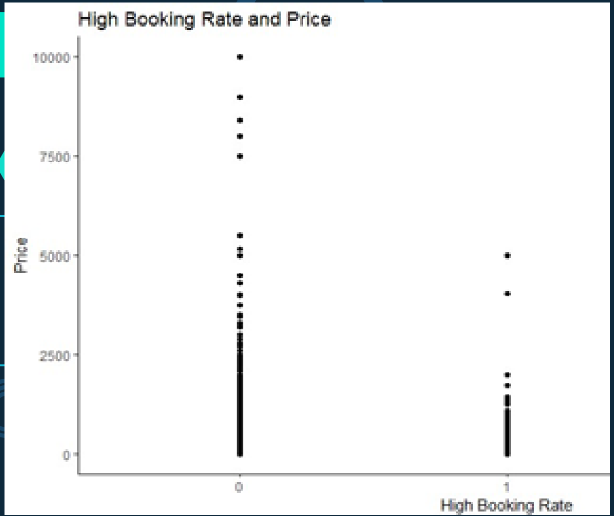
Predicting High Booking Rates

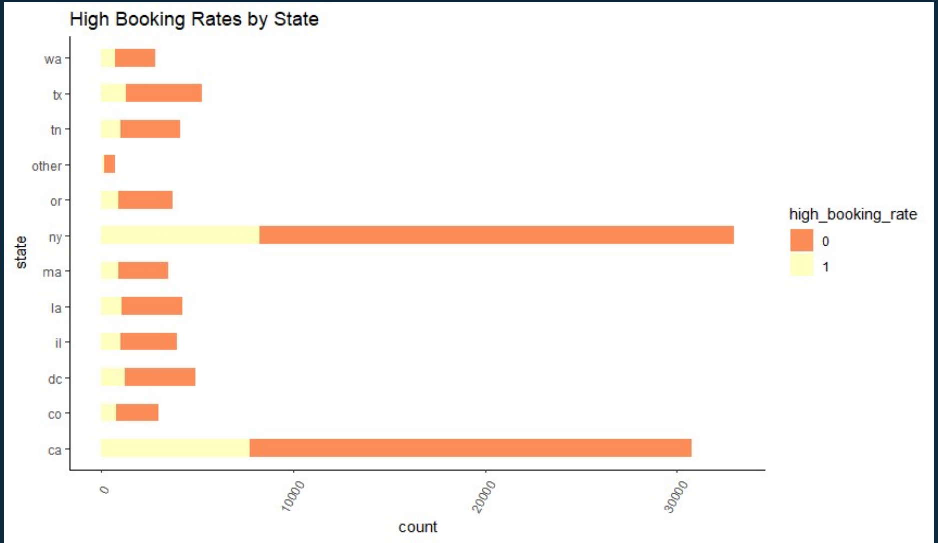
Bekzod Akramov | Akshay Havalgi | Austin Hom | Vivek Ramanathan | Shashank Manu Rao

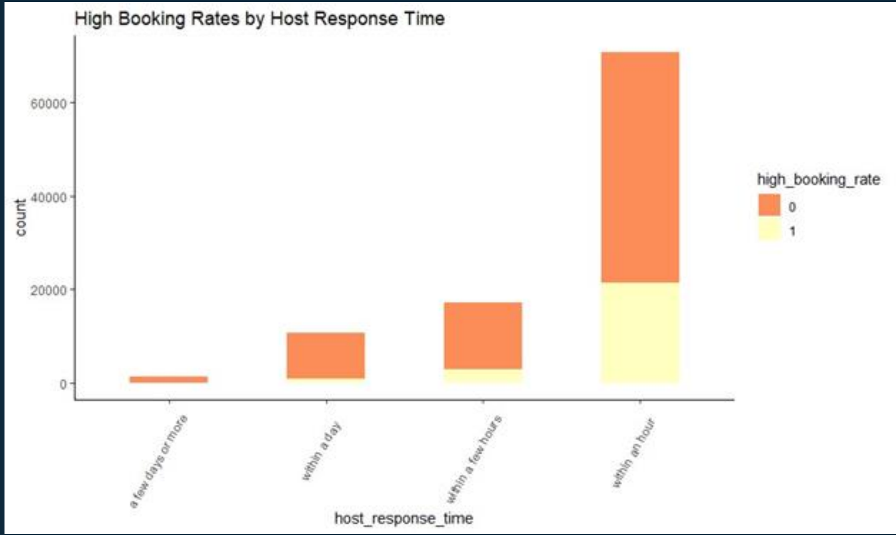
Group 5 | May 2020

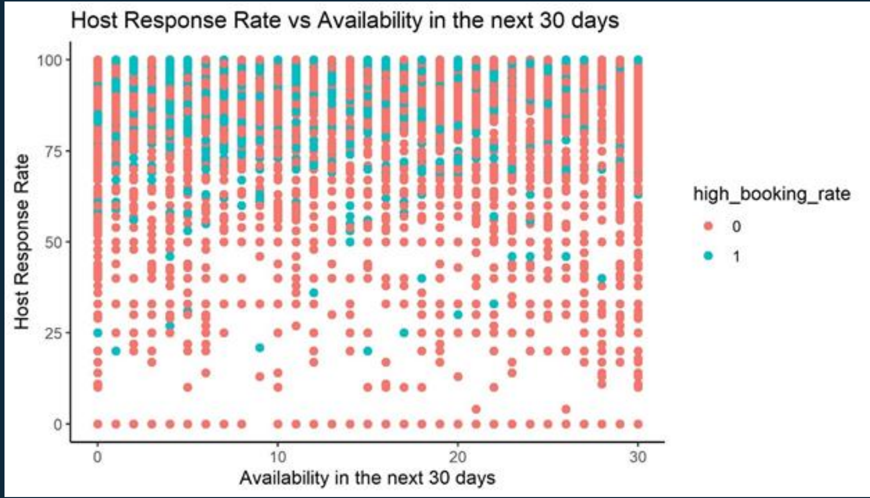
Exploratory Data Analysis

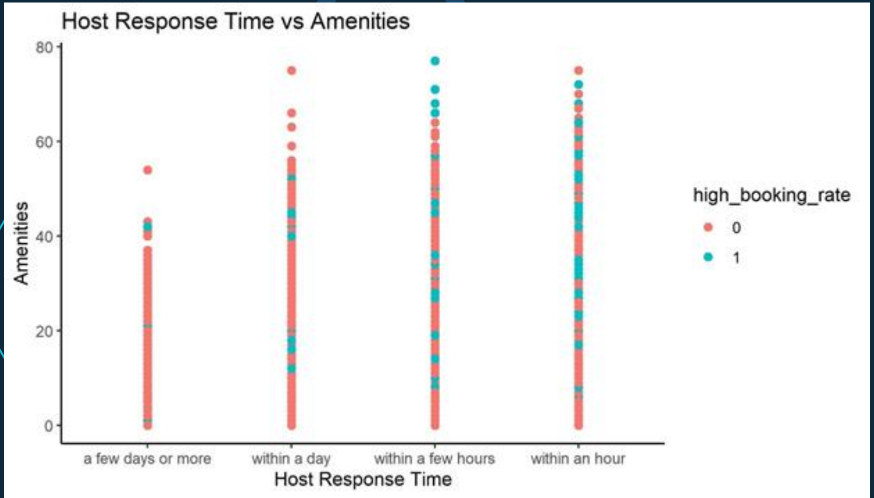
- Initial EDA helped us learn about the data types and structures of the objects in the dataset, as well as missing and problematic data
- Based on domain knowledge and research, we explored relationships between:
    - High booking rate and several variables of interest (e.g., host's response time, price, availability to book in 30, 60, 90, or 365 days, cleaning fee, room type, etc.)
    - Other predictors (e.g., price and host response rate, price and host response time, host response rate and availability, host response time and amenities, etc.)
- We got a good idea of which variables would be important to include in predictive models (including new variables to create)

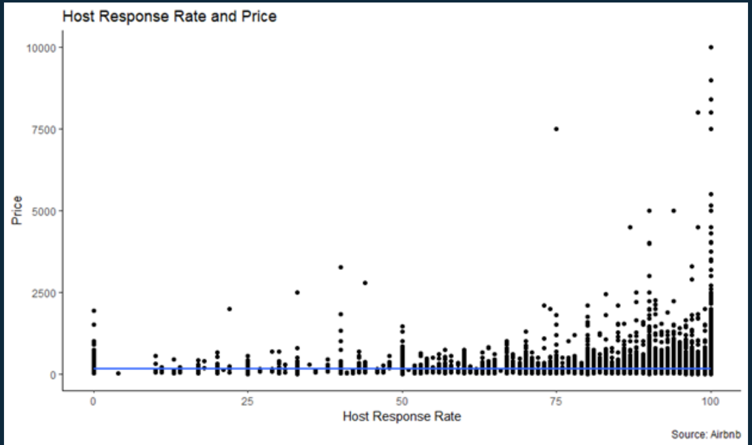

High Booking Rate and Price
Price
10000
7500
5000
2500
0
0
1
High Booking Rate

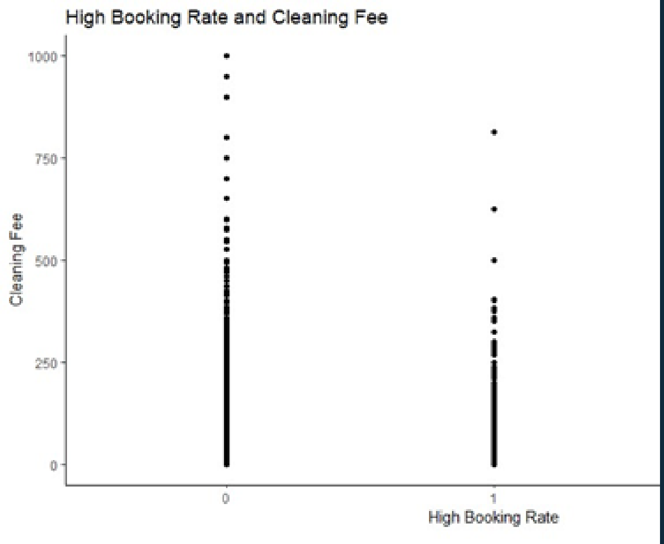

High Booking Rates by State
state
wa
tx
tn
other
or
ny
ma
la
il
dc
co
ca
0
10000
20000
30000
count
high_booking_rate
0
1

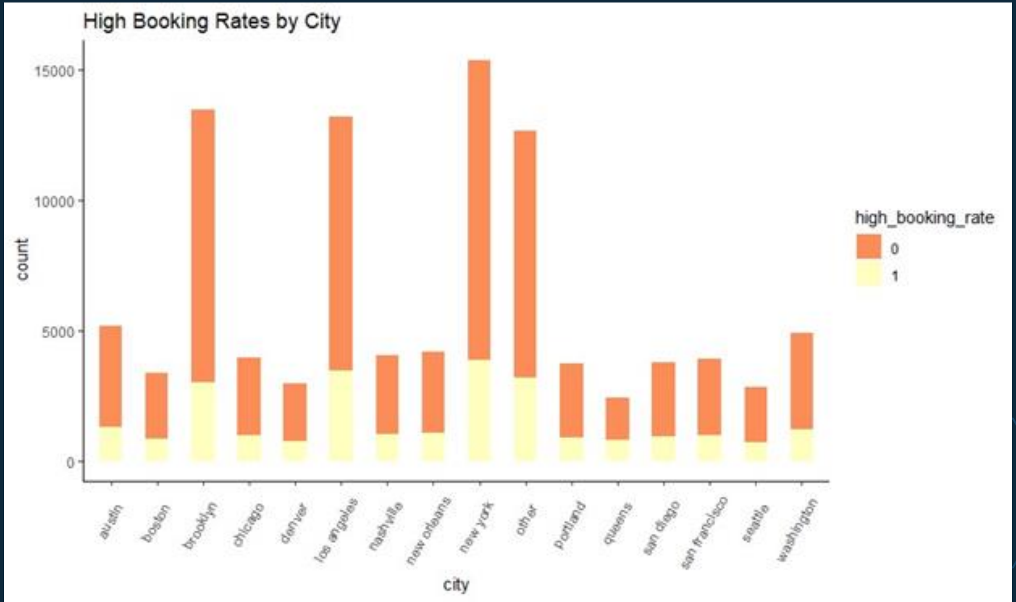

High Booking Rates by Host Response Time
count
60000
40000
20000
0
a few days or more
within a day
within a few hours
within an hour
host_response_time
high_booking_rate
0
1


Host Response Rate vs Availability in the next 30 days
Host Response Rate
100
75
50
25
0
0
10
20
30
Availability in the next 30 days
high_booking_rate
0
1


Host Response Time vs Amenities
Amenities
80
60
40
20
0
a few days or more
within a day
within a few hours
within an hour
Host Response Time
high_booking_rate
0
1


Host Response Rate and Price
Price
10000
7500
5000
2500
0
25
50
75
100
Host Response Rate
Source: Airbnb


High Booking Rate and Cleaning Fee
Cleaning Fee
1000
750
500
250
0
1
High Booking Rate


High Booking Rates by City
count
15000
10000
5000
0
austin
boston
brooklyn
chicago
denver
los angeles
nashville
new orleans
new york
other
portland
queens
san diego
san francisco
seattle
washington
city
high_booking_rate
0
1

Data Cleaning

- 2 self made helper functions for cleaning numeric columns and factor columns
- NA's were converted to be the median (numeric variables) or most common (factor variables) of the column manually
- Remove redundant Columns and data with over 80% NA values
- 1 function to clean entire dataset in the end

```
####-fn to convert to numeric
ConvertNumeric = function(dfCol, NAValue){
  val <- (gsub("\\$", "", dfCol))
  val <- (gsub("\\%", "", val))
  val <- suppressWarnings(as.numeric(gsub("\\,", "", val)))
  val[is.na(val)] = NAValue
  return(val)
}

#-fn to convert to factor
ConvertFactor = function(dfCol, GoodVector, NAValue){
  dfCol = trimws(dfCol)
  dfCol = tolower(dfCol)
  dfCol[!(dfCol %in% GoodVector)] = NAValue
  dfCol = as.factor(dfCol)
  return(dfCol)
}
```

Feature Engineering

- NumAmenities - The number of amenities at the listing
- NumHostVerification - The number of host verifications at the listing
- TransitText - 1 if there is a transit summary, 0 if there is not
- HouseRuleText - 1 if there are house rules, 0 if there is not
- Interaction sentiment- Positive, Negative, Neutral values on the interaction column

Logistic Regression

- Training: 75%, Validation: 25%
- Metric: Accuracy
- Baseline Model

```
#==============================================================================
# simple logistic regression
simpleLog <- glm(clean_train_labels ~., data = clean_train, family = binomial)
probabilities <- simpleLog %>% predict(clean_valid, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
lr_acc <- sum(ifelse(predicted.classes==clean_valid_labels,1,0))/valid_len
lr_acc #0.7884
```

Accuracy : 0.7884

Actuals\Predicted	0	1
0	17345	1274
1	4016	2365

Random Forest

- Beat Baseline Accuracy
- Ensemble method
- Tune Hyperparameters

```
#---------------------Random Forest model

#Get train labels
train_labels <- fread("airbnb_train_y.csv")
train_labels_hbr = ConvertFactor(train_labels$high_booking_rate, c('0','1'), '0')

#set seed
set.seed(12345) #tried different seeds - 123,42

#split train : valid - 75 : 25
valid_inst = sample(nrow(theCleanDF),0.25*nrow(theCleanDF))
clean_valid = theCleanDF[valid_inst,]
clean_train = theCleanDF[-valid_inst,]

clean_train_labels = train_labels_hbr[-valid_inst]
clean_valid_labels = train_labels_hbr[valid_inst]

#base model without grid search
rf_model=randomForest(clean_train_labels~.,data = clean_train, ntree=1000, mtry=7,importance=TRUE)
# rf_model

#predict on valid
pred1 = predict(rf_model, newdata = clean_valid)
pred1


#compute valid accuracy
valid_len = nrow(clean_valid) #get len of valid
rf_acc <- sum(ifelse(pred1==clean_valid_labels,1,0))/valid_len
rf_acc #0.837
```

Random Forest

Accuracy : 0.8370

Actuals\Predicted	0	1
0	17463	1156
1	2919	3462

XGBoost

Xtreme Gradient Tree Boosting

- Beats baseline with highest accuracy
- Boosting algorithms theoretically don't overfit
- Automatic feature selection

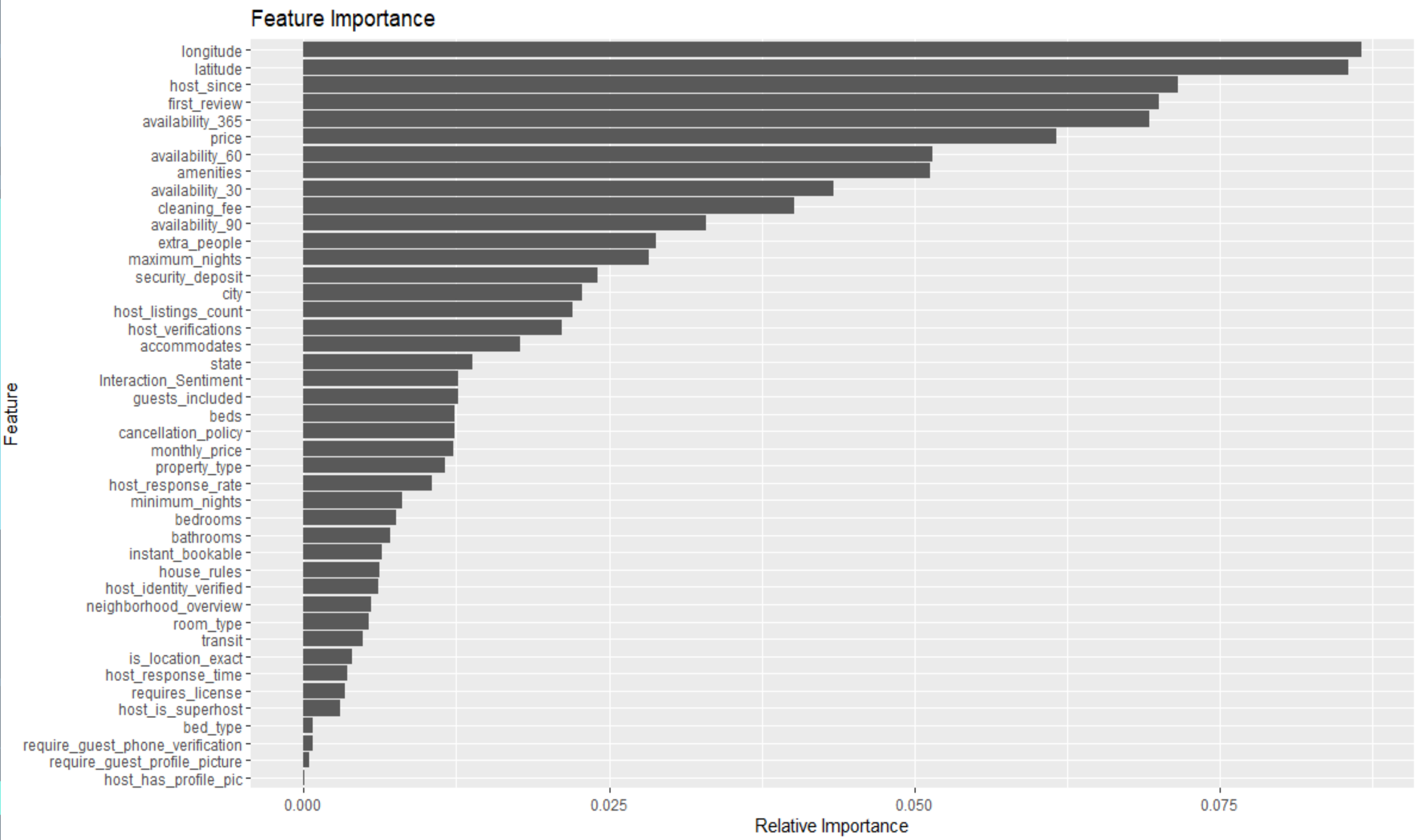
```r
#==========================================================================================
# XGBoost
library(xgboost)
# covert to matrix
dttrain = data.matrix(clean_train)
dtvalid = data.matrix(clean_valid)
dttest = data.matrix(theCleanDF_test)
# change to xgb matrix, note labels need to be numeric so we convert back to numeric then subtract 1
dtrain <- xgb.DMatrix(data = dttrain, label = as.numeric(clean_train_labels)-1)
dvalid <- xgb.DMatrix(data = dtvalid, label = as.numeric(clean_valid_labels)-1)
dtest <- xgb.DMatrix(data = dttest)
# make model
bstair<- xgboost(data=dtrain, max.depth = 14, eta =0.0390,min_child_weight = 2,nthread =8,nrounds = 3000, max_delta_step = 2,
                 colsample_bytree = 0.44, early_stopping_rounds = 400, objective = "binary:logistic",evalmetric = "auc")
pred1_XGB <- predict(bstair, dvalid)
```

XGBoost

Accuracy : 0.8459

Actuals\Predicted	0	1
0	17027	1592
1	2266	4115

XGBoost Model Feature Importance



Thanks!

Any questions?

Bekzod Akramov | Akshay Havalgi | Austin Hom | Vivek Ramanathan | Shashank Manu Rao

Group 5 | May 2020