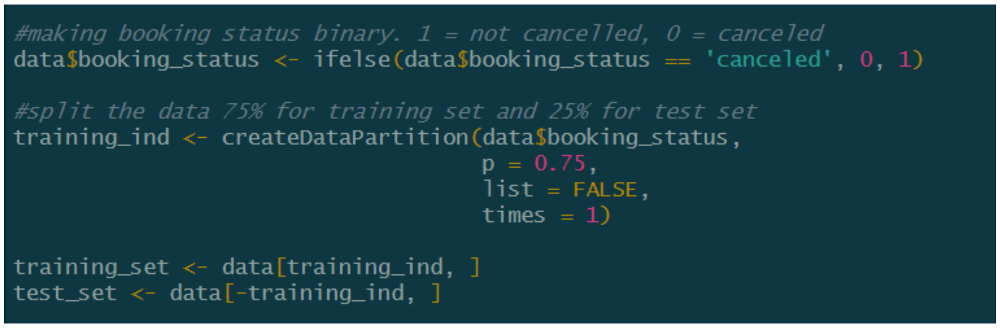
Mark Hoyt
2/18/24
DSE6211

Preliminary Results

In this project phase, we will process the data from ABC hotels and create the features identified in our analytical plan. Train a dense neural network model on the processed data to predict the label. Evaluate the neural network using learning curves to determine whether the model is overfitting or underfitting. We will identify necessary changes and steps for the final report from the results of the previous steps.

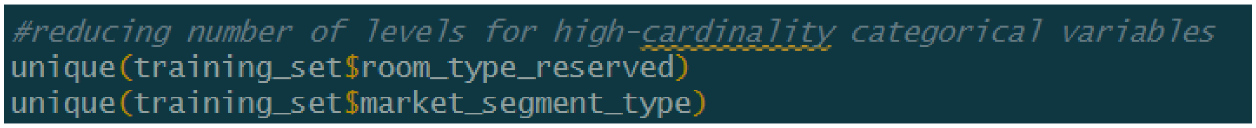
Data Preprocessing:

Step 1: split the original dataset 75% to 25%. 75% of the observations are put into a training dataset and the remaining 25% will be in a test dataset for when the model is trained.



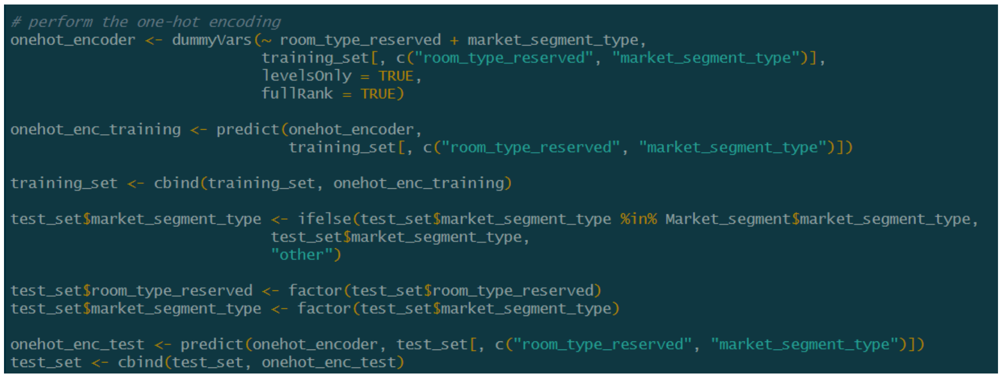
```r
#making booking status binary. 1 = not cancelled, 0 = canceled
data$booking_status <- ifelse(data$booking_status == 'canceled', 0, 1)

#split the data 75% for training set and 25% for test set
training_ind <- createDataPartition(data$booking_status,
                                    p = 0.75,
                                    list = FALSE,
                                    times = 1)

training_set <- data[training_ind, ]
test_set <- data[-training_ind, ]
```

Step 2: identify categorical variables and identify the number of levels in the categorical variables.



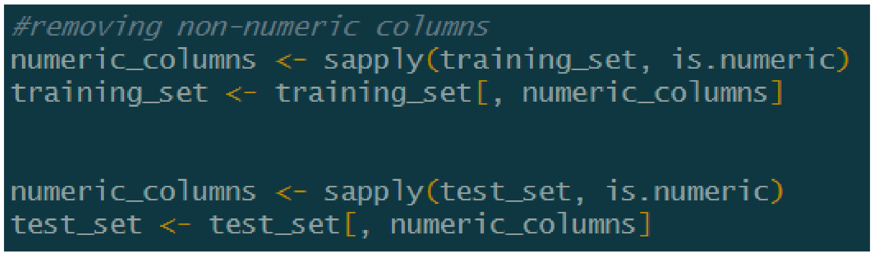
```r
#reducing number of levels for high-cardinality categorical variables
unique(training_set$room_type_reserved)
unique(training_set$market_segment_type)
```

Step 3: preformed one-hot encoding to the above-identified columns in order to process them for our model



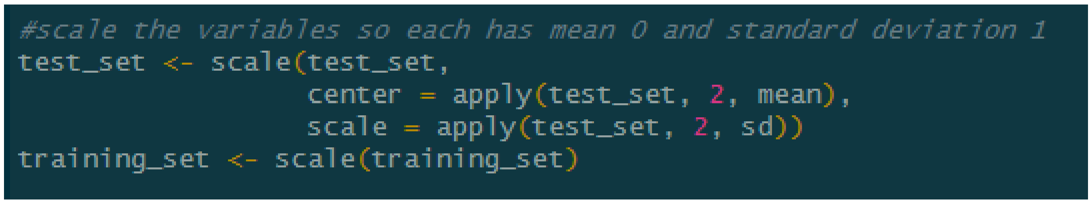
```r
# perform the one-hot encoding
onehot_encoder <- dummyVars(~ room_type_reserved + market_segment_type,
                            training_set[, c("room_type_reserved", "market_segment_type")],
                            levelsOnly = TRUE,
                            fullRank = TRUE)

onehot_enc_training <- predict(onehot_encoder,
                               training_set[, c("room_type_reserved", "market_segment_type")])

training_set <- cbind(training_set, onehot_enc_training)

test_set$market_segment_type <- ifelse(test_set$market_segment_type %in% Market_segment$market_segment_type,
                             test_set$market_segment_type,
                             "other")

test_set$room_type_reserved <- factor(test_set$room_type_reserved)
test_set$market_segment_type <- factor(test_set$market_segment_type)

onehot_enc_test <- predict(onehot_encoder, test_set[, c("room_type_reserved", "market_segment_type")])
test_set <- cbind(test_set, onehot_enc_test)
```

Step 4: removed remaining non-numeric columns and other unnecessary columns



```
#removing non-numeric columns
numeric_columns <- sapply(training_set, is.numeric)
training_set <- training_set[, numeric_columns]


numeric_columns <- sapply(test_set, is.numeric)
test_set <- test_set[, numeric_columns]
```

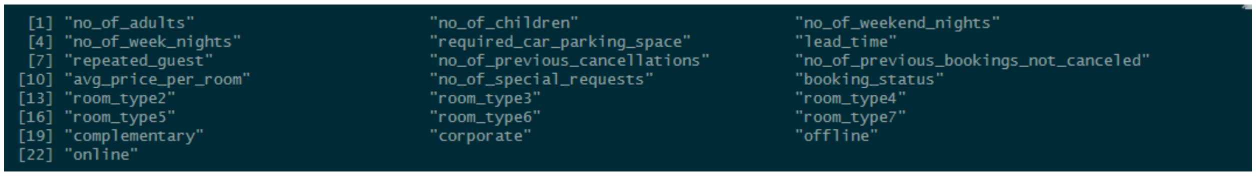
Step 5: Scaled the variables so each has a mean of 0 and a standard deviation of 1



```
#scale the variables so each has mean 0 and standard deviation 1
test_set <- scale(test_set,
                  center = apply(test_set, 2, mean),
                  scale = apply(test_set, 2, sd))
training_set <- scale(training_set)
```

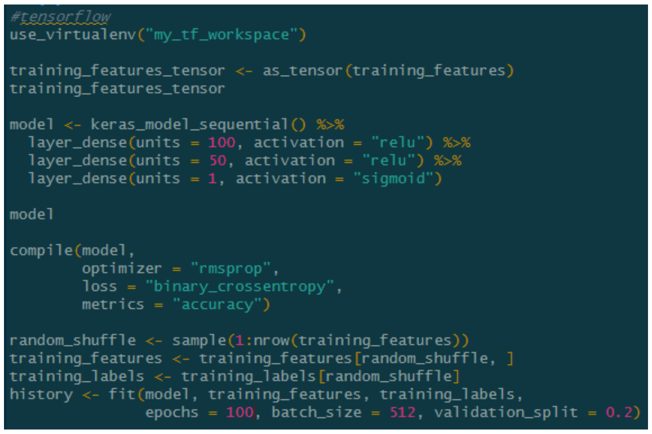
<u>Initial neural network finding:</u>

Features used in this neural network are;

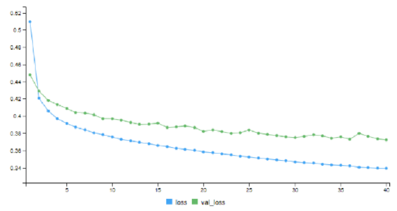


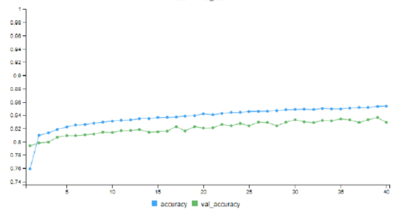
```
 [1] "no_of_adults"                     "no_of_children"                    "no_of_weekend_nights"
 [4] "no_of_week_nights"                "required_car_parking_space"        "lead_time"
 [7] "repeated_guest"                   "no_of_previous_cancellations"      "no_of_previous_bookings_not_canceled"
[10] "avg_price_per_room"               "no_of_special_requests"            "booking_status"
[13] "room_type2"                       "room_type3"                        "room_type4"
[16] "room_type5"                       "room_type6"                        "room_type7"
[19] "complementary"                    "corporate"                         "offline"
[22] "online"
```

The first neural network used leveraged ‘ReLU’ activation with an optimizer of ‘rmsprop’. There are three layers 100 unit, 50 units, and 1 unit. Overfitting can be observed within this model and ran 100 epoch iterations. Which resulted in an AUC score of .8875. I chose this number of unit in my layers as I wanted to keep this neural network simple to guage how it performs. With the results I have from a more basic neural network I feel confident in the changes I want to make in the future for more advanced models. My batch size is 512 and validation split is .25, but in the final model I plan to increase this to .33



```
#tensorflow
use_virtualenv("my_tf_workspace")

training_features_tensor <- as_tensor(training_features)
training_features_tensor

model <- keras_model_sequential() %>%
  layer_dense(units = 100, activation = "relu") %>%
  layer_dense(units = 50, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

model

compile(model,
        optimizer = "rmsprop",
        loss = "binary_crossentropy",
        metrics = "accuracy")

random_shuffle <- sample(1:nrow(training_features))
training_features <- training_features[random_shuffle, ]
training_labels <- training_labels[random_shuffle]
history <- fit(model, training_features, training_labels,
               epochs = 100, batch_size = 512, validation_split = 0.2)
```





<u>Calibration Curve:</u>

My Calibration Curve fell below the dashed line signifying that my model is over confident which in this case is good to see as it means I can better fit my data result in greater accuracy.

<u>Next Steps:</u>

- Improve model accuracy and maintaining a slightly over fit calibration curve inorder to keep refining my results
- Remove some of the clutter variable that negatively influenced my results such as; no_of_special_requests, Type_of_meal_plan, and properly formatting time variables.
- Make necessary changes to make my ROC curve shape better (this can be done with a deeper model and better preprocessing)
- Improve my FPR and TPR with a different threshold value
- Optimize my code through better practices and refinement of existing code.