**SRI SHANMUGHA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**Department of Biomedical Engineering**

**SMART PARKING**

**Phase 4 Submission Document**

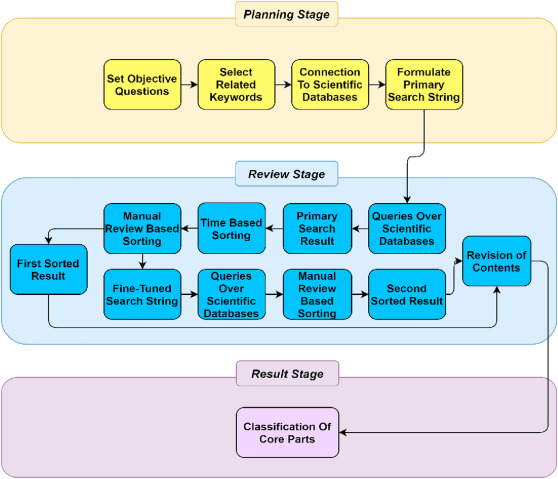
**Team members**

S. Swetha- (732721121055)  
 M. Sindhu - (732721121046)  
 T. Srimathi- (732721121051)  
 C. Chanthruba- (732721121007)

R. Bharathi- (732721121006)



**Introduction:**

 The goal of parking system project is to reserve parking spot for a car/vehicle before it arrives. One of the most problems that the driver faces is finding a free parking spot, so many driver stopping their cars at the edges of the street. Therefore, we choose this to prevent the frustration of finding a parking spot and they can reserve a spot when they stay at home. It is an IOT based project.

**Overview of the process:**

Smart parking systems aim to improve the efficiency and convenience of parking by leveraging technology and data to assist drivers in finding available parking spaces. The process of a typical smart parking system involves several key components and steps:

**Sensor Deployment**: Sensors, such as ultrasonic sensors, infrared sensors, or cameras, are installed in parking spaces or areas to monitor their occupancy status. These sensors can detect whether a parking spot is vacant or occupied.

**Data Collection:** The deployed sensors continuously collect data on the availability of parking spaces. This data is sent to a central server or cloud-based platform for processing and analysis.

**Data Processing and Analysis**: The data collected from sensors is processed and analyzed in real-time to determine the availability of parking spaces. Machine learning algorithms may be used to predict parking spot availability based on historical data and real-time sensor readings.

**User Interface:** A user-friendly interface, often in the form of a mobile app or a website, is made available to drivers. This interface provides real-time information about parking availability, including the number of available spots and their locations.

**Navigation and Booking:** Drivers can use the app or website to search for available parking spots near their destination. The system provides navigation instructions to guide drivers to the selected parking spot. Some systems also allow users to reserve parking spaces in advance.

**Payment and Billing:** Payment for parking can be integrated into the app or website. Drivers can make payments electronically, reducing the need for physical tickets or cash. Parking fees can be automatically calculated based on the duration of the stay.

**Feedback and Alerts:** The system can provide alerts to drivers, such as reminders of when their parking time is about to expire. Drivers can also provide feedback on the parking experience, helping operators improve the system.

**Management and Analytics:** Operators of the smart parking system have access to a management dashboard where they can monitor the status of parking spaces, gather analytics, and make data-driven decisions to optimize parking operations.

**Integration:** Smart parking systems can be integrated with other urban infrastructure systems, such as traffic management and public transportation, to create a more holistic approach to urban mobility.

**Performance Evaluation Techniuqes:**

To evaluate the performance of deep LSTM network we used the root mean square error (RMSE), mean absolute error (MAE), mean squared error (MSE), median absolute error (MdAE), and mean squared log error (MSLE). The mathematical formulation of these performance evaluation techniques is defined as:

RMSE= √1/𝑛𝑛 ∑ (𝑦𝑦𝑖𝑖 − 𝑦𝑦^𝑖𝑖) 𝑛𝑛 2 𝑖𝑖=1 (1)

MAE = 1/𝑛𝑛 ∑ |𝑦𝑦𝑖𝑖 − 𝑦𝑦^ 𝑖𝑖 | 𝑛𝑛 𝑖𝑖=1 (2)

MSE =1/𝑛𝑛 ∑ (𝑦𝑦𝑖𝑖 − 𝑦𝑦^𝑖𝑖) 𝑛𝑛 2 𝑖𝑖=1 (3)

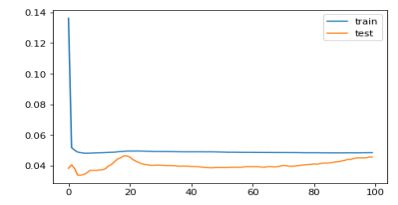
MdAE = median (∑ |𝑦𝑦𝑖𝑖 − 𝑦𝑦^ 𝑖𝑖 | 𝑛𝑛 𝑖𝑖=1 ) (4)

MSLE = 1/𝑛𝑛 ∑ log(𝑦𝑦𝑖𝑖) + 1 − log ( (𝑦𝑦^ 𝑖𝑖 ) + 1) 2 𝑛𝑛 𝑖𝑖=1 (5)

The symbol yi is the actual avail parking space that is computed by taking the difference in occupancy and capacity values. Whereas, yˆi is the predicted available parking space, predicted by deep LSTM network. The difference between actual available parking space and predicted parking space is computed by y y i i − ˆ . The experimental results achieved using a deep LSTM network.

**Model training:**

Model training in the context of the Internet of Things (IoT) involves training machine learning models to analyze data generated by IoT devices. IoT devices can produce vast amounts of data, and machine learning models can help extract valuable insights, make predictions, and enable automation based on this data.



**Data Collection:** IoT devices collect data, which can include sensor readings, images, audio, or other types of data. This data is typically stored and preprocessed before it's used for training.

**Data Preprocessing:** Raw IoT data often requires preprocessing to clean, format, and transform it into a suitable format for model training. Data preprocessing may involve handling missing values, normalizing data, and feature engineering.

**Data Labeling (if applicable):** In some IoT applications, data may need labeling, such as tagging anomalies in sensor data for predictive maintenance.

**Training Data Split:** The dataset is divided into training data, validation data, and test data to evaluate and fine-tune the model.

**Model Selection:** Choose an appropriate machine learning or deep learning model architecture based on the specific IoT application. Common models include decision trees, support vector machines, neural networks, and more.

**Feature Selection**: Identify relevant features or input variables that are important for making predictions. Feature selection can improve model efficiency and accuracy.

**Model Training:** The selected model is trained using the training data. During training, the model learns to make predictions or classifications based on the input data. Training may involve optimizing model parameters using optimization algorithms (e.g., gradient descent) to minimize a defined loss function.

**Hyperparameter Tuning:** Adjust hyperparameters like learning rate, batch size, and the number of layers or neurons in the model to optimize model performance. Hyperparameter tuning is often done using techniques like grid search or random search.

**Validation and Testing:** The trained model is evaluated using the validation dataset to assess its performance and generalization capability. Additional testing is performed on the test dataset to confirm the model's ability to make predictions on unseen data.

**Model Deployment:** Once the model demonstrates satisfactory performance, it can be deployed on edge devices, gateways, or cloud platforms to analyze real-time IoT data.

**Continuous Monitoring:** IoT models may need ongoing monitoring to ensure they continue to perform well over time. Drift detection mechanisms can be implemented to detect changes in data patterns that might require model retraining.

 The overall performance of deep LSTM is 0.068, 0.0411, 0.0046, 0.028, and 0.002 with RMSE, MAE, MSE, MdAE, and MSLE, respectively. From these results we can analyze that prediction accuracy of the proposed decision support system is 93.2%, 95.9%, 99.6%, 97.2%, and 99.8% respectively with five measures. The values presented in Table 3 demonstrate the reliability of the proposed decision support system. In general, from all performance measurement parameters, the minimum prediction accuracy is 93.2% (RMSE) and the maximum prediction accuracy is 99.8% (MSLE). By analyzing the performance of the proposed decision support system on individual parking slots we can notice that the average, maximum, minimum, and standard deviation with RMSE is 0.109, 0.177, 0.048, and 0.028, respectively.

**Model evaluation:**

Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data. There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:

**Mean squared error (MSE**): This metric measures the average

squared difference between the predicted and actual house prices.

**Root mean squared error (RMSE):** This metric is the square root

of the MSE.

**Mean absolute error (MAE):** This metric measures the average

absolute difference between the predicted and actual house prices.

**R-squared**: This metric measures how well the model explains the

variation in the actual house prices.

In addition to these metrics, it is also important to consider the following

factors when evaluating a house price prediction model:

**Bias:** Bias is the tendency of a model to consistently over- or

underestimate house prices.

**Variance:** Variance is the measure of how much the predictions of

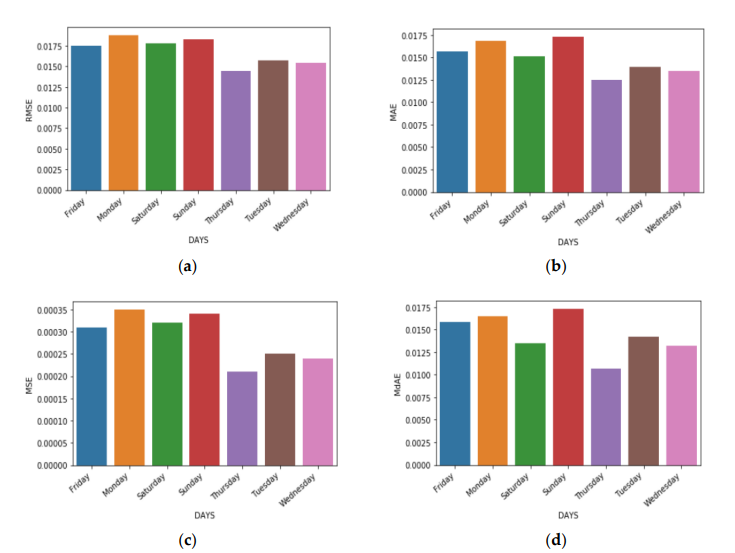
a model vary around the true house prices.

**Interpretability:** Interpretability is the ability to understand how

the model makes its predictions. This is important for house price

prediction models, as it allows users to understand the factors that

influence the predicted house prices.



1. Calculate the evaluation metrics. There are a number of different

evaluation metrics that can be used to assess the performance of a

machine learning model, such as R-squared, mean squared error

(MSE), and root mean squared error (RMSE).

1. Interpret the evaluation metrics. The evaluation metrics will

give you an idea of how well the model is performing on unseen data. If

the model is performing well, then you can be confident that it will

generalize well to new data. However, if the model is performing poorly,

then you may need to try a different model or retune the

hyperparameters of the current model.

**Program:**

#include <Ultrasonic.h>

#define NUM\_PARKING\_SPACES 4

Ultrasonic ultrasonic[NUM\_PARKING\_SPACES] = {

Ultrasonic(2, 3), // Trigger, Echo for Space 1

Ultrasonic(4, 5), // Trigger, Echo for Space 2

Ultrasonic(6, 7), // Trigger, Echo for Space 3

Ultrasonic(8, 9) // Trigger, Echo for Space 4

};

int parkingSpaceStatus[NUM\_PARKING\_SPACES] = {0}; // 0 = vacant, 1 = occupied

void setup() {

Serial.begin(9600);

for (int i = 0; i < NUM\_PARKING\_SPACES; i++) {

pinMode(i \* 2 + 2, OUTPUT); // Initialize LED pins for each parking space

}

}

void loop() {

for (int i = 0; i < NUM\_PARKING\_SPACES; i++) {

long distance = ultrasonic[i].timing();

int status = parkingSpaceStatus[i];

// Adjust the threshold based on your sensor placement and environment

int threshold = 20; // Distance threshold (in cm) to detect a car

if (distance < threshold && status == 0) {

// Vehicle detected

parkingSpaceStatus[i] = 1;

digitalWrite(i \* 2 + 2, HIGH); // Turn on LED for occupied space

Serial.print("Space ");

Serial.print(i + 1);

Serial.println(" is occupied.");

} else if (distance >= threshold && status == 1) {

// Space vacated

parkingSpaceStatus[i] = 0;

digitalWrite(i \* 2 + 2, LOW); // Turn off LED for vacant space

Serial.print("Space ");

Serial.print(i + 1);

Serial.println(" is vacant.");

}

}

}

**Feature Engineering:**

Feature engineering is a crucial aspect of building a house price

prediction model using machine learning. It involves creating new features, transforming existing ones, and selecting the most relevant variables to improve the model's predictive power. Here are some feature engineering ideas for house price prediction:

**1.Total Area Features:** Combine individual room areas to create features like "TotalLiving Area," "Total Bedroom Area," or "Total Bathroom Area." Thesecan be significant predictors of house price.

**2.Ratio Features**: Create features that represent ratios, such as the "Bedroom to Bathroom Ratio" or "Living Area to Lot Area Ratio." These ratios may capture the property's layout and functionality.

**3.Age of the Property:** Calculate the age of the property by subtracting the constructionyear from the current year. Newer properties might have higher values.

**4.Neighborhood Statistics:** Aggregate neighborhood-level statistics, such as the averageincome, crime rate, school ratings, or proximity to amenities, and usethese as features.

**5.Distance to Key Locations:** Calculate distances from the property to essential places likeschools, parks, shopping centers, or public transportation hubs. Closerproximity to such amenities can affect the price.

**6.Categorical Encodings:** Use techniques like one-hot encoding, label encoding, or targetencoding for categorical variables, such as property type, heating system,or garage type.

**7.Seasonal Features:** Create features indicating the season during which the house wassold. Seasonality can influence property demand and prices.

**8.Historical Data:** Incorporate historical data on house prices and local real estatemarket trends. This can help the model account for cyclical patterns.

**9.Exterior Features:** Develop features related to the property's exterior, such as thepresence of a swimming pool, patio, or garden. These features can bevaluable for determining a property's appeal.

**10.Quality Scores:** Create a combined quality score by aggregating the quality ratingsof various components of the property, such as kitchen quality,bathroom quality, and overall house quality.

**Challenges of the system:**

Smart parking systems face several challenges, both technical and operational, in their implementation and maintenance. Some of the common challenges include:

**Cost of Implementation:** The initial investment in sensors, infrastructure, and software can be substantial, and it may take time to realize a return on investment.

**Sensor Accuracy:** Parking sensors may provide inaccurate data at times, leading to incorrect information about parking space availability.

**Data Privacy:** Collecting and managing data related to vehicle movements, such as license plate recognition, raises privacy concerns. Safeguarding this data and ensuring compliance with data privacy regulations is crucial.

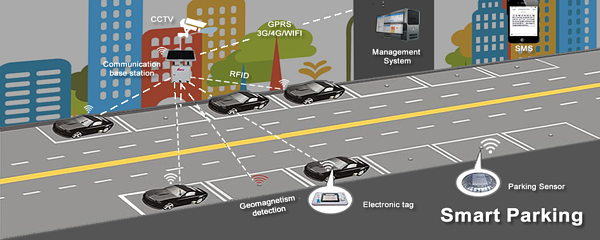
**Data Security:** Smart parking systems are susceptible to data breaches and cyberattacks. It is essential to implement robust security measures to protect sensitive information.

**Maintenance:** Regular maintenance of sensors, cameras, and other equipment is necessary to ensure system reliability and accuracy.

**Interoperability:** Ensuring that different components and devices from various vendors work seamlessly together can be challenging. Standardization is key to addressing interoperability issues.

**Traffic Management:** Smart parking systems can impact traffic flow, so integration with broader traffic management systems is vital to prevent congestion.

**Environmental Impact:** The energy consumption of sensors, lighting, and communication devices can have environmental implications, so implementing energy-efficient solutions is a consideration.



**Conclusion:**

Smart parking solutions offer a comprehensive approach to tackling the challenges associated with parking in today's urban environments. While initial implementation can be an investment, the long-term benefits in terms of reduced traffic congestion, environmental impact, and improved user experiences make smart parking a valuable addition to modern cities. As technology continues to advance, smart parking systems are likely to evolve and become even more integral to urban planning and sustainability efforts.