Predictive modeling of customer bookings

Rounak Saha

2025-01-26

Introduction

Customers are more empowered than ever because they have access to a wealth of information at their fingertips. This is one of the reasons the buying cycle is very different to what it used to be. Today, if you're hoping that a customer purchases your flights or holidays as they come into the airport, you've already lost! Being reactive in this situation is not ideal; airlines must be proactive in order to acquire customers before they embark on their holiday.

Task

With your predictive model, it is important to interpret the results in order to understand how "predictive" the data really was and whether we can feasibly use it to predict the target outcome (customers buying holidays). Therefore, you should evaluate the model's performance and output how each variable contributes to the predictive model's power.

```r
# Install all the neccessary packages
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("tidyverse")
install.packages("caret")        # split the data
install.packages("caTools")      # K-fold cross validation
install.packages("xgboost")      # Xgboost ml model
install.packages("countrycode")  #Convert Country Names and Country Codes
install.packages("ggplot2")      # plot the graph
install.packages("gridExtra") #arrange multiple grid-based plots on a page
install.packages("e1071") #for log transformation
install.packages("fastDummies") # for one-hot encoding
```

```r
# Load the library
library(tidyverse)
library(caret)
library(caTools)
library(xgboost)
library(countrycode)
library(ggplot2)
library(gridExtra)
library(e1071)
library(fastDummies)
```

About the data : To provide more context, below is a more detailed data description, explaining exactly what each column means:

num_passengers = number of passengers travelling

sales_channel = sales channel booking was made on

trip_type = trip Type (Round Trip, One Way, Circle Trip)

purchase_lead = number of days between travel date and booking date

length_of_stay = number of days spent at destination

flight_hour = hour of flight departure

flight_day = day of week of flight departure

route = origin = destination flight route

booking_origin = country from where booking was made

wants_extra_baggage = if the customer wanted extra baggage in the booking

wants_preferred_seat = if the customer wanted a preferred seat in the booking

wants_in_flight_meals = if the customer wanted in-flight meals in the booking

flight_duration = total duration of flight (in hours)

booking_complete = flag indicating if the customer completed the booking

```
#Load the dataset and explore
customer_booking <- read_csv("customer_booking.csv")
```

```
## Rows: 50000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): sales_channel, trip_type, flight_day, route, booking_origin
## dbl (9): num_passengers, purchase_lead, length_of_stay, flight_hour, wants_e...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#View(customer_booking)

head(customer_booking)
```

```
## # A tibble: 6 x 14
##   num_passengers sales_channel trip_type purchase_lead length_of_stay
##            <dbl> <chr>         <chr>             <dbl>          <dbl>
## 1              2 Internet      RoundTrip           262             19
## 2              1 Internet      RoundTrip           112             20
## 3              2 Internet      RoundTrip           243             22
## 4              1 Internet      RoundTrip            96             31
## 5              2 Internet      RoundTrip            68             22
## 6              1 Internet      RoundTrip             3             48
## # i 9 more variables: flight_hour <dbl>, flight_day <chr>, route <chr>,
## #   booking_origin <chr>, wants_extra_baggage <dbl>,
## #   wants_preferred_seat <dbl>, wants_in_flight_meals <dbl>,
## #   flight_duration <dbl>, booking_complete <dbl>
```

```
str(customer_booking)
```

```
## spc_tbl_ [50,000 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ num_passengers       : num [1:50000] 2 1 2 1 2 1 3 2 1 1 ...
##  $ sales_channel        : chr [1:50000] "Internet" "Internet" "Internet" "Internet" ...
##  $ trip_type            : chr [1:50000] "RoundTrip" "RoundTrip" "RoundTrip" "RoundTrip" ...
##  $ purchase_lead        : num [1:50000] 262 112 243 96 68 3 201 238 80 378 ...
##  $ length_of_stay       : num [1:50000] 19 20 22 31 22 48 33 19 22 30 ...
##  $ flight_hour          : num [1:50000] 7 3 17 4 15 20 6 14 4 12 ...
##  $ flight_day           : chr [1:50000] "Sat" "Sat" "Wed" "Sat" ...
##  $ route                : chr [1:50000] "AKLDEL" "AKLDEL" "AKLDEL" "AKLDEL" ...
##  $ booking_origin       : chr [1:50000] "New Zealand" "New Zealand" "India" "New Zealand" ...
##  $ wants_extra_baggage  : num [1:50000] 1 0 1 0 1 1 1 1 0 0 ...
##  $ wants_preferred_seat : num [1:50000] 0 0 1 0 0 0 0 0 0 0 ...
##  $ wants_in_flight_meals: num [1:50000] 0 0 0 1 1 1 1 1 1 0 ...
##  $ flight_duration      : num [1:50000] 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 ...
##  $ booking_complete     : num [1:50000] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   num_passengers = col_double(),
##   ..   sales_channel = col_character(),
##   ..   trip_type = col_character(),
##   ..   purchase_lead = col_double(),
##   ..   length_of_stay = col_double(),
##   ..   flight_hour = col_double(),
##   ..   flight_day = col_character(),
##   ..   route = col_character(),
##   ..   booking_origin = col_character(),
##   ..   wants_extra_baggage = col_double(),
##   ..   wants_preferred_seat = col_double(),
##   ..   wants_in_flight_meals = col_double(),
##   ..   flight_duration = col_double(),
##   ..   booking_complete = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
unique(customer_booking$sales_channel)
```

```
## [1] "Internet" "Mobile"
```

```
unique(customer_booking$trip_type)
```

```
## [1] "RoundTrip"  "CircleTrip" "OneWay"
```

```
unique(customer_booking$route)
```

```
##   [1] "AKLDEL" "AKLHGH" "AKLHND" "AKLICN" "AKLKIX" "AKLKTM" "AKLKUL" "AKLMRU"
##   [9] "AKLPEK" "AKLPVG" "AKLTPE" "AORICN" "AORKIX" "AORKTM" "AORMEL" "BBIMEL"
##  [17] "BBIOOL" "BBIPER" "BBISYD" "BDOCTS" "BDOCTU" "BDOHGH" "BDOICN" "BDOIKA"
##  [25] "BDOKIX" "BDOMEL" "BDOOOL" "BDOPEK" "BDOPER" "BDOPUS" "BDOPVG" "BDOSYD"
##  [33] "BDOTPE" "BDOXIY" "BKICKG" "BKICTS" "BKICTU" "BKIHND" "BKIICN" "BKIKIX"
##  [41] "BKIKTM" "BKIMEL" "BKIMRU" "BKIOOL" "BKIPEK" "BKIPER" "BKIPUS" "BKIPVG"
##  [49] "BKISYD" "BKIXIY" "BLRICN" "BLRMEL" "BLRPER" "BLRSYD" "BOMMEL" "BOMOOL"
##  [57] "BOMPER" "BOMSYD" "BTJJED" "BTUICN" "BTUPER" "BTUSYD" "BTUWUH" "BWNCKG"
##  [65] "BWNDEL" "BWNHGH" "BWNIKA" "BWNKTM" "BWNMEL" "BWNOOL" "BWNPER" "BWNSYD"
##  [73] "BWNTPE" "CANDEL" "CANIKA" "CANMEL" "CANMRU" "CANOOL" "CANPER" "CANSYD"
##  [81] "CCUMEL" "CCUMRU" "CCUOOL" "CCUPER" "CCUSYD" "CCUTPE" "CEBMEL" "CEBOOL"
##  [89] "CEBPER" "CEBSYD" "CGKCKG" "CGKCTS" "CGKCTU" "CGKDEL" "CGKHGH" "CGKHND"
##  [97] "CGKICN" "CGKIKA" "CGKJED" "CGKKIX" "CGKKTM" "CGKMEL" "CGKMRU" "CGKOOL"
## [105] "CGKPEK" "CGKPER" "CGKPUS" "CGKPVG" "CGKSYD" "CGKTPE" "CGKWUH" "CGKXIY"
## [113] "CKGCOK" "CKGDPS" "CKGJHB" "CKGKCH" "CKGLOP" "CKGMAA" "CKGMEL" "CKGMYY"
## [121] "CKGOOL" "CKGPEN" "CKGPER" "CKGPNH" "CKGSBW" "CKGSIN" "CKGSUB" "CKGSYD"
## [129] "CKGTGG" "CKGTRZ" "CKGTWU" "CMBCTS" "CMBCTU" "CMBHGH" "CMBHND" "CMBICN"
## [137] "CMBKIX" "CMBMEL" "CMBMRU" "CMBOOL" "CMBPEK" "CMBPER" "CMBPVG" "CMBSYD"
## [145] "CMBWUH" "CNXHND" "CNXICN" "CNXKIX" "CNXMEL" "CNXOOL" "CNXPEK" "CNXPER"
## [153] "CNXPVG" "CNXSYD" "CNXTPE" "COKCTU" "COKHGH" "COKICN" "COKKIX" "COKMEL"
## [161] "COKOOL" "COKPER" "COKPUS" "COKSYD" "COKTPE" "COKWUH" "CRKMEL" "CRKOOL"
## [169] "CRKSYD" "CSXPER" "CTSDMK" "CTSDPS" "CTSHKT" "CTSJHB" "CTSKBR" "CTSKCH"
## [177] "CTSKNO" "CTSLGK" "CTSMEL" "CTSMYY" "CTSOOL" "CTSPEN" "CTSPER" "CTSSGN"
## [185] "CTSSIN" "CTSSUB" "CTSSYD" "CTUDPS" "CTUHKT" "CTUIKA" "CTUJHB" "CTUKBV"
## [193] "CTUKCH" "CTUKNO" "CTUMAA" "CTUMEL" "CTUMRU" "CTUMYY" "CTUOOL" "CTUPEN"
## [201] "CTUPER" "CTUSBW" "CTUSIN" "CTUSUB" "CTUSYD" "CTUTGG" "CTUTRZ" "CTUTWU"
## [209] "CXRMEL" "DACHGH" "DACHND" "DACICN" "DACKIX" "DACMEL" "DACOOL" "DACPER"
## [217] "DACSYD" "DACTPE" "DADMEL" "DADOOL" "DADSYD" "DELDMK" "DELDPS" "DELHKG"
## [225] "DELHKT" "DELHND" "DELJHB" "DELJOG" "DELKBV" "DELKCH" "DELKIX" "DELKNO"
## [233] "DELLGK" "DELMEL" "DELMFM" "DELMNL" "DELMRU" "DELMYY" "DELOOL" "DELPEN"
## [241] "DELPER" "DELPNH" "DELSBW" "DELSGN" "DELSIN" "DELSUB" "DELSYD" "DELSZX"
## [249] "DMKHGH" "DMKHND" "DMKICN" "DMKIKA" "DMKKIX" "DMKKTM" "DMKMEL" "DMKMRU"
## [257] "DMKOOL" "DMKPEK" "DMKPER" "DMKPUS" "DMKPVG" "DMKSYD" "DMKTPE" "DPSHGH"
## [265] "DPSHND" "DPSICN" "DPSIKA" "DPSKIX" "DPSKTM" "DPSMEL" "DPSMRU" "DPSOOL"
## [273] "DPSPEK" "DPSPUS" "DPSPVG" "DPSSYD" "DPSTPE" "DPSXIY" "GOIKUL" "GOIMEL"
## [281] "GOIOOL" "GOIPER" "GOISYD" "HANKTM" "HANMEL" "HANOOL" "HANPER" "HANSYD"
## [289] "HDYHGH" "HDYKTM" "HDYMEL" "HDYOOL" "HDYPEK" "HDYPER" "HDYPVG" "HDYSYD"
## [297] "HDYTPE" "HGHHKT" "HGHJHB" "HGHJOG" "HGHKBR" "HGHKBV" "HGHKCH" "HGHKNO"
## [305] "HGHLGK" "HGHLOP" "HGHMAA" "HGHMEL" "HGHMYY" "HGHOOL" "HGHPEN" "HGHPER"
## [313] "HGHSBW" "HGHSUB" "HGHSYD" "HGHTRZ" "HKGIKA" "HKGKTM" "HKGMEL" "HKGMRU"
## [321] "HKGOOL" "HKGPER" "HKGSYD" "HKTHND" "HKTICN" "HKTKIX" "HKTKTM" "HKTMEL"
## [329] "HKTMRU" "HKTOOL" "HKTPEK" "HKTPER" "HKTPUS" "HKTPVG" "HKTSYD" "HKTTPE"
## [337] "HKTXIY" "HNDIKA" "HNDJOG" "HNDKBR" "HNDKBV" "HNDKCH" "HNDKNO" "HNDKTM"
## [345] "HNDLGK" "HNDLOP" "HNDMAA" "HNDMEL" "HNDMLE" "HNDOOL" "HNDPEN" "HNDPER"
## [353] "HNDPNH" "HNDREP" "HNDRGN" "HNDSBW" "HNDSGN" "HNDSIN" "HNDSUB" "HNDSYD"
## [361] "HNDTRZ" "HYDMEL" "HYDOOL" "HYDPER" "HYDSYD" "HYDWUH" "ICNIKA" "ICNJED"
## [369] "ICNJHB" "ICNKBR" "ICNKBV" "ICNKCH" "ICNKNO" "ICNKTM" "ICNLGK" "ICNMAA"
## [377] "ICNMEL" "ICNMLE" "ICNMYY" "ICNOOL" "ICNPEN" "ICNPER" "ICNREP" "ICNRGN"
## [385] "ICNSBW" "ICNSDK" "ICNSGN" "ICNSIN" "ICNSUB" "ICNSYD" "ICNTRZ" "ICNVTZ"
## [393] "IKAKCH" "IKAKIX" "IKALOP" "IKAMEL" "IKAMFM" "IKAMNL" "IKAOOL" "IKAPEK"
## [401] "IKAPEN" "IKAPER" "IKAPUS" "IKAPVG" "IKASGN" "IKASIN" "IKASUB" "IKASYD"
## [409] "IKATPE" "JEDJOG" "JEDKNO" "JEDMEL" "JEDMNL" "JEDPDG" "JEDPEN" "JEDSUB"
## [417] "JHBKIX" "JHBKTM" "JHBMEL" "JHBMRU" "JHBPEK" "JHBPUS" "JHBPVG" "JHBSYD"
## [425] "JHBTPE" "JHBWUH" "JHBXIY" "JOGKIX" "JOGKTM" "JOGMEL" "JOGOOL" "JOGPER"
## [433] "JOGPVG" "JOGSYD" "JOGTPE" "KBRKIX" "KBRKTM" "KBRMEL" "KBROOL" "KBRPEK"
## [441] "KBRPER" "KBRPVG" "KBRSYD" "KBRTPE" "KBVKTM" "KBVMEL" "KBVOOL" "KBVPEK"
## [449] "KBVPER" "KBVPVG" "KBVSYD" "KCHKIX" "KCHKTM" "KCHMEL" "KCHMRU" "KCHOOL"
## [457] "KCHPEK" "KCHPER" "KCHPUS" "KCHPVG" "KCHSYD" "KCHTPE" "KCHXIY" "KHHMEL"
## [465] "KHHOOL" "KHHPER" "KHHSYD" "KIXKNO" "KIXKTM" "KIXLGK" "KIXLOP" "KIXMAA"
## [473] "KIXMEL" "KIXMLE" "KIXMYY" "KIXOOL" "KIXPEN" "KIXPER" "KIXPNH" "KIXREP"
## [481] "KIXRGN" "KIXSBW" "KIXSGN" "KIXSIN" "KIXSUB" "KIXSYD" "KIXTGG" "KIXTRZ"
## [489] "KLOMEL" "KLOOOL" "KNOKTM" "KNOMEL" "KNOOOL" "KNOPEK" "KNOPER" "KNOPUS"
## [497] "KNOPVG" "KNOSYD" "KNOTPE" "KNOXIY" "KOSMEL" "KOSOOL" "KOSPEK" "KOSSYD"
## [505] "KTMMEL" "KTMMFM" "KTMMYY" "KTMPEN" "KTMPER" "KTMREP" "KTMSGN" "KTMSIN"
## [513] "KTMSUB" "KTMSYD" "KTMTGG" "KTMTPE" "KTMURT" "KWLPER" "LBUPER" "LGKMEL"
## [521] "LGKOOL" "LGKPER" "LGKPUS" "LGKPVG" "LGKSYD" "LGKTPE" "LOPOOL" "LOPPEK"
## [529] "LOPPVG" "LOPSYD" "LOPTPE" "LOPXIY" "LPQMEL" "LPQOOL" "LPQPER" "LPQTPE"
## [537] "MAAMEL" "MAAMRU" "MAAOOL" "MAAPER" "MAAPVG" "MAASYD" "MAATPE" "MAAWUH"
## [545] "MELMFM" "MELMLE" "MELMNL" "MELMRU" "MELMYY" "MELPEK" "MELPEN" "MELPNH"
## [553] "MELPUS" "MELPVG" "MELREP" "MELRGN" "MELSBW" "MELSGN" "MELSIN" "MELSUB"
## [561] "MELSWA" "MELSZX" "MELTGG" "MELTPE" "MELTRZ" "MELTWU" "MELURT" "MELUTP"
## [569] "MELVTE" "MELVTZ" "MELWUH" "MELXIY" "MFMOOL" "MFMPER" "MFMSYD" "MLEPEK"
## [577] "MLEPER" "MLESYD" "MNLMRU" "MNLOOL" "MNLPER" "MNLSYD" "MRUOOL" "MRUPEK"
## [585] "MRUPEN" "MRUPER" "MRUPVG" "MRUSGN" "MRUSIN" "MRUSUB" "MRUSYD" "MRUSZX"
## [593] "MYYOOL" "MYYPER" "MYYPUS" "MYYSYD" "MYYXIY" "NRTSYD" "OOLPEK" "OOLPEN"
## [601] "OOLPNH" "OOLPUS" "OOLPVG" "OOLREP" "OOLRGN" "OOLSBW" "OOLSDK" "OOLSGN"
## [609] "OOLSIN" "OOLSUB" "OOLSZX" "OOLTGG" "OOLTPE" "OOLTRZ" "OOLTWU" "OOLURT"
## [617] "OOLUTP" "OOLVTE" "OOLWUH" "OOLXIY" "PEKPEN" "PEKPER" "PEKREP" "PEKRGN"
## [625] "PEKSBW" "PEKSIN" "PEKSUB" "PEKSYD" "PEKTGG" "PEKTRZ" "PEKTWU" "PENPER"
## [633] "PENPUS" "PENPVG" "PENSYD" "PENTPE" "PENWUH" "PENXIY" "PERPNH" "PERPUS"
## [641] "PERPVG" "PERREP" "PERRGN" "PERSBW" "PERSDK" "PERSGN" "PERSIN" "PERSWA"
## [649] "PERSZX" "PERTGG" "PERTPE" "PERTRZ" "PERTWU" "PERUTP" "PERVTE" "PERVTZ"
## [657] "PERWUH" "PERXIY" "PNHSYD" "PNHTPE" "PNKTPE" "PUSRGN" "PUSSBW" "PUSSGN"
## [665] "PUSSIN" "PUSSUB" "PUSSYD" "PUSTRZ" "PVGREP" "PVGRGN" "PVGSIN" "PVGSUB"
## [673] "PVGSYD" "PVGTGG" "PVGTWU" "PVGURT" "REPSYD" "REPTPE" "RGNSYD" "RGNTPE"
## [681] "SBWSYD" "SBWTPE" "SBWXIY" "SDKSYD" "SGNSYD" "SGNXIY" "SINSYD" "SINTPE"
## [689] "SINWUH" "SINXIY" "SRGTPE" "SUBSYD" "SUBTPE" "SUBXIY" "SYDSZX" "SYDTPE"
## [697] "SYDTRZ" "SYDTWU" "SYDVTE" "SYDVTZ" "SYDXIY" "TGGTPE" "TGGXIY" "TPETRZ"
## [705] "TPEVTE" "TRZWUH" "TRZXIY" "TWUXIY" "HGHSGN" "ICNTGG" "JHBOOL" "KBRXIY"
## [713] "KBVTPE" "KIXTWU" "LBUTPE" "PVGSGN" "SBWWUH" "DELREP" "DPSWUH" "HKGJED"
## [721] "KBVKIX" "KBVPUS" "KIXLPQ" "LGKPEK" "LGKXIY" "LOPPER" "PEKSGN" "PERSUB"
## [729] "TPETWU" "BDOWUH" "BKIDEL" "CKGSGN" "CTUKBR" "CTULGK" "CTUREP" "DACMRU"
## [737] "DACPEK" "DELRGN" "HDYXIY" "HGHTGG" "HKTWUH" "ICNVTE" "KBRPUS" "KCHWUH"
## [745] "KLOSYD" "KNOWUH" "MLETPE" "SDKTPE" "SUBWUH" "TWUWUH" "AORPUS" "BTUCKG"
## [753] "BWNWUH" "CKGKNO" "CKGLGK" "CNXDEL" "CNXPUS" "CTSJOG" "CTSSBW" "CTUDMK"
## [761] "CTULOP" "DELKBR" "DELURT" "HDYKIX" "HGHSIN" "HGHTWU" "HYDMRU" "IKASZX"
## [769] "KBVWUH" "KBVXIY" "KIXLBU" "LGKWUH" "MELNRT" "MLEOOL" "MRUTPE" "TPEURT"
## [777] "URTXIY" "AORPER" "CKGHKT" "CKGMRU" "CNXXIY" "COKCTS" "CSXMRU" "CSXSYD"
## [785] "CTUMLE" "CTUSGN" "CTUSRG" "CTUURT" "DACPUS" "HGHMRU" "HKTIKA" "HKTJED"
## [793] "ICNMRU" "JEDMFM" "KBRWUH" "KIXMRU" "KTMTWU" "MLEPVG" "MRUXIY"
```

```
unique(customer_booking$booking_origin)
```

```
##  [1] "New Zealand"          "India"                "United Kingdom"
##  [4] "China"                "South Korea"          "Japan"
##  [7] "Malaysia"             "Singapore"            "Switzerland"
## [10] "Germany"              "Indonesia"            "Czech Republic"
## [13] "Vietnam"              "Thailand"             "Spain"
## [16] "Romania"              "Ireland"              "Italy"
## [19] "Slovakia"             "United Arab Emirates" "Tonga"
## [22] "R\xe9union"           "(not set)"            "Saudi Arabia"
## [25] "Netherlands"          "Qatar"                "Hong Kong"
## [28] "Philippines"          "Sri Lanka"            "France"
##  [31] "Croatia"              "United States"        "Laos"
##  [34] "Hungary"              "Portugal"             "Cyprus"
##  [37] "Australia"            "Cambodia"             "Poland"
##  [40] "Belgium"              "Oman"                 "Bangladesh"
##  [43] "Kazakhstan"           "Brazil"               "Turkey"
##  [46] "Kenya"                "Taiwan"               "Brunei"
##  [49] "Chile"                "Bulgaria"             "Ukraine"
##  [52] "Denmark"              "Colombia"             "Iran"
##  [55] "Bahrain"              "Solomon Islands"      "Slovenia"
##  [58] "Mauritius"            "Nepal"                "Russia"
##  [61] "Kuwait"               "Mexico"               "Sweden"
##  [64] "Austria"              "Lebanon"              "Jordan"
##  [67] "Greece"               "Mongolia"             "Canada"
##  [70] "Tanzania"             "Peru"                 "Timor-Leste"
##  [73] "Argentina"            "New Caledonia"        "Macau"
##  [76] "Myanmar (Burma)"      "Norway"               "Panama"
##  [79] "Bhutan"               "Norfolk Island"       "Finland"
##  [82] "Nicaragua"            "Maldives"             "Egypt"
##  [85] "Israel"               "Tunisia"              "South Africa"
##  [88] "Papua New Guinea"     "Paraguay"             "Estonia"
##  [91] "Seychelles"           "Afghanistan"          "Guam"
##  [94] "Czechia"              "Malta"                "Vanuatu"
##  [97] "Belarus"              "Pakistan"             "Iraq"
## [100] "Ghana"                "Gibraltar"            "Guatemala"
## [103] "Algeria"              "Svalbard & Jan Mayen"
```

Data Cleaning and Manupulation

```
# Encoding the categorical variables
#Label Encoding
customer_booking$sales_channel <- as.numeric(factor(customer_booking$sales_channel,
                                                    levels = unique(customer_booking$sales_channel)))-1
#,
                                            #levels = c('Internet', 'Mobile'),
                                            #labels = c(1,2))
#Mapping Encoding
trip_type_mapping <- c("RoundTrip" = 0, "OneWay" = 1, "CircleTrip" = 2)
customer_booking$trip_type <- trip_type_mapping[customer_booking$trip_type]
customer_booking$trip_type<- as.numeric(customer_booking$trip_type)

# Replace specific country name
customer_booking$booking_origin[customer_booking$booking_origin == "Myanmar (Burma)"] <- "Myanmar"

# The booking origin column also has many unique values,
# but because I don't want to delete the information on the origin of the booking,
# I will change the value of the booking origin, which initially contains
# the name of the country to the name of the continent

customer_booking$booking_origin <- countrycode(customer_booking$booking_origin, "country.name", "contin
customer_booking$booking_origin[is.na(customer_booking$booking_origin)] <- "Others"
```

```r
#customer_booking$booking_origin <- factor(customer_booking$booking_origin,
                                        #  levels = c("Oceania","Asia","Europe","Americas", "Africa","O
                                         # labels = c(1:6))
#(its better to do one-hot encoding when there are nominal cat values in col and more in number)

# Drop columns that has many unique values
length(unique(customer_booking$route))
```

```
## [1] 799
```

```r
customer_booking <- customer_booking[,-8]

#Make a new feature Because we want to know customer behavior to have a trip on holiday (weekend),
#so let's make a feature called is_weekend. if the flight day is Saturday or Sunday we give is_weekend
#for another flight day we give it 0
customer_booking <- customer_booking %>%
  mutate(is_weekend = ifelse(flight_day %in% c("Sat", "Sun"), 1, 0))
customer_booking <- customer_booking[,-7]

summary(customer_booking)
```

```
##  num_passengers  sales_channel      trip_type       purchase_lead
##  Min.   :1.000   Min.   :0.0000   Min.   :0.00000   Min.   :  0.00
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 21.00
##  Median :1.000   Median :0.0000   Median :0.00000   Median : 51.00
##  Mean   :1.591   Mean   :0.1124   Mean   :0.01238   Mean   : 84.94
##  3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:115.00
##  Max.   :9.000   Max.   :1.0000   Max.   :2.00000   Max.   :867.00
##  length_of_stay    flight_hour     booking_origin     wants_extra_baggage
##  Min.   :  0.00   Min.   : 0.000   Length:50000       Min.   :0.0000
##  1st Qu.:  5.00   1st Qu.: 5.000   Class :character   1st Qu.:0.0000
##  Median : 17.00   Median : 9.000   Mode  :character   Median :1.0000
##  Mean   : 23.04   Mean   : 9.066                      Mean   :0.6688
##  3rd Qu.: 28.00   3rd Qu.:13.000                      3rd Qu.:1.0000
##  Max.   :778.00   Max.   :23.000                      Max.   :1.0000
##  wants_preferred_seat wants_in_flight_meals flight_duration booking_complete
##  Min.   :0.000        Min.   :0.0000        Min.   :4.670   Min.   :0.0000
##  1st Qu.:0.000        1st Qu.:0.0000        1st Qu.:5.620   1st Qu.:0.0000
##  Median :0.000        Median :0.0000        Median :7.570   Median :0.0000
##  Mean   :0.297        Mean   :0.4271        Mean   :7.278   Mean   :0.1496
##  3rd Qu.:1.000        3rd Qu.:1.0000        3rd Qu.:8.830   3rd Qu.:0.0000
##  Max.   :1.000        Max.   :1.0000        Max.   :9.500   Max.   :1.0000
##    is_weekend
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2473
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

```r
# converting variables to factors
#customer_booking$wants_extra_baggage <- as.factor(customer_booking$wants_extra_baggage)
```

```
#customer_booking$wants_preferred_seat <- as.factor(customer_booking$wants_preferred_seat)
#customer_booking$wants_in_flight_meals <- as.factor(customer_booking$wants_in_flight_meals)
#customer_booking$is_weekend <- as.factor(customer_booking$is_weekend)
#customer_booking$booking_complete<- as.factor(customer_booking$booking_complete)

str(customer_booking)
```

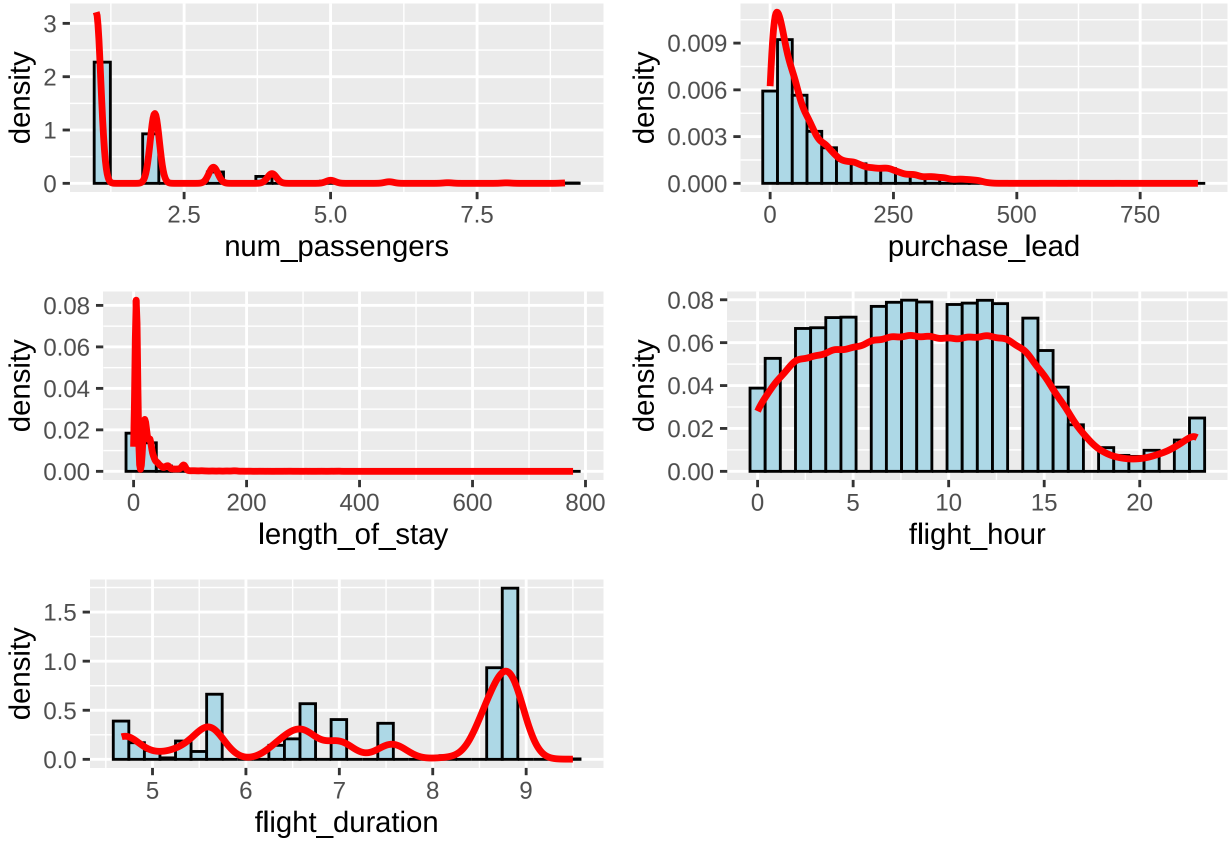
```
## tibble [50,000 x 13] (S3: tbl_df/tbl/data.frame)
##  $ num_passengers       : num [1:50000] 2 1 2 1 2 1 3 2 1 1 ...
##  $ sales_channel        : num [1:50000] 0 0 0 0 0 0 0 0 0 1 ...
##  $ trip_type            : num [1:50000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ purchase_lead        : num [1:50000] 262 112 243 96 68 3 201 238 80 378 ...
##  $ length_of_stay       : num [1:50000] 19 20 22 31 22 48 33 19 22 30 ...
##  $ flight_hour          : num [1:50000] 7 3 17 4 15 20 6 14 4 12 ...
##  $ booking_origin       : chr [1:50000] "Oceania" "Oceania" "Asia" "Oceania" ...
##  $ wants_extra_baggage  : num [1:50000] 1 0 1 0 1 1 1 1 0 0 ...
##  $ wants_preferred_seat : num [1:50000] 0 0 1 0 0 0 0 0 0 0 ...
##  $ wants_in_flight_meals: num [1:50000] 0 0 0 1 1 1 1 1 1 0 ...
##  $ flight_duration      : num [1:50000] 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 ...
##  $ booking_complete     : num [1:50000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ is_weekend           : num [1:50000] 1 1 0 1 0 0 0 0 0 1 ...
```

Data Transformation and Visualization

```
str(customer_booking)
```

```
## tibble [50,000 x 13] (S3: tbl_df/tbl/data.frame)
##  $ num_passengers       : num [1:50000] 2 1 2 1 2 1 3 2 1 1 ...
##  $ sales_channel        : num [1:50000] 0 0 0 0 0 0 0 0 0 1 ...
##  $ trip_type            : num [1:50000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ purchase_lead        : num [1:50000] 262 112 243 96 68 3 201 238 80 378 ...
##  $ length_of_stay       : num [1:50000] 19 20 22 31 22 48 33 19 22 30 ...
##  $ flight_hour          : num [1:50000] 7 3 17 4 15 20 6 14 4 12 ...
##  $ booking_origin       : chr [1:50000] "Oceania" "Oceania" "Asia" "Oceania" ...
##  $ wants_extra_baggage  : num [1:50000] 1 0 1 0 1 1 1 1 0 0 ...
##  $ wants_preferred_seat : num [1:50000] 0 0 1 0 0 0 0 0 0 0 ...
##  $ wants_in_flight_meals: num [1:50000] 0 0 0 1 1 1 1 1 1 0 ...
##  $ flight_duration      : num [1:50000] 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 5.52 ...
##  $ booking_complete     : num [1:50000] 0 0 0 0 0 0 0 0 0 0 ...
##  $ is_weekend           : num [1:50000] 1 1 0 1 0 0 0 0 0 1 ...
```

```
#check the distribution on numerical data
a1<- ggplot(customer_booking,                       # Initializes ggplot() using the dataset customer_b
            aes(x = num_passengers)) +              #Specifies that num_passengers is the variable for
  geom_histogram(aes(y = ..density..),              #By default, histograms show counts (frequency of o
                                                    #..density.. ensures that the histogram is scaled t
                                                    # making it comparable with a density curve
                 bins = 30,                         #Divides the x-axis range into 30 bins (intervals)
                 fill = "lightblue",                #Fills the bars with a light blue color.
                 color = "black") +                 #Adds black borders to each bin for better visibili
  geom_density(                                     #Adds a density curve (smoothed probability distrib
    color = "red",                                  #Makes the curve red for distinction.
    size = 1.2)                                     #Increases the line thickness for better visibility


a2<- ggplot(customer_booking, aes(x = purchase_lead)) +
  geom_histogram(aes(y = ..density..),
                 bins = 30,
                 fill = "lightblue",
                 color = "black") +
  geom_density(color = "red", size = 1.2)

a3<- ggplot(customer_booking, aes(x = length_of_stay)) +
  geom_histogram(aes(y = ..density..),
                 bins = 30,
                 fill = "lightblue",
                 color = "black") +
  geom_density(color = "red", size = 1.2)

a4<- ggplot(customer_booking, aes(x = flight_hour)) +
  geom_histogram(aes(y = ..density..),
                 bins = 30,
                 fill = "lightblue",
                 color = "black") +
  geom_density(color = "red", size = 1.2)

a5<- ggplot(customer_booking, aes(x = flight_duration)) +
  geom_histogram(aes(y = ..density..),
                 bins = 30,
                 fill = "lightblue",
                 color = "black") +
  geom_density(color = "red", size = 1.2)

grid.arrange(a1,a2,a3,a4,a5, nrow= 3, ncol=2)
```



```
###### OR #######

#variables1 <- c("num_passengers", "purchase_lead",
#                "length_of_stay", "flight_hour", "flight_duration")

#plots <- lapply(variables1, function(var){
#  ggplot(customer_booking, aes_string(x= var)) +
#    geom_histogram(aes(y = ..density..),
#                   bins = 30,
#                   fill= "lightblue",
#                   color = "black") +
#    geom_density(color= "red", size= 1.2)+
#    ggtitle(var)
# })
# grid.arrange(grobs= plots, nrow=3, ncol=2)
```

```
# Copy original data
df_transformed <- customer_booking

# Checking the skewness of the numeric variables
skewness(df_transformed$num_passengers)
```

```
## [1] 2.690747
```

```
skewness(df_transformed$purchase_lead)
```

```
## [1] 1.652936
```

```
skewness(df_transformed$length_of_stay)
```

```
## [1] 5.274426
```

```
skewness(df_transformed$flight_hour)
```

```
## [1] 0.3965994
```

```
skewness(df_transformed$flight_duration)
```

```
## [1] -0.3600581
```

```
#skewness(df_transformed$num_passengers) #log
#[1] 2.690747
#skewness(df_transformed$purchase_lead) #sqrt
#[1] 1.652936
#skewness(df_transformed$length_of_stay)# log
#[1] 5.274426
#skewness(df_transformed$flight_hour). #no need
#[1] 0.3965994
#skewness(df_transformed$flight_duration)
#[1] -0.3600581

#Skewness Range Recommended Transformation
#0 to ±0.5  Already normal (No transformation needed)
#0.5 to ±1.5    Log Transformation (log1p(x))
#1.5 to ±3.0    Square Root Transformation (sqrt(x))
#Above 3.0  Box-Cox or Log Transformation (log1p(x))

# Apply Log Transformation to Selected Columns
cols1 <- c("num_passengers", "length_of_stay")

for (col in cols1) {
  df_transformed[[col]] <- log1p(df_transformed[[col]])  # log(x + 1) to avoid log(0)
}                                             #[[col]] allows column selection dynamically inside a loop

# check skewness
skewness(df_transformed$num_passengers)
```

```
## [1] 1.498406
```

```
skewness(df_transformed$length_of_stay)
```

```
## [1] 0.4609978
```

```
# Apply Square Root Transformation to Selected Columns
df_transformed$purchase_lead <- sqrt(df_transformed$purchase_lead)
skewness(df_transformed$purchase_lead)
```

```
## [1] 0.6950384
```

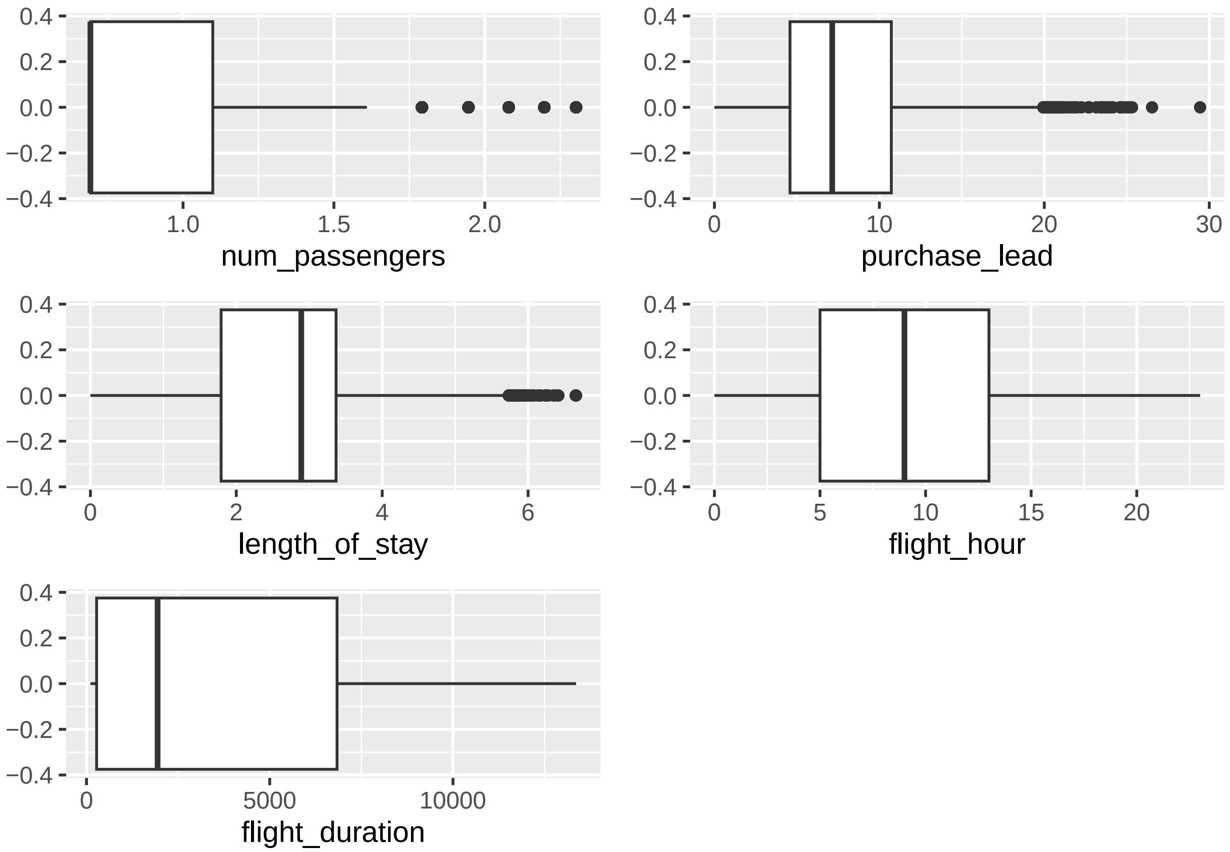
```
# there is no need to transform the flight hour column

# Exponential Transformation to flight_duration column since the data is left skewed
df_transformed$flight_duration <- exp(df_transformed$flight_duration)
summary(df_transformed$flight_duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   106.7   275.9  1939.1  3202.5  6836.3 13359.7
```

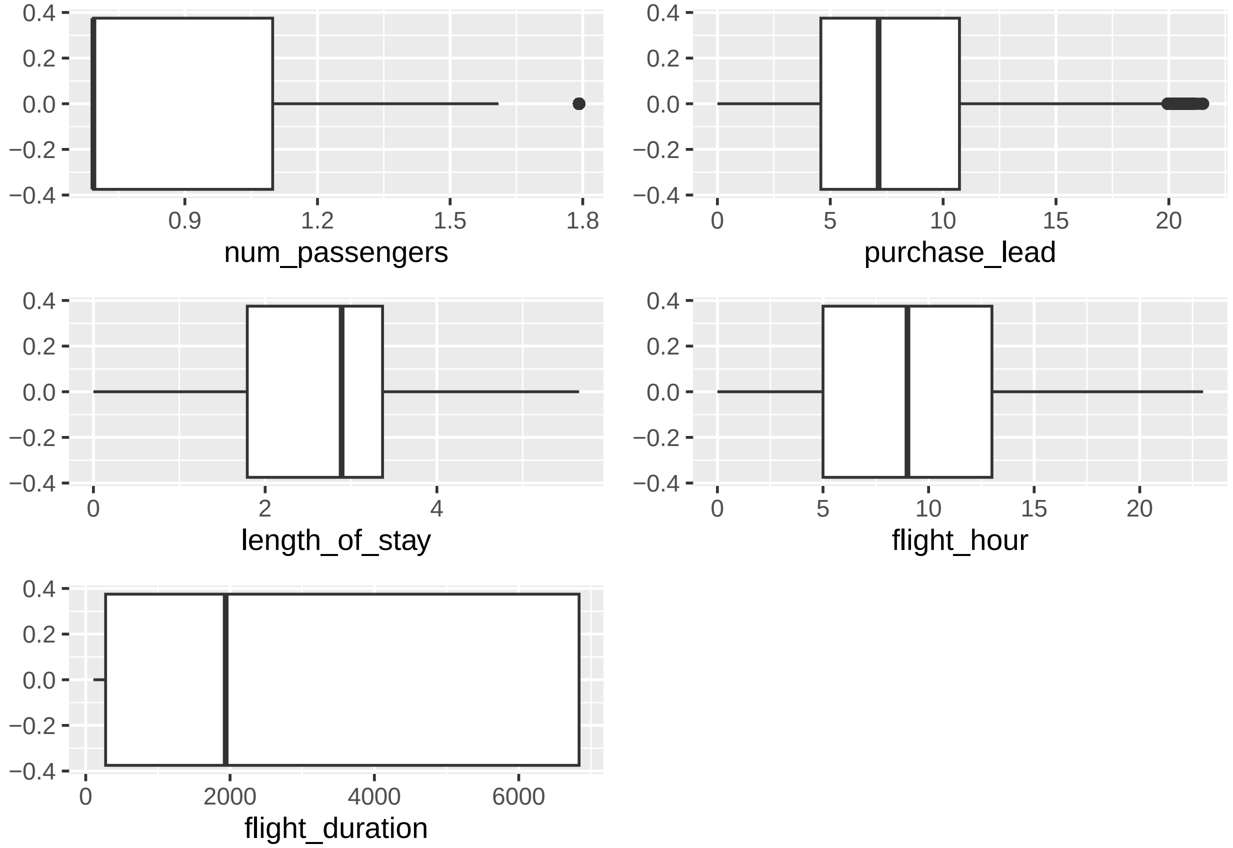
```
# plot boxplot
# Find outliers
# Let's see outlier on numeric column
variables2 <- c("num_passengers", "purchase_lead",
                "length_of_stay", "flight_hour", "flight_duration")

plots2 <- lapply(variables2, function(var){
  ggplot(df_transformed, aes_string(x= var)) +
    geom_boxplot()
})
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
grid.arrange(grobs = plots2, nrow= 3, ncol= 2)
```



```r
# as we can see, on purchase lead and length of stay have a lot of outlier values,
# so we will delete outlier wtih zscore so that not many columns are wasted
cols2 <- c("num_passengers", "purchase_lead",
           "length_of_stay", "flight_hour", "flight_duration")

for(col in cols2){
  zscore <- abs(scale(df_transformed[[col]]))
  df_transformed <- df_transformed[zscore <3,]
}

#Now checking the boxplot again
variables3 <- c("num_passengers", "purchase_lead",
                "length_of_stay", "flight_hour", "flight_duration")

plots3 <- lapply(variables3, function(var){
  ggplot(df_transformed, aes_string(x= var)) +
    geom_boxplot()
})
grid.arrange(grobs = plots3, nrow= 3, ncol= 2)
```



```r
# One-hot encoding for booking_origin column
df_encoded <- dummy_cols(df_transformed, select_columns = "booking_origin",
                         remove_selected_columns = TRUE)
df_encoded
```

```
## # A tibble: 49,241 x 18
##    num_passengers sales_channel trip_type purchase_lead length_of_stay
##             <dbl>         <dbl>     <dbl>         <dbl>          <dbl>
##  1          1.10              0         0         16.2            3.00
##  2          0.693             0         0         10.6            3.04
##  3          1.10              0         0         15.6            3.14
##  4          0.693             0         0          9.80           3.47
##  5          1.10              0         0          8.25           3.14
##  6          0.693             0         0          1.73           3.89
##  7          1.39              0         0         14.2            3.53
##  8          1.10              0         0         15.4            3.00
##  9          0.693             0         0          8.94           3.14
## 10          0.693             1         0         19.4            3.43
## # i 49,231 more rows
## # i 13 more variables: flight_hour <dbl>, wants_extra_baggage <dbl>,
## #   wants_preferred_seat <dbl>, wants_in_flight_meals <dbl>,
## #   flight_duration <dbl>, booking_complete <dbl>, is_weekend <dbl>,
## #   booking_origin_Africa <int>, booking_origin_Americas <int>,
## #   booking_origin_Asia <int>, booking_origin_Europe <int>,
## #   booking_origin_Oceania <int>, booking_origin_Others <int>
```

Make a machine learning Model- XGboost

```
#Split Data
str(df_encoded)
```

```
## tibble [49,241 x 18] (S3: tbl_df/tbl/data.frame)
##  $ num_passengers         : num [1:49241] 1.099 0.693 1.099 0.693 1.099 ...
##  $ sales_channel          : num [1:49241] 0 0 0 0 0 0 0 0 0 1 ...
##  $ trip_type              : num [1:49241] 0 0 0 0 0 0 0 0 0 0 ...
##  $ purchase_lead          : num [1:49241] 16.19 10.58 15.59 9.8 8.25 ...
##  $ length_of_stay         : num [1:49241] 3 3.04 3.14 3.47 3.14 ...
##  $ flight_hour            : num [1:49241] 7 3 17 4 15 20 6 14 4 12 ...
##  $ wants_extra_baggage    : num [1:49241] 1 0 1 0 1 1 1 1 0 0 ...
##  $ wants_preferred_seat   : num [1:49241] 0 0 1 0 0 0 0 0 0 0 ...
##  $ wants_in_flight_meals  : num [1:49241] 0 0 0 1 1 1 1 1 1 0 ...
##  $ flight_duration        : num [1:49241] 250 250 250 250 250 ...
##  $ booking_complete       : num [1:49241] 0 0 0 0 0 0 0 0 0 0 ...
##  $ is_weekend             : num [1:49241] 1 1 0 1 0 0 0 0 0 1 ...
##  $ booking_origin_Africa  : int [1:49241] 0 0 0 0 0 0 0 0 0 0 ...
##  $ booking_origin_Americas: int [1:49241] 0 0 0 0 0 0 0 0 0 0 ...
##  $ booking_origin_Asia    : int [1:49241] 0 0 1 0 1 0 0 1 0 1 ...
##  $ booking_origin_Europe  : int [1:49241] 0 0 0 0 0 0 0 0 0 0 ...
##  $ booking_origin_Oceania : int [1:49241] 1 1 0 1 0 1 1 0 1 0 ...
##  $ booking_origin_Others  : int [1:49241] 0 0 0 0 0 0 0 0 0 0 ...
```

```
# Convert all integer columns to numeric
df_encoded[] <- lapply(df_encoded, as.numeric) ##[] Preserves the data frame structure while applying la
                                               #Ensures that all columns are converted to numeric witho

#Splitting the data
set.seed(1234)
split <- sample.split(df_encoded$booking_complete, SplitRatio = 0.8)
train_set <- subset(df_encoded, split == TRUE)
test_set <- subset(df_encoded, split ==FALSE)

#Fitting Xgboost to training set
classifier <- xgboost(data = as.matrix(train_set[-11]),
                      label = train_set$booking_complete, nrounds = 10)
```

```
## [1]  train-rmse:0.426749
## [2]  train-rmse:0.385373
## [3]  train-rmse:0.362763
## [4]  train-rmse:0.350980
## [5]  train-rmse:0.344572
## [6]  train-rmse:0.341241
## [7]  train-rmse:0.339229
## [8]  train-rmse:0.337701
## [9]  train-rmse:0.336577
## [10] train-rmse:0.336055
```

```r
# Predicting the Test set results
y_pred <- predict(classifier, newdata = as.matrix(test_set[-11]))
y_pred <- (y_pred >= 0.4)

# Making the Confusion Matrix
cm <- table(test_set$booking_complete, y_pred)
Acc <- (cm[1,1]+cm[2,2])/ sum(cm)

# Applying k-Fold Cross Validation
set.seed(123)
folds = createFolds(df_encoded$booking_complete, k = 5)
cv = lapply(folds, function(x) {
  training_fold = df_encoded[-x, ]
  test_fold = df_encoded[x, ]
  classifier = xgboost(data = as.matrix(training_fold[-11]),
                       label = training_fold$booking_complete, nrounds = 10)
  y_pred = predict(classifier, newdata = as.matrix(test_fold[-11]))
  y_pred = (y_pred >= 0.4)
  cm = table(test_fold$booking_complete, y_pred)
  accuracy = (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  recall = cm[2,2] / (cm[2,1] + cm[2,2])
  return(c(accuracy, recall))
})
```

```
## [1]  train-rmse:0.426682
## [2]  train-rmse:0.385263
## [3]  train-rmse:0.362926
## [4]  train-rmse:0.350878
## [5]  train-rmse:0.344676
## [6]  train-rmse:0.341126
## [7]  train-rmse:0.339036
## [8]  train-rmse:0.337580
## [9]  train-rmse:0.336810
## [10] train-rmse:0.335875
## [1]  train-rmse:0.426298
## [2]  train-rmse:0.384504
## [3]  train-rmse:0.361727
## [4]  train-rmse:0.349717
## [5]  train-rmse:0.343103
## [6]  train-rmse:0.339638
## [7]  train-rmse:0.337583
## [8]  train-rmse:0.336075
## [9]  train-rmse:0.334918
## [10] train-rmse:0.334381
## [1]  train-rmse:0.427384
## [2]  train-rmse:0.386411
## [3]  train-rmse:0.364002
## [4]  train-rmse:0.352251
## [5]  train-rmse:0.346168
## [6]  train-rmse:0.342631
## [7]  train-rmse:0.340731
## [8]  train-rmse:0.339650
## [9]  train-rmse:0.338469
## [10] train-rmse:0.337606
## [1]  train-rmse:0.426472
## [2]  train-rmse:0.384973
## [3]  train-rmse:0.362570
## [4]  train-rmse:0.350361
## [5]  train-rmse:0.343911
## [6]  train-rmse:0.340411
## [7]  train-rmse:0.338369
## [8]  train-rmse:0.337065
## [9]  train-rmse:0.336216
## [10] train-rmse:0.335563
## [1]  train-rmse:0.426684
## [2]  train-rmse:0.385181
## [3]  train-rmse:0.362722
## [4]  train-rmse:0.350734
## [5]  train-rmse:0.344282
## [6]  train-rmse:0.340861
## [7]  train-rmse:0.338922
## [8]  train-rmse:0.337729
## [9]  train-rmse:0.336671
## [10] train-rmse:0.335844
```

```r
cv_results_df = do.call(rbind, cv)
colnames(cv_results_df) = c("Accuracy", "Recall")

mean_Accuracy = mean(cv_results_df[, "Accuracy"])
mean_Accuracy
```

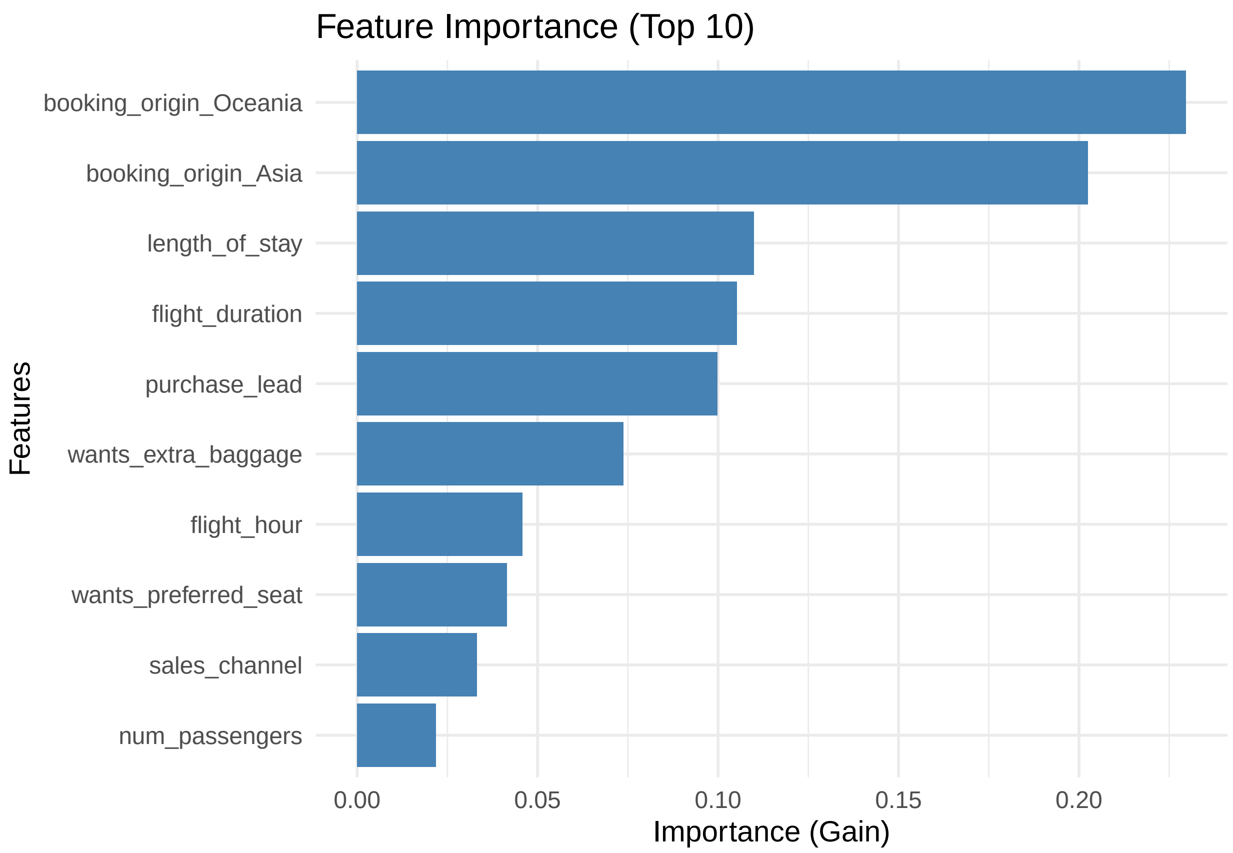
```
## [1] 0.8483376
```

```r
mean_recall = mean(cv_results_df[, "Recall"])
mean_recall
```

```
## [1] 0.0253033
```

Feature Importance

```r
# Get feature importance
feature_importance <- xgb.importance(model = classifier)

# Convert to dataframe and sort by importance
feature_importance_df <- feature_importance %>%
  arrange(desc(Gain)) %>%  # 'Gain' is the most important metric
  head(10)  # Select top 10 features

# Plot feature importance
ggplot(feature_importance_df, aes(x = reorder(Feature, Gain), y = Gain)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +  # Horizontal bar plot
  labs(title = "Feature Importance (Top 10)", x = "Features", y = "Importance (Gain)") +
  theme_minimal()
```


Feature Importance (Top 10)
booking_origin_Oceania
booking_origin_Asia
length_of_stay
flight_duration
purchase_lead
wants_extra_baggage
flight_hour
wants_preferred_seat
sales_channel
num_passengers
Features
0.00
0.05
0.10
0.15
0.20
Importance (Gain)