HOTEL NIGHTLY RATE AND CANCELLATION PREDICTION

Mayank Mishra[1] and Katy Luttrell[2]

[1]CSCI 4622 - Machine Learning, University of Colorado Boulder

1 PROBLEM SPACE

Hotels present a unique market problem due to price discrimination. That is, based on a variety of features ranging from the date of the booking to the method of payment hotel room rates can vary immensely. This challenge makes it difficult to predict the cost of a hotel room for consumers and disadvantages small, proprietary hotel owners. This variability has only been exacerbated by the inevitable shift to automated and digital reservations. It is within this problem space that this project will aim to elucidate three things. First, this project will identify the most relevant features that hotels use to determine pricing. Second, these features will be used to inform a predictive model that takes in a user's (a consumer or hotel managers) information and provides an estimate for the nightly rate of a hotel. Finally, as a stretch goal, this project will aim to provide hotel managers with a probability that a customer will cancel their booking, allowing hotels to maximize their bookings while minimizing risk.

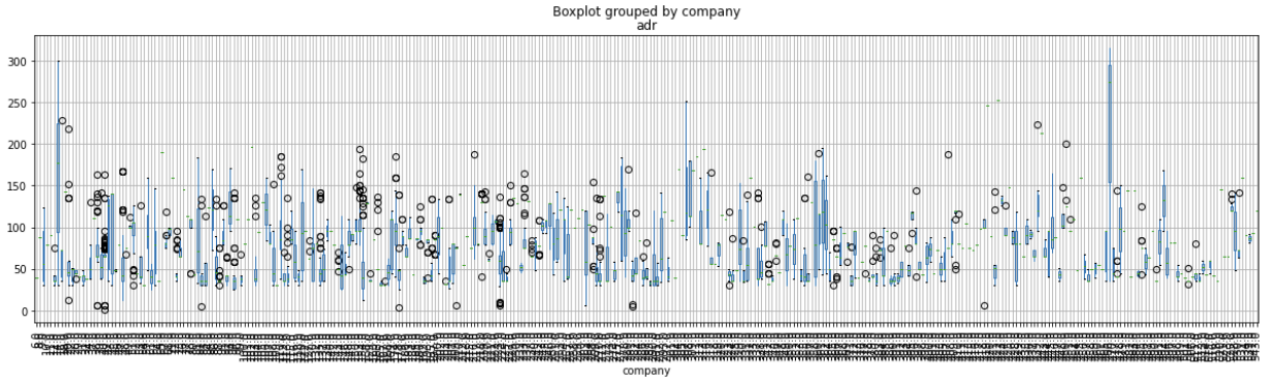
The use of machine learning for a decision support system in hotel revenue management has been well studied. In the International Journal Of Business And Economic Sciences Applied Research, prices for bed and breakfasts were predicted using a hedonic pricing model which relied on log-linear regression. They used mostly predictors having to do with characteristics of the property such as location, hotel category, and existence of certain amenities [1]. This can be good for predicting a general price point for a hotel compared to other hotels, but we are more focused on determining the pricing variability at one hotel from booking to booking. Our use of features having to do with time (date of stay, weekends vs weekdays, etc), information about the booking (direct booking vs travel agent, stay length) will make our model more useful for the application of discriminatory pricing.

With regards to cancellation, the Data Science Journal published a very impressive prediction model which, when used in practice, enabled hotels to reduce cancellations by 37 percentage points. This study used CRISP-DM and the gradient tree boosting algorithm XGBoost. In feature selection they found length of stay to be important in predicting cancellation [2]. The article discussed the issue of problem space of hotel booking cancellations having unbalanced classes and how they were unable to address this issue. We will be using K-nearest neighbors for our cancellation model to see if this approach can remedy the issue of unbalanced class sizes.

2 DATA

The data was sourced from Kaggle and it comes from two hotels in Portugal from the time period of 2015 to 2017 with one property being a resort and the other a city hotel.The dataset chosen for this project has more than 119,000 records, each with 32 features. Each record is a unique reservation

with information about when the reservation was made, when the booking was for, demographic information about the customer (age, number of people in the party etc.) and most importantly, the average daily rate the customer was charged for their booking. Once clean, the data was left with 72,514 samples and 39 features.
In cleaning the data any extreme outliers in the average daily rate were removed. Any rows containing 'nan' or 'unknown' values were dropped, as well as bookings that were complimentary or canceled. Many categorical features were one-hot encoded or dropped due to the sparsity of the data.



A few categories that this data didn't explicitly include, but were found to be important in other research were the length of the stay, if the stay was over a holiday, and if the stay was over a weekend. These features were extrapolated from the check-in and check-out dates as well in a combination with research on major holidays in Portugal gathered from `https://www.expatica.com/pt/lifestyle/holidays/portuguese-holidays-105482/`.

```
df.iloc[500]
```

lead_time	94.0
arrival_date_month	7.0
arrival_date_week_number	30.0
arrival_date_day_of_month	25.0
stays_in_weekend_nights	2.0
stays_in_week_nights	3.0
adults	2.0
children	0.0
babies	0.0
is_repeated_guest	0.0
previous_cancellations	0.0
previous_bookings_not_canceled	0.0
booking_changes	0.0
adr	134.0
required_car_parking_spaces	1.0
total_of_special_requests	1.0
is_resort	1.0
holiday	0.0
weekend_current_stay	1.0
bought_meal	1.0
is_international	0.0
aviation_segment	0.0
corporate_segment	0.0
groups_segment	0.0
offline_ta_to_segment	0.0
online_ta_segment	1.0
corporate_channel	0.0
direct_channel	0.0
gds_channel	0.0
ta_to_channel	1.0
no_deposit	1.0
refundable_deposit	0.0
nonrefundable_deposit	0.0
with_company	0.0
is_contract	0.0
is_group	0.0
is_transient	1.0
is_transient_party	0.0
stay_length	5.0

3 APPROACH

The code for the approach and results is located at our Github: `https://github.com/kalu6090/ML_Project.git`

3.1 PCA

After cleaning and exploding the data in more useful forms there were an excessive amount of features, totaling to 39 after the first pass. While this allows for more potential granularity in future

steps of estimating the average daily rate of a stay in a hotel, it significantly increases the complexity and runtime of these models. As such, the first step was to reduce the data. Principal Component Analysis (PCA) is an algorithmic process which takes all feature vectors of a dataset and reduces and condenses the data into its principal components. This process converges either by a discrete number of components or a threshold of explained variance. PCA accomplishes this by transforming data from its original n-dimensional coordinate frame into a smaller coordinate system such that the greatest variance in the data is projected onto a new coordinate, and each consecutive component is placed orthogonal to every other component. Put simply, if the data is imagined as an n-dimensional ellipsoid such that each consecutive axis of the ellipsoid explains some amount of variance in the data, PCA takes this variance, normalizes it and transforms this axis such that it is orthogonal to the other axises (by taking the eigenvector and values of the covariance explained by that axis), until a threshold is reached. By limiting this threshold dimensional reduction can be achieved by simply truncating the data once the desired threshold has been reached.
In this project, the approach taken was to use sklearn's default PCA library set at a threshold of 90%. This means the PCA will transform the data until 90% of the variance is explainable, and then truncates the remaining features. The first step in this process is to normalize the data. This allows the covariance matrices to not be swayed by different scales of data (for example lead-time is generally measured in days and has a range of 0 - 200 days, whereas ADR has a range of 0 - over 500). Once the data has been normalized, the PCA model can be fit to the scaled data until a 90 percent of the variance in the data is explained.

```
def make_PCA(X_train, X_test, features, thresh = .90):
    pca = PCA(thresh)
    pca.fit(X_train)

    var = pca.explained_variance_ratio_
    print("number components: ", len(var))
    evar = np.sum(var)
    print("Explained Variance:", evar)

    X_pcs_train = pca.transform(X_train)
    X_pcs_test = pca.transform(X_test)

    n_pcs= pca.components_.shape[0]
    most_important = [np.abs(pca.components_[i]).argmax() for i in range(n_pcs)]

    initial_feature_names = features
    most_important_names = [initial_feature_names[most_important[i]] for i in range(n_pcs)]
    dic = {'PC{}'.format(i): most_important_names[i] for i in range(n_pcs)}
```

3.2 Predictive Model

Building a predictive model required multiple iterations and multiple techniques until the best technique was identified. After cleaning and scaling the data, a separate function created a training and test set of the data by randomly selecting 70% of the rows to be reserved solely for training, and the remaining 30% to be tested on. This resulted in a training set of over 30,000 rows, and a testing set of over 13,000 rows. The performance of each of the techniques employed was measured using Root Mean Square Error between the predicted y values (ADR) on the test set versus the actual y values of the test set. The reasoning for this is some of the techniques required binning while others were continuous. An accuracy score like the harmonic score used in classification techniques would not be comparable to other techniques which operate on continuous data. RMSE gives the average distance between the predicted price and the actual price which is useful not only because it can be evaluated across models, but has significance in the model itself since the error is in units of dollars. Each approach was tried both with and without the data being run through PCA, to varying degrees of

success which will be discussed more in the results section. The models are discussed in more detail below, in the order they were explored.

3.2.1 Linear Regression

The first technique attempted was linear regression. Linear regression works by taking the dot product of some weight vector (m-dimensions), and the values of each feature in each row (m-dimensons) plus some constant to get a predicted y. This YHAT is then compared to the actual y value of that row using the residual sum of squares to compare the values. The weights are then adjusted accordingly to continue to reduce the error until the residual sum of squares between all YHATS and true Ys is minimized. To accomplish this, sklearn's Ordinary Least Squares Regression Library was used. First the data was scaled such that the means of each feature were centered at 0, and the variance was 1. The data was then fit to the model and evaluated using RMSE between the YHATs and Y values.

```
def lin_reg(X_train, y_train, X_test, y_test):
    reg = LinearRegression().fit(X_train, y_train)
    y_hat = reg.predict(X_test)
    score = reg.score(X_test, y_test)
    root_mse = calc_rmse(y_hat, y_test)
    print("score: ", score)
    print("Root Mean Square Error: ", root_mse)

    return y_hat, score, root_mse
```

3.2.2 Logistic Regression

Logistic Regression, similar to linear regression, operates off of determining the weights of each feature through an iterative process of comparing the output to the actual value. Importantly the data must first be binned in order to use logistic regression since this is a classification technique. In binary logistic regression, the dot product of each feature and the value of each feature is taken to give an e-score, which is put in a sigmoid function $\frac{1}{1+e^{-e_{score}}}$ estimating the probability of that row being classified in one bin or the other. In this project multi class logistic regression was employed with a bin size of $25, meaning that instead of a binary outcome, there were 20 bins that the model needed to classify each row into. The process uses a One-vs-Rest approach to classification, where each bin has its own set of weights, and the sigmoid gives the probability that each row is in a particular bin versus any other bin. Once the probability that a row is in each bin has been identified, the highest probability bin is selected as the classification.

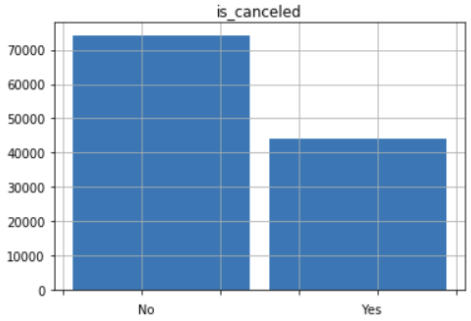
3.2.3 Decision Tree Classification

Decision Tree Classification also uses binned data, so much like in logistic regression the y values are binned. Unlike in the previous methods, the data does not need to be normalized first. Decision trees work by taking the features of the data, creating splits in the data, such that at each node the row of data follows a certain branch until it reaches a leaf node, which is a bin that the row is classified as. The decision tree is made by taking all the data at the root node, and then deciding which feature to split the data on. The best feature is determined by the amount of entropy, and the amount of information gain that is created if the data were to be split on each feature. Entropy is a mathematical

estimate of how poorly the final classification would be given the current split, and information gain takes the entropy values of the current split and all its children and provides an estimate of how well the split classifies the data. With PCA it was unnecessary to limit the max-depth of the tree. To implement this, sklearn's decision tree classifier library was used. This requires very little tweaking, though the data was still normalized before the model was trained on the data to maintain consistency with the other models. Once the model was fit on the training data, the test data was predicted on to yield y-hats from which we could draw the RMSE of the model.

3.3 Cancellation Prediction

Cancellation prediction was the stretch goal for this project, and fortunately time for it to be completed in addition to price prediction. The data for predicting cancellations instead of the average daily rate had to be cleaned differently. A big reason is that the data does not provide a checkout date for reservations that are canceled so features that rely on that, such as length of stay, can no longer be used. However, this problem space was able to have more samples due to it not requiring the removal of complimentary or canceled reservations. For this problem there were 118,215 samples with 37 features each. The data was randomly separated into training, validation, and testing sets with 60% in training, 30% in validation, and 30 % in testing. In cancellation prediction, there wasa very unbalanced dataset with only 37% of the bookings being canceled. For this reason, K-nearest neighbors was used as the prediction algorithm and an F1 score as the metric.



4 RESULTS

4.1 PCA

The PCA algorithm was effective at reducing the data. At 90 percent explained variance PCA maintained 22 features of the original 39. This reduces nearly 50% of the data which allows for far faster convergence in many of the models. A 90 percent threshold for evariance was chosen through brief experimentation with our best performing models, and below 90% evariance the predictive models started to perform worse. PCA was not as effective as expected. This is because all features in the dataset accounted for less than 10% of the e-variance, with most being lower than 5 percent. This means that the data is highly diverse, with no one feature swaying the clustering of the data one way or another. The impact of the low component e-variance is that any predictive model built on this data still required a significant amount of data in order to effectively classify the ADR. Included below is a list of the 22 most relevant features, in order of their e-variance.

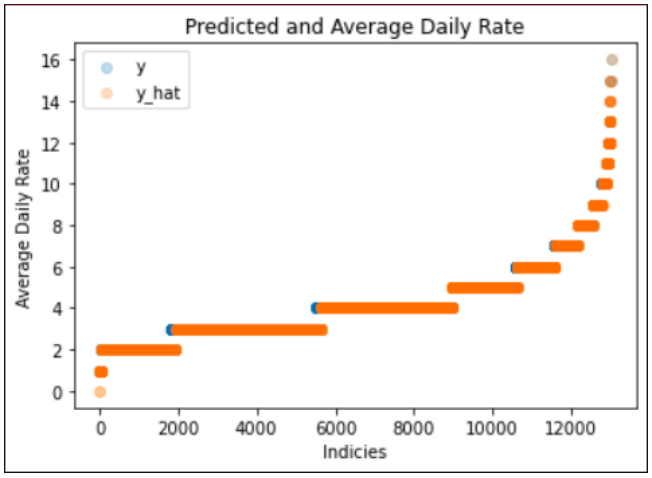
```
{'PC0': 'corporate_segment', 'PC1': 'is_transient', 'PC2': 'stay_length', 'PC3': 'direct_channel', 'PC4': 'arrival_date_week_number', 'PC5': 'no_deposi
t', 'PC6': 'offline_ta_to_segment', 'PC7': 'previous_cancellations', 'PC8': 'bought_meal', 'PC9': 'babies', 'PC10': 'is_contract', 'PC11': 'aviation_se
gment', 'PC12': 'is_group', 'PC13': 'arrival_date_day_of_month', 'PC14': 'children', 'PC15': 'children', 'PC16': 'required_car_parking_spaces', 'PC17':
'booking_changes', 'PC18': 'adults', 'PC19': 'lead_time', 'PC20': 'total_of_special_requests', 'PC21': 'is_repeated_guest', 'PC22': 'lead_time', 'PC2
3': 'groups_segment'}
```

4.2 Predictive Models

Below are the results of each of the predictive models performances. Each of the models were run 10 times on new splits of the training and test data. The average RMSE, score and other metrics are reported. While each model was run both with and without the PCA reduction of the data, it generally did not make much of a difference in the classification, though it did reduce the run time for some models. Since some of the models require the y values to be binned, the RMSE will be reported both in terms of the average bin distance, which will then be transformed into error in dollars to maintain a consistent measure of the performance of each model. A completely random prediction taken in the range of the ADR results in and RMSE of 154.

4.2.1 Decision Tree Classification

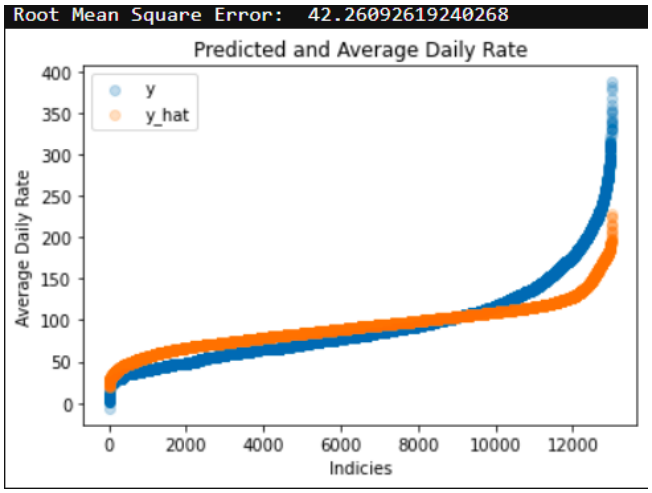
The best model was the decision tree classification without using PCA. The average RMSE was 1.13, with a bin size of $25. This means that on average the model gave an average daily rate within $28.25 of the actual price. There are two potential methods of reducing this error. The first would be to change the binning algorithm. Currently the bins are evenly distributed across the entire range of the data ($0-$500+), however the vast majority of the average daily rates in the dataset were between $100 and $300. This means there will be a higher weighting of accuracy towards the middle of the range of pricing over the outliers. This could potentially be solved by creating bins of the ADR with an even distribution of data in each bin, thus reducing the weight the middle values in the dataset have. Using the PCA data, the model converged about 5 percent faster, but more testing would be required in order to definitively guarantee this method. The PCA method had an average RMSE of 1.15, or $28.75. This difference is not significant enough to warrant prioritizing the unreduced data, though since the time differential is not massive it would only really make a difference at incredible scale.



4.2.2 Linear Regression

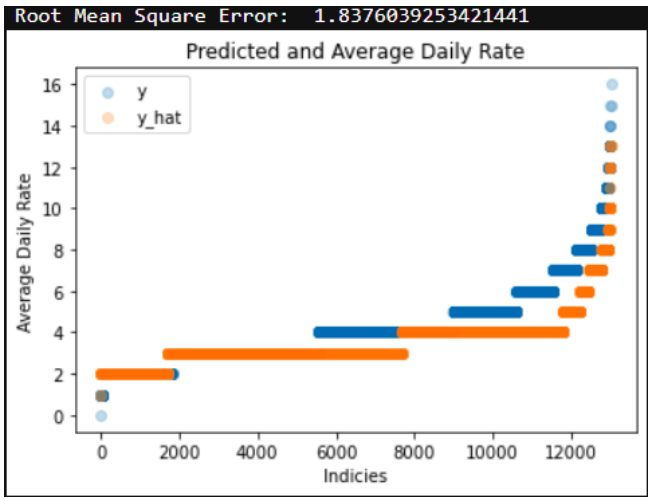
Linear regression performed rather poorly, both with and without PCA data. While the data did not to be binned, linear regression simply was unable to effectively predict the ADR values near the upper and lower ends of the data. This is again likely due to the heavy bias towards middle of the pack adr values. The impact of PCA was negligible in terms of time and the accuracy of the model. The RMSE of the predicted ADR versus the actual ADR without PCA was $42.26, whereas with PCA the

RMSE was $42.99. In perspective of completely random accuracy, this still model still reduces the error by around 72 percent. That being said, the value of this model is to try and help a human make a decision, and completely random guessing is not a great estimate of a human's decision making.



4.2.3 Logistic Regression

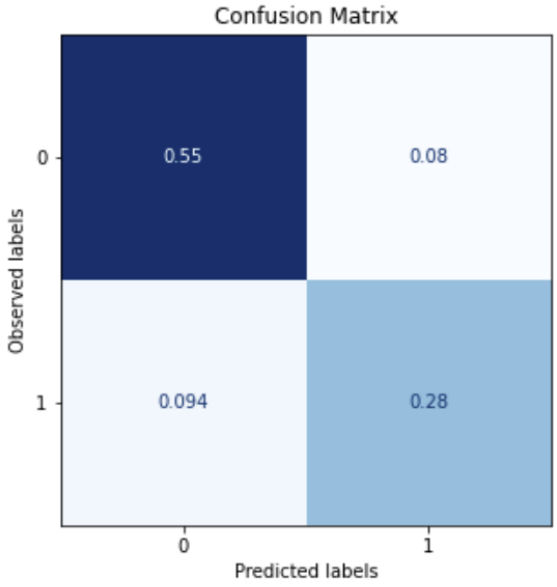
Logistic regression was by far the worst model. With an RMSE of 1.86 and 1.84 with and without PCA respectively, this model was off by more than $46 on average. While this is somewhat similar in performance to linear regression, the advantage linear regression has over logistic regression is that the data is not binned. Similar steps as those suggested under reducing error in binning for the decision tree classifier could be taken to reduce the error in this model. However unlike in other models, the logistic regression suffered most in classifying near the middle of the dataset. Further tests in increasing the penalty function (perhaps using a Ridge Regression penalty function since the data is somewhat exponential), may result in a far better accuracy.



4.3 Cancellation Prediction

K-nearest neighbors for cancellation prediction achieved a mean accuracy of 82.6% with the baseline accuracy of guessing that no reservation will cancel at 62.8%. The F1 score is 74.4%. In the "Limi-

tations and Future Studies" section of the Data Science Journal Article that also tackled the problem of cancellation predictions, they mentioned that they had a class imbalance that they did not address and that future research could address this issue. One of the benefits of K-nearest neighbors is that it is not sensitive to unbalanced classes. In this problem that fact is apparent from the confusion matrix of the KNN results showing that the rate of false positives and false negatives are about the same.



5 DISCUSSION

The prediction of hotel pricing is a difficult problem space due to the wide variety of factors that go into the determination of price point. Especially at small properties, a lot of negotiating can happen and salespeople can have considerable freedom in setting prices. This decision tree classification model works well with this sort of business strategy. A salesperson could enter in information about the stay and about the guest and recieve a $25 range within which to negotiate.

Antonio, de Almeida, and Nunes used the results of their cancellation predictions to have a hotel call guests who were predicted to be at risk of cancellation and try to salvage the booking [2]. This model can be used instead for hotels use the number of guests predicted to be at risk of cancellation to strategize how many overbookings should be accepted for a particular date to guarantee a full sell while minimizing the risk of having to walk a guest who they do not have space for.

One feature that was not included in the data but would have likely been very useful if it were is the state of the bookings at the time of sale. If the hotel were nearly sold out at the time of booking, there is much less risk in asking a higher price and possibly missing out on a sale. Similarly, if the hotel is quite empty for a particular night, salespeople are much more likely to give a considerable discount in order to get rooms filled. Information on how the current state of house for a particular night affects the booking price could help hotel managers to strategize on changing the price for a particular room over time to optimize getting the most rooms filled for the highest prices.

Future plans for this project are to research and apply transformation to date parameters to capture the cyclical nature of the months in a year, days in a month, etc. Additionally,a more robust dataset (ideally access to a hotel's PMS) which includes features such as state of house at time of sale, rate paid in past stays for a particular guest, and how many times a guest called the property before making a booking will likely improve the model.

6 CITATIONS

[1] Pawlicz, A., & Napierała, T. (2017). The determinants of hotel room rates: an analysis of the hotel industry in Warsaw, Poland. International Journal of Contemporary Hospitality Management, 29, 571-588.
[2] Antonio, N., de Almeida, A., & Nunes, L. (2019). An automated machine learning based decision support system to predict hotel booking cancellations. Data Science Journal, 18. https://doi.org/10.5334/dsj-2019-032