



*Lahiru Dissanayake*

# Smart Car Parking

- Predictive Availability System -

## Table of Contents

1. [Overview](#)
2. [Business Problem](#)
3. [Data Summary](#)
4. Data Quality & Preprocessing
5. Exploratory Data Analysis & Insights
6. Model Development
7. Results
8. Model Serving & System Design
9. Future Improvements
10. KPIs & Monitoring
11. Project Structure
12. Installation & Usage
13. Technologies
14. Contributing

### 1. Overview

This project is a **smart parking analytics and prediction system** that forecasts **available parking spots** for road segments in a city using **historical occupancy, weather, and location features**.

The goal is to support:

- **Drivers:** find parking faster and with less stress
- **Parking operators:** optimize capacity usage and revenue through dynamic pricing

- **City authorities:** reduce congestion and emissions
- **Local businesses:** increase foot traffic via accessible parking

The core of the solution is a **Random Forest regression model** that predicts the **number of available spots** at a given time and location, achieving **high accuracy and low error**, making it suitable for real-time smart parking applications.

## 2. Business Problem

### 2.1 Who benefits from a smart parking solution?

- **Drivers**
  - Reduce time and stress searching for a spot in busy urban areas, near offices, restaurants, and during events.
  - Plan better by knowing **real-time availability** and **expected cost**.
- **Parking lot operators**
  - Optimize revenue via **dynamic pricing**.
  - Improve utilization and planning of on-street and off-street capacity.
- **City authorities**
  - Reduce **traffic congestion** and **emissions** caused by cruising for parking.
  - Use insights for **urban planning** and infrastructure decisions.
- **Local businesses**
  - Benefit from increased **customer retention** when parking is easy and predictable.

### 2.2 User Pain Points and How the System Helps

Problem	Pain Reliever
Difficulty finding parking	Shows <b>real-time available spots</b> and navigates drivers directly to them.
Inconvenient payment methods	Integrates <b>digital and mobile payment</b> options.
Unexpected parking costs	Uses <b>dynamic pricing</b> and exposes expected prices upfront.
Limited parking during peak times	Manages demand with <b>price incentives</b> and visibility of alternative segments/lots.
Poor navigation within parking facilities	Provides guided navigation and <b>parking reminders</b> .
Environmental and traffic concerns	Reduces cruising time for parking, lowering <b>traffic and emissions</b> .
Unclear parking rules	Provides clear information on <b>rules, restrictions, and time windows</b> to avoid fines.

To act as both a **gain creator** and **pain reliever**, the product must combine:

- Accurate **availability predictions**
- **Real-time UX** (apps, web) for drivers
- **Dynamic pricing engine**
- Integrations with **events, weather, and public transport**

### 3. Data Summary

The analysis is based on **4 months of data** across multiple sources:

#### 3.1 Sources

- **Ground truth:** On-street parking data (capacity, occupied, available)
- **Weather features:** Temperature, wind speed, precipitation
- **Road features:** Commercial intensity, residential index, nearby facilities (schools, restaurants, shopping, offices, supermarkets), off-street parking capacity

#### 3.2 Key Dataset Characteristics

- **Time span:** 4 months
- **Spatial coverage:** 57 distinct road segments
- **Total parking slots represented:** 13,439,520
- **Observations:** ~1,251,720 records
- **Time-of-day coverage:** 06:00–21:00 (focus on active hours)

#### 3.3 Basic Descriptive Stats (Selected)

- **Average max capacity per road:** ~11 slots
- **Half of the roads:**  $\leq 4$  parking spaces
- **Weather**
  - Mean temperature: ~16 °C
  - Mean precipitation: 0.10 mm
  - Mean wind speed: 12.75 km/h

Environmental and location factors show **clear relationships** with occupancy and availability, which motivates using them as predictive features.

### 4. Data Quality & Preprocessing

#### 4.1 Identified Data Issues

Issue Type	Affected Variables	Count / Proportion	Handling Strategy
Missing values	commercial	21,960 rows	Median imputation
Outliers	available, precipMM, transportation	Up to 22.73% for precipMM	Capping at 99th percentile (where applicable)

Issue Type	Affected Variables	Count / Proportion	Handling Strategy
Logical inconsistencies	maxcapacity, occupied, available	1,353 negative available	Set negative available to 0
Invalid values	tempC	1 occurrence (-999)	Replaced with column mean
Duplicates	Join keys (roadsegmentid, timestamp)	0	No action required

Logical consistency rule:

For each record, **occupied + available = maxcapacity** should hold. Violations manifested as negative available values, which were corrected.

## 4.2 Preprocessing Steps

1. **Data loading and merging**
  - Merged ground\_truth, weather\_features, and road\_features on roadsegmentid and timestamp.
2. **Datetime handling**
  - Converted timestamp to datetime.
  - Extracted date, time, hour, day of week, month.
3. **Column cleanup**
  - Dropped auxiliary index columns such as Unnamed: 0x, Unnamed: 0y, etc.
4. **Missing values**
  - commercial filled with median.
5. **Outliers**
  - Capped precipMM and transportation at their 99th percentile where relevant.
6. **Data quality fixes**
  - Negative available values set to 0.
  - tempC == -999 replaced with mean temperature.
7. **Feature scaling**
  - Standardized selected numerical features (where required by experiments).
8. **Feature engineering**
  - Added lag features and rolling windows (details in Model Development).

## 5. Exploratory Data Analysis & Insights

### 5.1 Temporal Patterns

- **Peak days:** Weekdays, especially midweek.
- **Peak hours:**
  - 10:00-15:00

- 18:00-21:00
- **Event window:** 18:00-21:00 assumed to correspond to event times; these hours show **increased occupancy**.
- **Trend:** An upward trend in occupied spaces over the four months was observed, indicating **growing demand**.

#### Environmental Effects

- Occupancy **decreases** when:
  - Temperature  $< 7^{\circ}\text{C}$  or  $> 30^{\circ}\text{C}$ .
  - **Rain** or bad weather conditions are present.
- Suggestion: **Weather-aware pricing and incentives** (discounts on bad-weather days) to counter reduced demand.

#### Location-Based Behavior

- Roads near **restaurants, shopping, and residential areas** show:
  - Significantly higher occupancy.
  - Stronger correlation with demand.
- Only **~25% of segments** have public transport options, limiting park-and-ride opportunities. This is a **clear expansion opportunity**.

#### Correlations & Feature Importance

- **Max capacity vs available:** Strong positive correlation ( $\sim 0.95$ ).
- Feature importance (Random Forest):
  - maxcapacity
  - occupied
  - restaurant
  - Followed by residential, offstreetcapa, numoffstreetparking, etc.

These patterns support **data-driven decisions** in pricing, infrastructure, and prioritization of high-impact locations.

## 6. Model Development

### 6.1 Problem Definition

- **Task:** Predict **available parking spots** (continuous value) for a given road segment and timestamp.
- **Type:** Regression.

### 6.2 Target and Features

- **Target:** available
- **Core features:**
  - Capacity & usage: maxcapacity, occupied

- Weather: tempC, windspeedKmph, precipMM
- Location: commercial, residential, transportation, schools, eventsites, restaurant, shopping, office, supermarket, numoffstreetparking, offstreetcapa
- Temporal: dayofweek, isweekend, hour, month, iseventtime
- Engineered: lag and rolling statistics on occupied and maxcapacity

### 6.3 Baseline Model Choice

- Random Forest Regressor

#### Reasons

- Handles **non-linear relationships** and **interactions**.
- Works well with **mixed feature types**.
- Robust to **outliers**.
- Provides **feature importance**, aiding interpretability.
- Good balance between performance and implementation complexity for structured data.

### 6.4 Alternative Algorithms Considered

Algorithm	Pros	Cons
Gradient Boosting (GBM)	Often more accurate for structured data	Longer training time, requires parameter tuning
XGBoost	Fast, efficient, high accuracy	Sensitive to overfitting if not tuned properly
Linear Regression	Simple, interpretable, and fast to train	Limited in capturing non-linear relationships

### 6.5 Feature Engineering Experiments

Several experimental configurations were tested:

1. **Attempt 1 - Baseline (no advanced FE)**
  - Features: mostly raw numeric/location features.
2. **Attempt 2 - Specific feature subset**
  - Focus on location/weather features only.
3. **Attempt 3 - Full feature set**
  - All relevant raw features including temporal.
4. **Attempt 4 - Lag & rolling features (occupied)**
  - Lag-1, lag-3, lag-6, rolling means for multiple windows.
5. **Attempt 5 - Lag & rolling (occupied + maxcapacity) (BEST)**

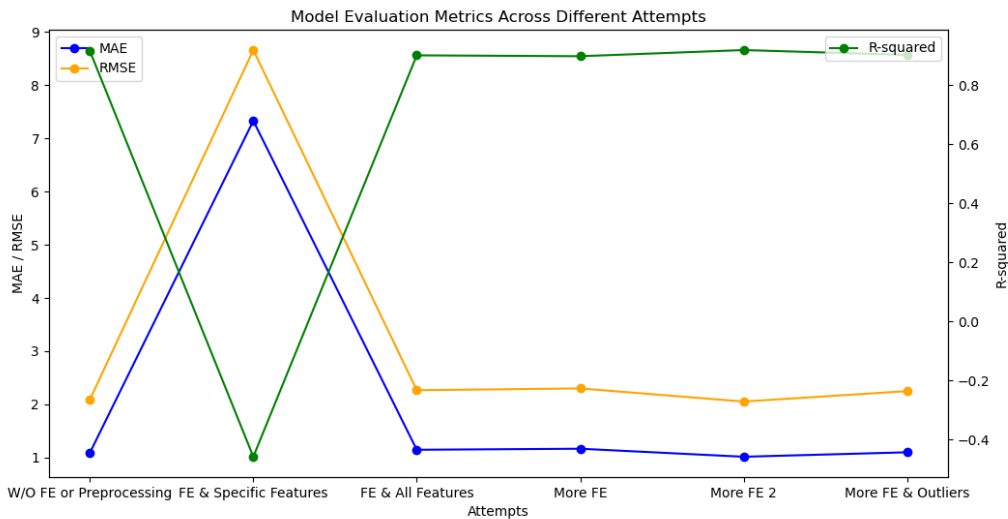
- Combined lag/rolling features on both occupied and maxcapacity.
6. **Attempt 6 - Same as 5 with aggressive outlier treatment**
- Additional capping on selected columns.

Train/validation split used chronological splitting (shuffle=False) to better reflect **time-series behavior**.

## 7. Results

### 7.1 Experiment Summary

Attempt	Description	MAE	RMSE	R <sup>2</sup>	Status
1	Baseline, all core features	1.09	2.09	0.915	Good
2	Selected features only (no occupancy/capacity)	7.33	8.66	-0.46	Poor
3	All raw features incl. temporal	1.15	2.27	0.900	Good
4	Lag & rolling on occupied + core features	1.17	2.30	0.897	Good
5	Lag & rolling on occupied & maxcapacity	<b>1.01</b>	<b>2.05</b>	<b>0.918</b>	<b>Best</b>
6	Attempt 5 + stronger outlier capping	1.10	2.25	0.902	Good



### 7.2 Final Model Performance (Attempt 5)

Mean Absolute Error (MAE):  $\approx 1.04$  spots

Root Mean Squared Error (RMSE):  $\approx 2.11$  spots

R<sup>2</sup> Score:  $\approx 0.916$

### 7.3 Interpretation

- On average, predictions are off by **about 1 parking spot**.

- The model explains **~91.6% of the variance** in availability.
- For a typical parking segment with ~10 spots, this level of error is **highly acceptable** for operational usage.
- The model is **robust, fast, and interpretable** for practical deployment.

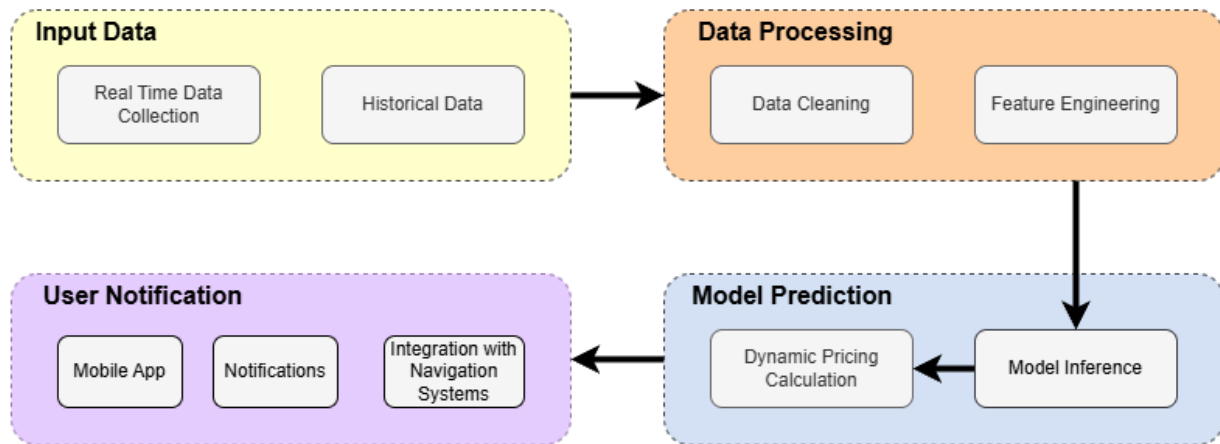
#### 7.4 Are the Results Satisfactory?

Yes. The model demonstrates:

- Low prediction errors and high explanatory power.
- Capture of **key determinants** of parking availability.
- Suitability for **real-time guidance systems** and **dynamic pricing engines**.
- Balance between accuracy and production readiness.

## 8. Model Serving & System Design

### 8.1 High-Level Workflow



### 8.2 Delivery & Deployment Components

#### 1. Real-Time Data Collection

- IoT sensors and systems collect:
  - Occupancy data
  - Weather data
  - Event indicators

#### 2. Feature Pipeline

- Join latest sensor data with road features.
- Generate temporal and lag-based features (where feasible).

#### 3. Prediction Service

- Random Forest model served via an API or batch job.
- Returns predicted available spots and confidence indicators (optional).

#### 4. **Dynamic Pricing Engine**

- Uses predictions + business rules to set real-time prices.
- Peak hours: increase pricing.
- Off-peak hours: reduce pricing to incentivize usage.
- Event-based: premium pricing during high-demand periods.
- Weather-based: discounts during adverse conditions.

#### 5. **User-Facing Applications**

- Mobile app / web interface providing:
  - Available spots in real-time
  - Dynamic pricing information
  - Navigation directions
  - Parking reminders

#### 6. **Monitoring & Feedback Loop**

- Logs predictions, actuals, and KPIs.
- Triggers model retraining when performance degrades.
- Feeds operational insights back to business teams.

## 9. Future Improvements

### 9.1 Model & Algorithm

- **Hyperparameter tuning**
- Use grid search or random search on:
  - max\_depth
  - min\_samples\_split
  - min\_samples\_leaf
  - max\_features(Careful optimization to avoid overfitting)
- **Advanced algorithms**
  - Gradient Boosting (GBM)
  - XGBoost / LightGBM
  - Neural networks (MLP, LSTM for time-series)
  - Ensemble methods combining multiple models
- **Richer feature engineering**
  - Interaction features (e.g., weather × time-of-day)
  - Seasonal decomposition and trends
  - Cyclical encoding for temporal features

### 9.2 Data Enrichment

- **Event data**

- Concerts, sports, festivals, major drivers of short-term demand spikes.
- Real-time event calendars and APIs.
- **Traffic volume**
  - Real-time traffic density/flow to link congestion with parking demand.
  - Traffic prediction APIs.
- **Nearby parking availability**
  - Occupancy in neighboring segments or off-street lots.
  - Overflow and substitution patterns.
- **Detailed weather**
  - Humidity, visibility, and real-time forecast.
  - Impact on short-term parking behavior.
- **Public transport schedules**
  - Integration with transit systems for park-and-ride behaviors.
  - Modal shift analysis.

### 9.3 Operational Improvements

- **Automated retraining**
  - Based on data drift detection, performance decay, or calendar schedule.
  - CI/CD pipelines for model deployment.
- **A/B testing**
  - Experiment with different pricing strategies.
  - UX flow variations.
- **User feedback loops**
  - Integrate app rating and feedback into model improvement.
  - Quality assurance checks.
- **Infrastructure expansion**
  - Off-street parking in high-demand residential/commercial zones.
  - EV charging stations.

## 10. KPIs & Monitoring

To ensure robust performance in production, monitor the following KPIs:

### 10.1 Prediction Quality

Metric	Target	Notes
MAE	< 1.5 spots	Currently 1.04
RMSE	< 2.5 spots	Currently 2.11
R <sup>2</sup>	> 0.90	Currently 0.916
Calibration	Prediction distribution $\approx$ actual	Quarterly review

## 10.2 System Performance

Metric	Target	Notes
Latency	< 500 ms	Per-request prediction time
Uptime	> 99.5%	System operational availability
Data freshness	< 5 min	Age of latest sensor data

## 10.3 User Engagement

Metric	Target	Notes
User retention rate	> 60%	App usage over time
Notification CTR	> 25%	Click-through on notifications
User satisfaction	> 4.0 / 5.0	App rating

## 10.4 Revenue & Utilization

Metric	Target	Notes
Revenue from dynamic pricing	TBD	Total dynamic pricing revenue
Occupancy rate	70–85%	Optimal range
Peak-time utilization	> 80%	Effectiveness during 10-15, 18-21 hours

## 10.5 Operational Efficiency

Metric	Target	Notes
Cost per prediction	TBD	Infrastructure + maintenance cost
Sensor uptime	> 98%	IoT device reliability
Model training time	< 1 hour	Retraining cycle efficiency

# 11. Project Structure

Suggested repository structure for this project:

```
smart-car-parking/  
├── data/  
│   ├── raw/  
│   │   ├── groundtruth.csv  
│   │   ├── weatherfeatures.csv  
│   │   └── roadfeatures.csv  
│   └── processed/  
│       └── combined_dataset.csv  
├── notebooks/  
└── smart_car_parking_analysis.ipynb
```

```
├── scripts/
│   ├── preprocess.py
│   ├── train_model.py
│   └── inference.py
├── models/
│   └── random_forest_model.pkl
├── reports/
│   ├── case_study_smart_car_parking.pdf
│   └── slides_smart_parking.pdf
├── requirements.txt
└── README.md
```

Adjust filenames and paths to match your actual repository structure.

## 12. Installation & Usage

### Requirements

- **Python:** 3.8+
- **Environment:** venv or conda recommended

### Install Dependencies

*# Clone the repository*

```
git clone <your-repo-url>
cd smart-car-parking
```

*# Create virtual environment*

```
python -m venv .venv
```

*# Activate virtual environment*

```
source .venv/bin/activate    # Linux/Mac
```

*# or*

```
.venv\Scripts\activate      # Windows
```

*# Install dependencies*

```
pip install -r requirements.txt
```

### Example requirements.txt

```
pandas>=1.3.0
numpy>=1.21.0
scikit-learn>=1.0.0
matplotlib>=3.4.0
seaborn>=0.11.0
jupyter>=1.0.0
```

### Run the Notebook

```
jupyter notebook notebooks/smart_car_parking_analysis.ipynb
```

Follow the notebook for: - Data loading and merging - Exploratory data analysis - Model training and evaluation - Results visualization

### Technologies

- **Language:** Python 3.8+
- **Data Processing:** pandas, numpy
- **Machine Learning:** scikit-learn (Random Forest Regressor)
- **Visualization:** matplotlib, seaborn
- **Notebooks:** Jupyter

### Optional / Future

- **Containerization:** Docker
- **Cloud:** AWS (SageMaker, S3)
- **APIs:** Flask or FastAPI for model serving
- **Orchestration:** Apache Airflow or Prefect

## 13. Future Developments

Contributions are welcome! Areas for enhancement include:

- Additional feature engineering and model experimentation
- Integration with real-time data sources (traffic, events, weather APIs)
- Deployment pipelines (API development, containerization, cloud setup)
- Interactive dashboards for operators and city planners
- Documentation improvements and tutorials
- Model interpretability (SHAP, LIME analyses)

## 14. Summary

This smart parking prediction system demonstrates the full ML lifecycle:

- **Business understanding:** Identified user pain points and opportunities
- **Data exploration:** Analyzed temporal, environmental, and location patterns
- **Data preprocessing:** Handled missing values, outliers, and quality issues
- **Modeling:** Selected Random Forest with strategic feature engineering
- **Evaluation:** Achieved 91.6%  $R^2$  with low error rates
- **Deployment strategy:** Defined KPIs and monitoring for production use

The system is ready for real-world deployment with clear paths for continuous improvement through additional data sources, algorithm enhancements, and operational feedback loops.