Ryan Canfield

Dr. Salemi

DSE6211

June 9th, 2024

Preliminary Results

**Introduction**

The purpose of this report is to provide an overview of the preliminary results obtained from the machine learning models utilized to meet the client's demands. After reviewing the results, the next objective of this report is to evaluate the models and identify potential areas for improvement. As a quick refresher, the client is ABC Hotels, and their requirement is for us to accurately estimate the probability of cancellation for each booking. This is crucial for them because by identifying bookings with high risks, the hotel can proactively target these bookings with tailored advertisements and offers, aiming to prevent cancellations. The dataset provided contains over 35,000 hotel bookings and various attributes about those bookings, such as the number of adults and children, meal plan type, average room cost, and booking status (canceled or not canceled).

**Approach**

The first step completed was addressing all the comments from the Analytic Plan. Instead of having a plethora of models, I limited it down to three to really focus on finding the best neural network possible. Other comments I addressed were dropping the “Booking\_ID” column since it was a unique identifier and switching from label encoding to one-hot encoding. The switch from one style of encoding to the other was to not impose a hierarchy on the data. Other preprocessing steps completed included checking and removing NA’s or missing values in the dataset, changing the date column to a season column, and splitting the data into a training set and testing set. The reason for removing the NAs was to ensure the integrity of the analysis. The reason for the change in columns was to make sure the dataset didn’t have high dimensionality and needless space when it got converted by the one-hot encoding. Lastly, the reason for the 80/20 training testing split was to allow us to train models on one set and evaluate the model's performance on the other. By doing this, it acts to give us an estimate of its real-world performance. After all the preprocessing was completed, a variety of machine learning methods were created to predict the booking status based on the given attributes. These models included logistic regression, support vector machine, and a dense neural network model. The purpose of this was to find the best possible model while still giving us a wide array of options.

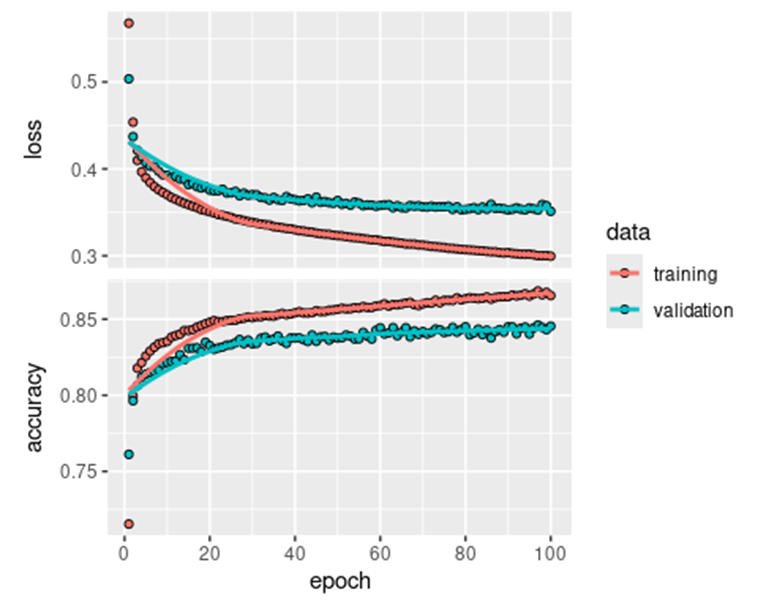
**Preliminary Results**

The initial analysis revealed promising results from the trained neural network model, which achieved an accuracy of 85.58% on the test set, with a sensitivity of 72.87% and specificity of 91.71%. The confusion matrix for all three models was able to provide insights into the model's performances. The neural network’s confusion matrix indicated a higher proportion of true negatives than false negatives, suggesting good predictive ability in identifying canceled bookings. Additionally, the other models yielded insightful results. Despite employing forward stepwise regression to identify the best possible features, the logistic regression appeared the least promising, likely due to having to fit many of its probabilities to zero or one because of one-hot encoding. The logistic regression model's metrics were an accuracy of 80.16% on the test set, with a sensitivity of 61.30% and a specificity of 89.26%. Conversely, the support vector machine demonstrated the highest performance, achieving an accuracy of 86.05% on the test set, with a sensitivity of 72.57% and a specificity of 92.55%. Below is a table comparing the performance of all three models.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Neural Network | Logistic Regression | Support Vector Machine |
| Accuracy | 85.58% | 80.16% | 86.05% |
| Sensitivity | 72.87% | 61.30% | 72.57% |
| Specificity | 91.71% | 89.26% | 92.55% |

Figure 1 Table of performance metrics of the three models.

While the support vector machine model performed better, I believe that the neural network will outperform the SVM in the final report due to its robustness and potential for greater fine-tuning. On its initial run, the NN achieved nearly the same level of accuracy as the SVM did after the hyperparameters were fine-tuned via a grid search. For the NN, I built a simple model with three layers, the first layer containing 50 nodes with a "relu" activation function, followed by another layer with 25 nodes with the same activation function, and a final layer with 1 node and a sigmoid activation function. I fully intend on optimizing this model before the final report to unlock its true accuracy potential. Finally, I anticipate the NN to deliver superior performance and accuracy due to it having more room to be tuned when compared to the SVM model.

**Discussion**

After building the model I plotted out the model’s accuracy and loss. This is important because it lets me better tune the model, and more easily communicate the performance of neural network models.

Figure 2 (right) loss and accuracy visual for the NN model.

It also helps me make decisions on how to improve the model’s effectiveness and plan for future steps. As you can see from the graph above the validation line on the loss graph is above the training line the entire time for each epoch the validation loss starts at 0.44 and ends at 0.35, while the training line starts at 0.43 and ends at a 0.30. The opposite can be said for the accuracy graph. Here the validation line is below the training one, with it starting at 0.80 and ending at 0.84 while the training line starts at 0.84 and ends at 0.89. With the validation curve being higher on the loss and lower on the accuracy it shows signs of overfitting. To try and prevent this I am going to try two different things. The first is to try a regularization technique and add a penalty term to the loss function. This will hopefully create a stronger model with less overfitting and redundancy. The other thing I would want to try is either make a simpler model or to monitor the model's performance during training and stop the training if the performance starts to go down. These things will hopefully minimize the overfitting the neural network is starting to have.

|  |  |  |
| --- | --- | --- |
|  | Loss | Accuracy |
| Validation | 0.44 - 0.35 | 80% - 84% |
| Training | 0.43 - 0.30 | 84% - 89% |

Figure 3 (left) table of values for loss and accuracy graph.

**Conclusion and Future Steps**

These preliminary findings suggest that the neural network has the potential to accurately predict hotel booking cancellations based on the given dataset. However, further tuning and improvement are necessary to increase the model's performance. To achieve this, I plan to conduct principal component analysis (PCA) on the dataset to maximize variance while minimizing attributes. PCA also helps to aid in reducing noise and mitigating multicollinearity, which likely exists in the dataset due to the additional columns created by one-hot encoding. Additionally, I want to optimize the model's hyperparameters through a grid search similarly to how it was done for the SVm model. This will give the model the most accurate predictions. Finally, I want to address the overfitting observed in the model by implementing previously mentioned strategies. In conclusion, these preliminary results suggest that the model still has room to grow but when it does it will be able to give promising performance and give the client accurate predictions for their hotel.

DSE6211 Preliminary Results

Ryan Canfield

2024-06-09

## R Markdown

library(glmnet) #For a Logistic Regression model.

# Addressed comment and removed extra models  
library(MASS)  
library(e1071) # For a Support Vector Machine model.  
library(neuralnet) # For a basic Neural Network model.  
library(keras) # For a TensorFlow Neural Network model.  
library(tensorflow) # For a TensorFlow framework.  
library(reticulate) # For a python interface.  
library(caret) # For model training and evaluation.

library(ROCR) # For ROC analysis.

library(tidyverse) # For data manipulation.

library(ggplot2) # For data visualizations.   
library(RColorBrewer) # For coloring visuals.  
library(forcats) # For Manipulating and working with categorical variables.

# Reading the CSV file into a data frame  
df <- read.csv("project\_data.csv")  
  
# Display a preview of the data frame.  
head(df)

### Preprocessing and addressing comments from Analytic Plan.  
  
# Checking for missing values.  
print(sum(is.na(df)))

# Addresses comment.  
# Excluding Booking\_ID since it is a unique identifier   
# Assuming 'df' is your dataframe  
df <- df[, !(names(df) %in% "Booking\_ID")]  
  
  
  
  
# Addresses comment.  
# Changes arrival date to season so when we use One Hot Encoding.   
# It doesn't create hundreds of extra columns, dimensionality, and space.  
df$arrival\_date <- as.Date(df[, "arrival\_date"], format = "%m/%d/%Y")   
df$season <- ifelse(month(df$arrival\_date) %in% c(12, 1, 2), "Winter",  
 ifelse(month(df$arrival\_date) %in% c(3, 4, 5), "Spring",  
 ifelse(month(df$arrival\_date) %in% c(6, 7, 8), "Summer",  
 ifelse(month(df$arrival\_date) %in% c(9, 10, 11), "Fall", NA))))  
df$arrival\_date <- df$season  
df <- df[, -which(names(df) == "season")]  
  
# This changes our target variable to numerical values without emposing heiracrhy or an extra column.  
df$booking\_status <- ifelse(df$booking\_status == "not\_canceled", 1, 0)

# Changed from label encoding to One Hot encoding  
for (col in names(df)) {   
 if (!is.numeric(df[[col]])) {  
 # Perform one-hot encoding  
 df <- cbind(df, model.matrix(~df[[col]] - 1))  
 df <- df[, -which(names(df) == col)]  
 cat("One-hot encoding applied to column:", col, "\n")  
 }  
}

## One-hot encoding applied to column: type\_of\_meal\_plan   
## One-hot encoding applied to column: room\_type\_reserved   
## One-hot encoding applied to column: arrival\_date   
## One-hot encoding applied to column: market\_segment\_type

# Split the dataset into a training and testing set with an 80 - 20 split  
set.seed(123)  
ind <- createDataPartition(df$booking\_status, p = 0.8, list = FALSE)  
train <- df[ind, ]   
test <- df[-ind, ]  
  
# Scaling the data  
# Define indices of columns to be scaled  
columns\_to\_scale <- setdiff(seq\_len(ncol(train)), 12)  
  
# Scale selected columns in the training set  
train[, columns\_to\_scale] <- lapply(train[, columns\_to\_scale], scale)  
  
# Scale selected columns in the test set  
test[, columns\_to\_scale] <- lapply(test[, columns\_to\_scale], scale)  
  
remove\_prefix <- function(df) {  
 colnames(df) <- gsub("df\\[\\[col\\]\\]", "", colnames(df))  
 return(df)  
}  
  
train <- remove\_prefix(train)  
test <- remove\_prefix(test)  
  
# Convert training and test features to arrays  
training\_features <- array(data = unlist(train[, -12]),  
 dim = c(nrow(train), 31))  
test\_features <- array(data = unlist(test[, -12]),  
 dim = c(nrow(test), 31))  
  
# Convert training and test labels to arrays  
training\_labels <- array(data = unlist(train[, 12]),  
 dim = c(nrow(train)))  
test\_labels <- array(data = unlist(test[, 12]),  
 dim = c(nrow(test)))

library(reticulate)  
library(tensorflow)  
library(keras)  
  
use\_virtualenv("my\_tf\_workspace")

model <- keras\_model\_sequential(list(  
 layer\_dense(units = 50, activation = "relu"),  
 layer\_dense(units = 25, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")  
))

compile(model,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")

NN <- fit(model, training\_features, training\_labels,  
 epochs = 100, batch\_size = 512, validation\_split = 0.33)  
  
plot(NN)

A graph of data and data

Description automatically generated

predictions4 <- predict(model, test\_features)  
head(predictions4, 10)

## [,1]  
## [1,] 0.87443310  
## [2,] 0.96991271  
## [3,] 0.97504157  
## [4,] 0.80336726  
## [5,] 0.90961266  
## [6,] 0.98129410  
## [7,] 0.99365830  
## [8,] 0.96050644  
## [9,] 0.99809760  
## [10,] 0.09230927

predicted\_class4 <- (predictions4[, 1] >= 0.5) \* 1  
head(predicted\_class4, 10)

conf\_matrix4 <- confusionMatrix(as.factor(predicted\_class4), as.factor(test\_labels))  
conf\_matrix4

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1719 405  
## 1 640 4483  
##   
## Accuracy : 0.8558   
## 95% CI : (0.8475, 0.8638)  
## No Information Rate : 0.6745   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6629   
##   
## Mcnemar's Test P-Value : 4.532e-13   
##   
## Sensitivity : 0.7287   
## Specificity : 0.9171   
## Pos Pred Value : 0.8093   
## Neg Pred Value : 0.8751   
## Prevalence : 0.3255   
## Detection Rate : 0.2372   
## Detection Prevalence : 0.2931   
## Balanced Accuracy : 0.8229   
##   
## 'Positive' Class : 0   
##

## First model: forward stepwise regression

# Forward Stepwise Logestic Regression   
  
# Fit an intercept-only model  
DF\_Null\_Model <- glm(booking\_status ~ 1, data = train, family = binomial)  
  
# fit a model with everything  
DF\_All\_Model <- glm(booking\_status ~ ., data = train, family = binomial)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Forward stepwise selection using AIC with both null and full models  
DF\_Final\_Model <- suppressWarnings(stepAIC(DF\_Null\_Model,  
 scope = list(lower = DF\_Null\_Model, upper = DF\_All\_Model),  
 direction = "forward",  
 trace = 0))  
  
  
# Display the final model summary  
summary(DF\_Final\_Model)

##   
## Call:  
## glm(formula = booking\_status ~ lead\_time + no\_of\_special\_requests +   
## online + avg\_price\_per\_room + Winter + required\_car\_parking\_space +   
## offline + repeated\_guest + no\_of\_weekend\_nights + room\_type1 +   
## aviation + not\_selected + room\_type4 + complementary + no\_of\_week\_nights +   
## no\_of\_previous\_cancellations + Spring + no\_of\_children +   
## room\_type2 + no\_of\_adults + room\_type7, family = binomial,   
## data = train)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.31219 1.18243 1.110 0.267113   
## lead\_time -1.32553 0.02041 -64.943 < 2e-16 \*\*\*  
## no\_of\_special\_requests 1.17650 0.02220 52.989 < 2e-16 \*\*\*  
## online -0.39963 0.04607 -8.675 < 2e-16 \*\*\*  
## avg\_price\_per\_room -0.59608 0.02357 -25.292 < 2e-16 \*\*\*  
## Winter 0.24289 0.02074 11.711 < 2e-16 \*\*\*  
## required\_car\_parking\_space 0.26801 0.02258 11.871 < 2e-16 \*\*\*  
## offline 0.44587 0.04464 9.987 < 2e-16 \*\*\*  
## repeated\_guest 0.45010 0.08075 5.574 2.49e-08 \*\*\*  
## no\_of\_weekend\_nights -0.11296 0.01622 -6.963 3.34e-12 \*\*\*  
## room\_type1 -0.33605 0.04834 -6.952 3.61e-12 \*\*\*  
## aviation -0.06405 0.01447 -4.427 9.54e-06 \*\*\*  
## not\_selected -0.08683 0.01716 -5.059 4.21e-07 \*\*\*  
## room\_type4 -0.22545 0.04393 -5.132 2.86e-07 \*\*\*  
## complementary 1.38734 11.39045 0.122 0.903058   
## no\_of\_week\_nights -0.06241 0.01628 -3.835 0.000126 \*\*\*  
## no\_of\_previous\_cancellations -0.10050 0.02814 -3.571 0.000356 \*\*\*  
## Spring -0.05139 0.01587 -3.239 0.001199 \*\*   
## no\_of\_children -0.06887 0.02060 -3.343 0.000830 \*\*\*  
## room\_type2 -0.06101 0.02168 -2.813 0.004901 \*\*   
## no\_of\_adults -0.05168 0.01834 -2.818 0.004836 \*\*   
## room\_type7 0.03751 0.01944 1.929 0.053669 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 36703 on 28990 degrees of freedom  
## Residual deviance: 24565 on 28969 degrees of freedom  
## AIC: 24609  
##   
## Number of Fisher Scoring iterations: 15

# Obtain predicted probabilities on the testing set  
predicted\_probs <- predict(DF\_Final\_Model, newdata = test, type = "response")  
  
# Assuming you have the true outcomes for the testing set (test\_data$output)  
observed\_responses <- as.factor(test$booking\_status)  
  
# Convert predicted probabilities to binary predictions (e.g., using a threshold of 0.5)  
predicted\_classes <- as.factor(ifelse(predicted\_probs >= 0.5, 1, 0))  
  
# Create and displaying the confusion matrix  
conf\_matrix <- confusionMatrix(predicted\_classes, observed\_responses)  
conf\_matrix

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1446 525  
## 1 913 4363  
##   
## Accuracy : 0.8016   
## 95% CI : (0.7922, 0.8107)  
## No Information Rate : 0.6745   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.528   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.6130   
## Specificity : 0.8926   
## Pos Pred Value : 0.7336   
## Neg Pred Value : 0.8270   
## Prevalence : 0.3255   
## Detection Rate : 0.1995   
## Detection Prevalence : 0.2720   
## Balanced Accuracy : 0.7528   
##   
## 'Positive' Class : 0   
##

# Training the best SVM model   
best\_svm\_model <- svm(booking\_status ~ .,   
 data = train,   
 kernel = "radial",   
 cost = 5,   
 gamma = 0.5)

# Make predictions on the test data  
predictions <- predict(best\_svm\_model, newdata = test)  
predicted\_classes2 <- as.factor(ifelse(predictions >= 0.5, 1, 0))  
# Create and displaying the confusion matrix  
conf\_matrix2 <- confusionMatrix(predicted\_classes2, as.factor(test$booking\_status))  
conf\_matrix2

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1712 364  
## 1 647 4524  
##   
## Accuracy : 0.8605   
## 95% CI : (0.8523, 0.8684)  
## No Information Rate : 0.6745   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6721   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7257   
## Specificity : 0.9255   
## Pos Pred Value : 0.8247   
## Neg Pred Value : 0.8749   
## Prevalence : 0.3255   
## Detection Rate : 0.2362   
## Detection Prevalence : 0.2865   
## Balanced Accuracy : 0.8256   
##   
## 'Positive' Class : 0   
##

# Create the neural network model  
#NNmodel <- neuralnet(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 +   
# no\_of\_previous\_bookings\_not\_canceled + avg\_price\_per\_room + no\_of\_special\_requests + meal\_plan\_1 +  
# meal\_plan\_2 + meal\_plan\_3 + not\_selected + room\_type1 + room\_type2 + room\_type3 + room\_type4 +  
# room\_type5 + room\_type6 + room\_type7 + Fall + Spring + Summer + Winter + aviation + complementary +  
# corporate + offline + online,  
# data = train,  
# hidden = c(5,3,1),  
# linear.output = TRUE)  
  
# Plot the neural network  
#plot(NNmodel)

# Make predictions on the test data  
#predictions3 <- predict(NNmodel, test)  
#predicted\_classes3 <- as.factor(ifelse(predictions3 >= 0.5, 1, 0))  
  
# Create and displaying the confusion matrix  
#conf\_matrix3 <- confusionMatrix(predicted\_classes3, as.factor(test$booking\_status))  
#conf\_matrix3