ST537 Final Project

Mana Azizsoltani & Chris Comora

04 November, 2020

Contents

Introduction

In the hospitality industry, there is currently a new push towards the implementation of different sorts of analytics solutions, especially in the realm of revenue management and optimization. Hotel rooms are a limited commodity, and once a room on a certain date is booked in advance, the hotel must guarantee the room to that customer. The problem is that the hotel takes on risk when they book the room, because the customer could cancel with relatively short notice, which leaves the hotel either with a vacant room or with no other choice than to downsell the room last minute. The impact on the bottom-line can be significant; research has shown that cancellations impacted "almost 40% of on-the-books revenue" in 2018 for some hotels.

For this project we will be assessing the accuracy, sensitivity, and specificity of various machine learning techniques when faced with predicting hotel cancellations. Essentially, these machine learning models will be attempting to classify a given hotel booking as either canceled or not canceled. We will be running the following four binary classification models:

- Basic Classification Tree
- Random Forest
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Logistic Regression

The data sets used come from two different hotels located in Southern Portugal. The first hotel is a resort hotel located along the coast and the second one is located within a city. They have 31 variables that correspond to different booking information. The first hotel has about 40,000 records, while the second has just under 80,000. Each record corresponds to a booking from the period between July 1, 2015 and August 31, 2017. The data are in .csv format. No-shows are considered cancellations. Because of limitations on our computational power, we used the first 5,000 records of each data set.

Required Libraries

To run the code for the project, the following libraries are required:

- `caret`: to do the heavy lifting of training and tuning the models
- `tidyverse`: for all the data reading and wrangling
- `knitr`: for rmarkdown table outputs
- `rmarkdown`: for output documents
- `rpart`: fitting the basic classification tree
- `randomForest`: fitting random forest models
- `kernlab`: fitting the support vector machine
- `class`: fitting the k-nearest neighbor model

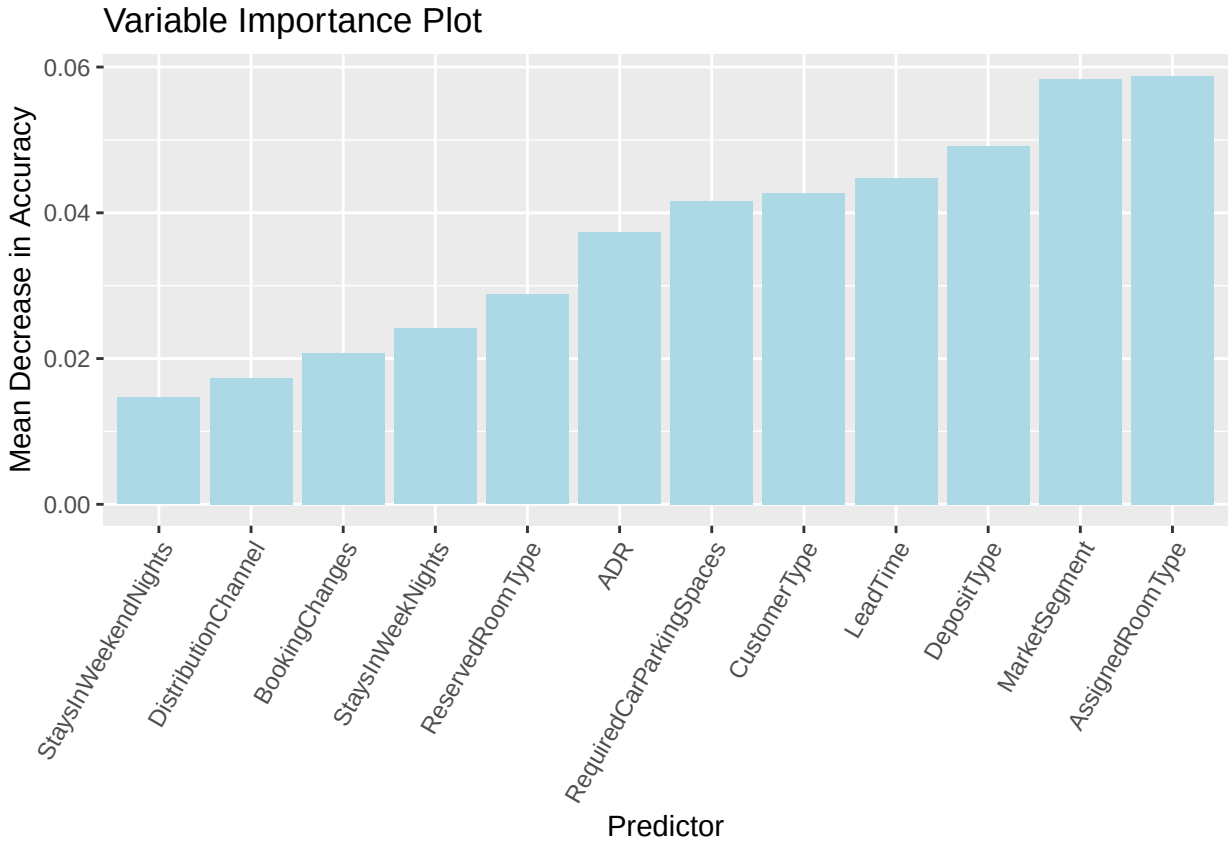
Data

Variable Descriptions

- IsCanceled: value indicating whether or not the booking was cancelled
- PreviousCancellations: number of previous bookings that were cancelled by the customer prior to the current booking
- ADR: average daily rate of the room
- AssignedRoomType: code for the type of room assigned to the booking
- CustomerType: type of booking
- DepositType: indication on if the customer made a deposit to guarantee the booking
  - Deposit can take on the values `No Deposit`, `Non Refund`, or `Refundable`
- LeadTime: number of days that elapsed between the entering date of the booking into the PMS and the arrival date
- MarketSegment: market segment designation
  - In categories, the term "TA" means "Travel Agents" and "TO" means "Tour Operators"
- RequiredCarParkingSpaces: number of car parking spaces required by the customer
- StaysInWeekNights: number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel

Variable Selection

After running an initial random forest model on the entire dataset with selected predictors, we looked at the variable importance of the predictors to decide which variables we would use in our final model. Variable importance, in the context on machine learning, refers to how much a given model "uses" that variable to make accurate predictions. In other words, the more a model relies on a variable to make predictions, the more important it is for the model.



Based on this plot, we chose the 8 predictors with the highest variable importance. The type of room being an important factor came as a surprise to us. We presume that it probably matters since basic, cheaper rooms will probably have more cancellations than suites. Similarly, the number of car parking spaces requested seemed like a rather bizarre predictor for cancellation. Maybe if a car space was requested, it makes the guest not at the mercy of an airline company when traveling to the hotel. We also decided to throw out the weekend stays variable on the basis that is was highly correlated with the weeknight stays variable as well as the reserved room type, since it was highly correlated with the assigned room type.

On the other hand, there were many variables that were highly important and *didn't* come as a surprise. For example, the average daily rate of the room and the deposit. When someone's money is on the line, it makes sense that they would take canceling their stay more serious, especially if they got an expensive room with an unfavorable cancellation policy. Likewise, customer type and market segment seemed intuitively important for prediction; certain clientele or certain sources of clientele could be more or less prone to cancellation.

Methods

As mentioned in the introduction, our focus is on the classification accuracy of machine learning methods. In particular, we will be using traditional cross-validation methods to assess the accuracy, sensitivity, and specificity of each of the five models. We split the data into a training and test data set in order to later

evaluate the model's prediction accuracy. For this project we used a 75/25 split, training the data on the 75% and testing the trained models on the withheld 25%. We will then repeat this process over 5 folds of the data, averaging the results.

The tree-based and the K-nearest neighbors models require parameters to be tuned (more on those later). Since we used the `caret` package in R to fit all of our models, we used the tunes of the parameters that we used were deemed the "best" by the `train()` function.

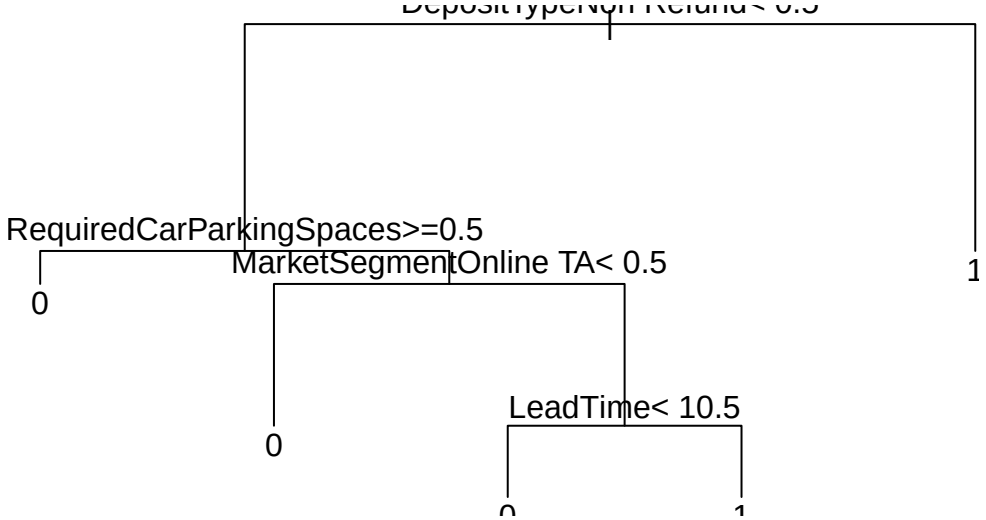
Models

Basic Classification Tree

The basic classification tree is based on partitioning the data into subgroups using simple binary splitting. Initially, all objects are considered a single group. Then, the group is binarily split into two subgroups based on the criteria of a certain variable. We then classify the observations in a specific region with majority vote. We want to grow the tree as big as we can, and then prune it back using cost-complexity pruning. This is done to not overfit the data, but pruning increases the bias. The pruning parameter needs to get tuned, which is done automatically in the `caret` package.

The advantage of using a basic classification is that it is easy to understand and has a good interpretability, which is not something that translates over to ensemble methods like the random forest. Additionally, the basic classification tree is not very computationally expensive, unlike the random forest and the support vector machine. We used the `caret` and `rpart` packages in R to fit our classification trees.

A visualization of the tree as well as the confusion matrices for each hotel can be found below.




```
## $`Confusion Matrix for H1`
##           Reference
## Prediction   0   1
##          0 527 193
##          1 148 381
##
## $`Confusion Matrix for H2`
##           Reference
## Prediction   0   1
##          0 474 203
##          1 201 371
```

Random Forest

The random forest model is an ensemble tree-based method, which creates multiple trees from bootstrap samples and averages the results. Many bootstrap samples are created with replacement and then a classification tree is fitted on each bootstrap sample with a random subset of the predictors. Once a prediction has been made by all of the bootstrapped trees, the final classification is based on majority vote of the bootstrap predictions.

The advantage for using an ensemble method over a regular classification tree is that because there are many bootstrap samples being averaged together, there is less variance. This is similar to how the variance of the sample mean goes down as the sample size increases. Although it will probably increase our prediction accuracy, the random forest loses the interpretability that the basic classification tree has. Furthermore, the algorithm that is used to fit the random forest is very computationally expensive.

To train our model, we used the `randomForest` and `caret` packages in R. The maximum number of predictors for a bootstrap sample as well as the number of trees in the forest are both parameters that need tuning. Again, we will use the "best" tune, automatically given to us by the `caret` package.

Below are the confusion matrices for both hotels.

```
## $`Confusion Matrix for H1`
##            Reference
## Prediction   0   1
##          0 565 110
##          1 110 464
##
## $`Confusion Matrix for H2`
##            Reference
## Prediction   0   1
##          0 542 126
##          1 133 448
```

Support Vector Machine

A support vector machine is a type of classification rule where it essentially maximizes the margin between groups by choosing the "line" with the widest "margin", but allowing for error. We put "line" in quotation marks because with higher-dimensional data, the "line" is actually a hyper-plane-thing. Similarly, the "margin" doesn't actually exist the in sense of a clean margin between two things, since we are allowing for some sort of error. The support vectors are the points closest to the middle that carry the most weight when classifying, since they are the most prone to error and are deciding factors of where the classification line could go. We use a radial kernel function to deal with the non-linearity and higher-dimensions. We fit our SVM using the `kernlab` and `caret` packages in R.

Below are the confusion matrices for both hotels.

```
## $`Confusion Matrix for H1`
##            Reference
## Prediction   0   1
##          0 493 109
##          1 182 465
##
## $`Confusion Matrix for H2`
##            Reference
## Prediction   0   1
##          0 492 129
##          1 183 445
```

K-Nearest Neighbor (KNN)

K-NN is a "model free" approach, meaning that it doesn't assume any probability model on the data. Given an observation, $\boldsymbol{x}$, we want to find the k training observations that are "closest" to $\boldsymbol{x}$, and then classify the new observation using majority vote among these k neighbors. We define "closeness" based on Euclidean distance in our model.

The number (k) of closest neighbors is a parameter that needs tuning. We used the `class` and `caret` packages in R to fit this model. Based on our cross-validation, we used a k value of 5.

Below are the confusion matrices for both hotels.

```
## $`Confusion Matrix for H1`
##           Reference
## Prediction   0   1
##          0 542 142
##          1 133 432
##
## $`Confusion Matrix for H2`
##           Reference
## Prediction   0   1
##          0 517 164
##          1 158 410
```

Logistic Regression

The logistic regression model uses the "logit" function $log(\frac{p}{1-p})$, which links the mean to the linear form of the regression model, $\boldsymbol{X\beta}$. Using it for binary classification, we round the fitted values either up or down to 1 or 0. The logistic regression function is defined as

$$\ln\left(\frac{p(x)}{1-p(x)}\right) = x'\beta, \qquad \text{where } p(x) = (1+e^{-x'\beta})$$

To fit the logistic regression model, we used the `glm()` function in base R. The advantages to using a logistic regression model are similar to those for the basic classification tree: it is not computationally expensive and easy to interpret the results. On the other hand, one of the major drawbacks to using a logistic regression model is that it is subject to distributional assumptions on the data and errors. If our assumptions are broken, our predictions may not be very reliable.

Below are the confusion matrices for both hotels.

```
## $`Confusion Matrix for H1`
##           Reference
## Prediction   0   1
##          0 494 161
##          1 181 413
##
## $`Confusion Matrix for H2`
##           Reference
## Prediction   0   1
##          0 501 199
##          1 174 375
```

Results

Below are two tables of the accuracy, sensitivity, and specificity of each model fit for each hotel.

Table 1: H1 Model Results

	Accuracy	Sensitivity	Specificity
Basic Tree	0.7270	0.7807	0.6638
Random Forest	0.8239	0.8370	0.8084
SVM	0.7670	0.7304	0.8101
KNN	0.7798	0.8030	0.7526
Logit. Reg.	0.7262	0.7319	0.7195

Table 2: H2 Model Results

	Accuracy	Sensitivity	Specificity
Basic Tree	0.6765	0.7022	0.6463
Random Forest	0.7926	0.8030	0.7805
SVM	0.7502	0.7289	0.7753
KNN	0.7422	0.7659	0.7143
Logit. Reg.	0.7014	0.7422	0.6533

Looking at the output of the machine learning models on the two data sets, we can see that the Random Forest model had the highest accuracy (about 80%), with the KNN and SVM models falling close behind it in the mid-70s. The worst performance in terms of prediction accuracy came from the logistic regression model, with an accuracy right around 70%. When considering the non-information rate of 54%, all of the models were effective in at least increasing the prediction accuracy by about 20%.

The fact that the random forest so drastically outperformed the basic classification tree did not come as a surprise, since as an average of many classification trees, ensemble methods generally improve prediction accuracy and reduce overall variance. We suspect that other ensemble methods, such as bagging or boosting, may also be able to get higher prediction accuracies. One of the main disadvantages to using ensemble techniques like the random forest is the fact that they tend to be very computationally intensive. The basic classification tree, logistic regression, and KNN models took significantly less time to run than the random forest and the SVM. If we wanted to run an ensemble method on a very large data set, it would be a nightmare with the minuscule computational power of our personal computers.