Preliminary Results

Adeline Casali

2024-02-15

# Data Preprocessing

# Loading Data  
data <- read.csv("project\_data.csv")  
  
# Loading packages  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(kableExtra)

##   
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':  
##   
## group\_rows

library(class)  
library(tree)  
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

library(reticulate)  
library(tensorflow)

##   
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':  
##   
## train

library(keras)  
library(MESS)  
  
# Explore the dataset  
na\_rows <- data[apply(is.na(data), 1, any), ]  
print(na\_rows)

## [1] Booking\_ID no\_of\_adults   
## [3] no\_of\_children no\_of\_weekend\_nights   
## [5] no\_of\_week\_nights type\_of\_meal\_plan   
## [7] required\_car\_parking\_space room\_type\_reserved   
## [9] lead\_time arrival\_date   
## [11] market\_segment\_type repeated\_guest   
## [13] no\_of\_previous\_cancellations no\_of\_previous\_bookings\_not\_canceled  
## [15] avg\_price\_per\_room no\_of\_special\_requests   
## [17] booking\_status   
## <0 rows> (or 0-length row.names)

unique(data$booking\_status)

## [1] "not\_canceled" "canceled"

# Remove unnecessary columns (Booking\_ID)  
data <- data[ , -1]  
  
# Convert booking\_status to 0 and 1  
data$booking\_status <- ifelse(data$booking\_status == "not\_canceled", 0, 1)  
  
# Calculate booking\_date based on arrival\_date and lead\_time  
data$arrival\_date <- as.Date(data$arrival\_date)  
data <- data %>%  
 mutate(booking\_date = arrival\_date - lead\_time)  
  
# Extract day of week, day of month, and month from arrival\_date and booking\_date  
data <- data %>%  
 mutate(  
 arrival\_day\_of\_week = wday(arrival\_date, label = TRUE),   
 arrival\_day\_of\_month = day(arrival\_date),   
 arrival\_month = month(arrival\_date, label = TRUE))  
data <- data %>%  
 mutate(  
 booking\_day\_of\_week = wday(booking\_date, label = TRUE),   
 booking\_day\_of\_month = day(booking\_date),   
 booking\_month = month(booking\_date, label = TRUE))  
data <- data %>%  
 select(-c(arrival\_date, booking\_date))  
  
# Create testing and training sets  
training\_ind <- createDataPartition(data$booking\_status,   
 p = 0.75,   
 list = FALSE,   
 times = 1)  
training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
  
# Assessing, grouping, and factoring categorical variables  
training\_set$booking\_day\_of\_week <- as.character(training\_set$booking\_day\_of\_week)  
training\_set$booking\_month <- as.character(training\_set$booking\_month)  
training\_set$arrival\_day\_of\_week <- as.character(training\_set$arrival\_day\_of\_week)  
training\_set$arrival\_month <- as.character(training\_set$arrival\_month)  
  
unique(training\_set$type\_of\_meal\_plan)

## [1] "meal\_plan\_1" "not\_selected" "meal\_plan\_2" "meal\_plan\_3"

unique(training\_set$room\_type\_reserved)

## [1] "room\_type1" "room\_type4" "room\_type2" "room\_type6" "room\_type7"  
## [6] "room\_type5" "room\_type3"

unique(training\_set$market\_segment\_type)

## [1] "offline" "online" "aviation" "complementary"  
## [5] "corporate"

unique(training\_set$booking\_day\_of\_week)

## [1] "Mon" "Tue" "Thu" "Wed" "Sat" "Sun" "Fri"

unique(training\_set$booking\_month)

## [1] "Feb" "Sep" "Mar" "Oct" "Aug" "Jul" "Jan" "May" "Apr" "Dec" "Jun" "Nov"

unique(training\_set$arrival\_day\_of\_week)

## [1] "Mon" "Wed" "Sun" "Fri" "Thu" "Tue" "Sat"

unique(training\_set$arrival\_month)

## [1] "Oct" "Feb" "Apr" "Jul" "Nov" "Aug" "Mar" "Sep" "May" "Dec" "Jan" "Jun"

training\_set$type\_of\_meal\_plan <- factor(training\_set$type\_of\_meal\_plan)  
training\_set$room\_type\_reserved <- factor(training\_set$room\_type\_reserved)  
training\_set$market\_segment\_type <- factor(training\_set$market\_segment\_type)  
training\_set$booking\_day\_of\_week <- factor(training\_set$booking\_day\_of\_week)  
training\_set$booking\_month <- factor(training\_set$booking\_month)  
training\_set$arrival\_day\_of\_week <- factor(training\_set$arrival\_day\_of\_week)  
training\_set$arrival\_month <- factor(training\_set$arrival\_month)  
  
class(training\_set$type\_of\_meal\_plan)

## [1] "factor"

class(training\_set$room\_type\_reserved)

## [1] "factor"

class(training\_set$market\_segment\_type)

## [1] "factor"

class(training\_set$booking\_day\_of\_week)

## [1] "factor"

class(training\_set$booking\_month)

## [1] "factor"

class(training\_set$arrival\_day\_of\_week)

## [1] "factor"

class(training\_set$arrival\_month)

## [1] "factor"

levels(training\_set$type\_of\_meal\_plan)

## [1] "meal\_plan\_1" "meal\_plan\_2" "meal\_plan\_3" "not\_selected"

levels(training\_set$room\_type\_reserved)

## [1] "room\_type1" "room\_type2" "room\_type3" "room\_type4" "room\_type5"  
## [6] "room\_type6" "room\_type7"

levels(training\_set$market\_segment\_type)

## [1] "aviation" "complementary" "corporate" "offline"   
## [5] "online"

levels(training\_set$booking\_day\_of\_week)

## [1] "Fri" "Mon" "Sat" "Sun" "Thu" "Tue" "Wed"

levels(training\_set$booking\_month)

## [1] "Apr" "Aug" "Dec" "Feb" "Jan" "Jul" "Jun" "Mar" "May" "Nov" "Oct" "Sep"

levels(training\_set$arrival\_day\_of\_week)

## [1] "Fri" "Mon" "Sat" "Sun" "Thu" "Tue" "Wed"

levels(training\_set$arrival\_month)

## [1] "Apr" "Aug" "Dec" "Feb" "Jan" "Jul" "Jun" "Mar" "May" "Nov" "Oct" "Sep"

# One-hot encoding the training set  
onehot\_encoder <- dummyVars(~ type\_of\_meal\_plan + room\_type\_reserved + market\_segment\_type + booking\_day\_of\_week + booking\_month + arrival\_day\_of\_week + arrival\_month,   
 training\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved", "market\_segment\_type",   
 "booking\_day\_of\_week", "booking\_month", "arrival\_day\_of\_week", "arrival\_month")],   
 levelsOnly = FALSE,   
 fullRank = TRUE)  
  
onehot\_enc\_training <- predict(onehot\_encoder,   
 training\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved", "market\_segment\_type",   
 "booking\_day\_of\_week", "booking\_month", "arrival\_day\_of\_week", "arrival\_month")])  
training\_set <- cbind(training\_set, onehot\_enc\_training)  
  
# One-hot encoding the test set  
test\_set$booking\_day\_of\_week <- as.character(test\_set$booking\_day\_of\_week)  
test\_set$booking\_month <- as.character(test\_set$booking\_month)  
test\_set$arrival\_day\_of\_week <- as.character(test\_set$arrival\_day\_of\_week)  
test\_set$arrival\_month <- as.character(test\_set$arrival\_month)  
  
test\_set$type\_of\_meal\_plan <- factor(test\_set$type\_of\_meal\_plan)  
test\_set$room\_type\_reserved <- factor(test\_set$room\_type\_reserved)  
test\_set$market\_segment\_type <- factor(test\_set$market\_segment\_type)  
test\_set$booking\_day\_of\_week <- factor(test\_set$booking\_day\_of\_week)  
test\_set$booking\_month <- factor(test\_set$booking\_month)  
test\_set$arrival\_day\_of\_week <- factor(test\_set$arrival\_day\_of\_week)  
test\_set$arrival\_month <- factor(test\_set$arrival\_month)  
  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved", "market\_segment\_type",   
 "booking\_day\_of\_week", "booking\_month", "arrival\_day\_of\_week", "arrival\_month")])  
test\_set <- cbind(test\_set, onehot\_enc\_test)  
  
# Scaling test and training sets  
test\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)] <- scale(test\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)],   
 center = apply(training\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)], 2, mean),   
 scale = apply(training\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)], 2, sd))  
training\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)] <- scale(training\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)])  
  
# Convert data sets to tensors  
training\_features <- array(data = unlist(training\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)]),   
 dim = c(nrow(training\_set), 42))  
training\_labels <- array(data = unlist(training\_set[, 15]),   
 dim = c(nrow(training\_set)))  
  
test\_features <- array(data = unlist(test\_set[, -c(5, 7, 9, 15, 16, 18, 19, 21)]),   
 dim = c(nrow(test\_set), 42))  
test\_labels <- array(data = unlist(test\_set[, 15]),   
 dim = c(nrow(test\_set)))  
  
# Remove unnecessary columns from training and test sets for use in linear models  
training\_set <- training\_set[ , -c(5, 7, 9, 16, 18, 19, 21)]  
test\_set <- test\_set[ , -c(5, 7, 9, 16, 18, 19, 21)]

# Building and evaluating models

# Building and evaluating a logistic regression model  
# Model with all predictors  
lm <- glm(booking\_status ~ ., data = training\_set, family = binomial)  
summary(lm)

##   
## Call:  
## glm(formula = booking\_status ~ ., family = binomial, data = training\_set)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.4200377 1.2139538 -1.170 0.242097   
## no\_of\_adults 0.0708876 0.0198360 3.574 0.000352 \*\*\*  
## no\_of\_children 0.0786922 0.0244028 3.225 0.001261 \*\*   
## no\_of\_weekend\_nights 0.1507545 0.0235582 6.399 1.56e-10 \*\*\*  
## no\_of\_week\_nights 0.0544262 0.0201135 2.706 0.006811 \*\*   
## required\_car\_parking\_space -0.3112423 0.0247718 -12.564 < 2e-16 \*\*\*  
## lead\_time 1.5004926 0.0275316 54.501 < 2e-16 \*\*\*  
## repeated\_guest -0.3400069 0.0833084 -4.081 4.48e-05 \*\*\*  
## no\_of\_previous\_cancellations 0.1334791 0.0375561 3.554 0.000379 \*\*\*  
## no\_of\_previous\_bookings\_not\_canceled -0.1582243 0.1325140 -1.194 0.232470   
## avg\_price\_per\_room 0.6982525 0.0277380 25.173 < 2e-16 \*\*\*  
## no\_of\_special\_requests -1.1885109 0.0233622 -50.873 < 2e-16 \*\*\*  
## arrival\_day\_of\_month 0.0480020 0.0172216 2.787 0.005315 \*\*   
## booking\_day\_of\_month 0.0303019 0.0174022 1.741 0.081637 .   
## type\_of\_meal\_plan.meal\_plan\_2 0.0562884 0.0196468 2.865 0.004170 \*\*   
## type\_of\_meal\_plan.meal\_plan\_3 0.0408737 12.4534226 0.003 0.997381   
## type\_of\_meal\_plan.not\_selected 0.1119736 0.0180543 6.202 5.57e-10 \*\*\*  
## room\_type\_reserved.room\_type2 -0.0269717 0.0182873 -1.475 0.140243   
## room\_type\_reserved.room\_type3 -0.0036730 0.0194510 -0.189 0.850222   
## room\_type\_reserved.room\_type4 -0.0853293 0.0197230 -4.326 1.52e-05 \*\*\*  
## room\_type\_reserved.room\_type5 -0.0734463 0.0176275 -4.167 3.09e-05 \*\*\*  
## room\_type\_reserved.room\_type6 -0.1692334 0.0244679 -6.917 4.63e-12 \*\*\*  
## room\_type\_reserved.room\_type7 -0.0915486 0.0198683 -4.608 4.07e-06 \*\*\*  
## market\_segment\_type.complementary -1.5008725 11.6149163 -0.129 0.897184   
## market\_segment\_type.corporate -0.1847399 0.0636818 -2.901 0.003720 \*\*   
## market\_segment\_type.offline -0.8773067 0.1224286 -7.166 7.73e-13 \*\*\*  
## market\_segment\_type.online -0.0105177 0.1279954 -0.082 0.934510   
## booking\_day\_of\_week.Mon -0.0754159 0.0219650 -3.433 0.000596 \*\*\*  
## booking\_day\_of\_week.Sat 0.0568772 0.0217147 2.619 0.008811 \*\*   
## booking\_day\_of\_week.Sun -0.0565001 0.0220988 -2.557 0.010567 \*   
## booking\_day\_of\_week.Thu -0.0576407 0.0229400 -2.513 0.011982 \*   
## booking\_day\_of\_week.Tue -0.0112044 0.0201814 -0.555 0.578768   
## booking\_day\_of\_week.Wed 0.0179344 0.0214026 0.838 0.402055   
## booking\_month.Aug 0.0484456 0.0240339 2.016 0.043829 \*   
## booking\_month.Dec -0.1350451 0.0233482 -5.784 7.30e-09 \*\*\*  
## booking\_month.Feb -0.0674523 0.0226445 -2.979 0.002894 \*\*   
## booking\_month.Jan -0.1207611 0.0246289 -4.903 9.43e-07 \*\*\*  
## booking\_month.Jul -0.0282708 0.0224206 -1.261 0.207334   
## booking\_month.Jun -0.0476325 0.0196917 -2.419 0.015567 \*   
## booking\_month.Mar -0.1013660 0.0205729 -4.927 8.34e-07 \*\*\*  
## booking\_month.May 0.0008268 0.0198802 0.042 0.966826   
## booking\_month.Nov -0.0910286 0.0239915 -3.794 0.000148 \*\*\*  
## booking\_month.Oct -0.0971161 0.0244568 -3.971 7.16e-05 \*\*\*  
## booking\_month.Sep -0.1676596 0.0274009 -6.119 9.43e-10 \*\*\*  
## arrival\_day\_of\_week.Mon -0.0769462 0.0259796 -2.962 0.003059 \*\*   
## arrival\_day\_of\_week.Sat -0.0684221 0.0235550 -2.905 0.003675 \*\*   
## arrival\_day\_of\_week.Sun -0.0327900 0.0247151 -1.327 0.184601   
## arrival\_day\_of\_week.Thu -0.0011100 0.0226071 -0.049 0.960840   
## arrival\_day\_of\_week.Tue -0.0649179 0.0289205 -2.245 0.024787 \*   
## arrival\_day\_of\_week.Wed -0.0217141 0.0250714 -0.866 0.386441   
## arrival\_month.Aug -0.1799858 0.0250317 -7.190 6.46e-13 \*\*\*  
## arrival\_month.Dec -0.4772847 0.0298915 -15.967 < 2e-16 \*\*\*  
## arrival\_month.Feb 0.1529759 0.0201365 7.597 3.03e-14 \*\*\*  
## arrival\_month.Jan -0.4018537 0.0440176 -9.129 < 2e-16 \*\*\*  
## arrival\_month.Jul -0.1143228 0.0223489 -5.115 3.13e-07 \*\*\*  
## arrival\_month.Jun -0.0501948 0.0227854 -2.203 0.027599 \*   
## arrival\_month.Mar 0.0802127 0.0205034 3.912 9.15e-05 \*\*\*  
## arrival\_month.May -0.1111325 0.0211689 -5.250 1.52e-07 \*\*\*  
## arrival\_month.Nov 0.0646236 0.0260541 2.480 0.013125 \*   
## arrival\_month.Oct -0.0809941 0.0296179 -2.735 0.006245 \*\*   
## arrival\_month.Sep -0.2014918 0.0284124 -7.092 1.32e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 34377 on 27178 degrees of freedom  
## Residual deviance: 22197 on 27118 degrees of freedom  
## AIC: 22319  
##   
## Number of Fisher Scoring iterations: 15

predict\_lm <- predict(lm, newdata = test\_set)  
binary\_predict\_lm <- ifelse(predict\_lm > 0.5, 1, 0)  
results <- data.frame(  
 Actual = test\_set$booking\_status,   
 Predicted = binary\_predict\_lm  
)  
results$Correct <- results$Actual == results$Predicted  
confusion\_matrix\_lm <- table(Predicted = results$Predicted, Actual = results$Actual)  
print(confusion\_matrix\_lm)

## Actual  
## Predicted 0 1  
## 0 5718 1449  
## 1 365 1527

accuracy\_lm <- (5691 + 1514) / (5714 + 1395 + 382 + 1568)  
error\_lm <- 1 - accuracy\_lm  
cat("Accuracy:", accuracy\_lm, "\n")

## Accuracy: 0.7953416

cat("Error Rate:", error\_lm, "\n")

## Error Rate: 0.2046584

# Model with only significant predictors  
sig\_lm <- glm(booking\_status ~ no\_of\_adults + no\_of\_children + no\_of\_weekend\_nights + no\_of\_week\_nights + required\_car\_parking\_space + lead\_time + repeated\_guest + no\_of\_previous\_cancellations + avg\_price\_per\_room + no\_of\_special\_requests + arrival\_day\_of\_month + type\_of\_meal\_plan.meal\_plan\_2 + type\_of\_meal\_plan.not\_selected + room\_type\_reserved.room\_type2 + room\_type\_reserved.room\_type4 + room\_type\_reserved.room\_type5 + room\_type\_reserved.room\_type6 + room\_type\_reserved.room\_type7 + market\_segment\_type.corporate + market\_segment\_type.offline + booking\_day\_of\_week.Mon + booking\_day\_of\_week.Sat + booking\_month.Dec + booking\_month.Feb + booking\_month.Jan + booking\_month.Jul + booking\_month.Jun + booking\_month.Mar + booking\_month.Nov + booking\_month.Oct + booking\_month.Sep + arrival\_day\_of\_week.Mon + arrival\_day\_of\_week.Sat + arrival\_month.Aug + arrival\_month.Dec + arrival\_month.Feb + arrival\_month.Jan + arrival\_month.Jul + arrival\_month.Jun + arrival\_month.Mar + arrival\_month.May + arrival\_month.Nov + arrival\_month.Oct + arrival\_month.Sep,  
 data = training\_set, family = binomial)  
summary(sig\_lm)

##   
## Call:  
## glm(formula = booking\_status ~ no\_of\_adults + no\_of\_children +   
## no\_of\_weekend\_nights + no\_of\_week\_nights + required\_car\_parking\_space +   
## lead\_time + repeated\_guest + no\_of\_previous\_cancellations +   
## avg\_price\_per\_room + no\_of\_special\_requests + arrival\_day\_of\_month +   
## type\_of\_meal\_plan.meal\_plan\_2 + type\_of\_meal\_plan.not\_selected +   
## room\_type\_reserved.room\_type2 + room\_type\_reserved.room\_type4 +   
## room\_type\_reserved.room\_type5 + room\_type\_reserved.room\_type6 +   
## room\_type\_reserved.room\_type7 + market\_segment\_type.corporate +   
## market\_segment\_type.offline + booking\_day\_of\_week.Mon + booking\_day\_of\_week.Sat +   
## booking\_month.Dec + booking\_month.Feb + booking\_month.Jan +   
## booking\_month.Jul + booking\_month.Jun + booking\_month.Mar +   
## booking\_month.Nov + booking\_month.Oct + booking\_month.Sep +   
## arrival\_day\_of\_week.Mon + arrival\_day\_of\_week.Sat + arrival\_month.Aug +   
## arrival\_month.Dec + arrival\_month.Feb + arrival\_month.Jan +   
## arrival\_month.Jul + arrival\_month.Jun + arrival\_month.Mar +   
## arrival\_month.May + arrival\_month.Nov + arrival\_month.Oct +   
## arrival\_month.Sep, family = binomial, data = training\_set)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.25962 0.02312 -54.473 < 2e-16 \*\*\*  
## no\_of\_adults 0.06896 0.01956 3.526 0.000422 \*\*\*  
## no\_of\_children 0.07733 0.02438 3.171 0.001517 \*\*   
## no\_of\_weekend\_nights 0.11819 0.01773 6.666 2.63e-11 \*\*\*  
## no\_of\_week\_nights 0.07958 0.01759 4.525 6.03e-06 \*\*\*  
## required\_car\_parking\_space -0.31201 0.02476 -12.603 < 2e-16 \*\*\*  
## lead\_time 1.49047 0.02638 56.493 < 2e-16 \*\*\*  
## repeated\_guest -0.38193 0.07844 -4.869 1.12e-06 \*\*\*  
## no\_of\_previous\_cancellations 0.11058 0.02953 3.745 0.000181 \*\*\*  
## avg\_price\_per\_room 0.72115 0.02715 26.566 < 2e-16 \*\*\*  
## no\_of\_special\_requests -1.18489 0.02329 -50.886 < 2e-16 \*\*\*  
## arrival\_day\_of\_month 0.04828 0.01711 2.822 0.004777 \*\*   
## type\_of\_meal\_plan.meal\_plan\_2 0.03995 0.01923 2.077 0.037792 \*   
## type\_of\_meal\_plan.not\_selected 0.11745 0.01800 6.527 6.73e-11 \*\*\*  
## room\_type\_reserved.room\_type2 -0.02517 0.01823 -1.381 0.167364   
## room\_type\_reserved.room\_type4 -0.08992 0.01964 -4.578 4.68e-06 \*\*\*  
## room\_type\_reserved.room\_type5 -0.07487 0.01757 -4.261 2.03e-05 \*\*\*  
## room\_type\_reserved.room\_type6 -0.17311 0.02441 -7.091 1.33e-12 \*\*\*  
## room\_type\_reserved.room\_type7 -0.09537 0.01988 -4.797 1.61e-06 \*\*\*  
## market\_segment\_type.corporate -0.17566 0.02364 -7.430 1.08e-13 \*\*\*  
## market\_segment\_type.offline -0.85909 0.02426 -35.407 < 2e-16 \*\*\*  
## booking\_day\_of\_week.Mon -0.05266 0.01789 -2.944 0.003245 \*\*   
## booking\_day\_of\_week.Sat 0.08019 0.01695 4.731 2.23e-06 \*\*\*  
## booking\_month.Dec -0.14185 0.02156 -6.580 4.70e-11 \*\*\*  
## booking\_month.Feb -0.07777 0.01988 -3.913 9.12e-05 \*\*\*  
## booking\_month.Jan -0.12742 0.02190 -5.819 5.93e-09 \*\*\*  
## booking\_month.Jul -0.05005 0.01909 -2.622 0.008735 \*\*   
## booking\_month.Jun -0.05732 0.01764 -3.250 0.001154 \*\*   
## booking\_month.Mar -0.10908 0.01805 -6.043 1.51e-09 \*\*\*  
## booking\_month.Nov -0.10417 0.02212 -4.709 2.49e-06 \*\*\*  
## booking\_month.Oct -0.11062 0.02166 -5.108 3.26e-07 \*\*\*  
## booking\_month.Sep -0.19154 0.02363 -8.105 5.27e-16 \*\*\*  
## arrival\_day\_of\_week.Mon -0.04620 0.01735 -2.663 0.007753 \*\*   
## arrival\_day\_of\_week.Sat -0.05155 0.01798 -2.867 0.004138 \*\*   
## arrival\_month.Aug -0.17666 0.02451 -7.207 5.70e-13 \*\*\*  
## arrival\_month.Dec -0.47041 0.02946 -15.966 < 2e-16 \*\*\*  
## arrival\_month.Feb 0.14362 0.01991 7.213 5.46e-13 \*\*\*  
## arrival\_month.Jan -0.39271 0.04357 -9.014 < 2e-16 \*\*\*  
## arrival\_month.Jul -0.11625 0.02213 -5.253 1.50e-07 \*\*\*  
## arrival\_month.Jun -0.05854 0.02249 -2.603 0.009236 \*\*   
## arrival\_month.Mar 0.07864 0.02044 3.847 0.000120 \*\*\*  
## arrival\_month.May -0.11633 0.02095 -5.552 2.82e-08 \*\*\*  
## arrival\_month.Nov 0.07166 0.02539 2.822 0.004774 \*\*   
## arrival\_month.Oct -0.07530 0.02873 -2.621 0.008776 \*\*   
## arrival\_month.Sep -0.19409 0.02735 -7.098 1.27e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 34377 on 27178 degrees of freedom  
## Residual deviance: 22259 on 27134 degrees of freedom  
## AIC: 22349  
##   
## Number of Fisher Scoring iterations: 7

predict\_sig\_lm <- predict(sig\_lm, newdata = test\_set)  
binary\_predict\_sig\_lm <- ifelse(predict\_sig\_lm > 0.5, 1, 0)  
results\_sig <- data.frame(  
 Actual = test\_set$booking\_status,   
 Predicted = binary\_predict\_sig\_lm  
)  
results\_sig$Correct <- results\_sig$Actual == results\_sig$Predicted  
confusion\_matrix\_sig\_lm <- table(Predicted = results\_sig$Predicted, Actual = results\_sig$Actual)  
print(confusion\_matrix\_sig\_lm)

## Actual  
## Predicted 0 1  
## 0 5704 1445  
## 1 379 1531

accuracy\_sig\_lm <- (5695 + 1518) / (5712 + 1395 + 384 + 1568)  
error\_sig\_lm <- 1 - accuracy\_sig\_lm  
cat("Accuracy:", accuracy\_sig\_lm, "\n")

## Accuracy: 0.7962247

cat("Error Rate:", error\_sig\_lm, "\n")

## Error Rate: 0.2037753

# Building and evaluating a K-Nearest Neighbors (KNN) model  
# Model with all predictors and K = 3  
predictors <- training\_set[, -which(names(training\_set) == "booking\_status")]  
label <- training\_set$booking\_status  
k <- 3  
knn\_model <- knn(train = predictors, test = predictors, cl = label, k = k)  
knn\_predictions <- knn(  
 train = training\_set[, -length(predictors)],  
 test = test\_set[, -length(predictors)],  
 cl = training\_set$booking\_status,  
 k = k  
)  
knn\_results <- data.frame(  
 Actual = test\_set$booking\_status,  
 Predicted = knn\_predictions  
)  
knn\_results$Correct <- knn\_results$Actual == knn\_results$Predicted  
knn\_confusion\_matrix <- table(Predicted = knn\_results$Predicted, Actual = knn\_results$Actual)  
print(knn\_confusion\_matrix)

## Actual  
## Predicted 0 1  
## 0 5443 872  
## 1 640 2104

accuracy\_knn <- (5520 + 2140) / (5519 + 849 + 577 + 2114)  
error\_knn <- 1 - accuracy\_knn  
cat("Accuracy:", accuracy\_knn, "\n")

## Accuracy: 0.8455679

cat("Error Rate:", error\_knn, "\n")

## Error Rate: 0.1544321

# Model with all predictors and K = 5  
k <- 5  
knn\_model <- knn(train = predictors, test = predictors, cl = label, k = k)  
knn\_predictions <- knn(  
 train = training\_set[, -length(predictors)],  
 test = test\_set[, -length(predictors)],  
 cl = training\_set$booking\_status,  
 k = k  
)  
knn\_results <- data.frame(  
 Actual = test\_set$booking\_status,  
 Predicted = knn\_predictions  
)  
knn\_results$Correct <- knn\_results$Actual == knn\_results$Predicted  
knn\_confusion\_matrix <- table(Predicted = knn\_results$Predicted, Actual = knn\_results$Actual)  
print(knn\_confusion\_matrix)

## Actual  
## Predicted 0 1  
## 0 5510 890  
## 1 573 2086

accuracy\_knn <- (5539 + 2072) / (5541 + 876 + 555 + 2087)  
error\_knn <- 1 - accuracy\_knn  
cat("Accuracy:", accuracy\_knn, "\n")

## Accuracy: 0.840159

cat("Error Rate:", error\_knn, "\n")

## Error Rate: 0.159841

# Model with all predictors and K = 10  
k <- 10  
knn\_model <- knn(train = predictors, test = predictors, cl = label, k = k)  
knn\_predictions <- knn(  
 train = training\_set[, -length(predictors)],  
 test = test\_set[, -length(predictors)],  
 cl = training\_set$booking\_status,  
 k = k  
)  
knn\_results <- data.frame(  
 Actual = test\_set$booking\_status,  
 Predicted = knn\_predictions  
)  
knn\_results$Correct <- knn\_results$Actual == knn\_results$Predicted  
knn\_confusion\_matrix <- table(Predicted = knn\_results$Predicted, Actual = knn\_results$Actual)  
print(knn\_confusion\_matrix)

## Actual  
## Predicted 0 1  
## 0 5580 956  
## 1 503 2020

accuracy\_knn <- (5616 + 2016) / (5629 + 962 + 467 + 2001)  
error\_knn <- 1 - accuracy\_knn  
cat("Accuracy:", accuracy\_knn, "\n")

## Accuracy: 0.8424771

cat("Error Rate:", error\_knn, "\n")

## Error Rate: 0.1575229

# Building and evaluating a classification tree model  
set.seed(123)  
rf <- randomForest(booking\_status ~ ., data = training\_set, mtry = 4, importance = TRUE, ntree = 25, type = "classification")

## Warning in randomForest.default(m, y, ...): The response has five or fewer  
## unique values. Are you sure you want to do regression?

rf

##   
## Call:  
## randomForest(formula = booking\_status ~ ., data = training\_set, mtry = 4, importance = TRUE, ntree = 25, type = "classification")   
## Type of random forest: regression  
## Number of trees: 25  
## No. of variables tried at each split: 4  
##   
## Mean of squared residuals: 0.09913929  
## % Var explained: 54.99

importance(rf)

## %IncMSE IncNodePurity  
## no\_of\_adults 7.7121676 6.853667e+01  
## no\_of\_children 6.1392842 2.499931e+01  
## no\_of\_weekend\_nights 9.8156294 9.066401e+01  
## no\_of\_week\_nights 8.7949003 1.326287e+02  
## required\_car\_parking\_space 8.1093038 2.935503e+01  
## lead\_time 16.2625047 9.212083e+02  
## repeated\_guest 4.3468597 8.649379e+00  
## no\_of\_previous\_cancellations 2.6700299 2.157938e+00  
## no\_of\_previous\_bookings\_not\_canceled 3.6685445 6.441178e+00  
## avg\_price\_per\_room 12.6012592 3.613485e+02  
## no\_of\_special\_requests 15.0696671 4.296998e+02  
## arrival\_day\_of\_month 16.5324035 1.753905e+02  
## booking\_day\_of\_month 13.0374162 2.059237e+02  
## type\_of\_meal\_plan.meal\_plan\_2 8.0110473 4.926116e+01  
## type\_of\_meal\_plan.meal\_plan\_3 0.0000000 0.000000e+00  
## type\_of\_meal\_plan.not\_selected 7.5873692 3.426121e+01  
## room\_type\_reserved.room\_type2 5.2492408 9.869068e+00  
## room\_type\_reserved.room\_type3 0.0000000 9.533189e-03  
## room\_type\_reserved.room\_type4 10.6843017 3.335472e+01  
## room\_type\_reserved.room\_type5 4.6162966 5.829810e+00  
## room\_type\_reserved.room\_type6 2.2321263 9.650285e+00  
## room\_type\_reserved.room\_type7 -0.4208946 1.237084e+00  
## market\_segment\_type.complementary 3.2603051 6.822365e+00  
## market\_segment\_type.corporate 4.7965452 2.918097e+01  
## market\_segment\_type.offline 9.6846817 7.641475e+01  
## market\_segment\_type.online 6.9401194 1.098621e+02  
## booking\_day\_of\_week.Mon 4.8252457 3.388445e+01  
## booking\_day\_of\_week.Sat 9.9089229 3.972156e+01  
## booking\_day\_of\_week.Sun 8.3415623 3.038232e+01  
## booking\_day\_of\_week.Thu 7.2341710 3.776506e+01  
## booking\_day\_of\_week.Tue 5.0073026 2.190781e+01  
## booking\_day\_of\_week.Wed 6.5195515 3.083216e+01  
## booking\_month.Aug 5.7991068 3.114505e+01  
## booking\_month.Dec 7.3403119 3.025253e+01  
## booking\_month.Feb 7.5510027 3.507522e+01  
## booking\_month.Jan 5.7560812 3.476318e+01  
## booking\_month.Jul 4.9756364 2.817256e+01  
## booking\_month.Jun 5.7940843 1.881424e+01  
## booking\_month.Mar 6.4304810 2.776489e+01  
## booking\_month.May 7.0310809 2.213184e+01  
## booking\_month.Nov 5.5539429 2.570861e+01  
## booking\_month.Oct 6.5281885 4.459255e+01  
## booking\_month.Sep 9.1810145 7.115736e+01  
## arrival\_day\_of\_week.Mon 10.9047952 2.939105e+01  
## arrival\_day\_of\_week.Sat 10.3946355 2.778081e+01  
## arrival\_day\_of\_week.Sun 8.8097260 3.301505e+01  
## arrival\_day\_of\_week.Thu 6.0681919 2.717870e+01  
## arrival\_day\_of\_week.Tue 8.5275074 2.815597e+01  
## arrival\_day\_of\_week.Wed 8.7189076 2.837586e+01  
## arrival\_month.Aug 4.3311332 2.583088e+01  
## arrival\_month.Dec 9.9322154 6.699225e+01  
## arrival\_month.Feb 6.1271789 1.673804e+01  
## arrival\_month.Jan 6.9756135 3.163992e+01  
## arrival\_month.Jul 7.5530006 2.537983e+01  
## arrival\_month.Jun 7.1975800 3.091923e+01  
## arrival\_month.Mar 4.0148751 2.204311e+01  
## arrival\_month.May 7.0374151 2.569660e+01  
## arrival\_month.Nov 7.3361546 2.878791e+01  
## arrival\_month.Oct 13.1056168 3.913780e+01  
## arrival\_month.Sep 10.7824909 3.029541e+01

rf\_predictions <- predict(rf, test\_set, type = "class")  
rf\_results <- data.frame(  
 Actual = test\_set$booking\_status,  
 Predicted = rf\_predictions  
)  
rf\_predictions <- factor(ifelse(rf\_predictions >= 0.5, 1, 0))  
test\_set$booking\_status <- factor(test\_set$booking\_status)  
levels(rf\_predictions) <- levels(test\_set$booking\_status)  
confusion\_mat <- confusionMatrix(rf\_predictions, test\_set$booking\_status)  
print(confusion\_mat)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 5794 815  
## 1 289 2161  
##   
## Accuracy : 0.8781   
## 95% CI : (0.8712, 0.8848)  
## No Information Rate : 0.6715   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7107   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9525   
## Specificity : 0.7261   
## Pos Pred Value : 0.8767   
## Neg Pred Value : 0.8820   
## Prevalence : 0.6715   
## Detection Rate : 0.6396   
## Detection Prevalence : 0.7296   
## Balanced Accuracy : 0.8393   
##   
## 'Positive' Class : 0   
##

accuracy\_tree\_rf <- (5800 + 2186) / (5893 + 804 + 250 + 2112)  
error\_tree\_rf <- 1 - accuracy\_tree\_rf  
cat("Accuracy:", accuracy\_tree\_rf, "\n")

## Accuracy: 0.8815543

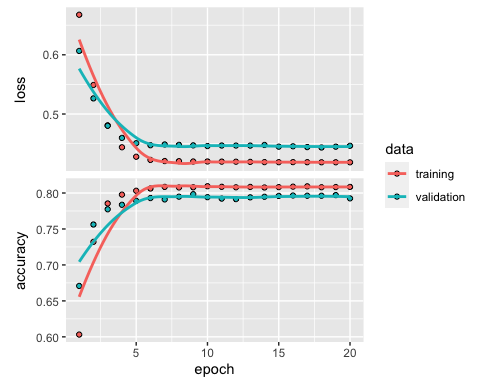
cat("Error Rate:", error\_tree\_rf, "\n")

## Error Rate: 0.1184457

# Building and evaluating a neural network model  
model <- keras\_model\_sequential(list(  
 layer\_dense(units = 20, activation = "relu"),   
 layer\_dense(units = 10, activation = "relu"),   
 layer\_dense(units = 1, activation = "sigmoid")  
))  
compile(model,   
 optimizer = "rmsprop",   
 loss = "binary\_crossentropy",   
 metrics = "accuracy")  
  
# Training the model  
history <- fit(model, training\_features, training\_labels,   
 epochs = 20, batch\_size = 512, validation\_split = 0.33)

## Epoch 1/20  
## 36/36 - 2s - loss: 0.6676 - accuracy: 0.6032 - val\_loss: 0.6065 - val\_accuracy: 0.6708 - 2s/epoch - 45ms/step  
## Epoch 2/20  
## 36/36 - 0s - loss: 0.5490 - accuracy: 0.7321 - val\_loss: 0.5264 - val\_accuracy: 0.7562 - 415ms/epoch - 12ms/step  
## Epoch 3/20  
## 36/36 - 0s - loss: 0.4809 - accuracy: 0.7854 - val\_loss: 0.4801 - val\_accuracy: 0.7774 - 423ms/epoch - 12ms/step  
## Epoch 4/20  
## 36/36 - 0s - loss: 0.4438 - accuracy: 0.7977 - val\_loss: 0.4597 - val\_accuracy: 0.7836 - 413ms/epoch - 11ms/step  
## Epoch 5/20  
## 36/36 - 0s - loss: 0.4278 - accuracy: 0.8031 - val\_loss: 0.4507 - val\_accuracy: 0.7892 - 398ms/epoch - 11ms/step  
## Epoch 6/20  
## 36/36 - 0s - loss: 0.4225 - accuracy: 0.8062 - val\_loss: 0.4473 - val\_accuracy: 0.7930 - 396ms/epoch - 11ms/step  
## Epoch 7/20  
## 36/36 - 0s - loss: 0.4206 - accuracy: 0.8085 - val\_loss: 0.4486 - val\_accuracy: 0.7911 - 427ms/epoch - 12ms/step  
## Epoch 8/20  
## 36/36 - 0s - loss: 0.4204 - accuracy: 0.8077 - val\_loss: 0.4479 - val\_accuracy: 0.7949 - 401ms/epoch - 11ms/step  
## Epoch 9/20  
## 36/36 - 0s - loss: 0.4198 - accuracy: 0.8080 - val\_loss: 0.4471 - val\_accuracy: 0.7983 - 395ms/epoch - 11ms/step  
## Epoch 10/20  
## 36/36 - 0s - loss: 0.4199 - accuracy: 0.8094 - val\_loss: 0.4459 - val\_accuracy: 0.7942 - 396ms/epoch - 11ms/step  
## Epoch 11/20  
## 36/36 - 0s - loss: 0.4196 - accuracy: 0.8086 - val\_loss: 0.4468 - val\_accuracy: 0.7922 - 398ms/epoch - 11ms/step  
## Epoch 12/20  
## 36/36 - 0s - loss: 0.4195 - accuracy: 0.8080 - val\_loss: 0.4467 - val\_accuracy: 0.7915 - 404ms/epoch - 11ms/step  
## Epoch 13/20  
## 36/36 - 0s - loss: 0.4191 - accuracy: 0.8084 - val\_loss: 0.4467 - val\_accuracy: 0.7940 - 395ms/epoch - 11ms/step  
## Epoch 14/20  
## 36/36 - 0s - loss: 0.4188 - accuracy: 0.8077 - val\_loss: 0.4478 - val\_accuracy: 0.7948 - 392ms/epoch - 11ms/step  
## Epoch 15/20  
## 36/36 - 0s - loss: 0.4187 - accuracy: 0.8082 - val\_loss: 0.4448 - val\_accuracy: 0.7960 - 402ms/epoch - 11ms/step  
## Epoch 16/20  
## 36/36 - 0s - loss: 0.4188 - accuracy: 0.8088 - val\_loss: 0.4455 - val\_accuracy: 0.7965 - 410ms/epoch - 11ms/step  
## Epoch 17/20  
## 36/36 - 0s - loss: 0.4187 - accuracy: 0.8093 - val\_loss: 0.4442 - val\_accuracy: 0.7962 - 392ms/epoch - 11ms/step  
## Epoch 18/20  
## 36/36 - 0s - loss: 0.4185 - accuracy: 0.8080 - val\_loss: 0.4434 - val\_accuracy: 0.7962 - 394ms/epoch - 11ms/step  
## Epoch 19/20  
## 36/36 - 0s - loss: 0.4185 - accuracy: 0.8083 - val\_loss: 0.4448 - val\_accuracy: 0.7971 - 396ms/epoch - 11ms/step  
## Epoch 20/20  
## 36/36 - 0s - loss: 0.4185 - accuracy: 0.8084 - val\_loss: 0.4464 - val\_accuracy: 0.7925 - 394ms/epoch - 11ms/step

plot(history)



# Using the model to make predictions  
predictions <- predict(model, test\_features)

## 284/284 - 0s - 431ms/epoch - 2ms/step

test\_set$p\_prob <- predictions[, 1]  
head(predictions, 10)

## [,1]  
## [1,] 0.10934297  
## [2,] 0.93055540  
## [3,] 0.96812302  
## [4,] 0.21342574  
## [5,] 0.23408312  
## [6,] 0.07456930  
## [7,] 0.87598109  
## [8,] 0.46272993  
## [9,] 0.21923907  
## [10,] 0.07389656

predicted\_class <- (predictions[, 1] >= 0.5) \* 1  
head(predicted\_class, 10)

## [1] 0 1 1 0 0 0 1 0 0 0

# Calculating accuracy  
accuracy <- mean(predicted\_class == test\_labels)  
accuracy

## [1] 0.8053869

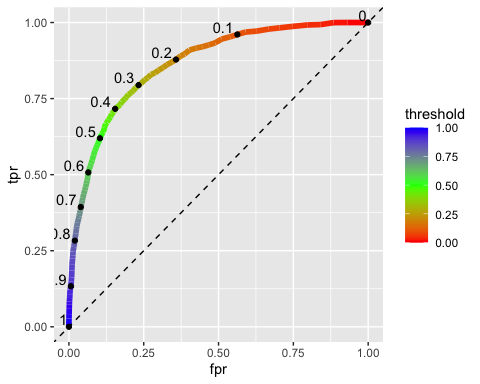
# Making predictions and calculating fpr and tpr rates at 0.5 threshold  
over\_threshold <- test\_set[test\_set$p\_prob >= 0.5, ]  
fpr <- sum(over\_threshold$booking\_status==0)/sum(test\_set$booking\_status==0)  
fpr

## [1] 0.1037317

tpr <- sum(over\_threshold$booking\_status==1)/sum(test\_set$booking\_status==1)  
tpr

## [1] 0.6196237

# Plotting ROC curve  
roc\_data <- data.frame(threshold = seq(1, 0, -0.01), fpr = 0, tpr = 0)  
for (i in roc\_data$threshold) {  
 over\_threshold <- test\_set[test\_set$p\_prob >= i, ]  
 fpr <- sum(over\_threshold$booking\_status==0)/sum(test\_set$booking\_status==0)  
 roc\_data[roc\_data$threshold==i, "fpr"] <- fpr  
 tpr <- sum(over\_threshold$booking\_status==1)/sum(test\_set$booking\_status==1)  
 roc\_data[roc\_data$threshold==i, "tpr"] <- tpr  
}  
ggplot() +   
 geom\_line(data = roc\_data,   
 aes(x = fpr, y = tpr, color = threshold), linewidth = 2) +   
 scale\_color\_gradientn(colors = rainbow(3)) +   
 geom\_abline(intercept = 0, slope = 1, lty = 2) +   
 geom\_point(data = roc\_data[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +   
 geom\_text(data = roc\_data[seq(1, 101, 10), ],   
 aes(x = fpr, y = tpr, label = threshold, hjust = 1.2, vjust = -0.2))



# Calculating the AUC  
auc <- auc(x = roc\_data$fpr, y = roc\_data$tpr, type = "spline")

## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to unique  
## 'x' values

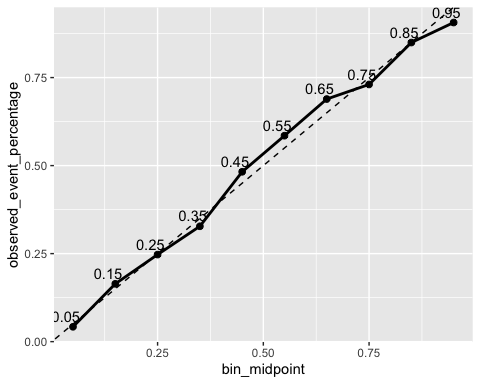
auc

## [1] 0.8632864

# Creating a calibration curve  
in\_interval <- test\_set[test\_set$p\_prob >= 0.7 & test\_set$p\_prob <= 0.8, ]  
nrow(in\_interval[in\_interval$booking\_status==1, ])/nrow(in\_interval)

## [1] 0.7305122

calibration\_data <- data.frame(bin\_midpoint=seq(0.05,0.95,0.1),  
 observed\_event\_percentage=0)  
for (i in seq(0.05,0.95,0.1)) {  
 in\_interval <- test\_set[test\_set$p\_prob >= (i-0.05) & test\_set$p\_prob <= (i+0.05), ]  
 oep <- nrow(in\_interval[in\_interval$booking\_status==1, ])/nrow(in\_interval)  
 calibration\_data[calibration\_data$bin\_midpoint==i, "observed\_event\_percentage"] <- oep  
}  
ggplot(data = calibration\_data, aes(x = bin\_midpoint, y = observed\_event\_percentage)) +  
 geom\_line(linewidth = 1) +  
 geom\_abline(intercept = 0, slope = 1, lty = 2) +  
 geom\_point(size = 2) +  
 geom\_text(aes(label = bin\_midpoint), hjust = 0.75, vjust = -0.5)



# Table with models and relative accuracies  
classification\_overview <- data.frame(  
 Method = c("Logistic Regression", "kNN (k = 3)", "Random Forest", "Neural Network"),  
 Accuracy = c("79.53%", "84.56%", "88.16%", "79.92%")  
)  
classification\_table <- kable(classification\_overview, "markdown") %>%  
 kable\_styling(full\_width = FALSE) %>%  
 column\_spec(1, bold = TRUE)

## Warning in kable\_styling(., full\_width = FALSE): Please specify format in  
## kable. kableExtra can customize either HTML or LaTeX outputs. See  
## https://haozhu233.github.io/kableExtra/ for details.

## Warning in column\_spec(., 1, bold = TRUE): Please specify format in kable.  
## kableExtra can customize either HTML or LaTeX outputs. See  
## https://haozhu233.github.io/kableExtra/ for details.

classification\_table

| Method | Accuracy |
| --- | --- |
| Logistic Regression | 79.53% |
| kNN (k = 3) | 84.56% |
| Random Forest | 88.16% |
| Neural Network | 79.92% |