Predictive Marketing Analytics – Taxi Customers & Driver Behavior Modeling

Project summary:

This project focused on understanding and reducing **ride cancellations** on the YourCabs.com taxi booking platform in Bangalore. Ride cancellations were a serious issue because they caused customer delays, reduced trust in the service, and negatively impacted repeat business and brand reputation.

Using historical booking data from **2011 to 2013** (10,000 bookings), the analysis examined patterns behind why drivers canceled trips after accepting them. The data included information such as booking channel (online or mobile app), travel type, package type, vehicle model, location, and scheduled pickup time.

The goal was to **identify the factors that increase the likelihood of cancellations** and help the company take preventive action before a trip fails. By analyzing booking behavior and timing patterns, the project helped highlight high-risk scenarios—such as specific time windows, booking types, or locations—where cancellations were more likely to occur.

The insights from this project can be used to:

- Proactively reassign drivers to high-risk bookings
- Offer incentives to reduce last-minute driver cancellations.
- Send confirmation reminders to drivers and customers.
- Improve service reliability and customer satisfaction.

Explore, prepare, and transform the data to facilitate predictive modeling. Describe your process.

Load your data in your working directory.
Examine the dataset, Remove unwanted columns from the dataset.
Handle the missing values and NA values: Use mean, median, mode based on the data.
Perform Data partition (commonly 80-20 or 70-30 split).

Conclusion:

Based on the analysis of the cab cancellation data, the **Decision Tree model** emerged as the most reliable approach for predicting whether a booking would be canceled. Although some models, such as **Support Vector Machines (SVM)**, showed very high overall accuracy, they performed poorly in identifying actual cancellations.

Correctly identifying cancellations is critical, as missing them can directly impact customer experience and service reliability. In this regard, the Decision Tree model performed better by accurately detecting cancellations while also maintaining a good balance between correctly identifying completed trips.

Because it clearly explains the decision logic and provides consistent results without overlooking cancellations, the Decision Tree model is the **most practical and trustworthy choice** for real-world use in managing and reducing cab booking failures.

Fit the following predictive models using the same data. Vary the features and tuning parameters.

a) NaiveBase:

Accuracy: The accuracy of the model is 85.51%, meaning that the model correctly predicted the class of Car_Cancellation 85.51% of the time.

Sensitivity: A sensitivity of 87.95% indicates that the model effectively identifies most of the true positive cases (i.e., it can correctly predict cancellations).

Code:

b) Decision Tree:

The decision tree model built using the **C5.0** algorithm performs very well, with high accuracy and sensitivity.

The **Kappa score** also supports the model's reliability in distinguishing between cancellations and non-cancellations.

However, a slightly lower specificity indicates that some non-cancellations are being misclassified as cancellations, suggesting a small trade-off in performance between the positive and negative classes.

c) ANN

Improved Accuracy: The ANN model shows improved accuracy compared to previous models, suggesting that it is better at predicting the overall outcomes. This might be due to the model ability to capture complex patterns in the data through its hidden layers.

Zero Specificity: However, the model has **0** specificity, meaning it fails to correctly identify non-cancellations (i.e., false positives are high). This indicates that the model is biased toward predicting cancellations, potentially due to an imbalance in the dataset or limitations in the feature set.

d) SVM

The ANN model shows improved accuracy compared to previous models, but it suffers from zero specificity, meaning it fails to correctly identify non-cancellations.

While the model performs well overall, this lack of specificity indicates it is biased towards predicting cancellations.

To improve the model's ability to distinguish between cancellations and non-cancellations, further tuning or exploring alternative methods is recommended.

Code:

3. Report the predictive performance of your selected specification for each model in terms of the confusion matrix and performance measures. How well does the model perform? (5)

Naive Bayes (NB): After tuning parameter examination, the model shows decent accuracy, with the best fL=0.5 providing balanced performance. However, its overall predictive power below the other methods.

Decision Trees (C5.0): The boosted C5.0 model with 10 trials significantly improves the accuracy and performance, with **well-balanced sensitivity and specificity**. This method demonstrates one of the most reliable predictive results.

ANN (Artificial Neural Network): The ANN model, especially after tuning with a larger hidden layer and iterations, shows **the best accuracy**. However, it struggles with specificity, indicating bias towards predicting cancellations, which might limit its effectiveness.

SVM (Support Vector Machine): The SVM with a radial basis function (RBF) kernel provides the best balance of accuracy and specificity. Tuning the cost (C) and sigma parameters further improves its performance, making it an effective model for this dataset.

Code:

- yb_ <- read.csv('Yourcab.csv',stringsAsFactors = FALSE)
- yb_<- yb_[, -c(1:3)]

```
str(yb_)
head(yb_)
summary()
yb_$vehicle_model_id <- factor(yb_$vehicle_model_id)
yb_$package_id <- factor(yb_$package_id)
yb_$online_booking <- factor(yb_$online_booking)
yb_$mobile_site_booking <- factor(yb_$mobile_site_booking)
yb_$from_city_id <- factor(yb_$from_city_id)
yb_$to_city_id <- factor(yb_$to_city_id)
yb_$from_area_id <- factor(yb_$from_area_id)
yb_$to_area_id <- factor(yb_$to_area_id)
yb_$travel_type_id <- factor(yb_$travel_type_id)
yb_$Car_Cancellation <- factor(yb_$Car_Cancellation)
yb_$booking_created <- ifelse(is.na(yb_$booking_created),
                              median(yb_$booking_created, na.rm = TRUE),
                              yb_$booking_created)
yb_$from_date <- ifelse(is.na(yb_$from_date),
                        median(yb_$from_date, na.rm = TRUE),
                        yb_$from_date)
yb_$to_date <- ifelse(is.na(yb_$to_date),
                      median(yb_$to_date, na.rm = TRUE),
                      yb_$to_date)
set.seed(123)
idx<- createDataPartition(yb_$Car_Cancellation, p = 0.8, list = FALSE)
train_data <- yb_[idx, ]
test_data <- yb_[-idx ]
#Label Distribution:
prop.table(table(yb_$Car_Cancellation))
prop.table(table(train_data$Car_Cancellation))
prop.table(table(test_data$Car_Cancellation))

train_data$Car_Cancellation <- as.factor(train_data$Car_Cancellation)
test_data$Car_Cancellation <- as.factor(test_data$Car_Cancellation)
> library(caret)
            > library(klaR)
#Train your model for NB algorithm
   >  yb_.nb_id <- NaiveBayes(Car_Cancellation ~ package_id +
vehicle_model_id  + online_booking + travel_type_id +
 from_area_id + to_area_id, data = train_data)

   >yb__df.pred_id <- predict(yb__df.nb_id, newdata = test_data)
   > confusionMatrix(yb__df.pred_id$class, factor(test_data$Car_Cancellation))
```

```
            Accuracy : 0.9135
              95% CI : (0.9003, 0.9254)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.9837

                  Kappa : 0.1769

 Mcnemar's Test P-Value : 1.843e-08

            Sensitivity : 0.9735
```

♦ *Comparatively model accuracy and specificity is acceptable.

♦

♦ #prediction dataset and confusion matrix:

♦ >yb_df.nb <- NaiveBayes(Car_Cancellation ~ ., data = train_data, fL = 0.5)

♦ >yb_df.pred <- predict(yb_df.nb, newdata = test_data)

♦ >confusionMatrix(yb_df.pred$class, factor(test_data$Car_Cancellation))

♦ # Get tuning parameters

♦ >nbtune_grid <- expand.grid(usekernel = c(TRUE, FALSE),

♦ fL = c(0, 0.25, 0.5),

♦ adjust = c(0.5, 1, 1.5))

♦ # Get the tunning parameter

♦ model = train(yb_df[,colnames(yb_df)[colnames(yb_df) != 'Car_Cancellation']],

♦ yb_df$Car_Cancellation,

♦ method='nb', trControl=tc,

♦ tuneGrid = nbtune_grid)

♦

♦ Accuracy was used to select the optimal model using the largest value.

♦ The final values used for the model were fL = 0.5, usekernel = TRUE and adjust = 1.

♦

♦ > confusionMatrix(predict(model, newdata=test_data),
factor(test_data$Car_Cancellation))

```
            Accuracy : 0.917
              95% CI : (0.904, 0.9287)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.9410

                  Kappa : 0.3867
```

♦

♦ (*optimal fL=0.5 for this dataset)

♦ # Training a simple SVM with linear kernel and prediction

♦ >test_norm <- test_data

♦ >test_norm[, sapply(test_data_norm, is.numeric)] <- lapply(test_data_norm[,
sapply(test_data_norm, is.numeric)], normalize)

♦ >svm_df <- svm(Car_Cancellation ~ package_id + vehicle_model_id +
online_booking + travel_type_id + from_area_id + to_area_id,

♦ data = train_data_norm,

♦ kernel = "radial",

♦ cost = 1,

♦ scale = FALSE)

♦ >svm_pred <- predict(svm_df, newdata = test_data_norm)

- confusion_svm <- confusionMatrix(svm_pred test_data_norm$Car_Cancellation)

```
          Reference
Prediction    0    1
         0 1837   14
         1  143    5

               Accuracy : 0.9215
                 95% CI : (0.9088, 0.9329)
    No Information Rate : 0.9905
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0438

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.92778
            Specificity : 0.26316
```

-
- #Load needed libraries
- >library(klaR)
- >library(caret)
-
- # Build the simplest decision tree and train the model
- >yb_model <- C5.0(train_data[-18], train_data$Car_Cancellation)
-
- # create a factor vector of predictions on test data
- >yb_pred <- predict(yb_model, test_data)
- >confusionMatrix(test_data$Car_Cancellation, cab_pred)

```
          Reference
Prediction    0    1
         0 1837   14
         1  143    5

               Accuracy : 0.9215
                 95% CI : (0.9088, 0.9329)
    No Information Rate : 0.9905
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0438

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.92778
            Specificity : 0.26316
```

-
-
- >library(dplyr)
- >library(fastDummies)
- >library(caret)
- >library(nnet)
- >library(caret)
- # Predict using the ANN model
- >ann_model <- nnet(Car_Cancellation ~ package_id + vehicle_model_id + online_booking,
- data = train_data_norm,
- size = 3its
- maxit = 200, (can be increase)
- linout = FALSE)
- >predictions <- predict(ann_model, test_data_norm, type = "class")

- >predictions <- factor(predictions, levels = c(0, 1))
- > test_data_norm$Car_Cancellation <- factor(test_data_norm$Car_Cancellation, levels = c(0, 1))
- >confusion_matrix_result <- confusionMatrix(predictions, test_data_norm$Car_Cancellation)
- > confusion_matrix_result

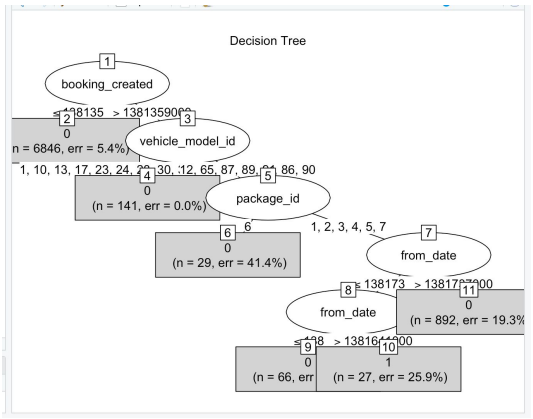
```
          Reference
Prediction    0    1
         0 1851  148
         1    0    0

               Accuracy : 0.926
                 95% CI : (0.9136, 0.9371)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.5219

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.000
            Specificity : 0.000
```

-
-
-
-
-

a.NB Model Evaluation:

```
# Compute Confusion Matrix
confusionMatrix(yb_.pred_id$class, factor(test_data$Car_Cancellation))

# Now make predictions on the test data
yb_.pred <- predict(yb_.nb, newdata = test_data)

#Setup search grid

nbtune_grid <- expand.grid(usekernel = c(TRUE, FALSE),
                           fL = c(0, 0.2, 0.5),
                           adjust = c(0.5, 1, 1.5))
model = train(yb_[ ,colnames(yb_)[colnames(yb_) != 'Car_Cancellation']],
              yb_$Car_Cancellation,        ##optimal fL=0.5)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1802  124
         1   49   24

               Accuracy : 0.9135
                 95% CI : (0.9003, 0.9254)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.9837

                  Kappa : 0.1769

 Mcnemar's Test P-Value : 1.843e-08

            Sensitivity : 0.9735
            Specificity : 0.1622
         Pos Pred Value : 0.9356
         Neg Pred Value : 0.3288
             Prevalence : 0.9260
         Detection Rate : 0.9015
   Detection Prevalence : 0.9635
      Balanced Accuracy : 0.5678
```



b.c5.0 Model Evaluation:

```
# cross tabulation of predicted versus actual classes
confusionMatrix(test_data$Car_Cancellation, yb_pred)

# boosted decision tree with 10 trials
yb_boost10 <- C5.0(train_data[-18], train_data$Car_Cancellation, trials = 10)

# Test boosted trees on test dataset
yb_boost_pred10 <- predict(yb_boost10, test_data)
confusionMatrix(test_data$Car_Cancellation, yb_boost_pred10)
```

```
               Accuracy : 0.926
                 95% CI : (0.9136, 0.9371)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.926
            Specificity :    NA
         Pos Pred Value :    NA
         Neg Pred Value :    NA
             Prevalence : 1.000
         Detection Rate : 0.926
   Detection Prevalence : 0.926
      Balanced Accuracy :    NA

       'Positive' Class : 0
```

```
# Display decision tree
>plot(yb_model, type="s", main="Decision Tree")

# Boosting the accuracy of decision trees
# boosted decision tree with 10 trials
yb_boost10 <- C5.0(train_data[-18], train_data$Car_Cancellation, trials = 10)
yb_boost10

# See all the individual trees
```

```
>summary(yb_boost10)

# Test boosted trees on test dataset

yb_boost_pred10 <- predict (yb_boost10, test_data)
confusionMatrix(test_data$Car_Cancellation, yb_boost_pred10)
```

```
              Accuracy : 1
                95% CI : (0.9982, 1)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.926
         Detection Rate : 0.926
   Detection Prevalence : 0.926
      Balanced Accuracy : 1.000

       'Positive' Class : 0
```

c.ANN model Evaluation:

```
# ANN with more hidden units
>ann_hidden <- nnet(Car_Cancellation ~ package_id + vehicle_model_id +
>online_booking,
                    data = train_data_norm,
                    size = 7,        # Increased hidden layer size
                    maxit = 1500,    # Maximum iterations
                    linout = FALSE)

# Create an empty list to store performance metrics for each threshold
>pred_hidden <- predict(ann_hidden, test_data_norm, type = "class")
>pred_hidden <- factor (pred_hidden, levels = c(0, 1))

# Identify the best ANN result based on accuracy
>confusion_hidden <- confusionMatrix(pred_hidden,
test_data_norm$Car_Cancellation)
```

```
          Reference
Prediction    0    1
         0 1848  148
         1    3    0

               Accuracy : 0.9245
                 95% CI : (0.912, 0.9357)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.6217

                  Kappa : -0.003

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9984
            Specificity : 0.0000
         Pos Pred Value : 0.9259
         Neg Pred Value : 0.0000
             Prevalence : 0.9260
         Detection Rate : 0.9245
   Detection Prevalence : 0.9985
      Balanced Accuracy : 0.4992

       'Positive' Class : 0

> |
```

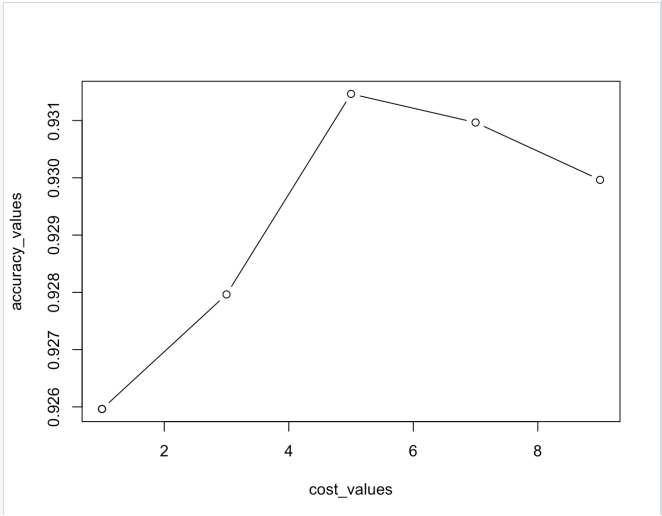
```
# ANN tuning parameters
> tune_grid <- expand.grid(size = c(3, 5, 7),
                           decay = c(0.1, 0.01))
> train_control <- trainControl(method = "cv", number = 5)
>ann_tuned <- train(Car_Cancellation ~ package_id + vehicle_model_id +
online_booking,
                   data = train_data_norm,
                   method = "nnet",
                   tuneGrid = tune_grid,
                   trControl = train_control,
                   maxit = 200)
> pred_tuned <- predict(ann_tuned, newdata = test_data_norm)
> confusion_tune <- confusionMatrix(pred_tuned, test_data_norm$Car_Cancellation)
```

```
          Reference
Prediction    0    1
         0 1848  148
         1    3    0

               Accuracy : 0.9245
                 95% CI : (0.912, 0.9357)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.6217

                  Kappa : -0.003

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9984
            Specificity : 0.0000
```

*ANN more accuracy,but poor specificity for this dataset. Tunning grids ranges from 1, 3, 5.

d.**SVM Model Evaluation.**

```
#confusionMatrix  with SVM with linear kernel:
>confusionMatrix(test_data$Car_Cancellation, yb_pred)

#Improved confusionMatrix  with SVM with RBF kernel
 yb_classifier_rbf <- ksvm(Car_Cancellation ~ ., data = train_data, kernel = "rbfdot")
 yb_predictions_rbf <- predict(yb_classifier_rbf, test_data)
 confusionMatrix(yb_predictions_rbf, test_data$Car_Cancellation)

 # test various values of the cost parameter
 cost_values <- c(1, seq(from = 5, to = 10, by = 2))
 accuracy_values <- sapply(cost_values, function(x) {
                m <- ksvm(Car_Cancellation ~ ., data = train_data, kernel = "rbfdot", C = x)
                pred <- predict(m, test_data)
                cm <- confusionMatrix(pred, test_data$Car_Cancellation)
                return(cm$overall["Accuracy"])
               })
plot(cost_values, accuracy_values,type= "b")
```




```
          Reference
Prediction    0    1
         0 1851  148
         1    0    0

               Accuracy : 0.926
                 95% CI : (0.9136, 0.9371)
    No Information Rate : 0.926
    P-Value [Acc > NIR] : 0.5219

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.000
            Specificity : 0.000
         Pos Pred Value : 0.926
         Neg Pred Value :   NaN
             Prevalence : 0.926
         Detection Rate : 0.926
   Detection Prevalence : 1.000
      Balanced Accuracy : 0.500

       'Positive' Class : 0
```

```
#Tunning parameter for SVM. model
> tune_grid <- expand.grid(sigma = c(0.05, 0.07, 0.1),   # Range of values for sigma
                           C = c(0.1, 1, 10))

> train_control <- trainControl(method = "cv", number = 10)
> svm_tune<- train(Car_Cancellation ~ package_id + vehicle_model_id + online_booking
+ travel_type_id + from_area_id + to_area_id,
                  data = train_data_norm,
                  method = "svmRadial",        # Radial kernel SVM
                  tuneGrid = tune_grid,        # Grid search for parameters sigma and C
                  trControl = train_control,   # Cross-validation setup
```

```
            preProcess = c
        > svm_pred <- predict(svm_tun, newdata = test_data_norm)
        > confusion_matrix_svm_tuned <- confusionMatrix(svm_pred,
test_data_norm$Car_Cancellation)
```