Oauth 2.0 to Access Google APIs"Somchart Fugkeaw, "Achieving Decentralized and Dynamic SSO-Identity Access Management System for Multi-Application Outsourced in Cloud", IEEE Access, vol.11, pp.25480-25491, 2023..

[19] "Google APIs Client Libraries" R Arthi., S Nishuthan, L Deepak Vignesh, "Smart Agriculture System Using IoT and ML", 2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT), pp.1-6, 2023.

[20] "Google Loader Developer's Guide". Archived from the original on January 26, 2013. Retrieved February 26, 2013..