HarvardX PH125.9x Data Science Capstone-Online Shoppers Intention Classification

Alvin Subakti

January 8, 2021

1. Introduction

Background

Coronavirus especially the new variant 2019-nCoV has cause a global pandemic issue since the end of 2019. One of the method to avoid further increase in number of positive infection, every country has issued their own method of social distancing. This decision has affected a lot of aspect in human activity especially everyone's economic condition. Due to the social distancing policy issued, it has become harder for business to be conducted offline and many of them have migrated and adapted an online system for their business.

The global pandemic has caused a significant increase in online activity especially ecommerce activity. Due to that, a proper consumer analysis need to be done to further increase the effectivity hence increasing sales revenue. An organization would like to focus more on online customers that have a higher tendency of purchasing a product. They will also like to have an eraly detection and behavioral prediction systems for future customer. In this project, we will try to classify the intention of an online shopper given their traits and charateristics by ultilizing various supervised model.

DataSet Information

The Dataset used in this problem is the 'Online Shoppers Purchasing Intention' dataset, this dataset can be found and downloaded from UCI Machine Learning Repository through this following link:

"https://archive.ics.uci.edu/ml/machine-learning-databases/00468/online_shoppers_intention.csv"

It is a multivariate dataset with a total of 12330 observations and 18 attributes including the class that will be predicted(Revenue). It is formed so that each session would belong to a different user in a 1-year period to avoid any tendency to a specific campaign, special day, user profile, or period. The attributes are as follows:

1. Administrative: the number of administrative page that is visited by the user on that session (discrete)
2. Administrative_Duration: the Duration of the time spent by the user in administrative pages in that session (continuous)
3. Informational: the number of information pages that is visited by the user on that session (discrete)
4. Informational_Duration: the Duration of the time spent by the user in information pages in that session (continuous)
5. ProductRelated: the number of product related pages that is visited by the user on that session (discrete)
6. ProductRelated_Duration: the Duration of the time spent by the user in product relalted pages in that session (continuous)

7. BounceRates: Ratio of user that immediately leaves a certain webpage after just opening one page (continuous)
8. ExitRates: Ratio of user that left a certain webpage by the end of the session (continuous)
9. PageValues: Average value for a webpage that a user visited before completing an ecommerce transaction (continuous)
10. SpecialDay: Correlation of webpage visit with a specific special day (continuous)
11. Month: The month of visit (categorical)
12. OperationSystems: Operation system used by the user in the session (categorical)
13. Browser: The type of browser used by the user (categorical)
14. Region:Geographical location of the user (categorical)
15. TrafficType: the type of source that sends the user to the webpage (categorical)
16. VisitorType: Type of user/visitor (categorical)
17. Weekend: Logical value whether the day of visit is in a weekend or not
18. Revenue: Label class whether the visit produce revenue or not

Goal

The goal of this project is to be able to visualize and gain insight of the dataset. Also to be able to construct a machine learning algorithm to correctly predicts the class future online ecommerce users. The binary classification in this project is whether the online ecommerce users make a purchase hence producing revenue or not.

The main parameter that will be used in this project is the F1 score that has range of 0 to 1. F1 score is used instead of the commmonly used accuracy due to the imbalance class dataset that we have. So by using F1 score as well as specificity as an additional parameter, it will be able to evaluate the performance of a model in classifying both class correctly better than accuracy.

2. Methods and Analysis

2.1. Data Preprocessing

First, the prevew of the dataset can be seen in the following table, where the dataset consisted of 18 different features including the class that will be predicted which is revenue

```
##   Administrative Administrative_Duration Informational Informational_Duration
## 1              0                       0             0                       0
## 2              0                       0             0                       0
## 3              0                       0             0                       0
## 4              0                       0             0                       0
## 5              0                       0             0                       0
## 6              0                       0             0                       0
##   ProductRelated ProductRelated_Duration BounceRates ExitRates PageValues
## 1              1                 0.000000  0.20000000 0.2000000          0
## 2              2                64.000000  0.00000000 0.1000000          0
## 3              1                 0.000000  0.20000000 0.2000000          0
## 4              2                 2.666667  0.05000000 0.1400000          0
## 5             10               627.500000  0.02000000 0.0500000          0
```

```
## 6             19              154.216667  0.01578947 0.0245614          0
##   SpecialDay Month OperatingSystems Browser Region TrafficType
## 1          0   Feb                1       1      1           1
## 2          0   Feb                2       2      1           2
## 3          0   Feb                4       1      9           3
## 4          0   Feb                3       2      2           4
## 5          0   Feb                3       3      1           4
## 6          0   Feb                2       2      1           3
##         VisitorType Weekend Revenue
## 1 Returning_Visitor   FALSE   FALSE
## 2 Returning_Visitor   FALSE   FALSE
## 3 Returning_Visitor   FALSE   FALSE
## 4 Returning_Visitor   FALSE   FALSE
## 5 Returning_Visitor    TRUE   FALSE
## 6 Returning_Visitor   FALSE   FALSE
```

Then we can check that there is no missing values in the dataset.

```
sum(is.na(online_shoppers_intention))
```

```
## [1] 0
```

Next we are going to change the data or encode the data into its proper data type. As it is noted in the previous explanation, there are both numerical and categorical data type in this dataset. Most of the categorical data in this dataset were not properly encoded yet. So we will encode these data. The 'Month' feature is encoded into numbers corresponding which month of the year it is. Then 'Weekend' and 'Revenue' feature which is originally in a boolean form is also encoded into 0 for `FALSE` and 1 for `TRUE`. Finally the 'VisitorType' feature is also encoded with 0 for returning visitors, 1 for new visitors, and 2 for others. Data type for every other features that is not modified above is also being redefined by using the definition in the dataset section to ensure that every feature is in the correct type.

```
# checking whether every month already have the proper abbreviation
unique(online_shoppers_intention$Month)

# changing "June" to its proper abbreviation which is "Jun"
online_shoppers_intention$Month[online_shoppers_intention$Month=='June']<-'Jun'

# changing month abbreviation to numbers
online_shoppers_intention$Month<-match(online_shoppers_intention$Month,month.abb)


# encoding Weekend column to 0 (FALSE) and 1 (TRUE)
online_shoppers_intention$Weekend<-factor(online_shoppers_intention$Weekend,levels = c(FALSE,TRUE),labe


# checking the levels in VisitorType
unique(online_shoppers_intention$VisitorType)
# encode VisitorType 0 for returning visitors, 1 for new visitors, 2 for other
online_shoppers_intention$VisitorType<-factor(online_shoppers_intention$VisitorType,levels=c("Returning

# encode revenue(target label) with 0 (FALSE) and 1 (TRUE)
online_shoppers_intention$Revenue<-factor(online_shoppers_intention$Revenue,levels = c(FALSE,TRUE),labe
```

Now we can check the final descriptive statistics for each feature after being modified, we can also verify that there is still no missing value in the dataset indicating proper preprocessing and encoding has been done.

```r
summary(online_shoppers_intention)
```

```
##  Administrative   Administrative_Duration Informational
##  Min.   : 0.000   Min.   :   0.00         Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.:   0.00         1st Qu.: 0.0000
##  Median : 1.000   Median :   7.50         Median : 0.0000
##  Mean   : 2.315   Mean   :  80.82         Mean   : 0.5036
##  3rd Qu.: 4.000   3rd Qu.:  93.26         3rd Qu.: 0.0000
##  Max.   :27.000   Max.   :3398.75         Max.   :24.0000
##
##  Informational_Duration ProductRelated   ProductRelated_Duration
##  Min.   :   0.00        Min.   :  0.00   Min.   :    0.0
##  1st Qu.:   0.00        1st Qu.:  7.00   1st Qu.:  184.1
##  Median :   0.00        Median : 18.00   Median :  598.9
##  Mean   :  34.47        Mean   : 31.73   Mean   : 1194.8
##  3rd Qu.:   0.00        3rd Qu.: 38.00   3rd Qu.: 1464.2
##  Max.   :2549.38        Max.   :705.00   Max.   :63973.5
##
##   BounceRates         ExitRates         PageValues        SpecialDay
##  Min.   :0.000000   Min.   :0.00000   Min.   :  0.000   Min.   :0.00000
##  1st Qu.:0.000000   1st Qu.:0.01429   1st Qu.:  0.000   1st Qu.:0.00000
##  Median :0.003112   Median :0.02516   Median :  0.000   Median :0.00000
##  Mean   :0.022191   Mean   :0.04307   Mean   :  5.889   Mean   :0.06143
##  3rd Qu.:0.016813   3rd Qu.:0.05000   3rd Qu.:  0.000   3rd Qu.:0.00000
##  Max.   :0.200000   Max.   :0.20000   Max.   :361.764   Max.   :1.00000
##
##      Month      OperatingSystems    Browser         Region      TrafficType
##  5      :3364   2      :6601     2      :7961   1      :4780   2      :3913
##  11     :2998   1      :2585     1      :2462   3      :2403   1      :2451
##  3      :1907   3      :2555     4      : 736   4      :1182   3      :2052
##  12     :1727   4      : 478     5      : 467   2      :1136   4      :1069
##  10     : 549   8      :  79     6      : 174   6      : 805   13     : 738
##  9      : 448   6      :  19     10     : 163   7      : 761   10     : 450
##  (Other):1337   (Other):  13     (Other): 367   (Other):1263   (Other):1657
##  VisitorType Weekend  Revenue
##  0:10551     0:9462   0:10422
##  1: 1694     1:2868   1: 1908
##  2:   85
##
##
##
##
```

```r
sum(is.na(online_shoppers_intention))
```

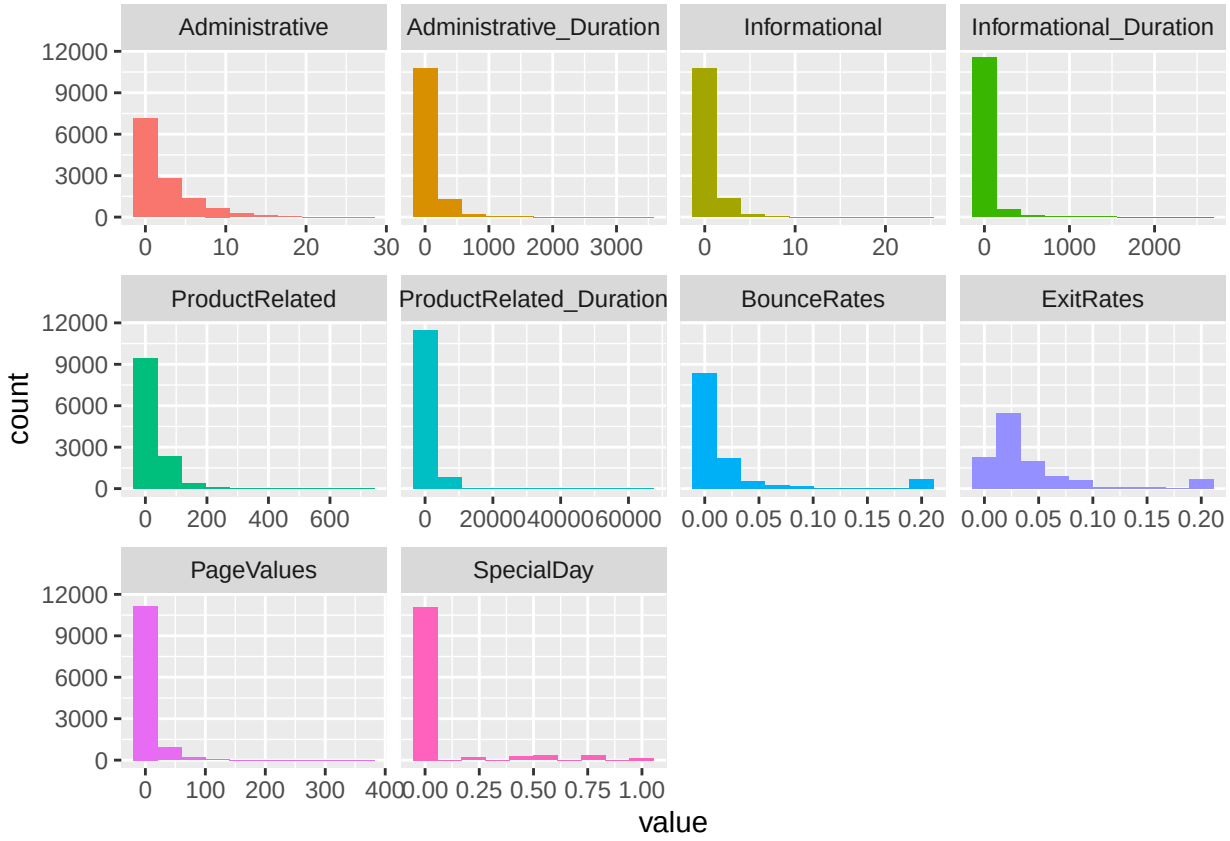
```
## [1] 0
```

2.2. Data Visualization

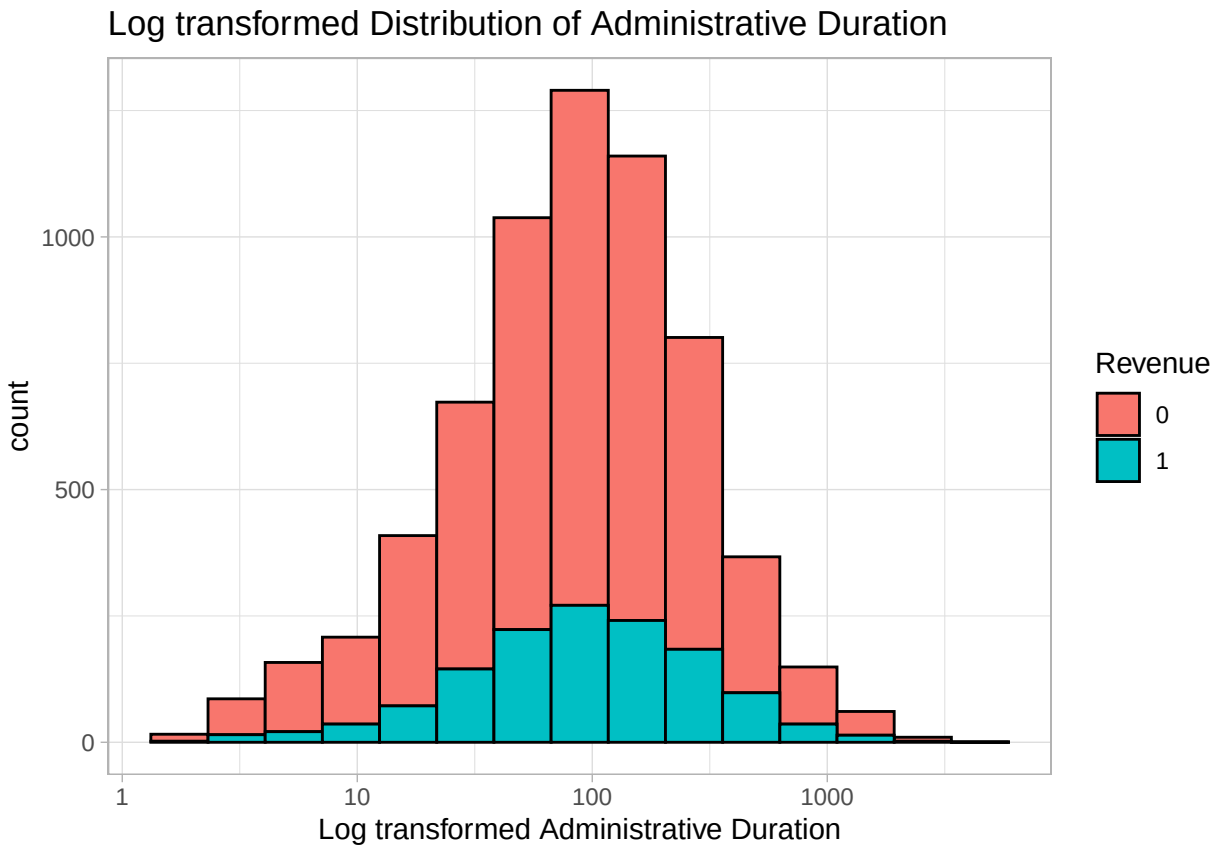
Numeric Feature

Data visualization is done to obtain more information regarding the dataset. First, all of the histogram for numeric type of feature is displayed below by ultizling the `plot_num` function from `funModeling` library. As

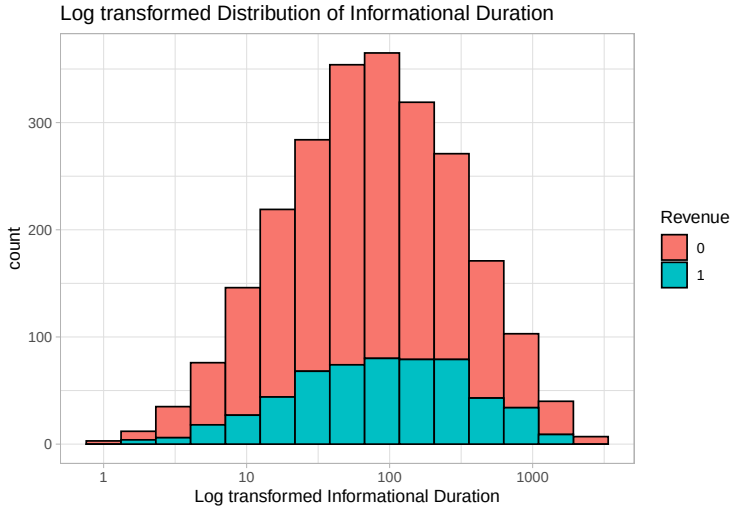
it is shown in the compilation of histograms in the figure below, most numeric feature has a tendency where the value of y axis will decrease exponentially as the x axis increase.

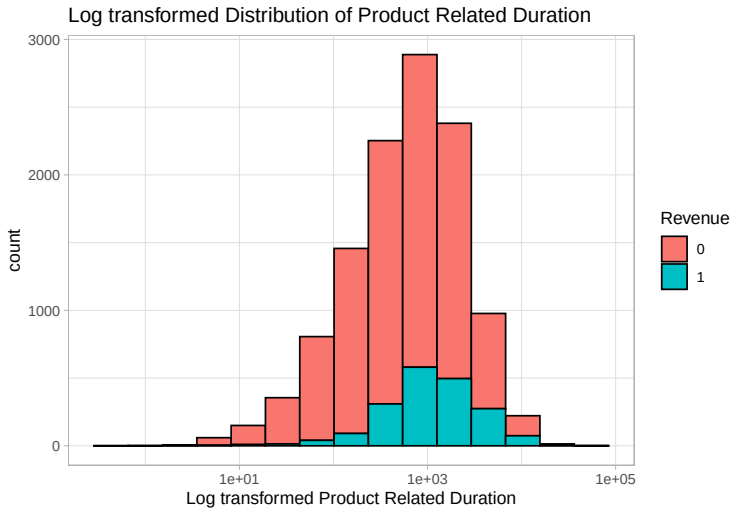


To obtain a much clearer relationship for the numeric features, the features will be transformed by using log transformation on the x axis. The following plot shows the distribution of 'Administrative_Duration' feature after being transformed. From the plot, we can say that after the transformation, the distribution is relatively similar to a normal distribution. Also, by dividing and stacking the count based on the class of Revenue, we can imply that both class has a similar distribution for this feature

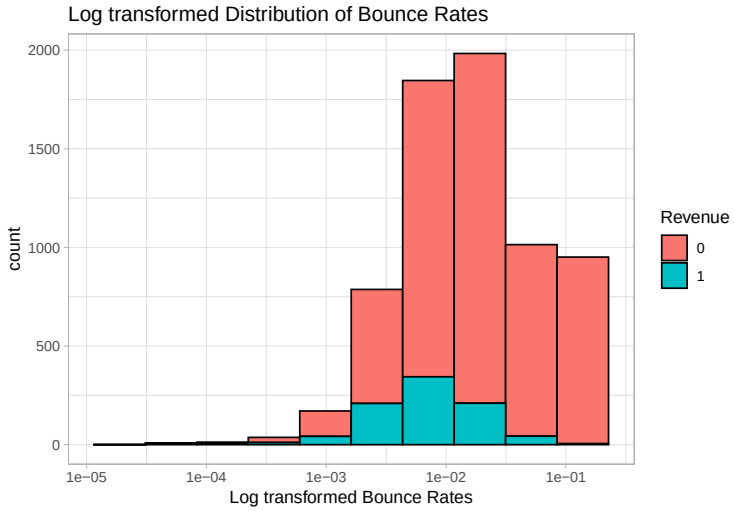


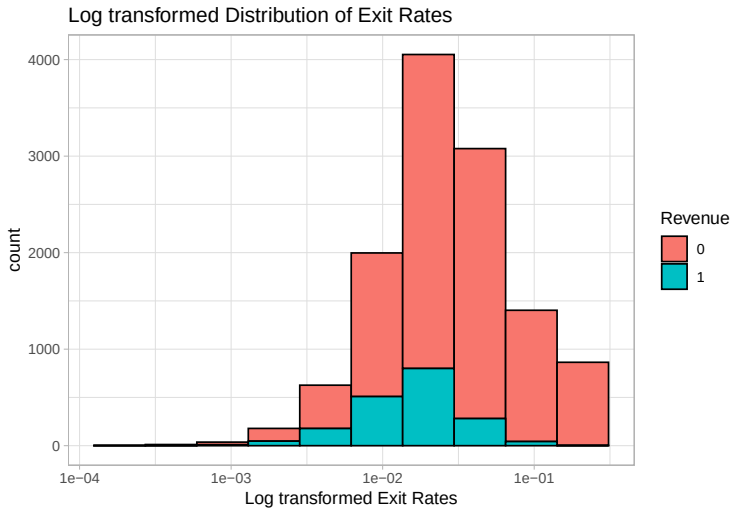
These findings will also occur in several other numeric features such as 'Informational_Duration' and 'ProductRelated_Duration'. One of the reason of these findings is due to the similar characteristics of the three feature which all of them are explaining about the duration or time spent in a type of pages.





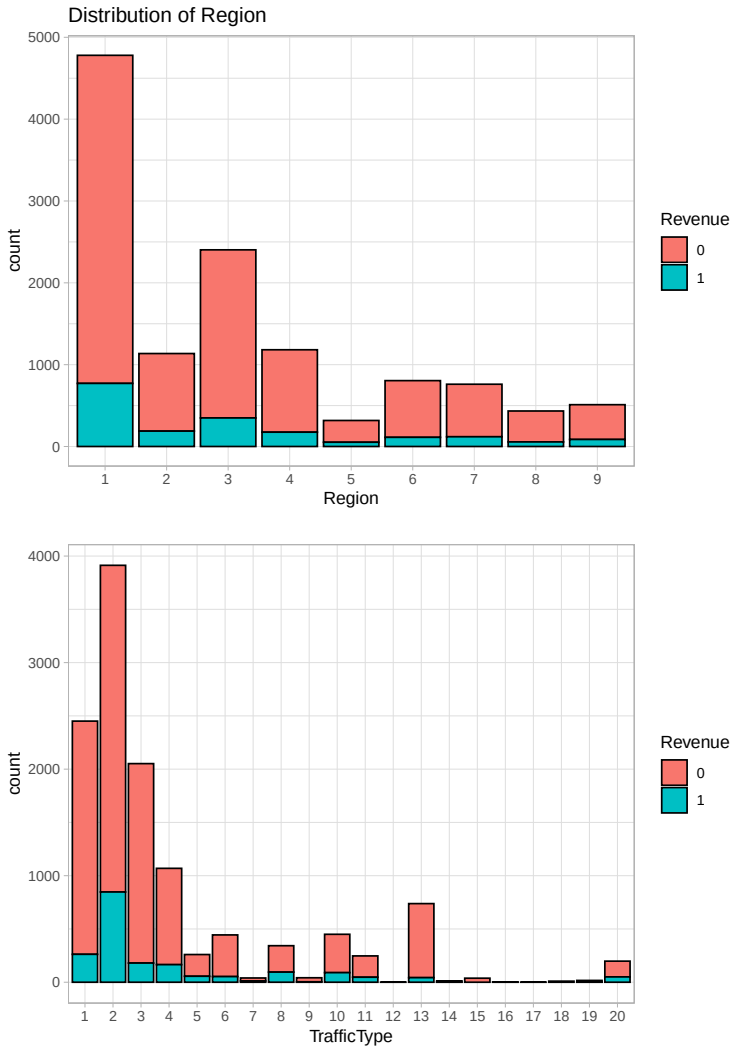
While for several other features such as 'BounceRates' and 'ExitRates', there is no clear relationship for the distribution. However, the fact that the distribution of both class if classified by Revenue remained similar



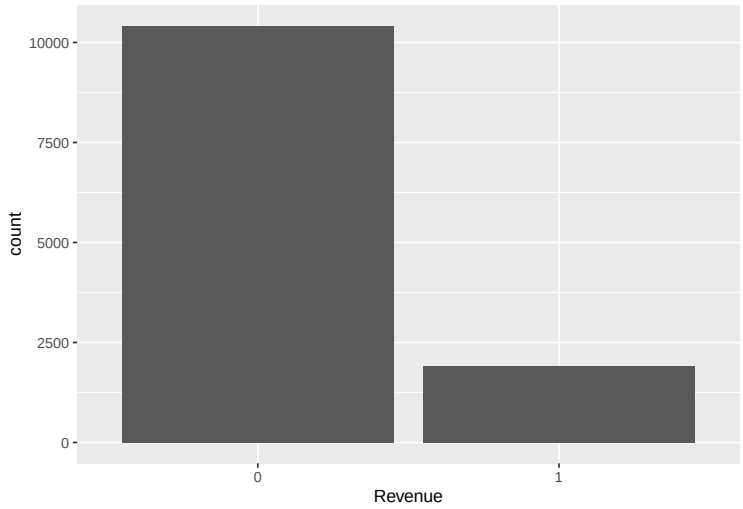


Categorical Features

The plot shown below are several histogram plot showing the distribution of categorical features in the datset binned by its respective categories. From the features 'Region' and 'TrafficType' we can still see that the distribution different class of revenue is relatively similar. There is no clear relationship between different categories in a feature besides that the first few categories has relatively more frequency than the others. With further obsevation from other categorical features and combined with the previous numerical features, we can imply that there are no clear distinction in distribution between different class of revenue.



For the distribution of frequency of Revenue as shown below, there is a clear diffence in amount of positive (1) class and negative (0) class. there are 10422 observations with negative class while only 1908 observations with positive class. The amount of positive class observation is nearly only 20% the amount of negative class observation. Therefore, a undersampling/oversampling method needs to be applied in order to increase the capability of our model to predict positive class.



The technique that will be applied later is SMOTE (Synthetic Minority Oversampling Techique). First introduced in 2002, it is a algorithm that can tackle imbalance dataset issue. The general idea of this method is to generate new examples for the minority classes by using neares neighbour while also undersample the majority class which produce a much more balanced dataset.

2.3. Feature Elimination and SMOTE

Feature Elimination need to be done in order to decrease the amount of ineffective or redundant feature. This is done so that the model that we obtain later will not overfit to the training data used. The training data generated will also only consisted of features that is considered important and able to increase the performance of a model.

The feature elimination method that will be applied is the RFECV (Reccursive Feature Elimination Cross Validation). It is a simple backwards selection algorithm that starts with the whole features used to train the model. It implements backward elimination based on predictor importance ranking. Each possible predictors are ranked and the predictors with the least importance are sequentially eliminated prior to modelling.

As the result of the RFE with cross validation method, 2 features are removed and we are going to proceed with 15 features.

```
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
```

```
results <- rfe(online_shoppers_intention[,-which(colnames(online_shoppers_intention)=='Revenue')], onli
```

Next, the dataset is splitted into training data and test data with 8:2 proportion, hence we obtained a `train_set` and `test_set`. Then both sets are used to generate new dataset and vector which is the independent variables(x) and dependent variables(y) for both train and test data. The dataset and vector created are `x_train`, `y_train`, `x_test`, `y_test`.

Then, SMOTE is performed for `train_set`. The specifications for this SMOTE are for each minority class sample, 2 new sample will be generated. 2 sample will also be sampled from the majority class for each new sample generated in the minority class.

The new training data named `newdata` showed that the classes are more balance. The minority class is now 75% the amount of the majority class, this is definitely better compared to the previous training data which have a minority class of roughly 18.3% the amount of majority class.

A new x and y for training is also generated namely `x_train_smote` and `y_train_smote` respectively.

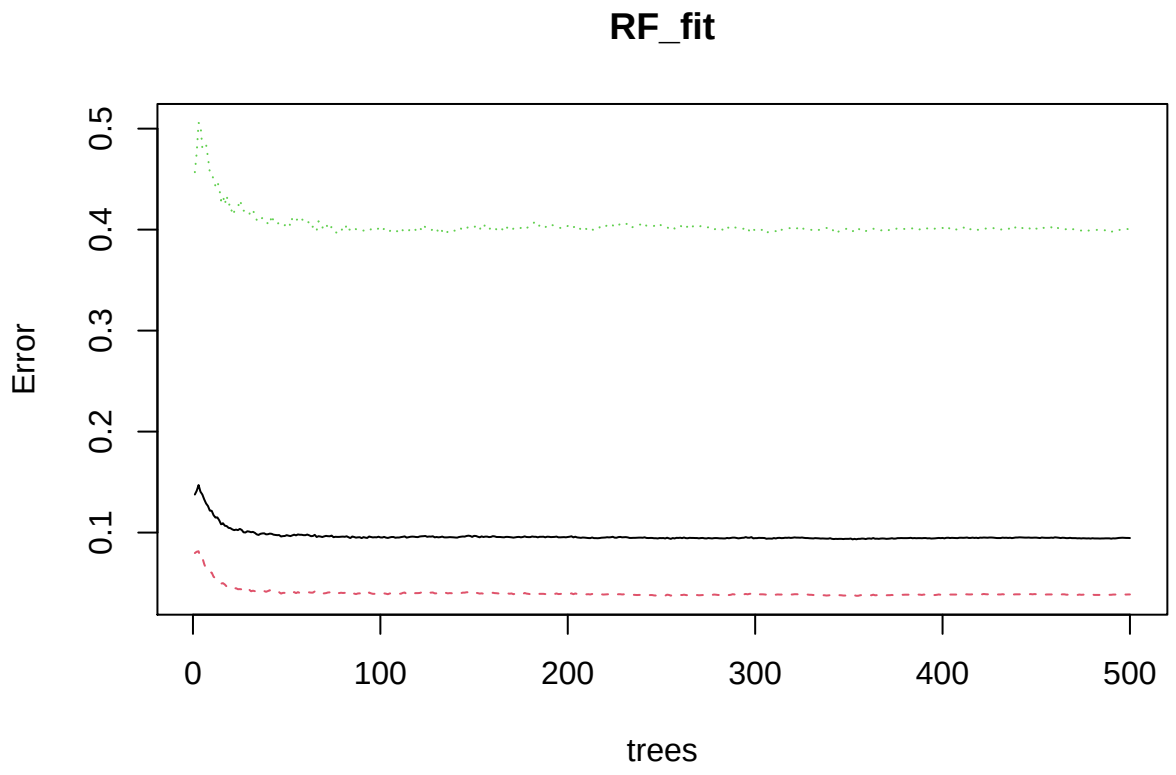
3. Modelling

3.1. Training Data COmparison

In order to show that the SMOTE generated training dataset is better than the orginial training dataset, a model is going to be trained using both model and compared its performance. The model used is Random Forest. Since the accuracy might still be high even though the ability to correctly predict positive class is low. Therefore, F1 score is going to be used.

Specificity is also going to be paid attention as an additional parameter because specificity is the ratio of positive class being correctly predicted and the minority class in this dataset is the positive class. Therefore, a higher specificity means a higher performance in correctly classifying positive class which is better because we can obtain a more accurate revenue and conversion rate. First, a random forest model using normal training dataset is trained and evaluated.

```
RF_fit <- randomForest(x=x_train,y=y_train)
plot(RF_fit)
```



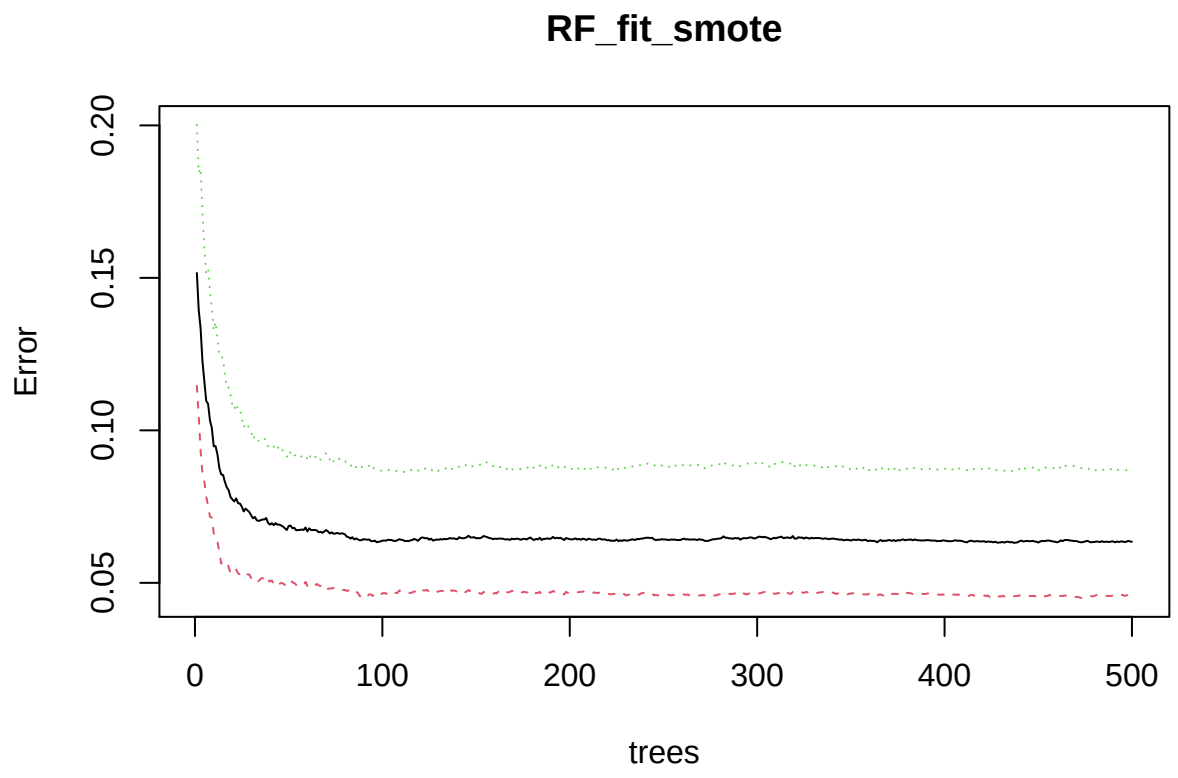

```
confusionMatrix(predict(RF_fit, x_test), y_test,positive='1')
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction    0    1
##          0 1997  167
##          1   88  215
##
```

```
##                Accuracy : 0.8966
##                  95% CI : (0.8839, 0.9084)
##     No Information Rate : 0.8452
##     P-Value [Acc > NIR] : 6.351e-14
##
##                   Kappa : 0.5686
##
##  Mcnemar's Test P-Value : 1.037e-06
##
##             Sensitivity : 0.56283
##             Specificity : 0.95779
##          Pos Pred Value : 0.70957
##          Neg Pred Value : 0.92283
##              Prevalence : 0.15484
##          Detection Rate : 0.08715
##    Detection Prevalence : 0.12282
##       Balanced Accuracy : 0.76031
##
##        'Positive' Class : 1
##
```

Then a random forest model using training dataset modified by SMOTE is trained and evaluated.

```
RF_fit_smote <- randomForest(x=x_train_smote,y=y_train_smote)
plot(RF_fit_smote)
```



```r
confusionMatrix(predict(RF_fit_smote, x_test), y_test,positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1891  101
##          1  194  281
##
##                Accuracy : 0.8804
##                  95% CI : (0.867, 0.893)
##     No Information Rate : 0.8452
##     P-Value [Acc > NIR] : 3.294e-07
##
##                   Kappa : 0.5844
##
##  Mcnemar's Test P-Value : 8.487e-08
##
##             Sensitivity : 0.7356
##             Specificity : 0.9070
##          Pos Pred Value : 0.5916
##          Neg Pred Value : 0.9493
##              Prevalence : 0.1548
##          Detection Rate : 0.1139
##    Detection Prevalence : 0.1925
##       Balanced Accuracy : 0.8213
##
##        'Positive' Class : 1
##
```

We can see clearly from both plot generated that random forest model by using SMOTE training dataset has a better performance because it has has a low error significantly different than the normal random forest that is unable to decrease too much error after using a lot of trees. From the confusion matrix of both model it can also be seen that even though the initial random forest has a better performance compared to the one using modified dataset, it has a much higher specificity. For better comparison, a table is generated. Another comparison with a naive prediction by predicting all class as '0' will also be added.

Method	F1	Accuracy	Sensitivity
Naive Predict '0'	0.9160808	0.8451561	0.0000000
RF Normal train set	0.9402072	0.8970409	0.5628272
RF SMOTE train set	0.9276429	0.8808269	0.7356021

From the table above, we can clearly see that even the naive prediction of prediciting the whole class as 0 has a fairly good accuracy of 84.51%. It also still have a fairly high F1 value of 0.916. However it has a very poor specificity of 0 which means it cannot predict positive class at all. In the other hand, even though the model using orginal training dataset has a sllightly higher F1 score of 0.9399 compared to 0.9276, the model using modified training dataset has a much better specificity of 0.7356 compared to only 0.5654. Due to these reasons, we will utilized the modified training dataset in further model training.

3.2. Algorithm Comparison

Random Forest

Before starting with training other machine learning models, the random forest model obtained from the previous section will be kept in a new table used to compare the performance of different models.

```
model_results <- data_frame(Algorithm = "Random Forest",F1 = F1_Score(y_test,predict(RF_fit_smote, x_tes
```

XGBoost

A matrix will be generated beforehand to be able to train an XGBoost model.

```
xgb_train<-xgb.DMatrix(data=data.matrix(x_train_smote),label=y_train_smote)
xgb_test<-xgb.DMatrix(data=data.matrix(x_test),label=y_test)
```

Training and evaluation of performance of XGBoost model is done. The method of adding F1 score results of different machine learning model is also going to be used in further models.

```
xgb_fit<-xgboost(xgb_train,nrounds=50)
model_results <- bind_rows(model_results,
                           data_frame(Algorithm="XGBoost",
                                      F1=F1_Score(y_test,as.factor(levels(y_test)[round(predict(xgb_fit
```

Logistic Regression (GLM)

Training and evaluation of performance of Generalized Linear Model is done.

```
train_glm <- train(x=x_train_smote,y=y_train_smote, method = "glm")
```

k-Nearest Neighbour

Training and evaluation of performance of k-Nearest Neighbour Model is done.

```
train_knn <- train(x=x_train_smote,y=y_train_smote, method = "knn")
```

Naive Bayes

Training and evaluation of performance of Naive Bayes model is done.

```
naive_fit<-naive_bayes(x=x_train_smote,y=y_train_smote)
```

Decision Tree

Training and evaluation of performance of Decision Tree model is done.

```
DT_fit<-train(x=x_train_smote,y=y_train_smote,method='rpart')
```

The final results of all the model can be seen in the table below

Algorithm	F1
Random Forest	0.9279058
XGBoost	0.9245375

Algorithm	F1
Generalized Linear Model	0.9165863
k-Nearest Neighbour	0.8548759
Naive Bayes	0.7368421
Decision Tree	0.9250671

We can conclude that Random Forest has the best F1 Score, therefore Random Forest model will be tuned by using hyperparameter tuning.

3.3 Hyperparameter Tuning Random Forest

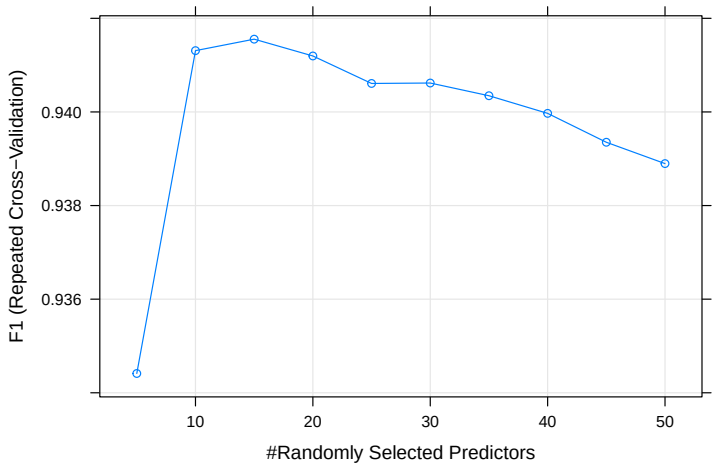
Before performing hyperparameter tuning for random forest model, a new function named `f1` function is defined as a metric to be optimized since there is no default metric for F1 score.

```
f1 <- function(data, lev = NULL, model = NULL) {
  f1_val <- F1_Score(y_pred = data$pred, y_true = data$obs, positive = lev[1])
  c(F1 = f1_val)
}
```

In the previous Random Forest plot it is clear that random forest model will not perform any better after `ntree` is bigger than 100. Performing grid cross validation with a lot of `ntree` will also increase computation time. Therefore, we will going to limit `ntree` by 100. Hyperperameter tuning to choose the best `mtry` is performed by first defining the candidate values of `mtry`.

```
control_rf <- trainControl(method="repeatedcv", number=10,
                           repeats=3,summaryFunction = f1)
tunegrid<-expand.grid(.mtry=seq(5,50,5))
set.seed(1,sample.kind = 'Rounding')
rf_gridsearch <- train(Revenue~.,data=newdata,
                       method="rf", metric='F1', tuneGrid=tunegrid, ntree=100,
                       trControl=control_rf)
```

From the plot below, we can see that this model perform best when `mtry=15`

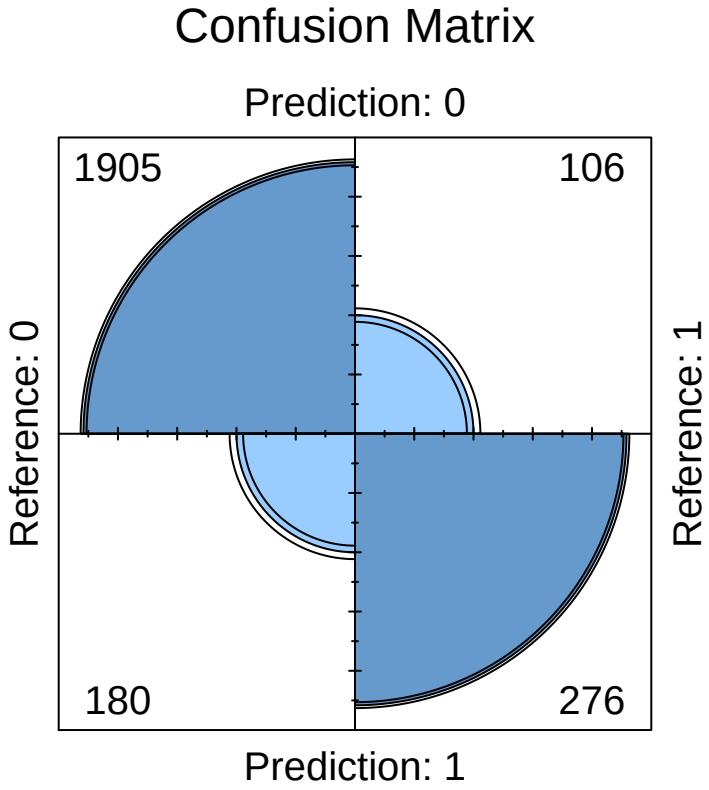


Algorithm	F1
Random Forest	0.9279058

Algorithm	F1
XGBoost	0.9245375
Generalized Linear Model	0.9165863
k-Nearest Neighbour	0.8548759
Naive Bayes	0.7368421
Decision Tree	0.9250671
Tuned Random Forest	0.9304708

3.4. Final Model Performance

Based on the result in the previous section, we obtain that our final model which is tuned Random Forest has the highes F1 score of 0.9284664. The detail of the model's performance can be seen below.



We can see that this model has a moderate accuracy of 88.49% and a relatively good sensitivity(0.7251) as well as specificity(0.9141). This means that 72.51% of the positive class and 91.41% of the negative class are correctly classified. From the confusion matrix figure we can also see that there are 276 observations that is correctly classified in positive value compared to 106 falsely classified. This numbers and confusion matrix implies that our model perform considerably well in classifying both negative and positive class correctly.

4. Conclusion

From this project, we can conclude that the dataset used which is 'Online Shoppers Purchasing Intention' has a similar distribution for all class in the Revenue Column. We can also find that SMOTE method could be a way to improve the performance of our model and to tackle imbalance dataset problem. From all the

model constructed, We can also conclude that Random Forest model has the best performance with the best hyperparmater of `mtry=15`. The final model obtained has an F1 score of 0.9304.

Further approach that can be done is to perform a one hot encoding in all categorical features which will make several machine learning models perform better. It is also possible to perform hyperparameter tuning in all of the machine learning models to properly search for the best model. Another approach that can be done is to perform feature elimination by utilizing correalation between features.

Appendix

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       i386-w64-mingw32
## arch           i386
## os             mingw32
## system         i386, mingw32
## status
## major          4
## minor          0.1
## year           2020
## month          06
## day            06
## svn rev        78648
## language       R
## version.string R version 4.0.1 (2020-06-06)
## nickname       See Things Now
```