Occupancy Prediction Using Building Operation Data

Phu Ngoc Dang

[a] *University of California San Diego*
[b] *Halicioglu Data Science Institute*
[c] *UC San Diego Urban Studies & Planning*

Abstract

The ability to forecast occupancy holds tremendous potential for resource-use efficiency and lowering O&M costs for buildings. We found the amount of indoor CO2 and number of WiFi-connected devices as informative predictors of occupancy with Linear Regression + RBFs delivering the best performance.

1. Introduction

This time-series predictive analytic project was made for Professor Julian McAuley at the UC San Diego Jacobs School of Engineering. The main goal is to discover variable(s) relevant to predicting building occupancy using data from a Library Space in the School of Design and Environment 4 (SDE4) building located at the National University of Singapore.

The dataset (Tekler et al., 2022) contains empirical sensor data readings of building operation metrics across 47 days, in 5-minute intervals, comprising 13,536 data points. Details and variable descriptions can be accessed at the cited source above in the *README.md* file.

2. Exploratory analysis

The dataset is comprised of three primary components: 1) timestamp, 2) occupant count, and 3) 33 building operation metrics (e.g., VOC [ppb], sound pressure [dba], air temperature [celsius], indoor CO2 [ppm], air handling unit fan speed [Hz], cooling coil valve position [%], etc.). Our target variable for prediction will be occupant count, which serves as the driver behind this exploratory analysis to discover relevant factors that seem to influence, or relate to, occupancy.

A heat map of Pearson's correlation coefficients (PCC) (Figure 1) was used to holistically identify any moderate to strong associations between the variables, with particular emphasis on the target variable. A quick glance, plus some inspections, shows the following variables as potential candidates with strong associations with occupancy, defined as:

$$|PCC| > 0.55$$

The variables are occupant presence (binary), WiFi-connected devices (number), plug load energy (kWh), supply air pressure (Pa), air handling unit fan speed (Hz), cooling coil valve command (%), cooling coil valve position (%), lighting energy (kWh), filter pressure (Pa), off-coil air temperature (Celsius), and supply air temperature (Celsius).

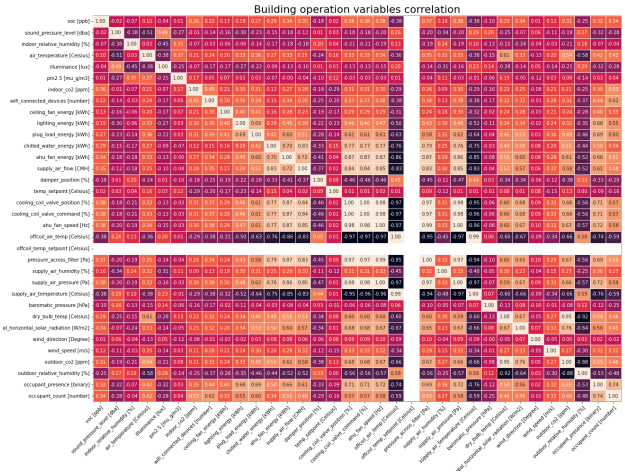

Figure 1: Variable Correlations Heat map

Note: From here, mentions of "variables" refer to the above 11 variables.

In the real world, some of the mentioned variables are more intuitive in their relation to occupancy than others, such as lighting or plug load energy. When visualizing the variables against occupancy across three different plot types (line, scatter, and kernel density estimate), the plots seldom show any clear associative pattern. Below are the observations:

1. The majority of observations appear to cluster at zero occupant count.
2. There are often two clusters in a plot that "force" a seemingly strong correlation when there is not.

Since the sensor data readings are continuous in 5-minute intervals across *days*, it makes sense for the majority of data points to have zero occupant count, which could "force" a strong correlation as long as there is *any* second cluster elsewhere in the plot. This is shown in the top plot of Figure 2, where the supply air pressure variable is mostly zero when occupancy is zero. When there are occupants, the system will supply approx. 125 Pascal of air, regardless of the specific occupant count; it is here that we observe a second, horizontal cluster at 125 Pascal. This behavior is attributable to the context of building operations where many processes' input/load is approximately constant. When combined with the mentioned cluster near the origin, a moderate correlation appears mathematically while there is not one, both visually and contextually. The bottom plot of Figure 2 illustrates the severe value imbalance in our target variable.

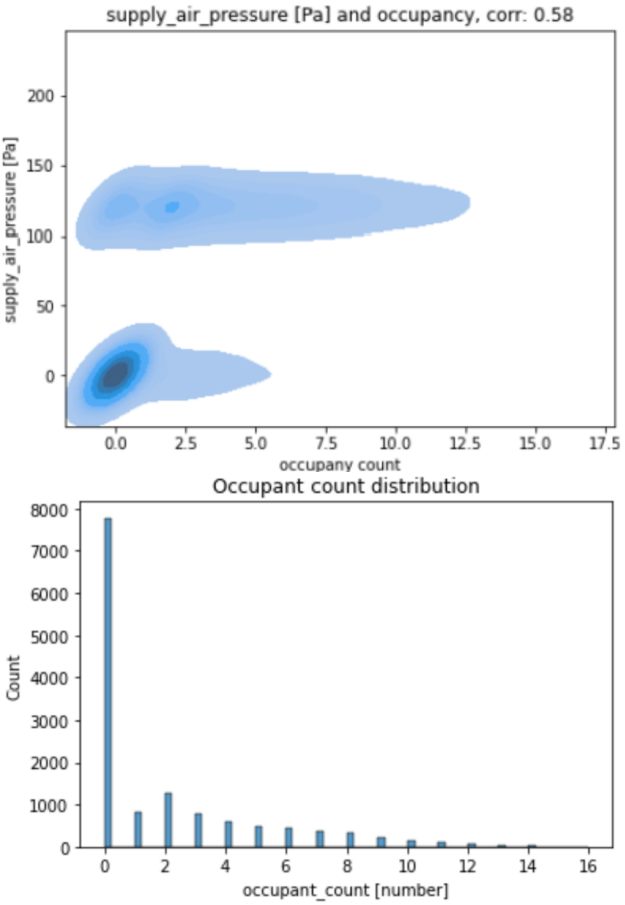

Figure 2: Invalid correlation and value imbalance

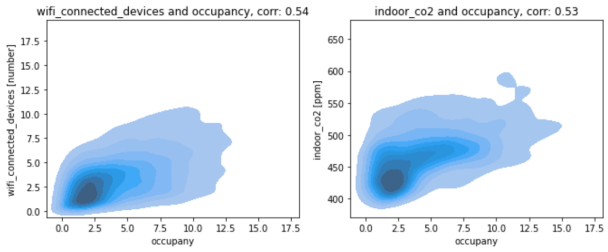

Figure 3: Kde plots of final two relevant variables

Solution & Practical use considerations: Data points with zero occupant count can be entirely excluded for two primary reasons. First, the validity of the predictions from our models will be jeopardized when there is a severe class imbalance, that is we do not want our models to predict zero all the time and still receive some promising performance evaluation. Second, the purposes and potential applications of our project in building management, fortunately, permit this exclusion as only predictions during times with *any* non-zero occupancy matter, whether the goal is energy demand forecasting, load allocation for increased efficiency, etc. because there are few worries about efficiency when occupancy is zero, or our building performance optimization problem becomes simpler (e.g., any energy-consuming devices not turned off, any occupancy-independent processes consuming too much energy, etc.).

With the zero-occupancy data points excluded and the same association analysis conducted on the new subset of data using a slightly lower threshold (0.5), as assessed, the majority of variables turned out to have no association with our target variable. Our list of potentially relevant variables narrowed down to only the number of WiFi-connected devices, and the amount of indoor CO2. In Figure 3, our data points are still clustered at lower occupancy counts, however, our correlations are now "valid" and definitively accurate. Figure 4 illustrates how well the two

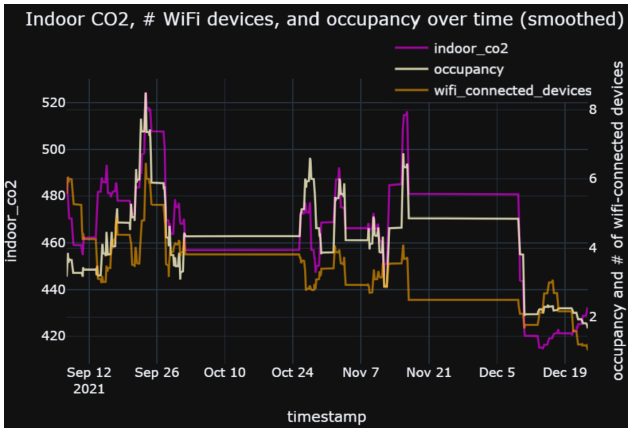

Figure 4: Time-series of final two relevant variables

variables' trends "follow" our target variable over time, as rolling averages across a 31-hour window. Interestingly, the indoor CO2 variable did not make our initial list of 11 variables.

Note: From here, mentions of "variables" refer to the above 2 variables.

3. Predictive analysis

3.1. Predictive task & Learning features choices

Our predictive task is to predict occupancy using the identified correlated features, indoor CO2 and # WiFi-connected devices, plus an encoding of timestamps to capture the temporal nature of our features. We will use the *root mean squared error* (*RMSE*) to evaluate performance across three model types: linear regression (baseline), auto-regression, and recurrent neural network. The LR models will be implemented across three temporal transformation techniques, whereas the AR and RNN models will not see any features, totaling five models.

3.2. Time-series feature engineering

To encode time, we consider *three* methods to transform timestamps as follows:

1. One-hot encoding
2. Sine / cosine transformations, and
3. Radial basis functions

With the 2nd and 3rd methods, our goal is to experiment with parametric functions to capture the cyclical nature of timestamps. We will first perform the encoding(s) at three tiers of time: month, day, and hour. To ensure their correct usage (Professor McAuley's notes), we will assess whether the resulting encoded values plot out their intended graph (e.g., smooth sine/cosine waves), in accordance with the original data.

3.2.1. *One-hot encoding*

We use dummy variables to represent each discrete value of time with one level dropped to prevent *multicollinearity*, where a value of 1 indicates the corresponding original value and the rest zero.

3.2.2. *Sine / cosine transformations*

We use the following functions to cyclically encode time with sine / cosine transformation:

$$y = \sin\left(\frac{x}{p} \times 2 \times \pi\right) \quad \text{and} \quad y = \cos\left(\frac{x}{p} \times 2 \times \pi\right)$$

where x is the original value, p is the period corresponding to a time tier (e.g., p=12 for month), and y is the resulting encoding.

As shown in Figure 5, our transformation technique seems to only work well for the hour attribute as our values form smooth sine and cosine waves as intended, whereas the graphs for the month and day attributes are a bit jagged (step-like) (Lewinson, 2022). One explanation involves data availability. The dataset has abundant recordings at the hourly level due to the high granularity of our timestamps (5-minute intervals), which allows our transformer to capture all hours in the 24h cycle. Contrarily, our data points only span four months from Sep 2021 to

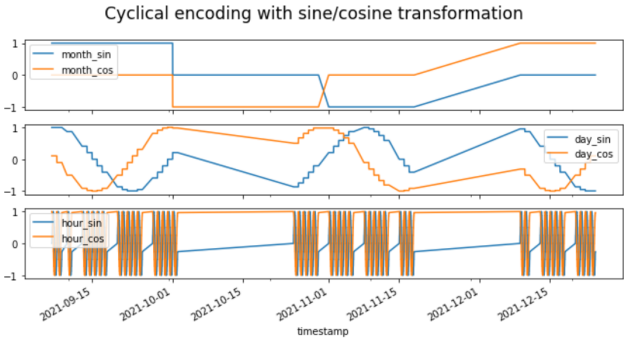

Figure 5: Encoding values visualized

Dec 2021, and every month has a decent amount of missing days, which rendered our transformers incapable of capturing all monthly and daily time frames. Hence, *we will proceed with only the hour attribute as a learning feature* to prioritize model simplicity and interpretability.

3.2.3. *Radial basis functions*

A detailed description of radial basis functions can be found in the following cited source (Wikipedia, 2023). Briefly, RBFs model the relative distances of each data point to a center value. Since one RBF equally spans the input data and repeatedly measures distances to instances of its respective "center" value, they are able to capture cyclicity. The number of RBFs for our transformer is a hyperparameter that can be chosen on a basis of interpretability, such as 12 RBFs to model days in a year where each function represents a month (Lewinson, 2022).

For comparability with the other two transformation techniques in later sections, we will only apply RBF transformation to the hour attribute. Using the RepeatingBasisFunction class from scikit-lego, our RBF transformer has four functions ($n\_periods = 4$) and an input range between 1 to 24 ($input\_range$) representing hours.

Figure 6 visualizes the encoded values of our four

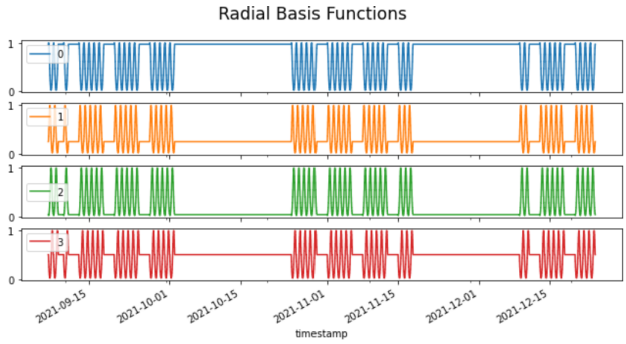

Figure 6: Radial basis functions

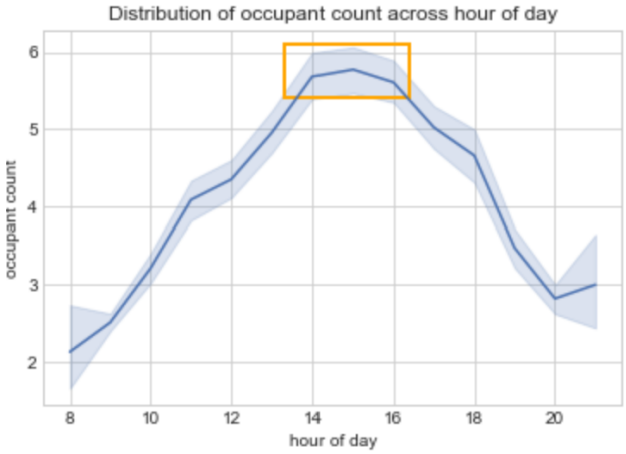

Figure 7: Occupant count distribution

RBFs. With four functions, we have four different centers representing four different times (approx. equally spaced) in a day. After inspecting the encoded values, function 0 measures distance from 1:30 AM of a day, function 1 measures from 7:30 AM, function 2 measures from 1 PM, and function 3 measures from 6:30 PM. Graphically, each function peaks at its respective "center" of every day, and decreases symmetrically as we move away from the center (Lewinson, 2022). A sanity check can be done by manually counting the number of peaks for each function, which equals 47, the number of days across which the data points were recorded.

3.2.4. Standardization

Since the attributes amount of indoor CO2 and number of WiFi-connected devices have different scales in their original values, we will standardize the two attributes by subtracting the mean from each value and dividing by the standard deviation. This transformation will allow our learning features to be in the same form ($\mu = 0, \sigma = 1$) and prevent any feature from being dominant or undermined.

4. Model & Performance evaluation

As noted, our model choices are linear regression, auto-regression, and a recurrent neural network.

Model	Train RMSE	Test RMSE
lin-reg + sin/cos	2.369	2.243
lin-reg + one-hot	2.350	2.233
lin-reg + rbf	2.355	2.226

Table 1: Linear regression models performance

4.1. Linear regression (LR)

With linear regression, the goal is to exploit the linear relationship each of our learning features has on occupancy (Figure 9). The modeling equation is as follows:

$$\hat{y} = bias + \sum_{t=1}^{\#coef.} value_t \times coef._t$$

Test Performance evaluation: (Table 1) Hour transformation using radial basis functions resulted in the best RMSE performance, followed by one-hot encoding, with sine/cosine transformations being the worst. Interestingly, with one-hot encoding, the hour indicators for hours between 1 PM to 4 PM have the highest predictive influence (highest coefficients); this observation matches the most popular

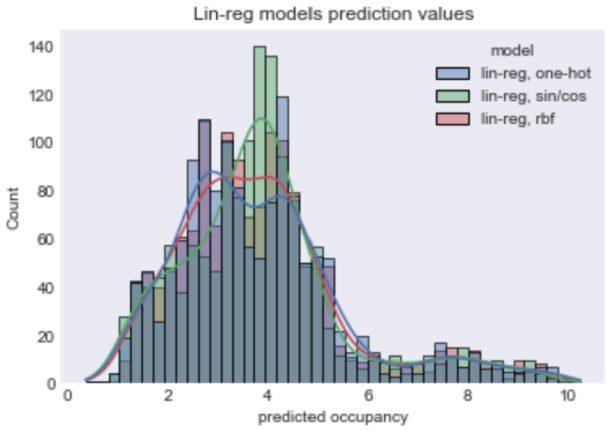

Figure 8: Lin-reg prediction values

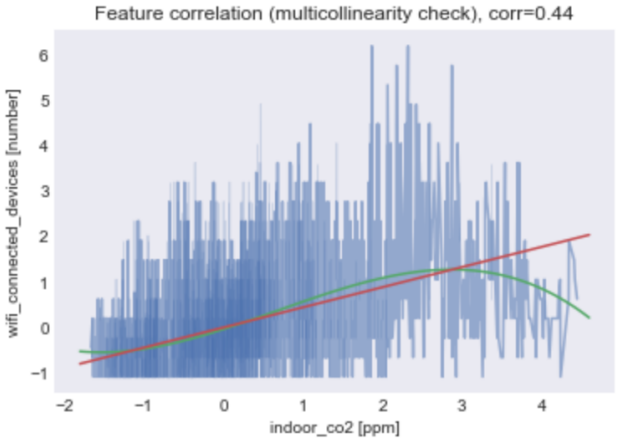

Figure 9: Features correlation

time window throughout the day (Figure 7). Similarly, with RBFs, the coefficients associated with functions 2 and 3 show the highest predictive influence. These functions measure distances from 1 PM and 6:30 PM (centers), respectively; this observation aligns with the daily most popular and high occupancy-shift times (Figure 7).

Validity check: Figure 8 shows the distributions of prediction values from our three linear regression models, which are relatively similar. The variety and range of values shown illustrate the models' general ability to capture the complexity of the data well; it is worth noting that the model with sine/cosine transformations appears to be much more clustered around 4 and has noticeably lower variety in prediction values, along with the worst performance in both the train and test RMSE. Additionally, to check for multicollinearity, Figure 9 shows the correlation between our two primary learning features, which is only 0.44 (linear).

For its interpretability and popularity, we will propose the best linear regression model (with RBF transformations) as the baseline.

4.2. Auto-regression (AR)

With auto-regression, we experiment with an alternative model architecture using previous values to predict the next value (no features are used). Our goal is to observe how such a model performs versus others (e.g., lin-reg, RNN) in the context of time-series. The modeling equation is as follows:

$$\hat{y}_x = bias + \sum_{t=1}^{T} value_{x-t} \times weight_t$$

where T represents the number of past values used for prediction. We used T=12 past values ($5 \times 12 = 60$ *minutes*), which equates to the previous hour of occupant counts used for prediction.

Test Performance evaluation: The AR model performed worse than any of the LR models, with a **Test RMSE = 3.093**. Different values of T were tested across a range between the previous 5 minutes to 5 hours of occupancy counts, which only resulted in small variations in performance. In the typical fashion of AR time-series models, the model's predictions were "unexciting" as they quickly converged to a mean prediction value, or constant for

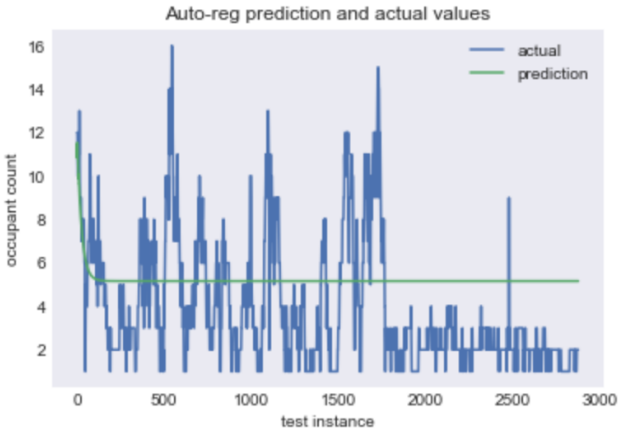

Figure 10: AR model prediction and real values

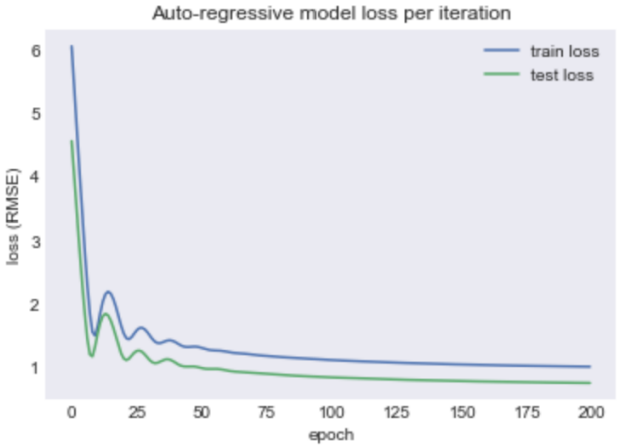

Figure 11: AR model losses per epoch

the majority of predictions (Figure 10). This shows the model is significantly less dynamic and is not as robust to changes in occupancy as the LR models. During training, a stopping tolerance was attempted as a convergence condition, which was unsuccessful as this resulted in strange behaviors towards the end of the train/test loss curves (a large loss drop). Figure 11 shows losses without a convergence condition with the expected steady decreases in loss per epoch.

4.3. Recurrent neural network (RNN)

With a recurrent neural network, the gist is similar to auto-regression in that we take advantage of the time-series structure of the data, where we make subsequent predictions using the immediate previous learning activity. That is, slightly unlike the traditional artificial neural network, the hidden feature (hidden state) depends on the previous hidden state, which is then used to calculate the target output. The modeling equations for our hidden and output layers are as follows:

$$h_t = W_{xh}^T \times x_t + W_{hh}^T \times h_{t-1} + bias_h$$

$$\hat{y}_t = W_o^T \times h_t + bias_o$$

where t is a time instance, x = input, h = hidden, o = output, xh = input-to-hidden, and hh = hidden-to-hidden (where we incorporate the previous hidden state to calculate a current hidden state). We used 5 hidden layers, 1 "recurrent" loop, and 12 previous occupancy counts for prediction (same as AR).

Test Performance evaluation: Comparable to the AR model, the Recurrent neural nets performed worse than any of the LR models and only slightly better than AR, with a **Test RMSE = 2.880**. Similar to AR, different ranges of past values used for prediction only resulted in slight performance fluctuations, including the convergence behavior in predicted values that renders the model essentially a constant predictor. The prediction against actual values graph and losses per iteration graph are similar to Figures 10 and 11, respectively. One probable cause of the RNN's poor performance relative to expectations, or the LR models, is *overparameterization*. In short, RNNs are likely to be overly complex for our predictive task as they possess levels of complexity and representational capacity to fit the data in many ways (Karlik et al., 2023), making our use case unreliable.

Model	Test RMSE
auto-reg	3.093
recurrent neural nets	2.880
lin-reg + sin/cos	2.243
lin-reg + one-hot	2.233
lin-reg + rbf	2.226

Table 2: All models performance

4.4. Overall

4.4.1. Comparisons, strengths, weaknesses

Table 2 shows the relative performance of our five models with linear regression using radial basis functions for hour encoding delivering the best performance; followed by the same model with one-hot encoding, sin/cos transformations, then the recurrent neural nets, and worst is auto-regression. The LR models are more intuitive and interpretable as they use features that are products of detailed association analysis; these features are comprised of diverse attributes that are relevant to occupancy, both by intuition and correlations. A possible weakness of the LR models is their limited capacity for hyperparameter tuning, if possible at all to any relevant extent. Contrarily, the AR and RNN models can use plentiful hyperparameter tuning work for further optimization, not to mention inspections of the gradients to remediate issues, such as diminishing gradients, despite an upfront poorer performance than LR.

4.4.2. Limitations

Fortunately, there were no issues of scalability, multicollinearity, or overfitting. One limitation of our model comparisons is an inconsistency in learning features. Given that the LR models used target-correlated features (informative) for prediction, whereas the AR and RNN models used 12 past occupancy counts for prediction (no features seen), this difference limits our ability to compare the models in greater detail.

4.4.3. Room for improvements

Our predictive task can be improved with a greater understanding of the schedule and operation at the subject library. Additional features can be incorporated to account for unique factors that may influence occupancy, such as special holidays or weather patterns in Singapore. As mentioned, finer model tuning can be done, especially for the AR and RNN models.

5. Literature

Some suggested uses for this dataset include "benchmarking and supporting data-driven approaches" in occupancy prediction, occupant behavior modeling, building simulation and control, energy forecasting, IoT, and construction management (Tekler et al., 2022). Although time-series forecasting is common in predictive analytics, its applications in real estate and building-related data are quite limited as the popular attention tends to be for stock/price forecasting. Amidst our digitizing world, the real estate industry is among the latest to begin digitizing with developments such as software modeling (e.g. Autodesk) and digital twins. Machine learning applications have gained popularity in building portfolio analytics and optimization (e.g. Introba). Research works have also emerged to study HVAC data, where a survey article emphasized the wealth of information from HVAC systems' manufacturing engineers for their products (Donlon, 2016), which can be incorporated to better design/tune models; other considerations include ASHRAE Standards and user preferences. Another article used deep reinforcement learning for building HVAC control that directly adjusts temperature ranges (Wei et al., 2017). Both articles demonstrated promising future growth for applying machine intelligence in building operations that are dynamic to varying conditions and user behaviors, versus rule-based approaches, which align with our narrative.

Unlike the supply-first approach to building operation optimization, our project tackles efficiency from a demand standpoint with occupancy prediction.

6. Results and conclusions

In general, our models seem to be fair predictors of occupancy. All three of the linear regression models achieved much lower RMSE values compared to the auto-regressive model and recurrent neural nets, with RBFs hour transformation resulting in the best performance. The sin/cos hour transformation turned out not as great as expected (considering its extra complexities) with a worse performance than one-hot encoding. The LR models hold high potential for actual application due to their demonstrated performance and interpretability (See Section 4.1 for LR model weights interpretations). All three temporal transformation techniques are equivalent in the interpretability of their weights, although the sin/cos method and RBFs can be slightly abstract for non-technical users.

The AR model also holds desirable interpretability with the 2 most recent values being the most influential, as indicated by their two highest weights; these 2 values represent the previous (latest) 10-minute portion of the hour-worth of values used for prediction.

The RNN model is where we see the greatest trade-off in parameter interpretability for model complexity, which has proven unworthy as our application likely suffered from overparameterization (Section 4.3), plus incomparable worse performance to LR. It is considerably challenging to interpret and assess the influence of the weights of the RNN model as they constitute multiple complex layers. Secondly, unlike AR where the previous values are multiplied with their corresponding weight and summed with a bias to obtain a prediction, the RNN model only "cares" about the last value, which is the final prediction. Thus, the trained (first) to (next-to-last) values are discarded, their utility was when their hidden states were used in the process of calculating the last value; this diminishes our ability to interpret the discarded values.

As presented, we proposes the Linear Regression model with RBF hour transformation as the final model for its performance, usage of cyclical features, simplicity, and interpretability.

Acknowledgements

Thanks to **Professor Julian McAuley** for teaching and guidance, Professor Ravi Bajaj, and Professor Justin Eldridge for inspiration in this project.

References

Tekler, Zeynep Duygu, et al. "ROBOD, room-level occupancy and building operation dataset." Building Simulation. Tsinghua University Press, 2022.

Lewinson, Eryk. "Three Approaches to Encoding Time Information as Features for ML Models." Data Science. NVIDIA Developer, 2022.

Wikipedia contributors. "Radial basis function." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 2023.

Edo Cohen-Karlik et al. "Learning Low Dimensional State Spaces with Overparameterized Recurrent Neural Nets." arXiv. arXiv:2210.1406. 2023.

Mike Donlon "Machine Learning in HVAC Controls" AutomatedBuildings. 2016.

T. Wei, Yanzhi Wang and Q. Zhu, "Deep reinforcement learning for building HVAC control," 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 2017, pp. 1-6.