Risk of Cancellations on Hotel Bookings

Nelson Tran

Master of Data Science, Merrimack College

DSE6211 Machine Learning and Artificial Intelligence

Dr. Peter Salemi

March 10th, 2024

**Summary**

Hotels want to eliminate the risk of cancellations on hotel bookings as much as possible. This report compares a few dense neural network models and recommends one model that will help target bookings that have a high probability of cancellation with additional advertisements and/or offers to prevent them from being canceled. This is a supervised classification problem with the target variable being “booking\_status” in the given data set. Before one hot encoding our data, we must clean and process it by removing data issues like ‘n/a” data points, converting our variables into a numerical format, or excluding variables that are not needed for the models. The most important data processing done was assigning whether a patron had "canceled" or "not\_canceled” by using 1’s for cancellation and 0’s for non-cancellations. Once the data is one hot encoded, 1 simple dense neural network and 2 more complex dense neural networks. The first model (model 1) had 3 hidden layers with 75, 50, and 25 nodes respectively. The second model (model 2) had 4 hidden layers with 1000, 500, 250, and 100 nodes respectively. The third model (model 3) had 3 hidden layers with 1000, 500, and 100 nodes respectively. All the models were compared using receiver operating characteristic (ROC) curves, area under the ROC curve (AUC) values, and calibration curves. The AUC values for each model are 0.9051597 (model 1), 0.9197194 (model 2), and 0.9217329 (model 3). The calibration curves showed that model 1's probabilities are under-confident, model 3's probabilities are overconfident and model 2's probabilities lie near the dashed diagonal line which corresponds to a well-calibrated classifier. The model that would be the best to provide a solution to this business need would be model 2. This model was chosen because though model 3 had a higher AUC value, the calibration curve showed that the probabilities for this model aren't well calibrated which makes model 2 the ideal model to solve the business needs of the hotel.

**Data & Approach**

The business needs help identifying hotel bookings that are at risk of cancellation. In the given data set, the target variable is “booking\_status.” This is a binary problem, so we assign cancellations as "1" and any bookings that weren't canceled as "0." If needed, other variables were changed into numerical format. Variables in this data set include “booking\_ID,” “no\_of\_adults,” “no\_of\_children,” “no\_of\_weekend\_nights,” “no\_of\_week\_nights,” “type\_of\_meal\_plan,” “required\_car\_parking\_space,” “room\_type\_reserved,” “lead\_time,” “arrival\_date,” “market\_segment\_type,” “repeated\_guest,” “no\_of\_previous\_cancellations,” “no\_of\_previous\_bookings\_not\_canceled,” “avg\_price\_per\_room,” “no\_of\_special\_requests,” and our target variable “booking\_status.” The only variable removed from this data set was “booking\_ID” as this was a unique identifier and was not needed. “Arrival\_date” was converted into one of the four seasons of when the patron will be arriving at the hotel. All other variables were factored in. The goal of the model is to help predict the probability of booking cancellation and assess the model's performance against other models to ensure that our proposed model has the highest chance of solving business needs. Other information goals are using the outcomes to understand the mode's reliability and limitations and identifying which features influence cancellations.

For each model, the hidden layers use the “relu” function because it helps the network learn non-linear relationships. For the output layer, since we are dealing with a binary classification problem, we are using the “sigmoid” function. The loss function that was used for each model is “binary\_crossentropy.” Lastly, the default optimizer for a binary classification problem is “rmsprop.”

The models built were obtained using a combination of different architectures and comparing the loss and accuracy curves for each model. Model 1 is a simpler dense neural network with 3 hidden layers with 75, 50, and 25 nodes respectively. This model does not appear to fit quickly while overfitting shortly after (as shown in Figure 1). This can be improved by making the next model more complex.

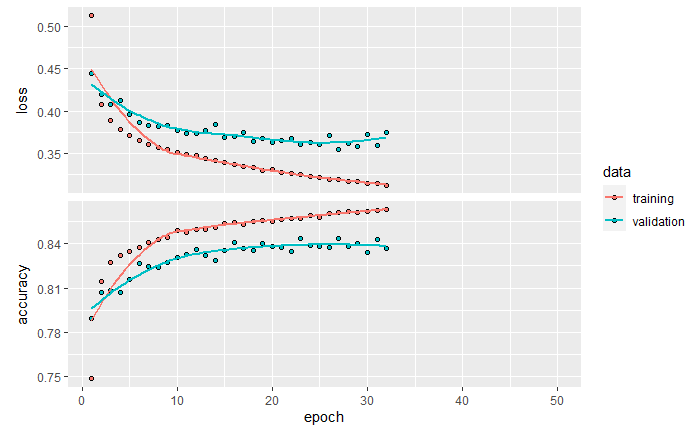


Figure 1: Loss and accuracy curve for model 1 (75, 50, and 25 nodes)

Model 2 has a more complex architecture with 4 hidden layers that have 1000, 500, 250, and 100 nodes respectively. This model appears to fit fast and starts to overfit after the 10th epoch (as shown in Figure 2). I wanted to make a slightly simpler model and see how that compared to this ideal model 2. This 3rd model resulted in an architecture with 3 hidden layers that have 1000, 500, and 100 nodes respectively. Model 3 did not fit as quickly as Model 2 did and took longer to start overfitting. The ideal model would fit quickly and start to overfit, but the model needs to be stopped early to prevent this overfitting. There is a function for this in Keras called “callback\_early\_stopping.” I used the “patience = 5” argument so that the model would stop after 5 epochs if the validation loss did not improve.

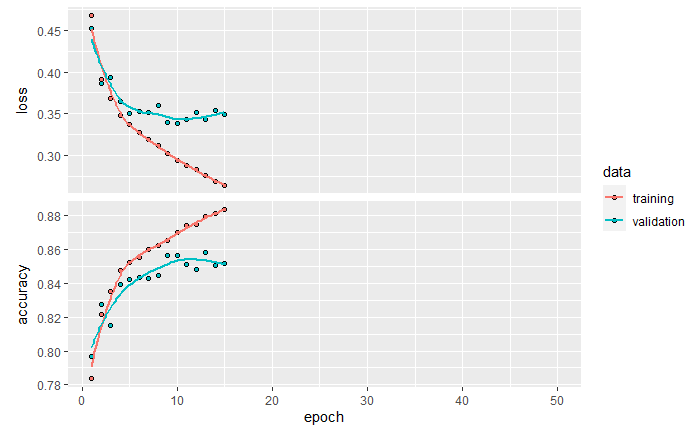


Figure 2: loss and accuracy curve of model 2 (1000, 500, 250, and 100 nodes)

**Detailed Findings and Evaluation**

There were 3 models built because comparing different architectures can help paint a picture as to why increasing the complexity will improve the performance of the model. The models will be compared using a ROC curve and the AUC values. After comparing the ROC curve and AUC values, the calibration curves will also be looked at to ensure that a model’s probabilities are well calibrated.

The ROC curve is an analytical method used to evaluate the performance of a model. The curve is a graph that shows the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis. The curve that we want would be as close to the top left corner of the graph as possible. This means that the model has a high TPR and a low FPR. The model's ROC curves looked very similar to each other but models 2 and 3 had a steeper slope earlier in the curve while model 1's curve looked more rounded. This can be seen when comparing Figure 3 and Figure 4.

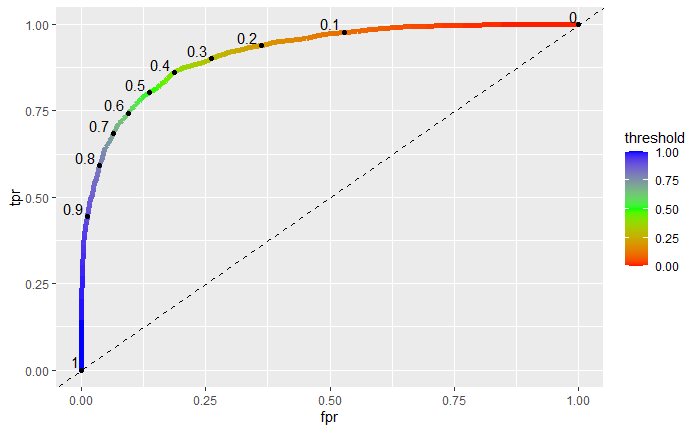


Figure 3: Model 1’s (75, 50, and 25 nodes) ROC curve

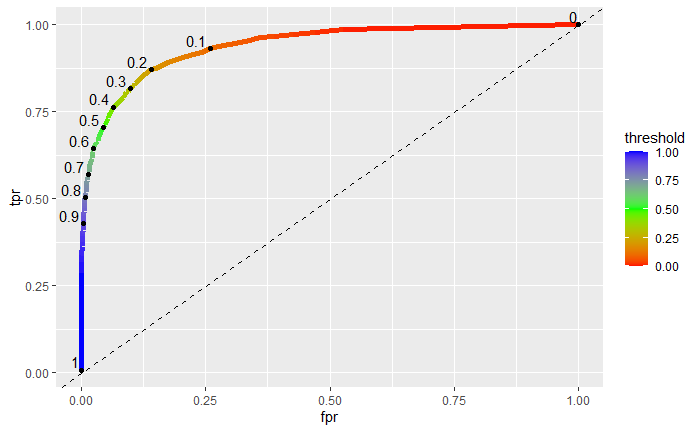


Figure 4: Model 3’s (1000, 500, and 100 nodes) ROC curve

Next, we want to find the AUC values for each curve. We did this using the “auc” function on the ROC curve data. Model 1 had an AUC value of 0.9051597, model 2 had an AUC value of 0.9197194, and model 3 had an AUC value of 0.921739. Models 2 and 3 have the best performances when comparing the ROC curves and AUC values. Before a model can be recommended, the calibration curves must be compared to show that these models are well-calibrated classifiers.

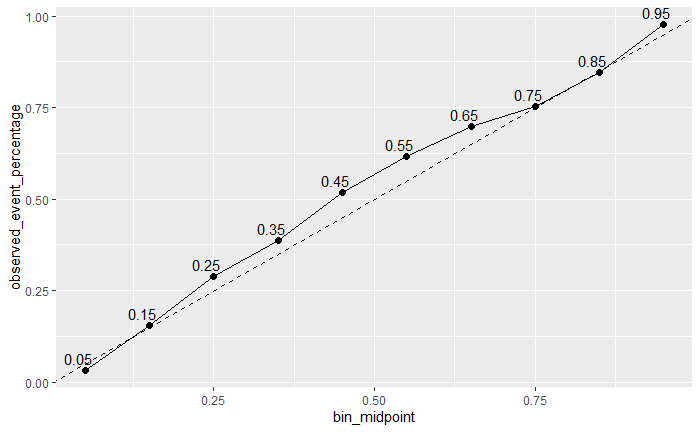


Figure 5: Model 2’s (1000, 500, 250, and 100 nodes) calibration curve

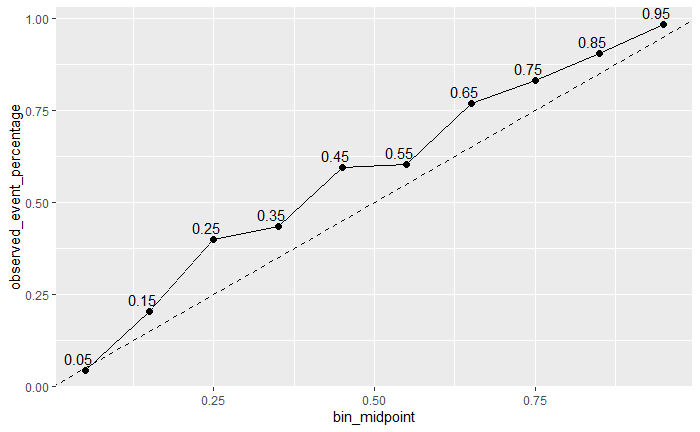


Figure 6: Model 3’s (1000, 500, and 100 nodes) calibration curve

Finally, when comparing the two curves model 2's is more well-calibrated with some slight under-confidence around the middle of the curve. There are under-confident probabilities in just about the whole curve for Model 3's calibration curve. The curves should be as close to the diagonal dashed line to be a well-calibrated classifier and model 2 shows the best example of this.

**Recommendations**

The model that's recommended for use is model 2 with 4 hidden layers and 1000, 500, 250, and 100 nodes respectively for identifying the risk of booking cancellations. Analytically, the model should be able to predict the probability of a booking being canceled and use this or other machine learning methods to assess this model's performance. Information that we want to attain using this model is understanding the reliability and limitations of the model itself. This model should also be able to identify the variables or features that have a significant influence on cancellations. With the success of identification, the business can offer promotions or advertisements to help prevent or lower the cancellation of a booking.

**Appendix**

Loading libraries that may be used.

library(lubridate)

library(reticulate)

library(keras)

library(tensorflow)

library(tidymodels)

library(caret)

library(MESS)

Data cleaning

data <- read.csv("./project\_data.csv", header = TRUE)

#removing Booking ID column from our data.

data <- subset(data, select = -c(0,1))

#seeing the length of each room type reserved

table(data$room\_type\_reserved)

##

## room\_type1 room\_type2 room\_type3 room\_type4 room\_type5 room\_type6 room\_type7

## 28105 692 7 6049 263 964 158

#From the table, we can see that 'room\_type1' is largely abundant compared to the other rooms.

#turning this variable into a factor 'room\_type1" and 'other

room\_type <- data %>%

group\_by(room\_type\_reserved) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

select(room\_type\_reserved) %>%

top\_n(-1)

## Selecting by room\_type\_reserved

data$room\_type\_reserved <- ifelse(data$room\_type\_reserved %in% room\_type$room\_type\_reserved,

data$room\_type\_reserved,

"other")

unique(data$room\_type\_reserved)

## [1] "room\_type1" "other"

unique(data$type\_of\_meal\_plan)

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

unique(data$market\_segment\_type)

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

## [5] "complementary"

#I will leave 'type\_of\_meal\_plan' and 'market\_segment\_type' as is but will still need to convert to a factor

#Converting dates from "arrival\_date" to seasons

data2 <- data

data2 <- data.frame(

arrival\_date = seq(as.Date("2017-01-01"), as.Date("2018-12-31"), by = "days")

)

get\_season <- function(month) {

case\_when(

month %in% 3:5 ~ "Spring",

month %in% 6:8 ~ "Summer",

month %in% 9:11 ~ "Fall",

TRUE ~ "Winter"

)

}

data2 <- data2 %>%

mutate(month = month(arrival\_date),

season = get\_season(month))

data <- data %>%

mutate(arrival\_date = get\_season(month(arrival\_date)))

unique(data$arrival\_date)

## [1] "Fall" "Winter" "Spring" "Summer"

Character variables will be changed to factors

data$type\_of\_meal\_plan <- factor(data$type\_of\_meal\_plan)

data$room\_type\_reserved <- factor(data$room\_type\_reserved)

data$arrival\_date <- factor(data$arrival\_date)

data$market\_segment\_type <- factor(data$market\_segment\_type)

#cancelled is = 1, not\_cancelled = 0

data$booking\_status <- as.integer(data$booking\_status == "canceled")

Onehot\_encoder

training\_ind <- createDataPartition(data$booking\_status,

p = 0.75,

list = FALSE,

times = 1)

training\_set <- data[training\_ind,]

test\_set <- data[-training\_ind,]

onehot\_encoder <- dummyVars(~ type\_of\_meal\_plan + room\_type\_reserved +

arrival\_date + market\_segment\_type,

training\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved",

"arrival\_date", "market\_segment\_type")],

levelsOnly = TRUE,

fullRank = TRUE)

onehot\_enc\_training <- predict(onehot\_encoder,

training\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved",

"arrival\_date", "market\_segment\_type")])

training\_set <- cbind(training\_set, onehot\_enc\_training)

onehot\_encoder <- dummyVars(~ type\_of\_meal\_plan + room\_type\_reserved +

arrival\_date + market\_segment\_type,

test\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved",

"arrival\_date", "market\_segment\_type")],

levelsOnly = TRUE,

fullRank = TRUE)

onehot\_enc\_test <- predict(onehot\_encoder,

test\_set[, c("type\_of\_meal\_plan", "room\_type\_reserved",

"arrival\_date", "market\_segment\_type")])

test\_set <- cbind(test\_set, onehot\_enc\_test)

cols\_to\_exclude <- c(5,7,9,10,16)

test\_set[, -(cols\_to\_exclude)] <- scale(test\_set[, -c(cols\_to\_exclude)],

center = apply(training\_set[, -c(cols\_to\_exclude)], 2, mean),

scale = apply(training\_set[, -c(cols\_to\_exclude)], 2, sd))

training\_set[, -c(cols\_to\_exclude)] <- scale(training\_set[, -c(cols\_to\_exclude)])

training\_features <- array(data = unlist(training\_set[, -c(cols\_to\_exclude)]),

dim = c(nrow(training\_set), 27))

training\_labels <- array(data = unlist(training\_set[, 16]),

dim = c(nrow(training\_set)))

test\_features <- array(data = unlist(test\_set[, -c(cols\_to\_exclude)]),

dim = c(nrow(test\_set), 27))

test\_labels <- array(data = unlist(test\_set[, 16]),

dim = c(nrow(test\_set)))

Loading our environment

use\_virtualenv("my\_tf\_workspace")

Testing different architectures to see which one will fit the best.

Model\_4 (75 units, 50 units, 25 units respectively)

model\_4 <- keras\_model\_sequential(list(

layer\_dense(units = 75, activation = "relu"),

layer\_dense(units = 50, activation = "relu"),

layer\_dense(units = 25, activation = "relu"),

layer\_dense(units = 1, activation = "sigmoid")

))

compile(model\_4,

optimizer = "rmsprop",

loss = "binary\_crossentropy",

metrics = "accuracy")

early\_stopping <- callback\_early\_stopping(patience = 5)

history\_4 <- fit(model\_4, training\_features, training\_labels,

epochs = 50, batch\_size = 512, validation\_split = 0.33,

callbacks = list(early\_stopping))

plot(history\_4)

A graph of data and data

Description automatically generated

model\_5 <- keras\_model\_sequential(list(

layer\_dense(units = 1000, activation = "relu"),

layer\_dense(units = 500, activation = "relu"),

layer\_dense(units = 250, activation = "relu"),

layer\_dense(units = 100, activation = "relu"),

layer\_dense(units = 1, activation = "sigmoid")

))

compile(model\_5,

optimizer = "rmsprop",

loss = "binary\_crossentropy",

metrics = "accuracy")

early\_stopping <- callback\_early\_stopping(patience = 5)

history\_5 <- fit(model\_5, training\_features, training\_labels,

epochs = 50, batch\_size = 512, validation\_split = 0.33,

callbacks = list(early\_stopping))

plot(history\_5)

A graph with red and blue lines

Description automatically generated

model\_6 <- keras\_model\_sequential(list(

layer\_dense(units = 1000, activation = "relu"),

layer\_dense(units = 500, activation = "relu"),

layer\_dense(units = 100, activation = "relu"),

layer\_dense(units = 1, activation = "sigmoid")

))

compile(model\_6,

optimizer = "rmsprop",

loss = "binary\_crossentropy",

metrics = "accuracy")

early\_stopping <- callback\_early\_stopping(patience = 5)

history\_6 <- fit(model\_6, training\_features, training\_labels,

epochs = 50, batch\_size = 512, validation\_split = 0.33,

callbacks = list(early\_stopping))

plot(history\_6)

A graph of data and data

Description automatically generated

ROC curves for each model

model\_4

predictions\_4 <- predict(model\_4, test\_features)

## 284/284 - 0s - 142ms/epoch - 498us/step

test\_set\_4 <- test\_set

test\_set\_4$p\_prob <- predictions\_4[,1]

over\_threshold <- test\_set\_4[test\_set\_4$p\_prob >= 0.5,]

roc\_data\_4 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)

for (i in roc\_data\_4$threshold) {

over\_threshold <- test\_set\_4[test\_set\_4$p\_prob >= i, ]

fpr <- sum(over\_threshold$booking\_status==0)/sum(test\_set\_4$booking\_status==0)

roc\_data\_4[roc\_data\_4$threshold==i, "fpr"] <- fpr

tpr <- sum(over\_threshold$booking\_status==1)/sum(test\_set\_4$booking\_status==1)

roc\_data\_4[roc\_data\_4$threshold==i, "tpr"] <- tpr

}

ggplot() +

geom\_line(data = roc\_data\_4, 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\_4[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +

geom\_text(data = roc\_data\_4[seq(1, 101, 10), ],

aes(x = fpr, y = tpr, label = threshold, hjust = 1.2 , vjust = -0.2))

A graph with a line and a point

Description automatically generated with medium confidence model\_5

predictions\_5 <- predict(model\_5, test\_features)

## 284/284 - 0s - 215ms/epoch - 757us/step

test\_set\_5 <- test\_set

test\_set\_5$p\_prob <- predictions\_5[,1]

over\_threshold <- test\_set\_5[test\_set\_5$p\_prob >= 0.5,]

roc\_data\_5 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)

for (i in roc\_data\_5$threshold) {

over\_threshold <- test\_set\_5[test\_set\_5$p\_prob >= i, ]

fpr <- sum(over\_threshold$booking\_status==0)/sum(test\_set\_5$booking\_status==0)

roc\_data\_5[roc\_data\_5$threshold==i, "fpr"] <- fpr

tpr <- sum(over\_threshold$booking\_status==1)/sum(test\_set\_5$booking\_status==1)

roc\_data\_5[roc\_data\_5$threshold==i, "tpr"] <- tpr

}

ggplot() +

geom\_line(data = roc\_data\_5, 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\_5[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +

geom\_text(data = roc\_data\_5[seq(1, 101, 10), ],

aes(x = fpr, y = tpr, label = threshold, hjust = 1.2 , vjust = -0.2))

A graph with a line graph

Description automatically generated with medium confidence model\_6

predictions\_6 <- predict(model\_6, test\_features)

## 284/284 - 0s - 193ms/epoch - 680us/step

test\_set\_6 <- test\_set

test\_set\_6$p\_prob <- predictions\_6[,1]

over\_threshold <- test\_set\_6[test\_set\_6$p\_prob >= 0.5,]

roc\_data\_6 <- data.frame(threshold=seq(1,0,-0.01), fpr=0, tpr=0)

for (i in roc\_data\_6$threshold) {

over\_threshold <- test\_set\_6[test\_set\_6$p\_prob >= i, ]

fpr <- sum(over\_threshold$booking\_status==0)/sum(test\_set\_6$booking\_status==0)

roc\_data\_6[roc\_data\_6$threshold==i, "fpr"] <- fpr

tpr <- sum(over\_threshold$booking\_status==1)/sum(test\_set\_6$booking\_status==1)

roc\_data\_6[roc\_data\_6$threshold==i, "tpr"] <- tpr

}

ggplot() +

geom\_line(data = roc\_data\_6, 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\_6[seq(1, 101, 10), ], aes(x = fpr, y = tpr)) +

geom\_text(data = roc\_data\_6[seq(1, 101, 10), ],

aes(x = fpr, y = tpr, label = threshold, hjust = 1.2 , vjust = -0.2))

A graph with a line and a point

Description automatically generated with medium confidence AUC for each model

Model\_4 AUC

auc\_4 <- auc(x = roc\_data\_4$fpr, y = roc\_data\_4$tpr, type = "spline")

auc\_4

## [1] 0.9078583

Model\_5

auc\_5 <- auc(x = roc\_data\_5$fpr, y = roc\_data\_5$tpr, type = "spline")

auc\_5

## [1] 0.921945

Model\_6

auc\_6 <- auc(x = roc\_data\_6$fpr, y = roc\_data\_6$tpr, type = "spline")

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

## 'x' values

auc\_6

## [1] 0.9198682

Calibration Curves for each model

Model\_4

calibration\_data\_4 <- 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\_4[test\_set\_4$p\_prob >= (i-0.05) & test\_set\_4$p\_prob <= (i + 0.05), ]

oep <- nrow(in\_interval[in\_interval$booking\_status==1, ])/nrow(in\_interval)

calibration\_data\_4[calibration\_data\_4$bin\_midpoint==i, "observed\_event\_percentage"] <- oep

}

ggplot(data = calibration\_data\_4, aes(x = bin\_midpoint, y = observed\_event\_percentage)) +

geom\_line() +

geom\_abline(intercept = 0, slope = 1, lty = 2) +

geom\_point(size = 2) +

geom\_text(aes(label = bin\_midpoint), hjust = 0.75, vjust = -0.5)

A graph with a line

Description automatically generated

model\_5

calibration\_data\_5 <- 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\_5[test\_set\_5$p\_prob >= (i-0.05) & test\_set\_5$p\_prob <= (i + 0.05), ]

oep <- nrow(in\_interval[in\_interval$booking\_status==1, ])/nrow(in\_interval)

calibration\_data\_5[calibration\_data\_5$bin\_midpoint==i, "observed\_event\_percentage"] <- oep

}

ggplot(data = calibration\_data\_5, aes(x = bin\_midpoint, y = observed\_event\_percentage)) +

geom\_line() +

geom\_abline(intercept = 0, slope = 1, lty = 2) +

geom\_point(size = 2) +

geom\_text(aes(label = bin\_midpoint), hjust = 0.75, vjust = -0.5)

A graph with a line and dots

Description automatically generated with medium confidence

model\_6

calibration\_data\_6 <- 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\_6[test\_set\_6$p\_prob >= (i-0.05) & test\_set\_6$p\_prob <= (i + 0.05), ]

oep <- nrow(in\_interval[in\_interval$booking\_status==1, ])/nrow(in\_interval)

calibration\_data\_6[calibration\_data\_6$bin\_midpoint==i, "observed\_event\_percentage"] <- oep

}

ggplot(data = calibration\_data\_6, aes(x = bin\_midpoint, y = observed\_event\_percentage)) +

geom\_line() +

geom\_abline(intercept = 0, slope = 1, lty = 2) +

geom\_point(size = 2) +

geom\_text(aes(label = bin\_midpoint), hjust = 0.75, vjust = -0.5)

A graph with a line graph

Description automatically generated