International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Hyperlocal Demand Forecasting and Dynamic Pricing for Commercial Spaces with LLM-Based Listing Generation

1st Somasekhar T
Associate Professor, Dept. of Computer Science and Engineering
K S Institute of Technology Bengaluru, India
somasekhart@ksit.edu.in

2nd Aditi S H
Dept. of Computer Science and Engineering
K S Institute of Technology
Bengaluru, India
aditishetty.1111@gmail.com

3rd Bhavani S
Dept. of Computer Science and Engineering
K S Institute of Technology
Bengaluru, India
bhavanishivu245@gmail.com

4th Bhoomika M H
Dept. of Computer Science and Engineering
K S Institute of Technology
Bengaluru, India
bhoomikamh24@gmail.com

5th Brinda G
Dept. of Computer Science and Engineering
K S Institute of Technology
Bengaluru, India
brindag160@gmail.com

***Abstract*— Commercial-space marketplaces depend on knowing how likely a listing is to be occupied and on presenting it to the right customers, yet a new platform starts without a long booking history. This paper presents SeekMySpace, a marketplace intelligence platform for commercial spaces in Bengaluru that addresses this cold-start problem with a source-tagged dataset and two AI modules. The forecasting module is a Random Forest regressor that predicts listing occupancy from nine features covering price, competitor price, footfall, booking count, location, category, events, holidays and customer segment. It is built on 199 records, 99 manually collected and 100 synthetic, of which 197 were usable, and achieves a mean absolute error of 0.0310 and an R^2 of 0.8194 on 40 held-out records. Each prediction reaches the owner with a demand score, a demand level and a suggested price. A supporting language-model module generates segment-specific, keyword-rich listing descriptions that remain tailored across property categories and customer segments. Both modules run behind a FastAPI service used by a React front end. By labelling every record as real, verified or synthetic, the approach gives early-stage platforms a transparent and reproducible way to start forecasting demand before large booking histories exist.**

Keywords— commercial real estate, demand forecasting, occupancy prediction, Random Forest, synthetic data, listing generation, search engine optimization.

I. INTRODUCTION

Commercial-space marketplaces list offices, coworking desks, event halls, warehouses, studios and retail units for short- and long-term use. Owners on these platforms face two practical questions. How likely is a listing to be occupied at a given price, and will the right customers find it when they search? Large platforms can answer both from years of booking records. A new platform cannot. SeekMySpace, an early-stage marketplace for commercial spaces in Bengaluru, began with a catalogue of listings and no long history of bookings to learn from.

Most published work on rental analytics deals with housing. Our earlier survey reviewed predictive analytics and generative AI for commercial space marketplaces but did not report a working system [1]. Related studies describe the move from brokers to online portals [2], apply ensemble models to apartment and house rents [3], [4], and combine recommendation, fraud detection and rent estimation for residential rentals [5]. On the language side, Shareef adapts large language models to customer support [6].

These studies leave two gaps for commercial spaces. First, they predict rent or price from structural attributes, while the owner of an event hall or a warehouse also needs to know how likely the space is to be occupied given its price, nearby competitor prices, footfall, local events and holidays. Second, they say little about how to build a usable dataset when real records are scarce, which is the normal situation for a startup.

In this paper we describe what we built with the data we could get. Occupancy forecasting is the main focus. We assembled a Bengaluru catalogue of 500 listings, with 210 records from real sources, 19 verified business records and 271 synthetic prototype records, each labelled by origin. For forecasting we prepared a separate dataset of 199 records, 99 collected manually and 100 synthetic, of which 197 were usable after cleaning. A Random Forest regressor [7] trained on nine features predicts occupancy rate, and a supporting language-model module writes keyword-rich descriptions for different customer segments.

Our contributions are as follows. First, we show how a labelled mix of real, verified and synthetic records can be used to bootstrap a commercial-space dataset when booking history is thin. Second, we report a Random Forest occupancy model with a mean absolute error of 0.0310, a root-mean-square error of 0.0413 and an R^2 0.8194 of on 40 held-out records. Third, we describe a working listing-generation module that we checked qualitatively; we have not yet measured its effect on search

ranking or clicks. Because about half of the forecasting records are synthetic, we present the results as early evidence and discuss this limit in Section VII.

The rest of the paper is organized as follows. Section II reviews related work, Section III states the problem, Section IV describes the datasets, Section V presents the methodology, Section VI discusses the results, Section VII covers limitations and future work, and Section VIII concludes.

II. Literature Review

A. Rental Platforms and Price Prediction

Rental analytics has mostly been studied for housing. The review by Singh et al. follows how flat and apartment rentals moved from brokers to web portals and mobile apps, and how machine learning is now applied to property search, tenant screening and lease management [2]. It also notes open problems such as data privacy, lagging rules and uneven online coverage [2].

For price prediction, Shanthi et al. compared ensemble machine learning models for forecasting rental apartment prices in an Indian metro city [3], and Singh et al. combined Random Forest and XGBoost to predict house rents from structural attributes such as location and size [4]. Both studies deal with residential rent. Random Forests suit this kind of tabular data because they handle a mix of numeric and categorical features without scaling [7].

SmartRent is the closest full-platform example [5]. It combines a hybrid recommendation engine, a multilingual chatbot, a fraud detector, a stacked rent-estimation model and blockchain contracts for residential rentals. Its authors report an R^2 of 0.87 and a user study with 200 landlords and tenants, and note that part of their data was synthetic [5]. Early-stage platforms thus often start from mixed real and synthetic data, although SmartRent targets homes and estimates rent rather than occupancy.

B. Language Models for Customer-Facing Tasks

Shareef adapted a large language model to customer support, training on the public Customer Support on Twitter dataset [6]. That work is about conversation quality, not marketing text, but it shows that language models can be tailored to a narrow business setting. Our earlier survey discussed how such models and predictive analytics might serve commercial-space marketplaces, without implementing a system [1].

C. Research Gaps

Taken together, this work leaves gaps for commercial spaces. The forecasting studies are residential and estimate price or rent from structural attributes; none forecasts occupancy for multi-category commercial spaces. Few describe how to build a dataset for a platform with little booking history, or say plainly how much of it is real and how much synthetic. Listing text for different customer segments is also rarely studied next to a demand model. This paper addresses these gaps in a small but transparent way.

III. Problem Statement and Objectives

A. Problem Context

Commercial spaces behave differently from homes. An event hall may be full at weekends and empty on weekdays, a coworking floor depends on the offices around it, and a warehouse is shaped by logistics and local industry. Demand for all of them moves with price, competitor offers, footfall, local events and holidays. A new platform such as SeekMySpace has listings, owners and customers but only a thin booking record, so it must learn from whatever it can collect and from synthetic records made to fill the gaps.

B. Problem Statement

We study two connected tasks. The first and main task is occupancy forecasting: given a commercial-space listing described by its monthly price, the average competitor price, footfall, booking count, location, category, whether an event or holiday coincides with the observation, and the customer segment, predict the occupancy rate of the listing on a scale from 0 to 1. The second, supporting task is listing generation: given a listing's category, location, amenities and target customer segment, write a description that fits that segment and carries relevant search keywords.

Four difficulties make this hard. Data are scarce. Sources are mixed, so real, verified and synthetic records must not be confused. Listings span very different categories, so one pattern does not fit all. And generated text must stay faithful to what a listing offers, because an invented amenity misleads the customer.

C. Research Questions

1) How well can the occupancy rate of a commercial-space listing be predicted from price, competitor, footfall, event and listing features when the training data are few and partly synthetic?

2) How can a catalogue that mixes real, verified and synthetic records be built and documented so that the origin of every record stays traceable?

3) Can a language-model module produce customer-segment-specific, keyword-rich descriptions that stay consistent with a listing's recorded details?

D. Objectives

1) Dataset construction: build a labelled, multi-category commercial-space catalogue for Bengaluru from real, verified and synthetic records, with a field that records the origin of each one.

2) Occupancy forecasting: develop and evaluate a regression model that predicts occupancy rate from the nine input features above, and report its error on a held-out test set using mean absolute error, root-mean-square error and R^2.

3) Listing generation: implement a language-model module that turns a listing's details and a target segment into an SEO-friendly title, description, keywords, highlights and a call to action, and review the output for segment fit and keyword use.

4) Integration: expose both services through one API so that the marketplace front end can request a prediction or a description for any listing.

5) Transparent reporting: state clearly how much of the data is real and how much is synthetic, so that readers can judge how far the results carry over to live conditions.

IV. Data Collection and Dataset Construction

A. Keeping the Origin of Every Record

SeekMySpace has no booking log of any real length, so its datasets were put together from several sources. Instead of merging them into one table, we kept the origin of each record visible. Every catalogue entry carries a data_source field and a data_status field that say whether the record was collected from a real source, verified as an actual business or place, or created synthetically for the prototype. Any figure in this paper can therefore be traced to the kind of data behind it.

B. Marketplace Catalogue

The catalogue holds 500 Bengaluru listings: 210 collected from real sources, 19 verified business or place records, and 271 synthetic prototype records added to widen the range of listings. Fig. 1 shows the split. Each listing has 30 attributes covering identity, location and access, commercial details (pricing, area, capacity, availability, booking options), amenities, online presence, images, contact and description text, reviews, and the two provenance fields. Categories range from coworking desks and offices to warehouses, studios, event venues, retail units, clinics and laboratories. The catalogue serves marketplace browsing and the listing generation module; it is not the training set for the forecasting model.

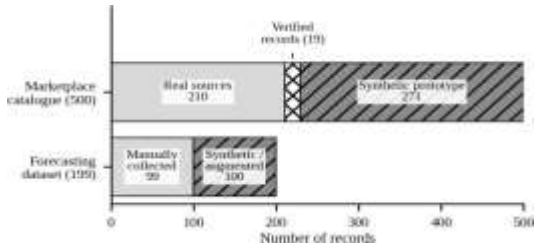


Fig. 1. Origin of the records in the marketplace catalogue and the forecasting dataset.

C. Forecasting Dataset

The forecasting model uses a separate table of 199 records and 29 columns. It combines 99 manually collected records with 100 synthetic ones and adds the demand variables the catalogue does not have: price, competitor price, footfall, booking count, occupancy rate, events, holidays and customer segment. The synthetic records are augmented entries that extend the manually collected ones. Two records had no usable occupancy rate and were removed, which leaves 197. We split these once into 157 training and 40 test records and kept that split fixed for all experiments.

Nine columns serve as inputs, and occupancy rate, a value between 0 and 1, is the target. Table I lists them. Numeric inputs are used as they are and categorical inputs are encoded before training.

TABLE I. Model Inputs and Target

Feature	Type and meaning
monthly_price_inr	Numeric: monthly listing price in rupees
competitor_avg_price_inr	Numeric: average price of competing listings
footfall	Numeric: footfall level around the listing
booking_count	Numeric: number of previous bookings recorded for the listing
location	Categorical: area of Bengaluru
category	Categorical: type of commercial space
event	Categorical: whether an event coincides with the observation
holiday	Categorical: whether a holiday coincides with the observation
customer_segment	Categorical: customer group the listing is aimed
occupancy_rate	Target: occupancy of the listing, from 0 to 1

V. Proposed Methodology

A. System Overview

Fig. 2 shows how the parts fit together. The user works in a React and TypeScript front end styled with TailwindCSS, which calls a FastAPI REST layer served by Uvicorn. The layer sends each request to one of two services. The demand forecasting service runs a Random Forest regressor on engineered demand features held in a CSV dataset. The listing generation service uses a language model to write SEO content from listing attributes. Structured listing data is stored in MongoDB, which feeds the marketplace interface. Table II lists the three endpoints.

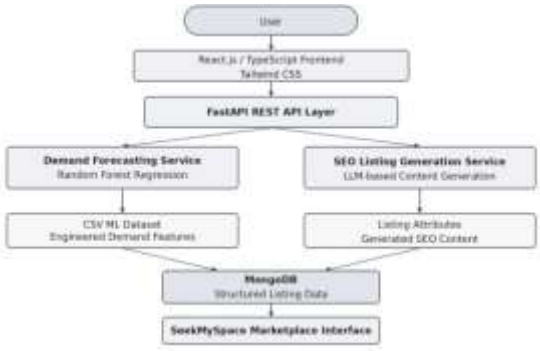


Fig. 2. SeekMySpace system architecture with the two services and their data.

TABLE II. API Endpoints

Endpoint	Input	Output
POST /api/predict	Location, category, capacity, average price, rating and prediction date	Predicted bookings and occupancy, demand score and level, suggested price
POST /generate-prompt	listing_id, target_audience	Assembled prompt text
POST /generate-description	listing_id, target_audience	SEO title, description, keywords, highlights and call to action (JSON)

B. Occupancy Forecasting Module

1) Prediction Flow and Event Context

Fig. 3(a) shows the flow of a prediction request. The user selects a property and enters the location, category, capacity,

average price, rating and the date to predict. The backend then retrieves what it holds for that property and place: previous bookings, demand patterns, seasonal trends and property characteristics. Demand for a coworking floor or an event hall tends to rise when something is happening nearby, so the module also collects live event information. It queries Google Events through SerpAPI for the same location, with the API key kept in the backend environment. A function, get_events_for_ date(check_date, days=7), returns the events for a seven-day window that starts on the selected date. An event with date t_e is kept when (1) holds, where d is the selected date.

$$d \leq t_e < d + 7\ days \tag{1}$$

For a request on 10 November, events from 10 to 16 November are kept, while those on 9 and 17 November are not. The kept events are then filtered by date, checked against the listing's location and category, and given an expected impact on demand, which the interface shows as High or Medium. Event data is kept in data/events.csv. These event-related features, together with the property features and the stored booking patterns, are the inputs of the model.

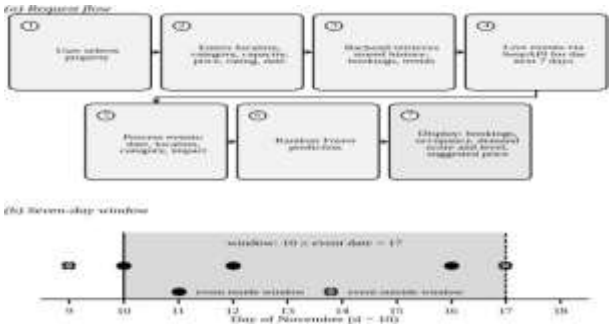


Fig. 3. Forecasting request flow and the seven-day event window.

The quantity the module predicts is the occupancy rate of a listing, defined as the share of its available spaces that are occupied, as given in (2).

$$Occupancy\ Rate = \frac{Occupied\ \ Spaces}{Total\ Available\ Spaces} \times 100 \tag{2}$$

2) Random Forest Model

Occupancy prediction is treated as a regression problem. We use the Random Forest regressor from scikit-learn [8], which averages the predictions of B decision trees, each trained on a bootstrap sample of the training data [7]:

$$\hat{y}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{3}$$

where T_b is the b-th tree and x is the vector of nine input features. We chose this model for practical reasons. It accepts a mix of numeric and categorical inputs without feature scaling, it copes reasonably with the noise that synthetic records bring in, and tree ensembles have worked well on tabular rent data [4]. The trained model is saved with joblib, so the API can load it without retraining.

3) Decision-Support Outputs

The prediction is not shown to the owner as a bare number. The interface returns predicted bookings, predicted occupancy, a demand score, a demand level such as High, a suggested price, upcoming events and short insights. These indicators are derived from the occupancy prediction by the platform's business logic; only the occupancy prediction itself is evaluated in this paper.

4) Evaluation Measures

We score the model on the 40 held-out records with three standard measures. Let y_i be the true occupancy rate, $\hat{y}_i$ the predicted one, $\bar{y}$ the mean of the true values and n the number of test records.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6}$$

MAE and RMSE are in the same units as occupancy rate, so smaller is better. R^2 compares the model with a baseline that always predicts the mean, and values closer to 1 are better.

C. Listing Generation Module

The second module writes listing content for a chosen audience. A request carries two values: a listing_id and a target_audience, for example Freelancers or IT Companies. As Fig. 4 shows, the service fetches the property record from properties.csv and builds a prompt that combines property details, the audience and the factuality instructions. The prompt gives the model a role, which is a marketing assistant for SeekMySpace,the recorded facts of the property (name, category, location, area, capacity, pricing, amenities and so on), and a fixed task: an SEO-friendly title, a personalised description, five SEO keywords, five highlights and a call to action. The text is generated by a language model reached through OpenRouter.



Fig. 4. Listing generation workflow with parsing and validation.

Most of the prompt is rules, because the main risk with generated listing text is invented detail. The model may use only the facts in the property record and must not invent, assume, exaggerate or change any of them, or add adjectives that alter the source: if the record says Wi-Fi, the text says Wi-Fi, not high-speed Wi-Fi. A missing detail is left out, not guessed, and the model may not create contact numbers, URLs, booking links or guarantees. The audience can change the perspective of the text but never the facts underneath. The model must return only valid JSON in a fixed structure, and the /generate-prompt endpoint returns the assembled prompt so that we can inspect exactly what the model is asked.

The reply then goes through a parse-and-validate step. The service converts it to JSON and checks its structure and key claims against the property data. If validation passes, the content is marked as valid and ready for use. If it does not, the content is still returned but carries warnings about unsupported or uncertain claims, so the owner can review it.

D. *Implementation*

The backend is written in Python with FastAPI, Pandas, NumPy and scikit-learn, and the front end uses React.js with TypeScript and TailwindCSS. Model data live in CSV files and listing data in MongoDB. Events come from SerpAPI and listing text from a language model reached through OpenRouter. The endpoints were tested through the Swagger interface that FastAPI generates. Fig. 5 shows the property selection screen and Fig. 6 the resulting forecast with local events. Fig. 7 shows the sign-in screen of the platform and Fig. 8 the detail page of a listing.

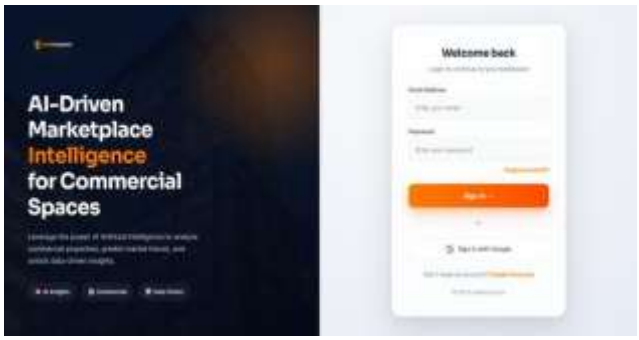


Fig. 5. Login screen of the SeekMySpace platform.



Fig. 6. Property detail page of a coworking listing (BLR-0002).



Fig. 7. Property selection screen of the demand prediction tool.



Fig. 8. Forecast for a selected date with demand score and local events.

VI. Results and Discussion

A. *Occupancy Forecasting*

Table III reports the three measures on the 40 test records. Because occupancy rate runs from 0 to 1, an MAE of 0.0310 means the predictions are off by about 3.10 percentage points on average. An R^2 of 0.8194 means the model accounts for about 81.94% of the variation in occupancy across the test listings. RMSE (0.0413) is never smaller than MAE because it weights large misses more heavily, so the gap between the two shows how much of the error comes from a few listings.

TABLE III. Forecasting Results on the 40 Test Records

Measure	Value
MAE	0.0310
RMSE	0.0413
R^2	0.8194

These figures come from a small test set drawn from data that is about half synthetic. We read them as evidence that the pipeline learns a usable signal from limited data, not as a forecast of live accuracy. At this stage the output suits decision support, for example flagging listings that look underpriced or under-promoted, and it should not change prices automatically.

B. *Listing Generation*

We tested the module through its Swagger interface on listing BLR-0001, KOLAB Coworking Spaces in Jayanagar. For the audience IT Companies, the service returned a title, a two-sentence description, five keywords, five highlights and a call to action. The title reads “KOLAB Coworking Spaces Jayanagar Bengaluru - 1400 sq.ft, 46 Capacity, ₹4450” and the call to action reads “Enquire now for daily or monthly booking”.

We then compared every fact in the output with the source record, as Table IV shows. Each one matched, and nothing was added. This is a single worked example, not a statistical evaluation.

TABLE IV. FACT CHECK OF ONE GENERATED LISTING (BLR-0001)

Attribute	Source record	Generated text
Name, place	KOLAB Coworking Spaces, Jayanagar	Same
Area	1400.0 sq.ft	1400 sq.ft
Capacity	46.0	46
Price	₹4450	₹4450
Amenities	Wi-Fi, Power backup, Meeting rooms, Reception, CCTV	Same five, wording unchanged
Nearby metro	Jayanagar, 800 m	800 m from Jayanagar metro station
Booking	Daily / Monthly	Daily and monthly

Two things stand out. First, the grounding rules worked: the wording stays close to the record, and Wi-Fi and meeting rooms appear exactly as recorded. Second, that fidelity has a price. The IT Companies description is mostly a plain restatement of the facts, and the audience shows up in what is emphasised, namely capacity, location and booking flexibility, rather than in persuasive language. Loosening the rules would add flair but bring back the risk of invented claims, so the balance is a design choice that an owner review step can adjust. For the Freelancers audience, the prompt returned by /generate-prompt carries the same facts and rules with only the audience changed.

C. Discussion

Building with scarce data taught us three things. Provenance tags cost almost nothing and make every later number easier to defend, so we would add them from the first day of any new marketplace. Event context is cheap to obtain and gives the model a view that a booking-only approach lacks, although we have not yet measured how much it moves the predictions. And grounded prompts trade style for safety, with validation flagging claims it cannot support, which is usually the right trade where a wrong amenity or price can mislead a customer.

VII. LIMITATIONS AND FUTURE WORK

A. Limitations

Several limits affect how far the results can be trusted. About half of the forecasting records are synthetic, the test set holds only 40 records, and everything is scoped to Bengaluru, so the metrics may not carry over to live conditions. We did not compare the Random Forest with other models, so we cannot say how it stands against a simpler baseline. The event module is described as designed: we have not measured how often live SerpAPI requests succeed or how much event information changes a prediction. The listing module was checked by hand on one listing, and its warning rate and effect on search ranking or click-through are untested.

B. Future Work

The next steps follow from these limits. We plan to replace synthetic records with real booking and pricing data as the platform collects it, extend the dataset to other cities, and compare the Random Forest with a linear baseline and gradient boosting, with feature importance and an ablation that removes the event inputs. For listing generation, we plan a larger test of the validation step across many listings and audiences, a controlled before-and-after test of search visibility, and a safe fallback for when the event service is unavailable.

VIII. CONCLUSION

This paper described how SeekMySpace approached occupancy forecasting and listing generation for Bengaluru's commercial spaces without a long booking history. The datasets keep every record labelled as real, verified or synthetic. A Random Forest predicts occupancy from nine features, with event information gathered for a seven-day window, and it reached an MAE of 0.0310,an RMSE of 0.0413 and an R^2 of 0.8194on 40 held-out records. A grounded prompt design lets a language model write audience-specific listing content without adding facts, as a worked example showed. The results are early and rest on partly synthetic data, but the approach offers a practical and transparent starting point for marketplaces that have listings before they have history.

REFERENCES

[1] T. Somasekhar, S. H. Aditi, S. Bhavani, M. H. Bhoomika, and G. Brinda, "AI-driven marketplace intelligence for commercial spaces," *Int. J. Research and Scientific Innovation (IJRSI)*, vol. 13, no. 5, pp. 809–814, May 2026, doi: 10.51244/IJRSI.2026.1305000076.

[2] N. T. Singh, M. Wadhwa, S. Kumar, V. M. Shrimal, U. Shrivastava, and Himanshu, "A study of the evolution of the real-estate flats and apartment rental system," in *Proc. 5th Int. Conf. Smart Electronics and Communication (ICOSEC)*, 2024, pp. 1846–1851, doi: 10.1109/ICOSEC61587.2024.10722232.

[3] J. M. Shanthi, P. S. Rao, V. R. Kumar, G. Nithya, J. Kartigeyan, and A. Gupta, "Ensemble machine learning models used for forecasting in rental apartment price prediction of metro city in India," in *Proc. 6th Int. Conf. Emerging Technology (INCET)*, 2025, doi: 10.1109/INCET64471.2025.11140248.

[4] H. V. Singh, S. K. Srivastava, N. K. Pandey, and N. Kumar, "Integrating Random Forest and XGBoost for accurate house rent prediction," in *Proc. Int. Conf. Pervasive Computational Technologies (ICPCT)*, 2025, pp. 956–957.

[5] L. Pallavi, D. Rishika, G. B. S. Sree, J. Ruthwik, and J. Yashaswini, "SmartRent: AI-driven online house rental platform," in *Proc. 5th Int. Conf. Soft Computing for Security Applications (ICSCSA)*, 2025, pp. 346–353, doi: 10.1109/ICSCSA66339.2025.11170770.

[6] F. Shareef, "Enhancing conversational AI with LLMs for customer support automation," in *Proc. 2nd Int. Conf. Self Sustainable Artificial Intelligence Systems (ICSSAS)*, 2024, pp. 239–244, doi: 10.1109/ICSSAS64001.2024.10760403.

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.