Classifying Hotel Cancellation

Connor O'Shea, James Lee, Kevin Kuo, Mani Singh

4/13/2020

Introduction of the Problem

In 2020, COVID-19 has significantly impacted the day-to-day life of people everywhere. As the majority of the country has been placed in a shelter-in-place-esce way of life, the economic impact of this virus has began to show its true form. Millions of people have been left unemployed, small business have been shut down, and the market has struggled to return back to normalcy. During all this economic downturn, one of the hardest hit industries that have been hit by this is the hotel industry. STR, a hospitality research company, recently reported that about occupancy is down nearly 70% compared to last year[^1]:{https://www.usatoday.com/story/travel/hotels/2020/04/09/coronavirus-leaves-hotel-rooms-empty-str-data/5120441002/}. When everything returns to a place of normalcy, every hotel will be looking for ways to save as much profits as possible. The problem that we have attempted to solve this semester focuses on saving hotels money through an attempt at being able to figure out when a consumer is going to cancel a hotel reservation. While hotels receive a cancellation fee from a consumer opting out of a reservation, this is small fee that only represents approximately 1% of total operating costs for a single year, meaning that high cancellations lead to inability for a company to function[^2]:{https://www.viggosmarthotel.com/hotel-booking-cancellations-challenge/}. As there has been a rise in the use of Online Travel Agencies (also known as OTAs), there has also been a rise of cancellations as the expereince has become easier, leading to more strain on the industry. We hope to be able to provide some relief through our ability to classify and the possible implications that are associated with successful classification.

Dataset

We found our dataset from Kaggle.com[^3]:{https://www.kaggle.com/jessemostipak/hotel-booking-demand}. After further research into the dataset, we found that it originated from an experiment conducted by Nuno Antoin, Ana de Almeida and Luis Nunes who had published their work "Hotel Booking Demand Datasets" in the February 2019 issue of *Data in Brief*. The original purpose of this dataset was to determine the number of bookings a hotel would have on a certain day, which we believed made it perfect for classifying cancellations. The dataset consisted of data from two hotels located in Portugal. One was a resort located in Algarve, while the other was a city hotel located in Lisbon. It contained booking data for reservations from July 1, 2015 to August 31, 2018, containing 119390 observations (66% of which came from the city hotel). The dataset contained 33 variables, which included the Booking Channel from which the reservation was made, the lead time which represented the time before the reservation, and the country the individual was from.

Data Collection

In addition to the dataset we had received from Kaggle, we wanted to find some supplemental data that we believed would have influence on whether an individual would cancel. For this we focused on four pieces of supplemental data: *Temperature* Due to the fact that one of the major reasons for travel is related to vacation, we decided that data related to temperature may be important to record. To do so, we recorded the high, average, and low temperature for both Lisbon and Algarve by gathering data that was easily copied and pasted from a weather site from July 1, 2015 to August 31, 2018[^4]:{https://www.wunderground.com/history/monthly/pt/montenegro/LPFR/date/2017-4}. *Precipitation* As we had numerous people traveling,

we felt that precipation could be a reason people cancel as the flight could be delayed/canceled and also if it is a rainy week, an individual may cancel their upcoming trip. To collect this data, we began by manually grabbing the first 400 days from a weather website[^5]:{https://www.worldweatheronline.com/lang/en-us/lisbon-weather-history/lisboa/pt.aspx}. Realizing this process was too tedious and taking hours to collect, we developed a code in Python to scrape the maximum points of rain we would from the API (1000 points a day). We have included the code below to serve as a point of reference. From this point we created a binary variable where if it rains there is a 1 for if it rained that day, 0 if it had not read that day. The reason we did that was due to issues we had in recording exact measurements of the total of rain from a singular day.

```
from darksky.api import DarkSky
from darksky.types import languages, units, weather
from datetime import datetime as dt
import numpy as np


# key needed to access dark sky api
API_KEY = 'e7a30ac223ddd32b8afd34c868279d7f'

# dark sky api object
darksky = DarkSky(API_KEY)


# Function that returns csv containing the daily precipitation values
def get_data(year, latitude, longitude, start_month, end_month):
    # Storing the daily precipitaion values
    daily_precip = []
    # Accounting for leap years
    feb_days = 28
    if year % 4 == 0:
        feb_days = 29

    # Getting a subset of the months that we want
    wanted_keys = [i for i in range(start_month, end_month + 1)]

    months = {1: 31, 2: feb_days, 3: 31, 4: 30, 5: 31, 6: 30, 7: 31, 8: 31, 9: 30, 10: 31, 11: 30, 12: 3

    # Creating a new dictionary with the subset
    months_subset = dict((k, months[k]) for k in wanted_keys if k in months)

    for i in months_subset:
        for j in range(1, months_subset[i] + 1):
            t = dt(year, i, j)
            forecast = darksky.get_time_machine_forecast(
                latitude, longitude,
                extend=False, # default `False`
                lang=languages.ENGLISH, # default `ENGLISH`
                values_units=units.AUTO, # default `auto`
                exclude=[weather.MINUTELY, weather.ALERTS], # default `[]`,
                timezone='UTC', # default None - will be set by DarkSky API automatically
                time=t
            )


            # Adding the precipitation to the list
```

```
            daily_forecast = forecast.daily
            daily_data = daily_forecast.data
            print(daily_data)
            daily_precip.append(daily_data[0].precip_intensity_max)
            print(daily_precip)

    # Exporting results to a csv
    daily_precip_np = np.array(daily_precip)
    file_name = str(start_month) + "-" + str(end_month) + "-" + str(year) + ".csv"
    np.savetxt(file_name, daily_precip_np, delimiter=",")

    return daily_precip


function_test = get_data(2015, 38.722252, -9.139337, 4, 12)
print(function_test)

test2 = get_data(2016, 38.722252, -9.139337, 4, 12)
test3 = get_data(2017, 38.722252, -9.139337, 4, 9)
test4 = get_data(2015, 38.638760, -9.037660, 1, 12)
test5 = get_data(2016, 38.638760, -9.037660, 1, 12)
test6 = get_data(2017, 38.638760, -9.037660, 1, 12)
```

KM from Portugal We believed that an individual who lived further away had a higher likelihood to cancel due to the fact that a small issue such as sickness or family could quickly deter plans. The first step in being able to determine if this would be true, was determining which countries we had in our dataset. Using the table function, we found the 178 different country codes used in our dataset. As the country code provided is 3 letters, we had to determine which code corresponded to what country. Using the World Integrated Trade Solution website [^6]:{https://wits.worldbank.org/wits/wits/witshelp/content/codes/country_codes.htm}, we went through and found each code individually and wrote in an Excel document what country corresponded to which code. After we took this step, we next looked up the distance that each country was from Portugal in kilometers [^7]:{https://www.google.com/maps and https://www.rome2rio.com/}. Once again, each country look up had to be done at an individual level as there was no dataset that contained this information for us that we could find. In the case of Portugal, we set the distance as 0 due to the fact that this was the country the hotel reservations were made for and we were not provided more granular information at the specific city level of travel. *European Union* We wanted to test to see if a person who booked a reservation came from the European Union, as we felt that this could lead to an person being able to freely travel in all other member states. To construct this variable, we created a new column and set it to be a binary variable where if a country was in the EU, they were given the value of 1 and if a country was not it was given a value of 0 [^8]:{https://europa.eu/european-union/about-eu/countries_en}. Overall, we have 28 EU countries, which included the United Kingdom as our data was from 2015 through 2018 and the United Kingdom did not leave the European Union until January 2020. *Joining All The Data* Once all the supplemental data was collected by our team, we attached this to our original data using the merge function, as demonstrated below:

```
df <- merge(x = df, y = weather, by.x = c('arrival_date', 'city'), by.y = c('date', 'city'), all.x = TRU
```

Other Data Consideration In a perfect world, we had two other variables we would have considered:

Economic GDP Factor In our dataset, we had reservations made from over 100 countries. As we were a team of four, we would have struggled to capture the GDP of each country for day, month, or even yearly. We felt that without getting this granular type of data, it would be not worth incorporating this information.

Reason for Hotel Cancellation One of the biggest things that we wish we had in our dataset was the reason for hotel cancellation. This would help us be able to know if there were multiple cancellations over a period

of time due to a specific issue and filter out outliers. Unfortunately, the dataset kept the name of the hotels anonymous and as students, we felt we would not be able to get this data from the creators, if this is something they had even recorded. We would hope in the future hotels would keep this information as it is something that could help us possibly identify trends instead of just cancellations.

Data Cleaning

In order to make the dataset is a useable one for cancellation prediction: *Country Code Null* In our dataset, we had 488 country codes that were NULL. Since we had added a KM from Portugal variable, we need to have data that we can match to this variable. It is for this reason we removed these 488 data points. *Balancing Cancellation / No Cancellation* Originally we had 119,390 datapoints in our dataset. This contained 75,166 non-cancellations and 44,224 cancellations. IWe wanted to make sure that the model did not show bias towards non-cancellation. We did this by nmaking sure our dataset contained equal distribution of cancellations and non-cancellations. *Elimination of Reservation_Status* In our dataset, we realized we had a variable that in reality we would not have in real-life. The reservation status and reservation status date indiciated that a reservation was cancelled. It was for this reason that we removed these two columns from our dataset. *Day of Week* One of the things that we wanted to test was to see if the specific day of the week had any effect. To do this, we set a numerical value for each day.
Normalization We normalized the numerical values so that those columns change the columns in the dataset to the same scale. This will make sure columns with intrinsically large values does not carry larger influence in the dataset.

Analysis

Variable Selection Initially, we used a correlation matrix along with the corresponding correlation coefficients to select variables. We believed that variables that had strong positive or negative correlations with the output variable, which we defined as having a correlation coefficient greater than 0.2, would best help the model make decisions. However, we saw that many of the predictors did not have strong correlations with the outcome variables. We then researched other methods for variable selection and we found that a random forest can be used for variable selection. The random forest decides how important a feature is based on how much a feature decreases node impurity. So after running a random forest with all the features, we were able to see the mean decrease in error for all the features. So, the variables that decreased the error the most were more important. We chose the 20 features with the highest mean decrease in error because there was a substantial drop off after the 20th highest feature.

Model Selection Including our baseline model, which was a decision tree, we ran in total 6 models. These models are logistic regression, random forest, decision tree, boosted decision tree, and naive bayes. We used nested cross validation to select the best model because nested cross validation gives us a general idea of how well the model will perform with optimal parameters.

```
# formula used after variable selection
baseline_f1 <- as.formula(is_canceled ~  deposit_type + km + lead_time + market_segment + total_of_spec

# Repeated CV
fitControl <- trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  ## repeated ten times
  repeats = 10)

# boosted tree
```

```
b.tree <- train(baseline_f1, data = df.train.sample,
                method = "gbm",
                trControl = fitControl,
                verbose = FALSE, preProc = 'zv')
b.tree.train <- predict(b.tree, df.train)
confusionMatrix(table(b.tree.train, y.train))

# lasso
lasso <- train(baseline_f1, data = df.train.sample,
                method = "glmnet",
                family = "binomial",
                trControl = fitControl,
                verbose = FALSE, preProc = 'zv')
lasso.train <- predict(lasso, df.train)

# decision tree
dtree <- train(baseline_f1, data = df.train.sample,
                      method = "rpart",
                      trControl = fitControl,
                      tuneLength = 10
                      )
dtree.train <- predict(dtree, df.train)
confusionMatrix(table(dtree.train, y.train))

# random forest
rf <- train(baseline_f1, data = df.train.sample,
                    method = "rf",
                    trControl = fitControl,
                    verbose = FALSE, preProc = 'zv')
rf.train <- predict(rf, df.train)
confusionMatrix(table(rf.train, y.train))


# Naive bayes
bayes <- train(baseline_f1, data = df.train.sample,
                      method = "nb",
                      trControl = fitControl,
                      tuneLength = 10,
               verbose = FALSE, preProc = 'zv'
)
```

Implications

The ability to predict the cancellation of hotel reservations is important as it presents a potential recipe to a problem of loss of revenue. If a hotel had the ability to determine the number of hotel cancellations will happen in a singular night, the hotel can potentially overbook with some certainity to decrease the number of empty room and maximize revenue. One industry that has done this already is the airport industry. According to the Bureau of Transportation Statistics, the number of trips where people denied board, whether for overbooking purposes or for other reasons was 0.09% in 2015[^8]:{https://www.ft.com/content/e4cb5744-1e9d-11e7-a454-ab04428977f9}. In the cases of overbooking, passengers are offered vouchers and potentially a complimentary flight. While this does demonstrate a small loss, 99.91% of the time, companies will remain significantly profiting from this strategy. Lastly, we would hope that by including other variables

as we have in our models, we will be able to reduce this 0.09% overbooked rate as we know that there are possible backlashes with this, as demonstrated by the United Continental public relation disaster in April of 2017, where a customer was forcibly removed. We would also need to develop a backup plan for what would happen if there was overbooking. With our current model, we would begin with using the model to allow for stand-bys. Our reason for this is that we can provide a certain estimate to the customer of a chance that they will have an ability to get a room. We could potentially charge a small fee to be put on the wait-list based off of that for the spot on the waitlist. The reason we are not jumping straight into double-booking and following the same models airlines follow is because our model still does not have a high enough accuracy rate. We know that there is a high risk for the fact that we will have false-negatives and for this reason too many double-bookings. This will result in a lot of backlash for the company that we would be way too much of a risk at this current moment. As we get better and better with our model down the rode though, we believe it is something that could be brought into discussion. One of our biggest reasons we choose this problem is that fact it has so many implications in numerous industries besides hospitality. By having the ability to overbook, more customers can be served. This is something that can have as little of an effect of a couple gets to have a dinner reservation instead of being on a waitlist or as big as a patient getting to see a doctor that would not have been able to for months. We hope that by demonstrating that this method can have success, that eventually this is something that will be accepted over time.

GitHub

All the code for the project can be found on GitHub: https://github.com/realmanisingh/hotel-cancellations-predictor