A Machine Learning Model for Occupancy Rates and Demand Forecasting in the Hospitality Industry

Conference Paper · November 2016

DOI: 10.1007/978-3-319-47955-2_17

2 authors, including:

William Caicedo
Universidad Tecnológica de Bolívar
11 PUBLICATIONS **18** CITATIONS

A Machine Learning Model for Occupancy Rates and Demand Forecasting in the Hospitality Industry

William Caicedo-Torres and Fabián Payares

Department of Computer Science - Universidad Tecnológica de Bolívar,Parque Industrial y Tecnológico Carlos Vélez Pombo, Km 1 Vía Turbaco, Cartagena, Colombia
wcaicedo@unitecnologica.edu.co

Abstract. Occupancy rate forecasting is a very important step in the decision-making process of hotel planners and managers. Popular strategies as Revenue Management feature forecasting as a vital activity for dynamic pricing, and without accurate forecasting, errors in pricing will negatively impact hotel financial performance. However, having accurate enough forecasts is no simple task for a wealth of reasons, as the inherent variability of the market, lack of personnel with statistical skills, and the high cost of specialized software. In this paper, several machine learning techniques were surveyed in order to construct models to forecast daily occupancy rates for a hotel, given historical records of bookings and occupation. Several approaches related to dataset construction and model validation are discussed. The results obtained in terms of the Mean Absolute Percentage Error (MAPE) are promising, and support the use of machine learning models as a tool to help solve the problem of occupancy rates and demand forecasting.

Keywords: Machine Learning · Forecasting · Hotel Occupancy · Demand · Neural Networks · Ridge Regression · Kernel Ridge Regression

1 Introduction

In hospitality, occupancy rate forecasting has central importance in planning and decision making, since anticipating demand allows managers plan accordingly on issues as inventory, workforce, supplies, financial budgeting and pricing; all of this in order to maximize revenue and minimize costs.

Currently, the hospitality industry has adopted strategies to maximize income per room, and the set of these strategies fall into the discipline of Revenue Management, which allows selling each room at the highest price customers are willing to pay, so the highest income can be achieved [7].

In Revenue Management, forecasting is the essential element in pricing [11] [16]. According to this, better pricing policies can be developed if better forecasting methods are developed. For this reason the main goal of this paper is to

present the development and validation of a machine learning model to forecast room demand and occupation rate in hotels, taking into account that a sufficient low error is necessary to avoid a negative impact on revenue.

Basically, approaches to occupation forecasting can be grouped in two categories: historical booking models and advanced booking models[16]. Historical booking models try to solve the forecasting problem as a time series modelling problem, and advanced booking problems use reservations data, and the concept of "Pick-Up", which is the increment of bookings N from now to a day T (for which there are already K reservations) in the future, resulting in an occupation forecast of K + N for day T.

For instance, [3] used ARIMA and Holt-Winters exponential smoothing to forecast monthly hotel occupation, showing that both methods yield low Mean Squared Error. In [14], long-term forecast via Holt-Winters is combined with short-term forecast obtained from observations, to obtain ensemble predictions. [10] used Neural Networks for Time Series Prediction, where Japanese citizen travel to Hong Kong was forecasted with Neural Networks outperforming other techniques like Multiple Regression, Moving Average and Exponential Smoothing. Examples of the Pick-Up method can be found in [13], where sold rooms with Pick-Up are forecasted to 7, 14, 30, and 60 days of reserves, with day of week as additional input. [17] proposes a Monte-Carlo model for occupation forecasting at an Egyptian hotel.

As we have shown, statistical techniques can predict, sometimes with good accuracy, occupancy rates and demand. However, some of the aforementioned techniques require important statistical skills and lengthy procedures to be applied in order for then to function correctly (e.g. building and analyzing correlograms); or the use of expensive commercial software. We propose the use of Machine Learning algorithms to build predictive models for the occupancy rates and demand in the hospitality industry, that can be readily used by hotel personnel, without advanced training in statistics. Such models can be packaged into relatively inexpensive applications or executed on the cloud, to provide the hospitality sector with cost-effective forecasts.

The rest of the paper is organized as follows: First, a brief introduction to the algorithms used is given, then the general structure and characteristics of the data set are presented; next the experimental results are shown and discussed, followed by our conclusions.

2 Machine Learning Algorithms

In this work, four algorithms were considered and used. A brief description of each is given next.

2.1 Ridge Regression

Ridge Regression [9] is a regression algorithm that includes a complexity penalty (called a regularization term) in its cost function in order to better approximate the out-of-sample error. Ridge Regression penalizes the the algorithm by a factor proportional to the Euclidean (L2) norm of its parameter vector,

$$J(\theta) = \sum_{i=1}^{N} (\theta^T x_i - y_i)^2 + \alpha ||\theta||^2 \tag{1}$$

Where regularization factor α controls the relative importance of the complexity penalty. Solving for optimal parameters involve finding the derivative of the cost function and setting it to zero. According to this, the parameter vector that minimizes the error is

$$\theta = (X^T X + \lambda I)^{-1} X^T y \tag{2}$$

New predictions can be calculated as follows

$$h(x) = \theta^T x = \sum_{i=1}^{n} \theta_i x_i \tag{3}$$

2.2 Kernel Ridge Regression

Kernel Ridge Regression [12] is a Kernel Method that projects the original data to a feature space, without explicitly computing the transformations, via the Kernel Trick [6]. The transformation allows for a linear model to correctly adjust the data in a higher dimensional space, while remaining a well regularized regressor. As previously stated, Ridge Regression optimal weights are defined as

$$\theta = (X^T X + \lambda I)^{-1} X^T y \tag{4}$$

After some algebraic manipulation, we can state that

$$\theta = X^T (X X^T + \lambda I)^{-1} y \tag{5}$$

So, a prediction can be obtained by

$$h(x') = \theta^T x' = y^T (X X^T + \lambda I)^{-1} X x' \tag{6}$$

A Kernel function evaluates to a dot product of the form

$$f(x_1, x_2) = \phi(x_1).\phi(x_2) \tag{7}$$

without explicitly computing the transform ϕ. If we define $K_{i,j} = f(x_i, x_j)$ and $k_i = f(x_i, x')$, were x_i is the i-th row of matrix X, then

$$h(x') = \theta^T x' = y^T (K + \lambda I)^{-1} k \tag{8}$$

We fitted a model in a higher dimensional space induced by ϕ without incurring in additional computational complexity. Several Kernels can be used, being popular choices the polynomial Kernel

$$k(x_1, x_2) = (\gamma x_1^T x_2 + \beta)^\delta \tag{9}$$

and the Gaussian Kernel

$$k(x_1, x_2) = \exp(-\frac{||x_1 - x_2||^2}{2\delta^2}) \tag{10}$$

2.3 Multilayer Perceptron

The Multilayer Perceptron [15] is a feed forward neural network model capable of classification and regression. A multilayer perceptron is composed of fully connected layers of artificial neurons forming a connected graph (in the simplest case one hidden, one final layer), where neurons are modelled very loosely on biological neurons; and as a feed forward net, the output of each layer becomes the input of the next one. The activation of each neuron is given by

$$y = f(\sum_i^n w_i x_i + b) \tag{11}$$

where f is the activation function. Popular choices include the logistic sigmoid function $y = (1 + \exp(-x))^{-1}$, and the linear function $y = x$. The choice of activation function in the final layer dictates if the network will be able to adequately perform regression or classification.

Multilayer Perceptrons can be trained through the Backpropagation Algorithm [15]. The Backpropagation Algorithm is an application of gradient descent that takes into account the contribution to the error E by the neurons of the hidden layer. Through differential calculus the gradients of all neurons can be calculated and weights updated accordingly. Gradient descent update rule is

$$w_{t+1} = w_t - \eta \nabla E(w_t) \tag{12}$$

where w is the weight vector of any particular neuron. It remains then to specify the gradient. For neuron k in the final layer, the gradient components can be expressed as

$$\frac{\partial E(w)}{\partial w_{jk}} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k} \frac{\partial z_k}{\partial w_{jk}} \tag{13}$$

where w_{jk} are the weights connecting hidden layer neuron j to neuron k, z_k total input to neuron k, and y_k the output of neuron k. Partial$\frac{\partial E}{\partial y_k}$ depends on the cost function used to assess network performance (for regression and time series problems, the Mean Squared Error is often used, with Weight Decay as

the complexity penalization strategy).

Assuming only one hidden layer, the gradient for neurons in that hidden layer can be expressed as

$$\frac{\partial E(w)}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} \tag{14}$$

where where w_{ij} are the weights connecting input component i to neuron j, z_j total input to neuron j, and y_j the output of neuron j. The partial $\frac{\partial E}{\partial y_j}$ represents the variation of the error with respect to the output of neuron j, and can be expressed as

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial y_j} \tag{15}$$

which captures the way neurons from the hidden layer affect the total error, by summing up the contributions from each connection to the neurons in the final layer. The algorithm gets its name from the fact that the partial

$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k} \tag{16}$$

was already calculated in the final layer and now re-used for the neurons in the hidden layer, i.e. it is being "back-propagated" from the final layer.

Now, with all the corresponding gradients known, the gradient descent update rule becomes

$$w_{j,k,t+1} = w_{j,k,t} - \eta\frac{\partial E}{\partial y_k}y_k(1-y_k)y_j \tag{17}$$

for the weights of neurons in the final layer and

$$w_{i,j,t+1} = w_{i,j,t} - \eta\sum_k \frac{\partial E}{\partial z_k}w_{jk}y_j(1-y_j)x_i \tag{18}$$

for the weights of neurons in the hidden layer.

2.4 Radial Basis Function Networks

Radial Basis Function (RBF) Networks [5] are Neural Networks with usually one hidden layer, that employ a set of functions called Radial Basis Functions as feature detectors. RBF nets can be used to perform regression and classification as well. Each neuron in the hidden layer uses a a RBF as activation function ϕ, with the Gaussian function

$$\phi_j(x) = \exp(-\frac{||x-\mu_j||^2}{2\sigma_j^2}) \tag{19}$$

being a popular choice. The network output can be represented by

$$y(x) = \sum_{j=0}^{M} w_{kj}\phi_j(x) \tag{20}$$

with w_{jk} representing the weights connecting hidden layer neuron j to output layer neuron k.

Training consists in finding a set of values for all μ_j, σ_j, and some set of w_{kj} that maximize network performance. First, the Gaussian RBF parameters are selected according to some procedure (μ_j can be selected randomly or to coincide with any training point; and σ_j can be the maximum or average distance between data points), and then solving for w_{kj} to minimize an error function. For the Mean Squared Error as cost function, it can be shown that the gradient is

$$\frac{\partial E}{\partial w_{kj}} = \sum_{n}(\sum_{j=0}^{M} w_{kj}\phi_j(x_n) - y_{kn})\phi(x_n) \tag{21}$$

Setting the gradient to zero, we can solve analytically for the optimal weights. Expressed in matrix form,

$$w^T = \Phi^{\dagger} y \tag{22}$$

where

$$\Phi^{\dagger} = (\Phi^T\Phi)^{-1}\Phi^T \tag{23}$$

is the pseudo-inverse of Φ.

3 Model Development

3.1 Dataset Construction

Reservations and occupation data from July 1, 2008 to June 30, 2014 (2191 days), were gathered from a hotel in Cartagena, Colombia. Using these, three datasets were constructed using different schemes: The first, arranges as inputs the observations from earlier days $(x_{i+0}, x_{i+1}, ..., x_{i+n-1})$ and the output as the current day (x_{i+n}). Occupation in day i corresponds to x_i, and n represents the number of lag variables being used (a classic time series dataset). The second scheme organizes inputs as before but it also adds other variables as day of week, number of holidays present between the previous and following seven days, and whether or not it is tourism season. The third dataset is generated using the number of reservations made for the current day with several days in advance (7, 10, 15, 20, 30, 60 and 90 days), month of the year, day of the week, number of holidays present between the previous and following seven days, and whether or not it is tourism season. Datasets were then partitioned into three subsets, i.e. training, validation and test sets. The test set comprises the 8.26% of the original data, corresponding to all the observations from year 2014 (181); 73.4%

Table 1. Time series forecasting results.

Horizon	Algorithm	MAPE	Parameters
1-step-ahead	Kernel Ridge Regression (Gaussian)	11.2055	– 21 lag variables – $\delta = 2300$ – $\lambda = 0.015$
	Kernel Ridge Regression (Poly)	11.1799	– 20 lag variables – $\lambda = 0.1$ – $d = 1$ – $\beta = 0.1$ – $\gamma = 1$
	RBF Network	11.2301	– 20 lag variables – 150 hidden neurons – $\lambda = 0.0015$ – $\beta = 1.5 \times 10^{-7}$
7-step-ahead	Ridge Regression	19.5595	– 22 lag variables – $\lambda = 0.1$
	Kernel Ridge Regression (Gaussian)	19.6819	– 22 lag variables – $\delta = 3000$ – $\lambda = 0.15$
	Kernel Ridge Regression (Poly)	19.55972	– 22 lag variables – $\lambda = 0.1$ – $d = 1$ – $\beta = 0.1$ – $\gamma = 1$

of the data was used for the training set, and the remaining 18.34% was used for the validation set.

Table 2. Time series plus additional variables forecasting results.

Horizon	Algorithm	MAPE	Parameters
1-step-ahead	Kernel Ridge Regression (Poly)	10.2351	– 5 lag variables – $\lambda = 0.1$ – $d = 2$ – $\beta = 0.1$ – $\gamma = 1$
	RBF Network	11.23586	– 20 lag variables – 150 hidden neurons – $\lambda = 0.0015$ – $\beta = 1.5 \times 10^{-7}$
7-step-ahead	Ridge Regression	19.6819	– 23 lag variables – $\delta = 3000$ – $\lambda = 0.15$
	RBF Network	19.7113	– 22 lag variables – 150 hidden neurons – $\lambda = 0.0015$ – $\beta = 1.5 \times 10^{-7}$

3.2 Training and Validation

Training and validation were performed using Rolling Origin Update Cross Validation [8]. Forecasts are made using One-Step-Ahead (predict next day occupation) and Multi(h)-Step-Ahead procedures, with $h = 7$ (predict occupation seven days in advance), and model performance is measured through Mean Absolute Percentage Error (MAPE), defined as

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{y_t - h_t}{y_t} \right| \tag{24}$$

We used the Mlpy Python library [2] implementation of Ridge Regression and Kernel Ridge Regression, the ALGLIB library [4] implementation of Multilayer Perceptron and Thomas Rückstieß Radial Basis Function Networks implementation [1].

Table 3. Ridge Regression and Kernel Ridge Regression results on the reservations dataset. It is not surprising that Kernel Ridge Regression and Ridge Regression obtained the same MAPE, given that the use of a Polynomial Kernel with $d = 1$ in Kernel Ridge Regression, equals to the standard Ridge Regression.

Algorithm	Kernel/Degree	λ	β	MAPE
Ridge Regression	1	0.15	n/a	8.6962
Ridge Regression	2	0.15	n/a	8.2012
Kernel Ridge Regression	Poly($d = 1$)	0.15	0.1	8.69622

Table 4. Neural Networks results on the reservations dataset.

Model	Hidden nodes	β	MAPE
Multilayer Perceptron	50	5	12.89174
RBF Network	150	1.5×10^{-9}	26.32086

Table 1 shows results for the first data set, where data is arranged as a time series. Table 2 shows results for the second data set, arranged as a time series plus additional variables.

Experiments show that usage of additional variables appear to slightly improve model performance and offer better results than using the time series data only in the one-step-ahead forecast problem, as Polynomial Kernel Ridge Regression using additional variables offered a MAPE of 10.2351% versus 11.1799 % when using only the time series. However in the 7-step-ahead situation, best performance was exhibited by Ridge Regression using only time series data and no additional variables, with a MAPE of 19.5595%.

In the case of the third dataset, inputs are comprised of reservations data at multiple dates before the actual date for which occupation is to be predicted. Table 3 and Table 4 show best validation set results for Ridge Regression, Kernel Ridge Regression and Neural networks respectively.

As shown in Table 3, Ridge Regression with a second degree polynomial transform scored better than the other alternatives, with a MAPE of 8.2012 %.

3.3 Test

We decided to assess the performance of the best model on the reservations dataset, using the test set. Results show a MAPE of 8.65615% on the test set, for the Ridge Regression model with quadratic polynomial features. Results are in line with the validation phase, and there is no evidence of overfitting as validation and test scores are quite close from each other. Figure 1 shows model behavior on the test set.

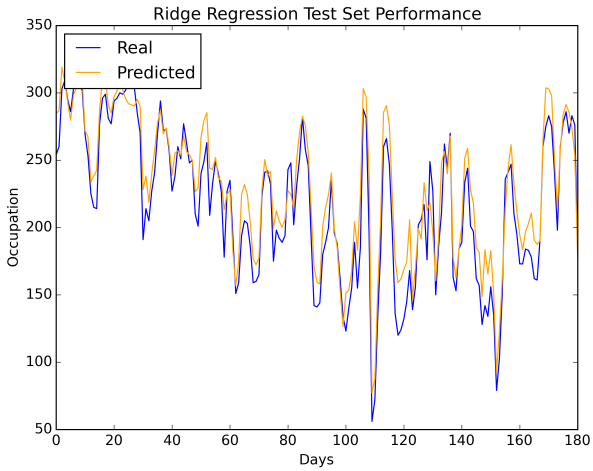

Fig. 1. Test set results for Ridge Regression with quadratic features.

4 Conclusion

In this work, several machine learning techniques were compared. Different models were trained and validated using Ridge Regression, Kernel Ridge Regression, Multilayer Perceptron and Radial Basis Function Networks. Three data sets (each including training set, validation and test set) were constructed using occupation time series data, occupation times series data plus additional variables, and reservations data. Grid search was employed to find optimal parameters for the models. Results show a Ridge regression model with quadratic features trained on the reservations data set, outperforms the other models considered, with a validation set MAPE of 8.2012% and a test set MAPE of 8.6561 %. Test set results show no evidence of overfitting. Also, it is worth noticing that models trained on time series plus additional variables data showed an modest increase in performance, compared to those trained on time series data only. The presence of additional inputs allowed the models to leverage contextual information and improve their predictions.

Finally, the use of bookings and reservations known in advance offered the best performance. The results obtained are promising and support the use of black-box Machine Learning based tools for estimating hotel occupation, which require little statistical expertise by the hotel staff; allowing for a more effective deployment of Revenue Management techniques in the hospitality sector.

References

1. Thomas rückstieß rbf network implementation. http://www.rueckstiess.net/research/snippets/show/72d2363e, accessed: 2016-05-04
2. Albanese, D., Visintainer, R., Merler, S., Riccadonna, S., Jurman, G., Furlanello, C.: mlpy: Machine learning python (2012)
3. Andrew, W.P., Cranage, D.A., Lee, C.K.: Forecasting hotel occupancy rates with time series models: An empirical analysis. Journal of Hospitality & Tourism Research 14(2), 173–182 (1990), http://jht.sagepub.com/content/14/2/173.abstract
4. Bochkanov, S.: Alglib. http://www.alglib.net, accessed: 2016-04-26
5. Broomhead, D., Lowe, D.: Multivariable functional interpolation and adaptive networks. Complex Systems 2, 321–355 (1988)
6. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297, http://dx.doi.org/10.1007/BF00994018
7. El-Gayar, N., Hendawi, A., Zakhary, A., El-Shishiny, H.: A proposed decision support model for hotel room revenue management. ICGST Int J Artif Intell Mach Learn 8(1), 23–8 (2008)
8. Gilliland, M., Sglavo, U., Tashman, L.: Business Forecasting: Practical Problems and Solutions. John Wiley & Sons (January 2016)
9. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 42(1), 80–86 (Feb 2000), http://dx.doi.org/10.2307/1271436
10. Law, R., Au, N.: A neural network model to forecast japanese demand for travel to hong kong. Tourism Management 20(1), 89–97 (1999)
11. Lee, A.O.: Lee, A. O. 1990. Airline reservations forecasting: Probabilistic and statistical models of the booking process. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA (1990)
12. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press (2012)
13. Phumchusri, N., Mongkolku, P.: Hotel room demand forecasting via observed reservation information. In: Kachitvichyanukul, V., Luong, H., Pitakaso, R. (eds.) Proceedings of the Asia Pacific Industrial Engineering & Management Systems Conference. pp. 1978–1985 (2012)
14. Rajopadhye, M., Ghalia, M.B., Wang, P.P., Baker, T., Eister, C.V.: Forecasting uncertain hotel room demand. Inf. Sci. 132(1-4), 1–11 (2001), http://dx.doi.org/10.1016/S0020-0255(00)00082-7
15. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation, pp. 318–362. MIT Press, Cambridge, MA, USA (1986), http://dl.acm.org/citation.cfm?id=104279.104293
16. Weatherford, L.R., Kimes, S.E.: A comparison of forecasting methods for hotel revenue management. International Journal of Forecasting 19(3), 401–415 (2003)
17. Zakhary, A., El Gayar, N., Atiya, A.F.: A comparative study of the pickup method and its variations using a simulated hotel reservation data. ICGST International Journal on Artificial Intelligence and Machine Learning 8, 15–21 (2008)