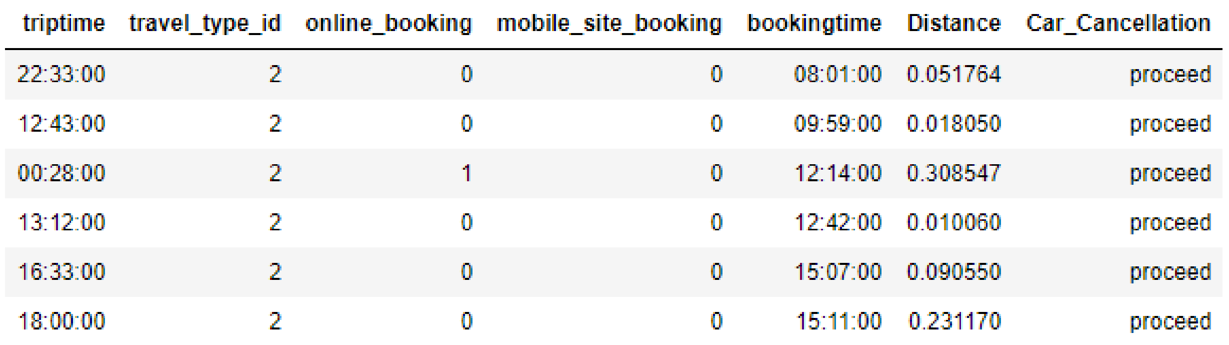
1. Predictive Modelling: The aim of Predictive modelling is to predict new records accurately. In Yourcabs.com, there was a problem. If there occurred any cancellation, then company had to lose money. There were no predictive mechanism to predict cancellations. As predictive models optimize predictive accuracy, it would reduce costs if there are cancellations with the help of the model. With the help of predictive model, we can predict which calls may be canceled and we can build a model according to our findings. This may reduce the company loss.
   In practice, I would try to build a predictive model which will predict the possible cancellations and reduce the cancellation cost. It will also help the organization to manage its vendors and drivers by providing up-to-date information regarding cancellation and it will reduce cost. It will also save time of drivers. Drivers will be able to pick up confirmed passengers instead of wasting time to pick up passengers who are not sure about the ride.

2. From the given dataset, I selected few columns as predictor variables. They are ‘triptime’,’travel_type_id’,’online_booking’,’mobile_site_booking’,’booking_time’ and distance between two pairs of co-ordinate. ‘Car_Cancellation’ column is Target Variable. At first I deleted columns which contains no values. After that I modified date column. I sliced time from date. Then I omitted rows with null values. I also calculated the distance between 2 points. I turned car_cancellation column as factor variable. Here is the final version of first 6 rows:



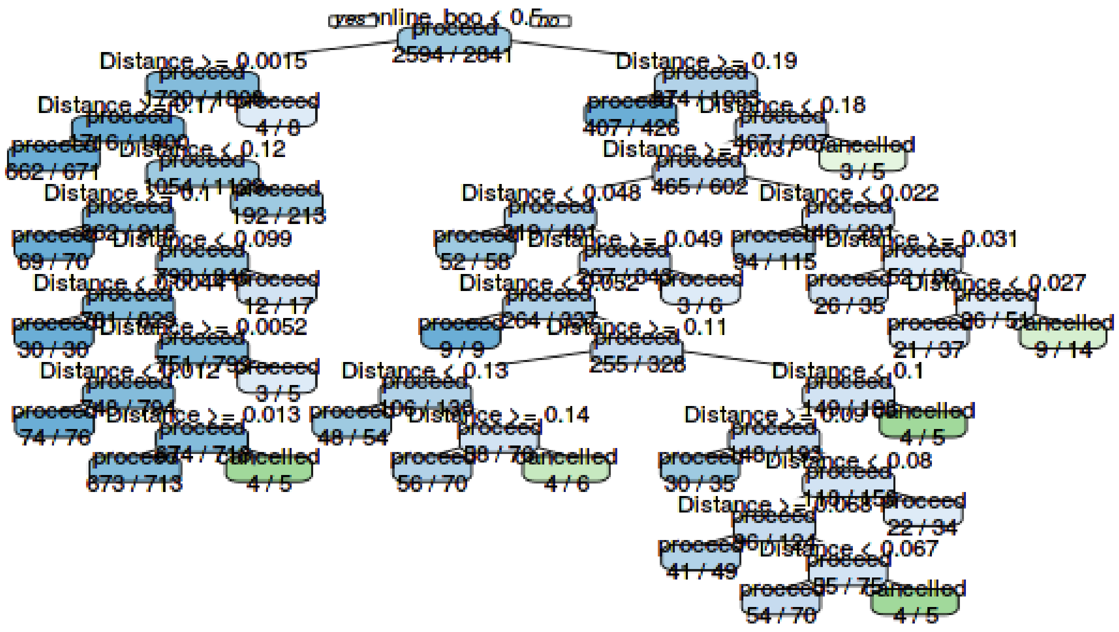
triptime	travel_type_id	online_booking	mobile_site_booking	bookingtime	Distance	Car_Cancellation
22:33:00	2	0	0	08:01:00	0.051764	proceed
12:43:00	2	0	0	09:59:00	0.018050	proceed
00:28:00	2	1	0	12:14:00	0.308547	proceed
13:12:00	2	0	0	12:42:00	0.010060	proceed
16:33:00	2	0	0	15:07:00	0.090550	proceed
18:00:00	2	0	0	15:11:00	0.231170	proceed

3. Classification Tree: To create a classification tree, I’ve taken consideration of ‘Distance’ and ‘online_booking’. These two predictor variables have role on car cancellations. If distance and online booking status don’t meet the condition then the status will be cancelled. If the condition is met, then the status will be proceed.

```r
tree_model<-rpart(Car_Cancellation ~ Distance + online_booking,
                  data=taxi.df,
                  method="class",
                  control=rpart.control(minsplit  = 50, #the minimum number of observations tha
                                        minbucket = 5, # the minimum number of observations in
                                        xval = 3, cp=0.0002)) # to use all samples for the firs
tree_model
```

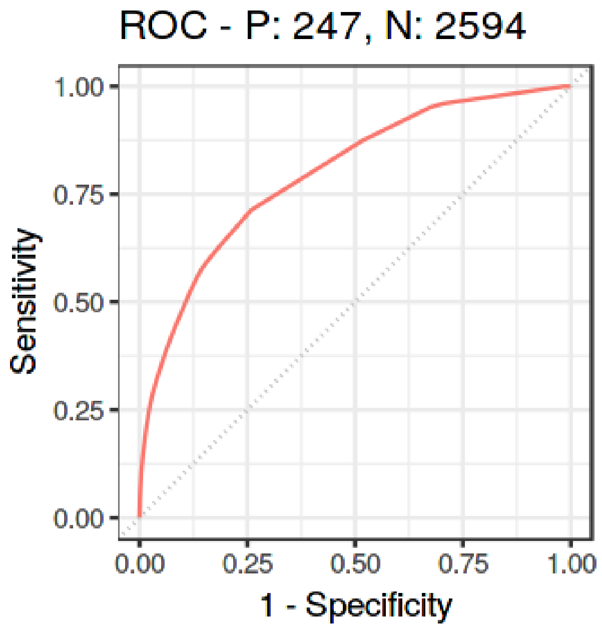
```
n= 2841

node), split, n, loss, yval, (yprob)
      * denotes terminal node

    1) root 2841 247 proceed (0.91305878 0.08694122)
      2) online_booking< 0.5 1808  88 proceed (0.95132743 0.04867257)
        4) Distance>=0.001535 1800  84 proceed (0.95333333 0.04666667)
          8) Distance>=0.17197 671   9 proceed (0.98658718 0.01341282) *
          9) Distance< 0.17197 1129  75 proceed (0.93356953 0.06643047)
           18) Distance< 0.1183135 916  54 proceed (0.94104803 0.05895197)
             36) Distance>=0.1028035 70   1 proceed (0.98571429 0.01428571) *
             37) Distance< 0.1028035 846  53 proceed (0.93735225 0.06264775)
               74) Distance< 0.099141 829  48 proceed (0.94209891 0.05790109)
                148) Distance< 0.0044385 30   0 proceed (1.00000000 0.00000000) *
                149) Distance>=0.0044385 799  48 proceed (0.93992491 0.06007509)
                  298) Distance>=0.0051685 794  46 proceed (0.94206549 0.05793451)
                    596) Distance< 0.0124565 76   2 proceed (0.97368421 0.02631579) *
                    597) Distance>=0.0124565 718  44 proceed (0.93871866 0.06128134)
```

```r
rpart.plot(tree_model,
           fallen.leaves = FALSE, # to position the leaf nodes at the bottom of the graph c
           type  = 1, # 1 Label all nodes, not just leaves.
           extra = 2, # 1 Display the number of observations that fall in the node
           split.font = 1, # Font for the split labels. 1=normal 2=bold
           varlen = -10) # Length of variable names in text at the splits (and, for class r
```



4. <u>Predictive Performance in terms of Error Rate:</u>

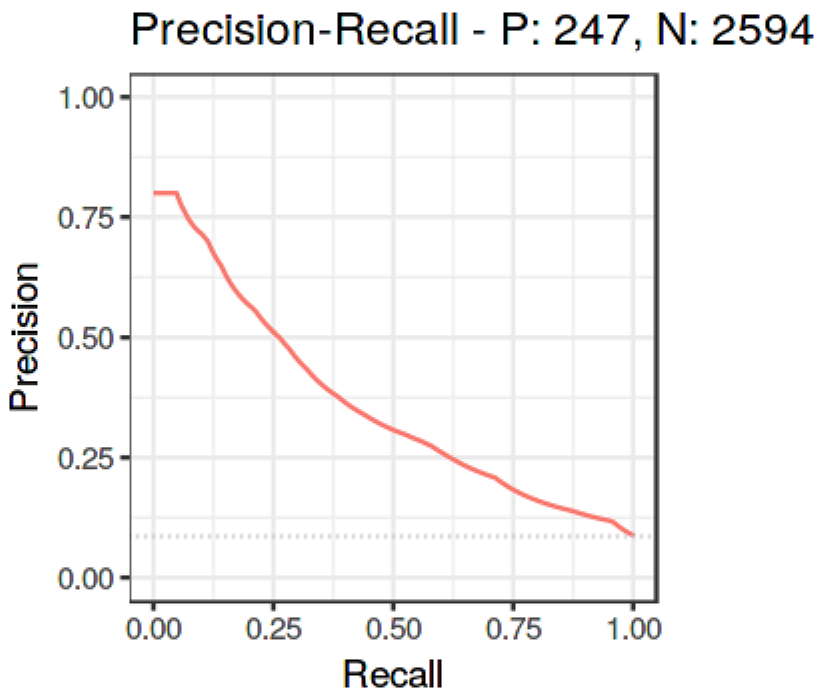
modnames	dsids	curvetypes	aucs
m1	1	ROC	0.7979673



From the above figure, we can see that the AUC is 0.7979673. If AUC value is going towards 1, then it can be said that the model has good measure to distinguish among different classes. So, we can say that this model is good.

5. <u>Predictive Performance in terms of Ranking(Lift):</u>

```
autoplot(sscurves, "PRC")
```



This model shows lower recall and higher precision values. If it would closer to 1, it could be called ideal model. But its far away from a good model.

Appendix

1. taxi_cancel.df <- read.csv(file = "Taxi-cancellation-case.csv",

   header = TRUE,

   sep = ",")

taxi_cancel.df

2.
   ```
   taxi_cancel.df <- taxi_cancel.df[-c(4,8,9,11) ]
   head(taxi_cancel.df)
   ```

3.
   ```
   triptime <- format(as.POSIXct(strptime(taxi_cancel.df$from_date,"%d/%m/%Y %H:%M",tz="")) ,format = "%H:%M")
   bookingtime <- format(as.POSIXct(strptime(taxi_cancel.df$booking_created,"%d/%m/%Y %H:%M",tz="")) ,format = "%H:%M")
   taxi_cancel.df$triptime <- triptime
   taxi_cancel.df$bookingtime <- bookingtime
   taxi_cancel.df
   ```

4.
   ```
   taxi.df <- na.omit(taxi_cancel.df)
   head(taxi.df)
   ```

5.
   ```
   taxi.df$Car_Cancellation <- factor(taxi.df$Car_Cancellation, levels=c(0,1), labels = c("proceed", "cancelled"))
   head(taxi.df)
   ```

6.
   ```
   taxi.df <- subset(taxi.df, select = c(16,4,8,9,17,11,12,13,14,15))
   head(taxi.df)
   ```

7.
   ```
   triptime1 <- as.ITime(taxi.df$triptime)
   bookingtime1 <- as.ITime(taxi.df$bookingtime)
   taxi.df$triptime <- triptime1
   taxi.df$bookingtime <- bookingtime1
   taxi.df
   ```

8.
   ```
   distance <- abs((taxi.df$from_lat-taxi.df$to_lat) + (taxi.df$from_long-taxi.df$to_long))
   taxi.df$Distance <- distance
   head(taxi.df)
   ```

9.
   ```
   taxi.df <- taxi.df[c(1,2,3,4,5,11,10) ]
   head(taxi.df)
   ```

10.
    ```
    tree_model<-rpart(Car_Cancellation ~ Distance + online_booking,
                data=taxi.df,
                method="class",
                control=rpart.control(minsplit  = 50,
                                minbucket = 5,
                                xval = 3, cp=0.0002))
    tree_model
    ```

11.
```
rpart.plot(tree_model,
          fallen.leaves = FALSE,
           type  = 1,
          extra = 2,
         split.font = 1,
        varlen = -10)
```

12.
```
treePred.class <- predict(tree_model, data = train.df, type = "class")
head(treePred.class)
```

13.
```
treePred.score <- predict(tree_model, data = train.df, type = "prob")
head(treePred.score)
```

14.
```
cancelledScore <- treePred.score[,2]
head(cancelledScore)
```

15.
```
sscurves <- evalmod(scores = cancelledScore, labels = taxi.df$Car_Cancellation)
autoplot(sscurves, "ROC")
auc(sscurves) %>% filter(curvetypes=='ROC')
```

16.
```
confusionMatrix(taxi.df$Car_Cancellation,factor( ifelse(cancelledScore > 0.3,
"cancelled", "proceed") ),positive = 'cancelled')
```

17.
```
a.df <- data.frame(cancelledScore,taxi.df$Car_Cancellation)
a.df %>% arrange(desc(cancelledScore)) %>% mutate(id=1:2841)
```

18.
```
autoplot(sscurves, "PRC")
```