Math 533 Final Project - Statistical Learning

Janice Ferrer

December 11, 2020

Online Shopper's Intention - Predicting Purchasing Intention

Abstract

As the population of the internet grows, online shopping becomes a growing trend. However, in the experience of online shopping, it is difficult to predict the purchasing intention of the shopper without face-to-face interaction. Thus, machine learning and data analytics have become important tools to predict the purchase intention of online shoppers. In this paper, we extract features from the data provided by the visits of uses and feed it into machine learning classification algorithms to construct a predicting model. The classification methods we will implement include random forests, support vector machines (SVM), logistic regression, decision trees, and neural networks. In the efforts of finding the best model among each method, we have conducted a repeated 10 fold cross-validation for each classification method and get the average accuracy among the validated models. From our findings, the best model overall is Random Forest using 10-fold cross validation with 90.46% accuracy. We can also say that a possible strategy to gain higher revenue are focusing on increasing the page value, lowering the bounce rate, administrated related pages type '5' and below, and more marketing in November to drive revenue based on exploratory data analysis and model findings.

Introduction

Over the years, the phenomenon of online shopping has increasingly grown more than in-person shopping. Many consumers turn to shopping online due to convenience and accessible information on desired products. The fact that all the information and consumer reviews are readily available online helps consumers come to a final decision on what products to purchase. Thus, some pros of online shopping include saving time and energy, 24/7 availability, and the ability to compare prices of products. However, online shopping can be difficult on the sales and marketing side since it can be difficult to predict the purchasing intention of customers without the personal interaction that a salesperson has in physical retailing. "Therefore, many e-commerce and information technology companies invest in early detection and behavioral prediction systems to imitate the behavior of a sales person in the virtual shopping environment" [1]. Additionally, in academics, there have been approaches to the problem using machine learning methods.

In this project, we will analyze the Online Shopper's Intention data set from the online UCI Repository to address the problem of predicting the purchasing intention online consumers. The data is structured with 10 quantitative variables and 8 qualitative or categorical variables belonging to a total of 12,330 sessions, where each session corresponds to a different user in a one-year period to refrain from any bias to a particular campaign, special day, user profile, or period. We can view the variable names and descriptions in Table 1 and Table 2 below.

Table 1 Quantitative Variables in Online Shopper's Intention Data Set	
Variables	Description
Administrative	Number of pages visited by the visitor about account management
Administrative duration	Total amount of time (in seconds) spent by the visitor on account management related pages
Informational	Number of pages visited by the visitor about Web site, communication and address information of the shopping site
Informational duration	Total amount of time (in seconds) spent by the visitor on informational pages
Product related	Number of pages visited by visitor about product related pages
Product related duration	Total amount of time (in seconds) spent by the visitor on product related pages
Bounce rate	Average bounce rate value of the pages visited by the visitor
Exit rate	Average exit rate value of the pages visited by the visitor
Page value	Average page value of the pages visited by the visitor
Special day	Closeness of the site visiting time to a special day

Table 2 Categorical Variable in Online Shopper's Intention Data Set		
Variable	Description	Number of Levels
OperatingSystems	Operating system of the visitor	8
Browser	Browser of the visitor	13
Region	Geographic region from which the session has been started by the visitor	9
TrafficType	Traffic source by which the visitor has arrived at the Web site (e.g., banner, SMS, direct)	20
VisitorType	Visitor type as "New Visitor," "Returning Visitor," and "Other"	3
Weekend	Boolean value indicating whether the date of the visit is weekend	2
Month	Month value of the visit date	12
Revenue	Class label indicating whether the visit has been finalized with a transaction	2

From the quantitative variables, we have **Administrative**, **Administrative Duration**, **Informational**, **Informational Duration**, **Product Related** and **Product Related Duration** depicting the number of different types of pages visited by the visitor in that session and the total duration of time spent in each of these types of pages. The values of each of these variables come from the URL information of the pages by the user and are updated in real time when the user takes some course of action (e.g. clicking from page to page). Then the **Bounce rate**, **Exit Rate**, and **Page Value** variables depict the metrics measured by "Google Analytics" for each page in the website. **Bounce rate** indicates the percentage of visitors who enter the site from that page and then leave without triggering any requests to the server during that session. Thus, this is out of the number of single-page visits by the visitors of the website. The **Exit rate** denotes the percentage of all visitors who were last in the session indicating the number of exits from the websites out of all page views to the page. The **Page Value** variable tells you which particular page offer the most value before completing an online transaction. Finally, the **Special Day** variable depicts the closeness of the website visiting time to some special day such as Valentine's Day which takes in account the duration between the date which the product was ordered and the date when the product was delivered.

Among the qualitative variables, we have **Operating System**, **Browser**, **Region**, **Traffic Type**, **Visitor Type** with values such as returning or new visitor, **Weekend** which is a Boolean value indicating whether the date of the visit is weekend, and **Month**. Note that most of the categorical variables have levels labeled by a number. For example, the first level of the **Operating System** variable is denoted as '1' meaning that the operating system of the visitor during that session is labelled as Operating System 1. Now out of all the variables, the most valuable qualitative variable we will focus on to help predict purchasing intention is the variable **Revenue** which indicates whether the consumer has purchased a product on the website. We will see however that there is an imbalanced proportion of categories (Revenue to No Revenue) in the **Revenue** variable which may affect the performance of our prediction models.

Literature Review

Among all the literature studies reviewed for this project, they all proposed methods to predict purchasing intention of customers using popular classification algorithms such as decision trees and random forests as well as deep learning methods such as neural networks. In one of the studies, Sakar et. al proposed a real-time online shopper behavior system composing of two modules which both predicts the visitor's shopping intent and Web site abandonment likelihood [6]. The first module focuses on predicting the purchasing intention of the visitor using aggregated pageview data collected during the visit as well as some session and user information. Methods in this module include random forest, support vector machines, and multilayer perceptron neural network model. In the second module, the focus shifts to the sequential clickstream data by training a long short-term memory-based recurrent neural network. Together, these modules "determine the visitors which have purchasing intention but are likely to leave the site in the prediction horizon and take actions accordingly to improve the Web site abandonment and purchase conversion rates" [6].

In the second study, Muda et al. compared classification models predicting purchasing intentions implementing Logistic Regression, Decision Tree, and Random Forest models [5]. After implementing these classification algorithms on the data set, they compared the performance of each using accuracy rate. Also, in the process of tuning the models for more insightful result, they pre-processed the data through oversampling and feature selection. Finally, in the third study, Baati K. and Mohsil M. suggested a real-time online shopper behavior prediction system by relying on session and visitor information using naive Bayes classifier, C4.5 decision tree and random forest [1]. Similarly to [5], [1] employed oversampling to boost model performance and scalability of each classifier.

Methodologies

Exploratory Data Analysis

We will now begin to conduct some exploratory data analysis which will further help us determine what types of methodologies or algorithms to predict purchasing intention. To begin, we will examine the distribution of values of the numerical variables by plotting histograms below.

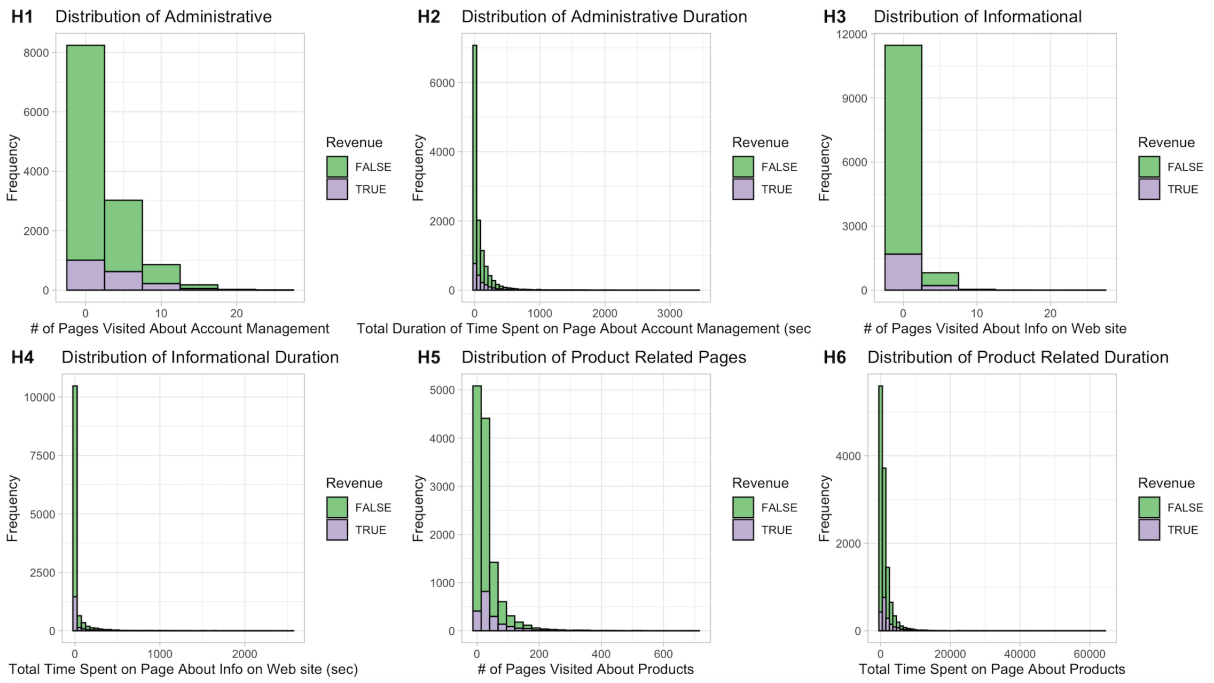

Figure 1: Histogram depicting distribution of different types of pages in data set: **Administrative**, **Administrative Duration**, **Informational**, **Informational Duration**, **Product Related** and **Product Related Duration**

From the histograms above, we see that all of the distributions are right skewed indicating that the mode of the data will cater to the left tail of the data. If we also look at a summary of those quantitative variables, we see that there are a many 0's in the distribution of the data which makes the data skewed. We can also observe that majority of visitors spend more time in the `Product Related` and `Administration` pages than in the `Informational` pages.

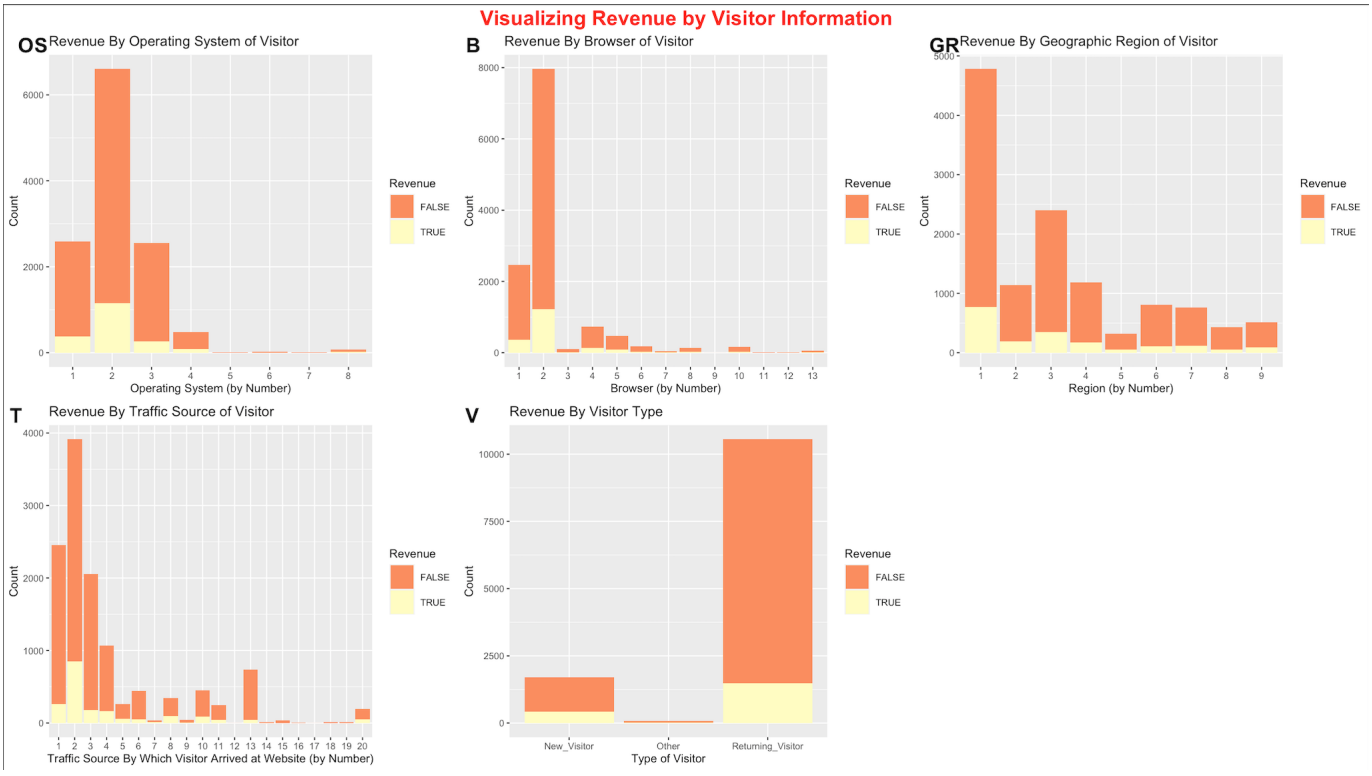

Figure 2: Bar plots depicting information for the visitor sessions: **Operating System**, **Browser**, **Region**, **Traffic Type**, **Visitor Type**

Next, we have observed the distribution of values of each of the categories of the qualitative variables depicting information on each of the visitor such as the operating system or browser they use or if they are a returning or new visitor. The bar plot above tells us that most visitors are returning visitors using operating system #2, browser #2, live in region #1, and come from Traffic Source type #2.

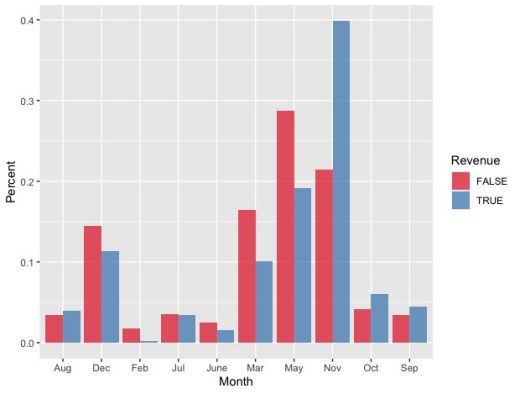

Figure 3: Distribution of Revenue by Month

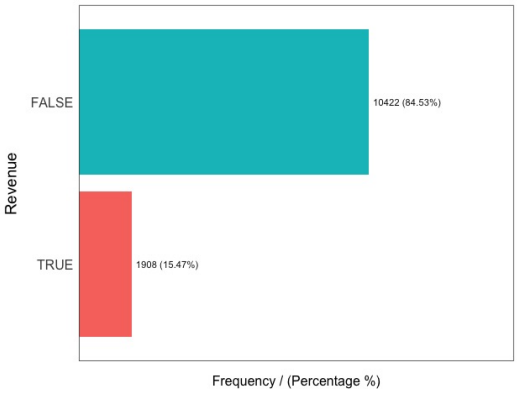

Figure 4: Distribution of Revenue: Imbalanced

In addition to the bar plots of the categorical variables concerning information about the visitor, we took a look at the distribution of revenue in each month. We see that more visitors bought a product on the Website in the month of November than any other month making up about 40% of consumers purchasing a product in November. There is also a significant amount of consumers visiting the website in May but it depicts that more of them do not end up purchasing an item making up about 30% of consumers in the data. Also, a significant observation from the distribution of the levels of Revenue is that we have an imbalanced structure of 'FALSE' Revenue meaning the client did not end up purchasing a product which can have an affect on the classification accuracy on the models.

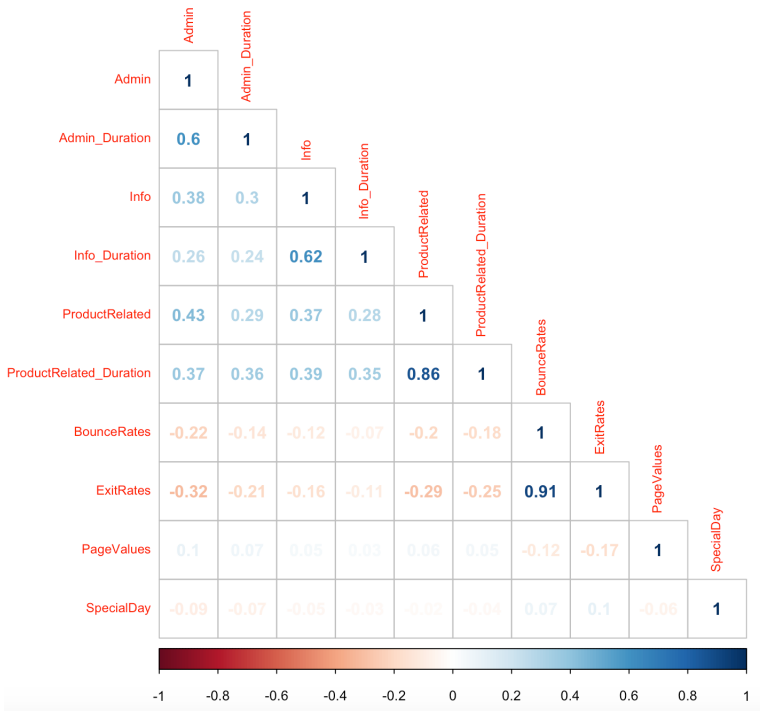

Figure 5: Correlation Plot showcasing correlation relationship between all numerical variables

Next, we would like to look at the correlation between all the numerical variables with a correlation plot as well as a multi-collinearity test using variance inflation factor. Since we may want to fit a logistic regression model on the data, we will need to watch out for two or more predictor variables in the multiple regression model which are high correlated. Looking at the correlation plot above, we see that the variables that are correlated with each other include `Administration` and `Administration Duration`, `Information` and `Information Duration`, `Product Related` and `Product Related Duration`, and `Bounce Rate` and `Exit Rate`. The most weighty of these correlation is the `Bounce Rate` and `Exit Rate` relationship with a value of 0.91. However after conducting a multi-collinearity test with VIF, we see in Table 3 below that none of the variables had a VIF higher than 5 which is the cut-off to determine multi-collinearity. Thus, we will not remove any of the variables from the data before fitting the models. Note that `OperatingSystems` variable was removed due to linear dependence.

Table 3 VIF Test	
Variable	VIF
Administrative	1.826564
Administrative duration	1.671059
Informational	1.876327
Informational duration	1.728954
Product related	4.988631
Product related duration	4.952398
Bounce rate	2.105241
Exit rate	2.315668
Page value	1.082516
Special day	1.264542
Browser	2.900356
Region	1.173374
TrafficType	2.654994
VisitorType	2.574195
Weekend	1.031928
Month	2.072540

Principal Component Analysis (PCA)

Another method of exploring the data for analysis is conducting Principal Component Analysis (PCA) which can serve as a tool for data visualization. We have performed a full PCA analysis using all the numerical predictors to predict the response variable `Revenue`. The objective we are aiming at is to see if we can load as much information as possible from a collection of predictors onto a new collection of predictors where we have $\vec{a}_1, ..., \vec{a}_p$ be our principal component vectors to rotate our existing p random variables $\vec{x}$ onto p new random variables $\vec{y}$:

$$y_1 = \vec{a}_1^T \vec{x}$$
$$.$$
$$.$$
$$.$$
$$y_p = \vec{a}_p^T \vec{x}$$

, in order to maximize the variance for every observation $\vec{a}_i^T \vec{x}$. We want to the minimum number of principal components to explain a good amount of variability in the data. Below is

a plot of the number of principal components plotted against the proportion of variance each component explains. We look for a point where the proportion of the variance explained by each principal component levels off. We can see that it will be sufficient to choose at least 5 principal components which will explain at least 80% of the variability of the data.

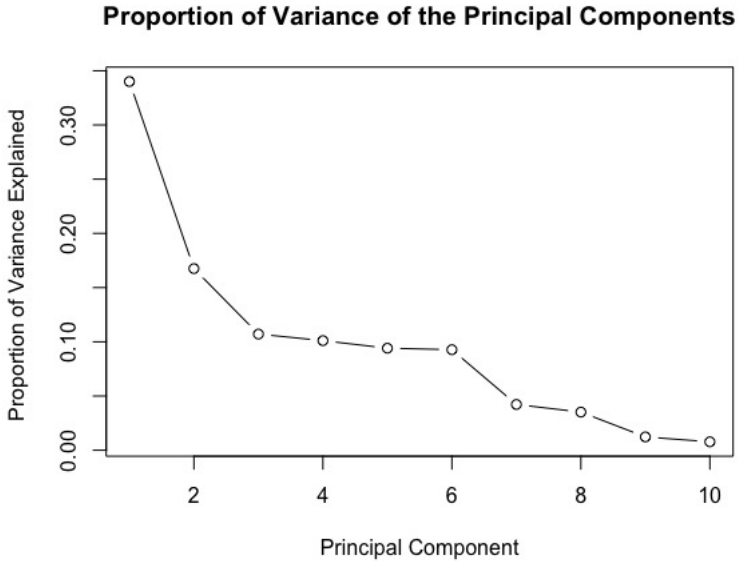

Figure 6: Scree Plot: Proportion of Variance Explained by Each Principal Component

Even though we need at least 5 principal components to explain a majority of the variability in the data, for visualization and interpretability, we have constructed a biplot (Figure 7) to observe which variables are higher in which principal components. We see that if we are higher in the variables associated with the different types of pages (`Administrated`, `Administrated Duration`,..., `Informational Duration`), we have a higher value in principal component 1 score and some value in principal component 2 score. Whereas if we are higher in variables `Bounce Rate`, `Exit Rate`, and `Special Day`, we have a higher principal component 2 score. Also, we know that `Page Values` is associated with a lower principal component 2 score.

The plot also tells us which variables are associated with either 'FALSE' or 'TRUE' `Revenue` values by the color of the data points. We see that `Page Values` are more associated with 'TRUE' `Revenue` values and variables `Bounce Rate`, `Exit Rate`, and `Special Day` are more associated with 'FALSE' `Revenue` values. Note however that some of the 'FALSE' or 'TRUE' `Revenue` values in each cluster are merged together so the variables `Administrated`, `Administrated Duration`, ...,`Informational Duration` are split between the 'FALSE' or 'TRUE' `Revenue` values. However, we can determine that the `Administrated`, `Administrated Duration`, `Product Related`, and `Product Related Duration` variables are a little more associated with 'TRUE' `Revenue` values whereas the `Informational` and `Informational Duration` variables are more associated 'FALSE' `Revenue` values.

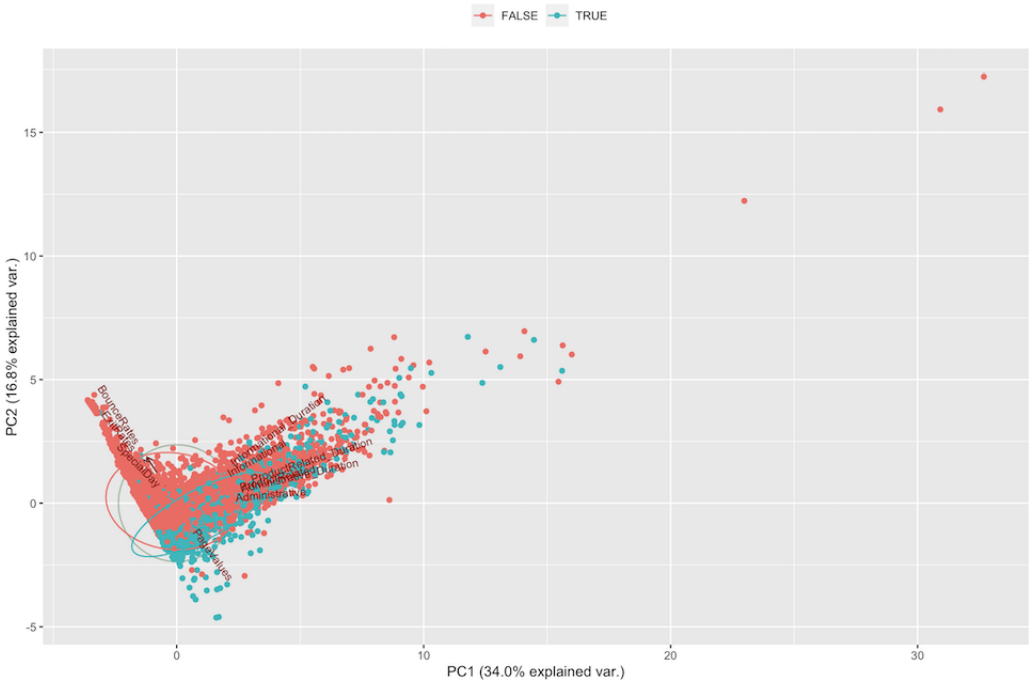

Figure 7: PCA Biplot of Numerical Variables

K-Medoids Clustering

The final method of visualizing the data was a clustering algorithm called 'K-Medoids' clustering. This form of clustering is similar to K-means in which it divides the data set into k number of clusters but in k-medoids, but it attempts to minimize the sum of dissimilarities between points labeled to be in a cluster and a point designated as the center of that cluster called a medoid. To minimize the sum of dissimilarities between points, we measure Gower's distance which takes the average of the partial dissimilarities across points. This is an appropriate measure of distance as compared to Euclidean and other forms of distance since Gower's distance can be used to compute distance between two entities with attributes that can have either categorical and numerical values. We use the following formula for Gower's Distance:

$$d(i,j) = \frac{1}{p}\sum_{f=1}^{p} d_{ij}(f),$$

where $d_{ij}(f) = \frac{|x_{if} - x_{jf}|}{R_f}$ and $R_f = \max(x_{.f}) - \min(x_{.f})$. For qualitative features, $d_{ij}(f)$ is either 0 or 1. In `R`, we compute the gower distances using the `daisy()` function, which computes all pairwise distances between the observations in the data set.

To implement k-medoids clustering, we also need determine the optimal number of clusters k. The approach to determine k is using the silhouette method. The silhouette coefficient is the metric used to measure how similar an object is to its own cluster as compared to other cluster. It measures from -1 to +1 where -1 represent that the cluster assignment is incorrect, 0 representing that the clusters are indifferent, and 1 representing that the clusters are well apart from each other indicating good clustering. Therefore, to find the best number of clusters, we want k to be associated with the highest sihouette width. Below in Figure 8, we see that the optimal number of clusters is $k = 2$.

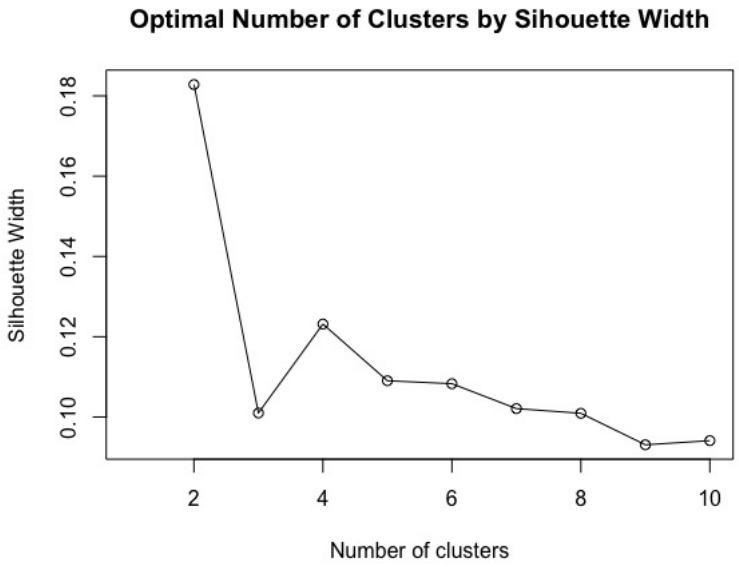

Figure 8: Plot of Number of Clusters vs. Silhouette Width: Optimal Number of Clusters has highest Silhouette Width

We will use the `pam()` function to create the clusters by inputing the calculated gower distances and the number of clusters $k = 2$. PAM stands for "Partitioning Around Medoids" and forms the clusters by assigning each observation to the closest medoid. It we will continue switching medoids with non-medoids until the sum of the dissimilarities of all the data points to ther closes medoid no longer decreases. After creating the clusters, we obtained medoids at data points '6101' in cluster 1 and '4346' in cluster 2.

We now plot a visualization of our clusters using the `Rtsne()` function in which we input our gower distances. We can see that the clusters have some overlap with each other which means we may need more observations to have more distinct clusters.

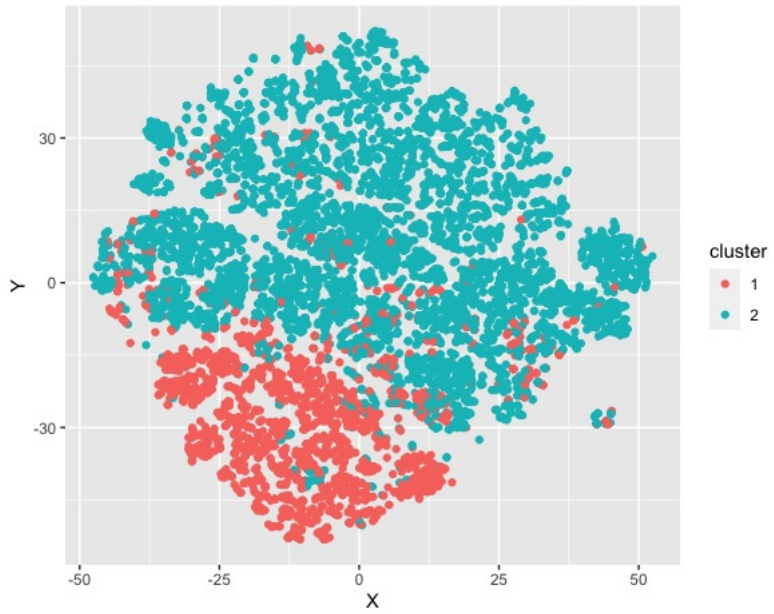

Figure 9: Visualization plot (tSNE) of Clusters using Gower distance for K-Medoids Clustering

Below we have a table of the differences between the clusters and depicts what is the major category (level - TRUE or FALSE) for `Revenue`:

Table 4 - Cluster Differences		
	Cluster 1 - 3234	Cluster 2 - 9096
Month	Nov - 38.28%	May - 30.76%
Operating Systems	1 - 75.45%	2 - 71.99%
Browser	1 - 74.34%	2 - 82.27%
Revenue	FALSE - 82.28%	FALSE - 85.32%

Table 4 shows there is a distinction between the clusters in terms of Month, Operating System, and Browser. But for Revenue, both clusters have majority of 'FALSE' Revenue which is due to the present imbalanced structure of the data.

Preprocessing the Data for Methodologies

After reviewing the literary studies on the data set and exploring the data set, we have chosen to conduct classification algorithms random forest, logistic regression, support vector machines (SVM), decision tree, and neural network and compare each performance using accuracy, F-statistic, and the area under the receiver operating curve (AUC, ROC). Accuracy determines the overall predicted accuracy of the model. F-score is a measure of a model's accuracy on a data set and is a preferred metric when there are imbalanced case. Then the AUC determines the accuracy of a classification model at a user defined threshold value which means that it is calculated on the predicted scores whereas accuracy is predicted based on the predicted classes. The best model is preferred to be the one with highest accuracy, F-statistic, and/or AUC.

We will also test and train data to prepare for modelling by training the data on a random sample of 75% of the data and the test data as the rest of the 25% of the data. The training data set has 97247 observations while the test data set has 3083 observations. Additionally, we also conduct a repeated 10-fold cross-validation three times for selection of best parameters and obtain classification accuracy rate with those tuning parameters for each of the validated models.

Based on the exploratory data analysis, we noticed that we many levels for variables `Traffic Type` and `Browser` after looking at the bar graphs. Thus, we merge some levels to decrease the number of levels for each variable and a more even distribution of values for each level. For `Browser`, we see that there aren't many values for levels 3-13. Thus, we will merge levels 3-13 with level 1 to have a combined level 1 so we will have 2 levels for `Browser` where Level 1 has 4369 observations and Level 2 has 7961 observations. Finally for `TrafficType`, we see that there are a small number of values for levels 6-13. So, we will combine levels 6-13 with level 5 to have a combined level 5. This indicates we have a total of 5 levels for `TrafficType` where level 1 has 2451 observations, level 2 has 3913 observations, level 3 has 2052 observations, level 4 has 1069 observations, and level 5 has 2845 observations.

Support Vector Machines

The first method we will implement on the data is Support Vector Machines (SVM), which is a method for classification purposes developed in the 1990s and is noted as one of the best "out of the box" classifiers [4]. SVM is a discriminant-based algorithm which aims to find the optimal separation boundary called hyperplane to discriminate the classes from each other [4]. In this model, we will implement the radial basis kernel after looking at a pairwise plot of the numerical variables and the response `Revenue`. Below is the formula for the radial basis kernel function:

$$K\left(x_i, x_{i'}\right) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2\right), \tag{1}$$

where γ is a positive constant. If a given test observation $x^* = (x_1^* \dots x_p^*)^T$ is far from a training observation x_i in terms of Euclidean distance, then $\sum_{j=1}^{p}(x_j^* - x_{ij})^2$ will be large and so (1) will be very small. This means that the training observations that are from x^* will essentially not play any role in the predicted class label for x^*.

Now, before fitting an SVM using a radial kernel on the data, we need to determine the best choice of γ and `cost`, which are tuning hyperparameters for the SVM function. We will use a repeated ten-fold cross-validation (3 times repeated) to obtain the best values for σ and `cost` by using the `train()` function where we input a grid of values for each parameter to search through. For σ, we have chosen the sequence of `c(0.25,0.5,1,2)`, and for `cost`, we have chosen the sequence of `c(0.1, 1, 10, 100)` to get an appropriate range of small to large values. This is key since if the cost value is small, then we know that the margin(s) will tend to not overfit the data, and more violations of the margin(s) are possible; however, if the cost is huge, fewer violations will be made, but overfitting of the data is possible. After tuning for parameters, we have obtained final values of `cost = 1` and `gamma = 0.25`.

Since we have our appropriate values `cost = 1` and `gamma = 0.25`, we will input these values into the `svm()` with `Revenue` as the response. After fitting the model, we will evaluate the predictive performance on the training set and the test set. We constructed the following confusion matrix to show how well the model classified the data:

Table 5 - Confusion Matrix for SVM			
Prediction	Reference		
		FALSE	TRUE
	FALSE	2519	245
	FALSE	79	240

Table 6 Predictive Performance for SVM	
Accuracy	89.49%
Precision	91.14%
Recall	96.96%
F-Score	93.95%

From the confusion matrix, we can compute the accuracy, precision, recall, and F-score to evaluate the predictive performance. In terms of accuracy, it is calculated as

$$\text{Accuracy} = \frac{(\text{True Positives} + \text{True Negatives})}{(\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives})}.$$

In terms of precision, it determines how many values, out of all the predicted positive values ('FALSE'), are actually positive. It is calculated as: $(TP/TP + FP)$. In terms of recall, it determines how many positive values, out of all the positive values, have been correctly predicted so the formula is the same as the true positive rate: $(TP/TP + FN)$. Finally, in terms of F-score, it is essentially the harmonic mean of precision and recall, which is formulated as $2((\text{precision} * \text{recall})/(\text{precision} + \text{recall}))$. The F-score is useful to calculate for imbalanced data set since class imbalance affects the accuracy of the models. Table 6 displays all the calculated values for accuracy, precision, recall, and F-Score. We see that we have a fairly high accuracy rate which shows that 89.49% of the `Revenue` predictions were correctly predicted. In terms of precision, we see that 91.14% of all the predicted `Revenue` values of 'FALSE' were actually 'FALSE'. The recall rate of 96.96% indicates that 96.96% out of all 'FALSE' Revenue values have been correctly predicted. With F-Score, we see that we have a high value of 93.95% which indicates that the model did pretty well in predictive performance.

To graphically depict the relationship between the true positive rate and the false positive rate, we have the Receiver Operating Curve plotted below:

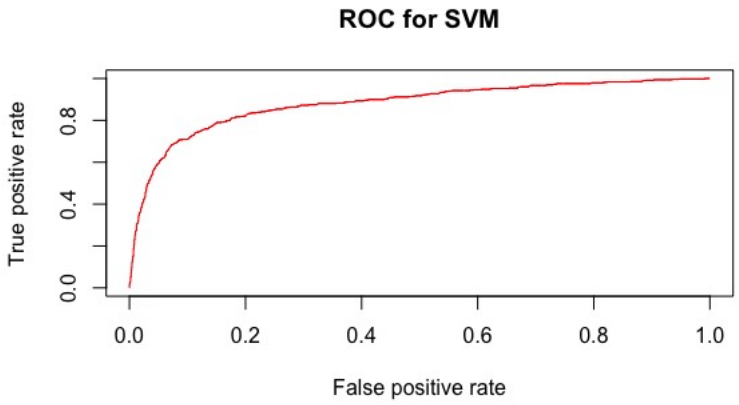

Figure 10: Receiver Operating Curve (ROC) for Support Vector Machines (SVM)

ROC represents the accuracy of a classification model at a user defined threshold value. It determines the model's accuracy using Area Under Curve (AUC). The area under the curve was calculated as 0.8792.

Logistic Regression

Next classification algorithm we have fit on the model was a full logistic regression model with `Revenue` as the response. To fit the logistic regression model, we resample to find the best model using a repeated ten-fold cross-validation (3 times repeated) with the `train()` function. As a result, we have the following significant predictors according to the summary results:

Table 7 Logistic Regression Significant Predictors			
Variables	Coefficient Estimate	Standard Error	P-Value
Exit Rates	-13.08	2.759	2.15e-06
Page Values	0.08505	0.002905	<2e-16
MonthDec	-0.6849	0.2123	0.00126
MonthMay	-0.5957	0.2003	0.00294
VisitorTypeReturning_Visitor	-0.269	0.1021	0.00843

According to the table above, we can see that variables `Exit Rates` , `Page Values`, `Month` with levels 'Dec' and 'May', and `VisitorType` with level `Returning_Visitor` seem to be statistically significant with low p-values. In particular, `Page Values` seems to be the most statistically significant of the predictors having an extremely low p-value $< 2e-16$. Thus, for `Page Values`, we say that a one unit increase in `Page Values` results in a 0.08505 increase in the log odds. In other words, on average, a one unit increase in `Page Values` results in a 108.88% increase in odds of Revenue having 'TRUE' value. This means that increasing page values will increase the chance of transaction. For the case of `Exit Rates`, we say that on average, a one unit increase in `Exit Rates` results in a $e^{(-13.08)}$ decrease in the odds of Revenue having 'TRUE' value. For `MonthDec`, we say that on average, the effect of the visitor session on the site being in the Month of December has 50.41% lower odds than any other month for the chance of a Revenue transaction. For the rest of the categorical predictors, we can interpret in a similar way. Overall, if we are interested in increasing the chance for a Revenue transaction based on the logistic regression model, we want to increase the `Page Values`.

We can also conduct some 95% confidence intervals to investigate the chance of Revenue transaction. For the effect of `Page Values`, we calculate that we are 95% confident that on average, holding all else constant a one unit increase in `Page Values` results in a 108.26

$(e^{(0.08505)-1.96\times0.002905})$ to 109.5% increase in odds of a Revenue transaction. We can conduct the 95% for `Exit Rates` in a similar way. Then for our categorical predictor `Month` (December) over the rest of the 12 months, we are 95% confident that on average and holding all else constant the visitor session taking place in the Month of December as opposed to other months is a 33.25% to 76.43% decrease in odds of a Revenue transaction. Again, we can compute the rest of the 95% confidence intervals for the categorical predictors in a similar manner.

Table 8 - Confusion Matrix for Log Reg			
Prediction	Reference		
		FALSE	TRUE
	FALSE	2533	289
	FALSE	65	196

Table 9 Predictive Performance for Log Reg	
Accuracy	88.52%
Precision	89.76%
Recall	97.50%
F-Score	93.47%

To further examine the fit of the logistic regression model, we calculated the confusion table above showing the true positive and negative values along the diagonal of the table which are 2535 for positive ('FALSE') and 196 for negative ('TRUE') for `Revenue`. We can calculate the overall accuracy, sensitivity, recall, and F-Score which are calculated as 88.52%, 89.76%, 97.50%, and 93.47%. This means that overall the logistic regression model is accurate 88.52% of the time. But terms of precision, we see that 89.76% of all the predicted `Revenue` values of 'FALSE' were actually 'FALSE'. The recall rate of 97.50% indicates that the model correctly predicted 97.50% 'FALSE' Revenue out of all 'FALSE' Revenue values. With F-Score, we see that we have a high value of 93.47% which indicates that the model did pretty well in predictive performance. We have also plotted the ROC curve illustrating the relationship between False Positive Rate and True Positive Rate (Sensitivity/Recall). The area under the ROC curve for Logistic Regression is 0.8813, which is slightly higher than the AUC for SVM.

Figure 11: Receiver Operating Curve (ROC) for Logistic Regression

Random Forests

This next method we will fit on the dataset is Random Forest, which is a classification and regression technique and machine learning model that gathers weak learners (group of individual trees) to produce a stronger learner. This technique is also a version of 'bagging', which builds a number of decision trees on bootstrapped training samples. But in random forests, when building the decision trees, a random sample of m predictors is chosen each time a split in a tree is considered. This random sample are chosen as split candidates from the entire set of

p predictors. Typically, the value of m used for the number of predictors considered at each split is approximately equal to the square root of the total number of p predictors ($m \approx \sqrt{p}$). Thus, we will use this value as a starting value in our resampling method of repeated ten-fold cross-validated models. So this means $m \approx \sqrt{18} = 4.24 \approx 4$ is our starting value.

After resampling for our final Random Forests model, we computed the following confusion matrix and statistics for predictive performance below:

Table 10 - Confusion Matrix for RF			
Prediction	Reference		
		FALSE	TRUE
	FALSE	2498	195
	FALSE	100	290

Table 11 Predictive Performance for RF	
Accuracy	90.43%
Precision	92.76%
Recall	96.15%
F-Score	94.42%

The confusion table above showcases that the true positive (Revenue - 'FALSE') value is 2498 and true negative value 290 (Revenue - 'TRUE'). This means that it correctly predicted 2498 FALSE `Revenue` values and 290 TRUE verb—Revenue— values. Then we have computed the overall accuracy, sensitivity, recall, and F-Score as 90.43%, 92.76%, 96.15%, and 94.42%. Note that the values of accuracy and F-value are the highest so far indicated that the Random Forest performed the pretty well. All in all, the random forest model accurately predicted 90.43% of all its predicted values. If we would like to see how much the model predicted `Revenue` values of 'FALSE' that were actually 'FALSE', we see that our precision rate or True Positive rate is 92.76%. Then our recall rate of 96.15% indicates that the model correctly predicted 96.15% 'FALSE' Revenue out of all 'FALSE' Revenue values. Finally, taking the middle-ground between both precision and rate With F-Score, we see that we have a high value of 94.42%. We have also plotted the ROC curve illustrating the relationship between False Positive Rate and True Positive Rate (Sensitivity /or Recall) which has an area of 0.9246, which is higher than both SVM and Logistic Regression ROC's. Note all values of predictive performance for Random Forests have been the highest so far except for Recall. The highest Recall rate belonged to Logistic Regression which signifies that Logistic Regression did better in predicting its 'FALSE' Revenue out of all 'FALSE' Revenue values.

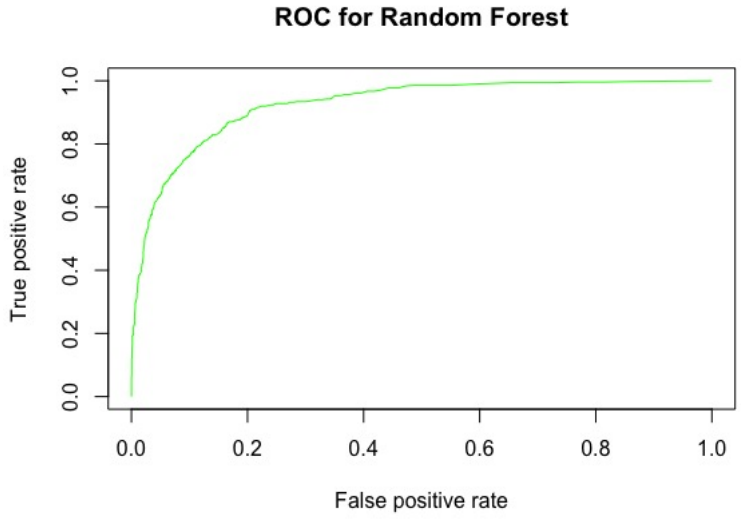

Figure 12: Receiver Operating Curve (ROC) for Random Forests

Additionally, we can also plot a Variable Importance plot to see which variables have more value or importance in the Random Forests model. This plot is helpful since we do not have

an attractive tree diagram to interpret for the case of a Random Forests model which bags a larger number of trees. Thus, we have the variable importance plot below which shows the mean decrease in Gini index. The Gini index is defined by the following formula:

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right).$$

This measure represents a measure of the total variance across the K classes. According to the variance importance plot below, we see that the `Page Values` feature has the greatest `MeanDecreaseGini` followed by `ProductRelated_Duration`, `ExitRates`, and so on. We will see that in the decision tree model we fit in the next section, the `Page Values` feature has a significant impact in decision-making.

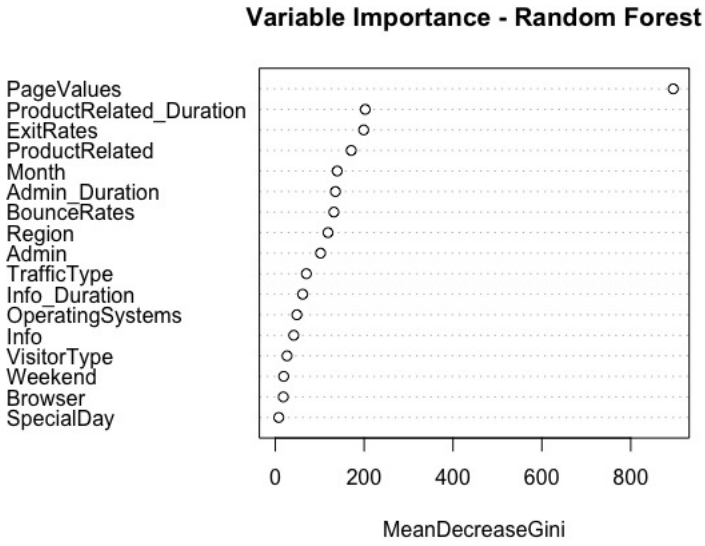

Figure 13: Variable Importance Plot for Random Forests

Decision Tree

This method called Decision Trees or Classification Trees since we are predicting a categorical response is similar to Random Forests except that Random Forests is essentially a collection of decision trees. Thus, in this model, we plot a single general decision tree that we can easily interpret its result. For classification tree, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs [4]. When we make an interpretation of the plot, we are interested in both the class prediction correlating to a specific terminal node region and the class proportions among the training observations that fall into that region.

To grow the classification tree, there are a few measures to consider which include *classification error rate*, *Gini index*, or *cross-entropy*. Since we aim to classify an observation in a given region to the most commonly occurring error rate class of training observations in that region, the classification error rate is:

$$E = 1 - \max_{k} \left(\hat{p}_{mk}\right),$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. For Gini index, it is a measure of total variance across K classes or as a measure of node *purity* (a small value denoting that a node mostly contains observations from a single

class). Gini index is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk} \left(1 - \hat{p}_{mk}\right).$$

Now an alternative to Gini index is *cross-entropy* which is defined as

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

We can see that like Gini index, the cross-entropy will be small if the mth node is pure. Out of all these measures, Gini index and cross-entropy is preferred to evaluate the quality of a specific split due to their sensitivity to node purity whereas the classification error rate is not sensitive to tree-growing.

In this paper, we use the function `rpart()` to construct the decision tree model, which by default uses the Gini impurity to choose splits. Like the other methods, we run the `train()` function to perform repeated ten-fold cross-validation three times to obtain the decision tree model. After running the model, we obtain the following decision tree plot below (Figure 14) using the `rpart.plot()` function. Since we are interested in a strategy to increase `Revenue`, we will focus on decisions that lead to TRUE values. From the decision tree plot, it shows that `Page Values` greater than 0.93 lead to a TRUE `Revenue` 57% of the time. Then after this, we consider the `Bounce Rate` where an effective Bounce Rate above 0 boosts our TRUE `Revenue` to 73% and as an added point pages of Administrative type '5'or below (0,1,2,3,4,5) result in a TRUE 83% of the time.

Additionally, we also that the month of November is significant for shopper conversion. If `MonthNov=0` meaning when it is not November, it leads to FALSE value of 35% which then we would need to focus on `ProductRelated` pages, the time an online consumer will spend on the `ProductRelated` pages, and whether or not they are a returning visitor or not. If it is not the month of November, then the focus should be on decreasing the number of `ProductRelated` pages the consumer visits from 33 below and decreasing the amount of time the consumer spends on those `ProductRelated` pages (below 513 seconds or 8.55 minutes). If the online consumer spends more than 8.55 minutes, then based on the decision tree plot, it is preferred that the consumer not be a returning visitor for the session to lead to a transaction (TRUE 85%).

On the other hand, if it is the month of November, then we would consider a successful `Exit Rate` to lead to 'TRUE' value of 72% if the `Exit Rate` is less than 0.021. But if it is higher than or equal to 0.021, then like before, we need to consider the amount of time spent on `ProductRelated` pages. Here, the duration of time should be either over 4994 seconds (83.23 minutes) to lead to TRUE value of 72% or less than 1020 seconds (17 minutes) to lead to a TRUE value of 62%.

Overall, based on the plot, we can come up with a strategy to increase Revenue. The focus should be on the following metrics: increasing the page value, decreasing the bounce rate, and paying attention to the pages of administrative type '5' and below which are pages about account management. Then in November, to drive revenue, the focus should be on decreasing exit rates and focusing on Product related pages and the amount of time the consumer spent on them.

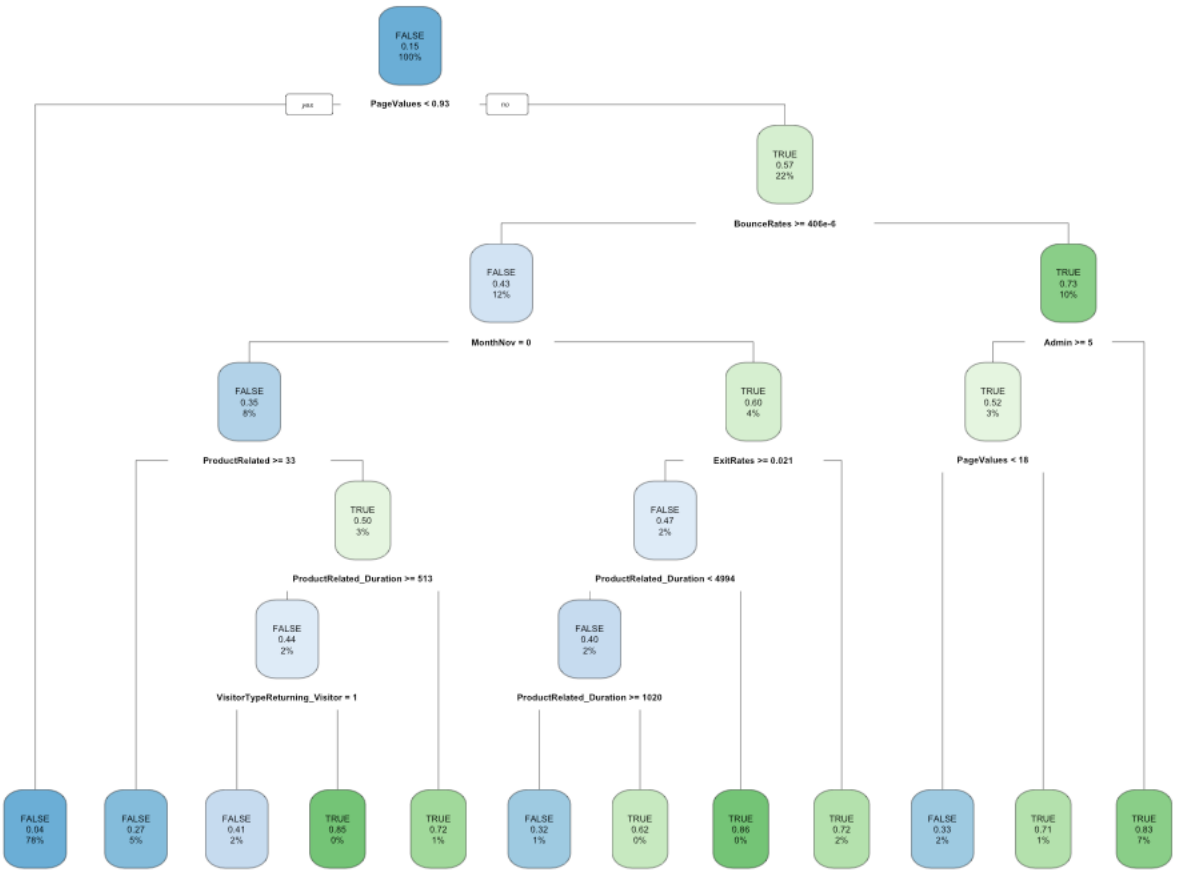

Figure 14: Decision Tree Plot (rpart) from Best Cross-Validated Model

Now, we will examine the predictive performance by observing the classification accuracy, precision, recall, and the F-Score AUC. We start by making a confusion matrix of our predictions for Revenue from test data using the cross-validated model and compare them to the actual values as we see in Table 11 below: We have computed the overall accuracy, sensitivity, recall,

Table 12 - Confusion Matrix for Decision Trees			
Prediction	Reference		
		FALSE	TRUE
	FALSE	2494	214
	FALSE	104	271

Table 13 Predictive Performance for Dec Tree	
Accuracy	89.69%
Precision	92.10%
Recall	96.00%
F-Score	94.01%

and F-Score as 89.69%, 92.10%, 96.00%, and 94.41%. These values are pretty comparable to the Random Forest model but lower in value. This is understandable since Random Forests is an extension of Decision Tree as it builds a number of decision trees and splits the trees based on a random sample of m predictors. This builds diversity among the trees. Also decision trees are often pruned to avoid over-fitting whereas random forest models are unpruned and fully grown which results in a feature space divided into additional and smaller regions. Note that for our decision tree to be adequately pruned and have the best accuracy rate, a complexity parameter (`cp`) was chosen as 0.002810963 from our `train()` function.

Altogether, the decision tree model accurately predicted 89.69% of all its predicted values. Then the precision rate or True Positive rate turned out to be 92.10% indicating that how much of the model predicted `Revenue` values of 'FALSE' that were actually 'FALSE'. To display how the model correctly predicted 96.15% 'FALSE' Revenue out of all 'FALSE' Revenue values, our recall rate of 96%. Finally, as the harmonic mean between both precision and rate, we have

a high F-score of 94.01%. The plot of ROC curve illustrating the relationship between False Positive Rate and True Positive Rate (Sensitivity /or Recall) has an area of 0.8523, which makes it our lowest AUC among the models.

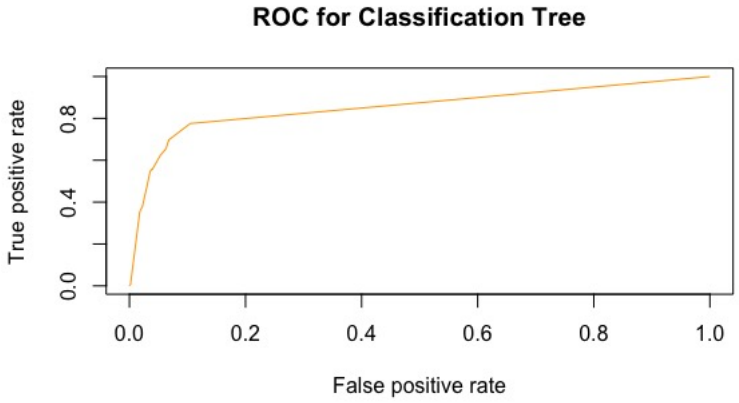

Figure 15: Receiver Operating Curve (ROC) for Random Forests

Neural Network

The final model we will fit on the data is Neural Networks (NNs). Neural Networks are highly parameterized models inspired by the structure of the human brain and was widely promoted as a *universal approximator* (a machine that with enough data could learn any smooth predictive relationship [2]. In this project, we will implement a feed-forward neural network with a single hidden layer using the method `nnet` in our `train()`. NNs is a supervised learning algorithm in which the memory units called *neurons* automatically learn new features from the data, and each neuron a_l is connected to the input layer via a vector of parameters or *weights* $\left\{w_{\ell j}^{(1)}\right\}_1^p$. The weight $\left\{w_{\ell j}^{(1)}\right\}_1^p$ is located in the first layer as indicated by the (1), and the ℓ_j indicates the jth variable and ℓth unit. The intercept terms are called *bias* and the function g is a nonlinearity function such as the sigmoid function $1/(1+e^{-t})$. There are also weights in the final output layer with an output function h. In this case, since we have a binary response `Revenue`, the output function will be the sigmoid function $1/(1+e^{-t})$.

We start by resampling using a repeated ten-fold cross-validation three times to be consistent with other models and inputting a `tuneGrid` for the `size` and `decay` parameters where size represents the number of units in the hidden layer and decay is the regularization parameter to avoid over-fitting. We plug in a sequence from 1 to 10 for `size` and a sequence from 0.1 to 0.5 spaced by 0.1 units for `decay`. Then after resampling for the best cross-validated model, we obtained a model with 2 units in the hidden layer and the decay parameter of 0.4. Below, we have a visual representation of the neural network model:

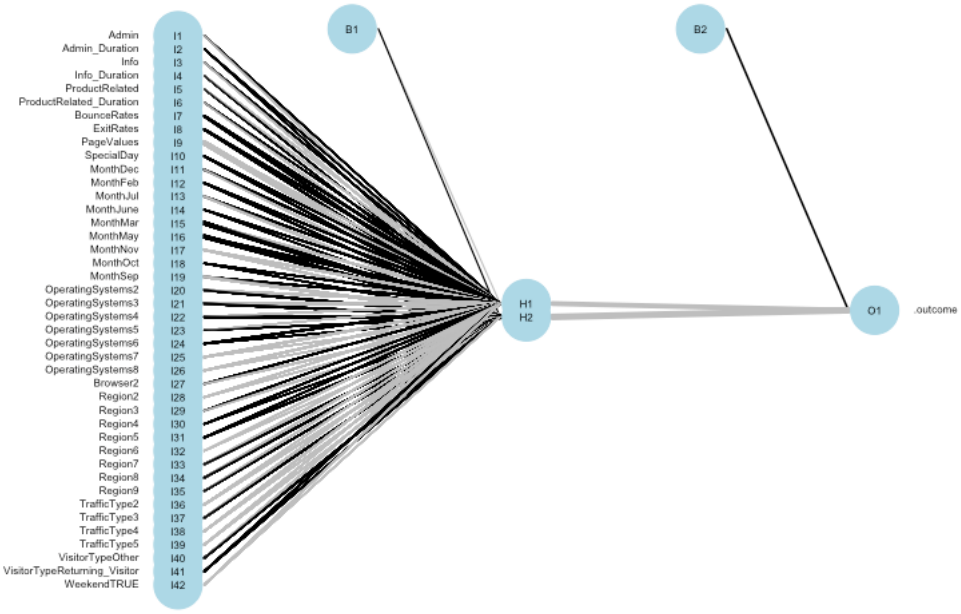

Figure 16: Feed Forward Neural Network Model (Single Layer) with 42 input features (I1 - I42), 2 units in the hidden layer (H1, H2), weights (B1, B2), and the output function denoted as O1

After fitting the model, we can inspect the predictive performance of the model using accuracy rate, precision, recall, F-Score, and the AUC under ROC. We created a confusion matrix of the predicted values from the model on the test set against the actual values from the test set and obtained the following values below in Table 14:

Table 14 - Confusion Matrix for Neural Network			
Prediction	Reference		
		FALSE	TRUE
	FALSE	2402	150
	FALSE	196	335

Table 15 Predictive Performance for NNs	
Accuracy	88.78%
Precision	94.12%
Recall	92.46%
F-Score	93.28%

We computed the confusion table above showing the true positive and negative values along the diagonal of the table which are 2402 for positive ('FALSE') and 335 for negative ('TRUE') for `Revenue`. We can calculate the overall accuracy, sensitivity, recall, and F-Score which are calculated as 88.52%, 94.12%, 92.46%, and 93.28%. This means that overall the Neural Networks model is accurate 88.52% of the time. But terms of precision, we see that 94.12% of all the predicted `Revenue` values of 'FALSE' were actually 'FALSE'. The recall rate of 92.46% indicates that the model correctly predicted 92.46% 'FALSE' Revenue out of all 'FALSE' Revenue values. With F-Score, we see that we have a high value of 93.28% which indicates that the model did pretty well in predictive performance. We have also plotted the ROC curve illustrating the relationship between False Positive Rate and True Positive Rate (Sensitivity/Recall). The area under the ROC curve for Neural Networks is 0.909891, which makes it the second highest AUC among all the models.

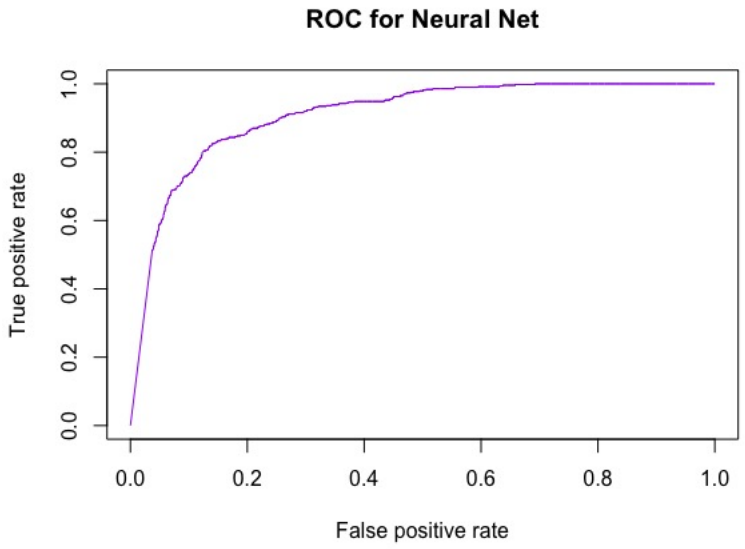

Figure 17: Receiver Operating Curve (ROC) for Neural Networks

Results

Shop Model Results

Model	F_Score	Accuracy	Lower_Conf_Interval	Higher_Conf_Interval	Area_Under_Curve
Support Vector Machines	0.9395748	0.8949076	0.8835429	0.9055164	0.8792410
Decision Tree	0.9400678	0.8968537	0.8855761	0.9073717	0.8522706
Logistic Regression	0.9346863	0.8851768	0.8733938	0.8962231	0.8881320
Random Forests	0.9442449	0.9043140	0.8933812	0.9144724	0.9246435
Neural Net	0.9328155	0.8877717	0.8760976	0.8987040	0.9095910

Figure 18: Model Results for all Classification Algorithms

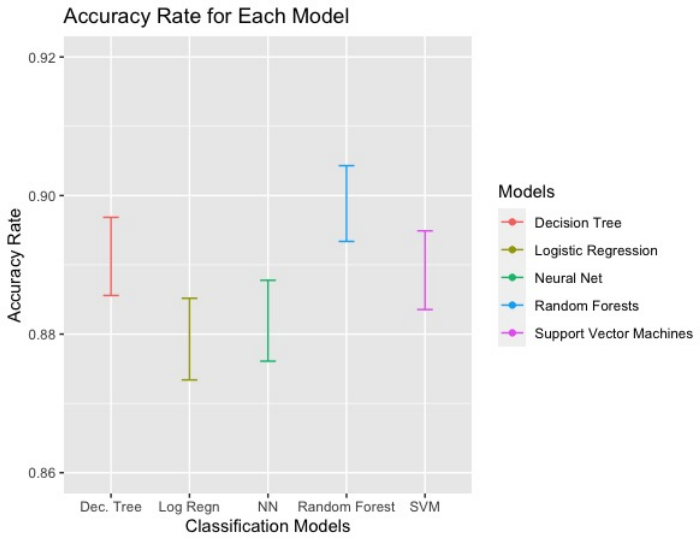

Figure 19: Visualization of the 95% Confidence Intervals for Accuracy

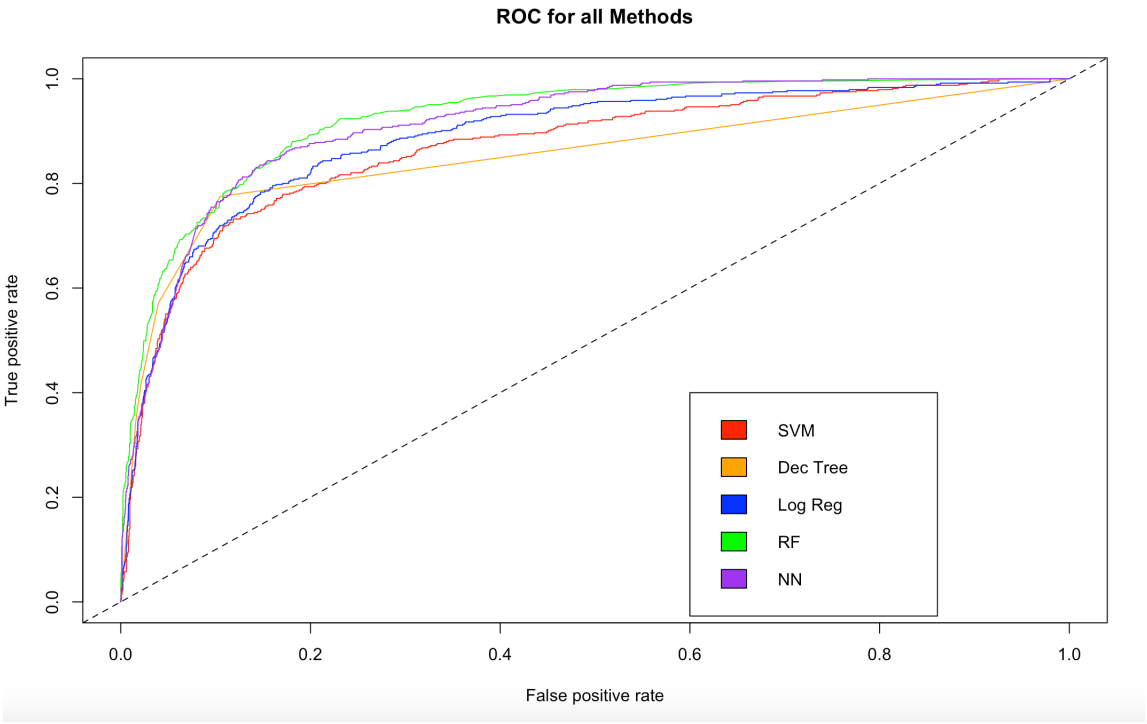

Figure 20: ROC Curves for All Classification Algorithms

Now that we have fit all the models with resampling method three times repeated ten-fold cross-validation for the best model, we have constructed a table of all the final results according to F-Score, Accuracy, a 95% Confidence Interval of Accuracy rate and the Area under the ROC for each classification model. Among all the models, we see that Random Forests performed the best in all categories for predictive performance. We also see in Figure 19 that the ROC Curve for Random Forest has the most area under the curve as the curve is closer to the left top corner of the plot with Neural Networks following closely afterwards. However, based on the values and the visualization plot of the confidence intervals for each model, we see that in terms of accuracy, the confidence intervals each overlap in other confidence beds indicating that we do not have a clear winner among all the classification models.

Conclusions and Future Work

As the Internet population increased over the years, the phenomenon of online shopping becomes has grown. However, in the event of online shopping, it is difficult to predict the purchasing intention of the shopper without a personal interaction between the consumer and the salesperson. Thus, machine learning and data analytics have become popular devices to predict whether or not a consumer will buy a product on the website or not. In this paper, we take features from the data provided by the visits of uses and feed it into machine learning classification algorithms to construct a model to predict our response `Revenue`. The classification methods we have implemented are support vector machines (SVM), logistic regression, random forests, decision trees, and neural networks. To search for the best model among each method, we executed a resampling method via a three times repeated 10 fold cross-validation for each classification method and get the average accuracy among the validated models. From our findings, the best model overall is Random Forest using 10-fold cross validation with 94.42% accuracy and 0.9328 for F-Score. However, since the accuracy rate is within the other confidence interval beds of other models, particularly Decision Tree and Support Vector Machines, we cannot accurately declare which of the models performed the best. From all the methods, we also have a possible

strategy to increase `Revenue` by focusing on increasing the page value, lowering the bounce rate, administrated related pages type '5' and below, product related pages and the amount of time spent on the product related pages. Also, more marketing in November can likely drive `Revenue` based on exploratory data analysis and model findings.

In terms of future work on the data set, since we have an imbalanced classification for Revenue and the accuracy rates were within each of the model's confidence beds, there can be some strategies to deal with the imbalanced structure and have a better distinction of accuracy between each model before fitting the models. From papers by Sakar, Baati, and Muda et. al. ([1], [5], and [6]), a possible method to fix the imbalance problem would be to balance the data using the "Synthetic Minority Overampling Technique" (SMOTE). The central point of SMOTE is to introduce synthetic examples, and this new data is generated by interpolation between several minority class instances that are within a defined neighborhood. Thus, we can say that SMOTE is concentrated on the "feature space" rather than on the "data space", in other words, the algorithm is based on the values of the features and their relationship, instead of considering the data points as a whole [3]. Thus, after implementing the SMOTE method on the unbalanced the data the classification ratio from TRUE to FALSE becomes more even as seen in [5]. Additionally, more collection of data can also help balance the classification ratio of TRUE and FALSE in the `Revenue` response. Also more future work to have better distinct accuracy is to have a more robust resampling method for our models such as to repeat the ten-fold cross-validation more than 3 times and to implement more methods such as Boosting, Naive Bayes Classifier, or Neural Network (Multi-layer Perceptron) instead of single-layer to compete against other methods we have performed on the data set.

References

1. Baati K., Mohsil M. (2020) Real-Time Prediction of Online Shoppers' Purchasing Intention Using Random Forest. In: Maglogiannis I., Iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 583. Springer, Cham. https://doi.org/10.1007/978-3-030-49161-1_4

2. Efron, Bradley, and Trevor Hastie. Computer Age Statistical Inference: Algorithms, Evidence, and Data Science. Cambridge University Press, 2016.

3. Fernandez, Alberto, et al. "SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-Year Anniversary." Journal of Artificial Intelligence Research, vol. 61, 20 Apr. 2018, pp. 863–905., doi:10.1613/jair.1.11192.

4. Gareth, James, et al. An Introduction to Statistical Learning: with Applications in R. Science+Business Media New York, 2013.

5. Muda, Muhammad & Iswari, Radha & Ahsan, Muhammad. (2020). Prediction of Online Shopper's Purchasing Intention Using Binary Logistic Regression, Decision Tree, and Random Forest. 10.13140/RG.2.2.16567.55209.

6. Sakar, C.O., Polat, S.O., Katircioglu, M. et al. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Comput & Applic 31, 6893–6908 (2019). https://doi.org/10.1007/s00521-018-3523-0

R Code Appendix

```
# Load Data
rm(list = ls())
setwd("/Users/macbookpro/Documents/Janice/Cal-State Fullerton/Fall 2020 Math
   533")
shop = read.csv("online_shoppers_intention.csv", header = TRUE)

# Libraries
library(corrplot)
library(ggplot2)
library(ggpubr)
library(factoextra)
library(rpart)
library(rpart.plot)
library(caret)
library(randomForest)
library(e1071)
library(ROCR)
library(Metrics)
library(plyr)
library(gridExtra)
library(kableExtra)
library(cluster)
#library(neuralnet)
library(funModeling)
library(car)
library(Rtsne)

# Clean Data
## variables like OperatingSystems expressed as int
## but should be expressed as categorical (factor)
## will change variable to factor

# rename some long named variables
library(tidyverse)
shop <- shop %>%
  rename(
    Admin = Administrative,
    Admin_Duration = Administrative_Duration,
    Info = Informational,
    Info_Duration = Informational_Duration,
  )

cols = c("OperatingSystems","Browser","Region","TrafficType","Revenue","
   Weekend")
shop[cols] <- lapply(shop[cols], factor)
sapply(shop, class)

# Check for any NA values
indx <- apply(shop, 2, function(x) any(is.na(x) | is.infinite(x)))
colnames = names(shop)
colnames[indx]


# EDA
pairs(shop[,c(1:10)], col = shop$Revenue, pch = 16)
plot(shop$ExitRates, shop$BounceRates, xlab = "Exit Rates",
     ylab = "Bounce Rates", col = "Red", pch = 16,
     main = "Exit Rates vs Bounce Rates")
options(repr.plot.width = 8, repr.plot.height = 5)
```

```r
ggplot(data = shop, mapping = aes(x = BounceRates, y = ExitRates)) +
  geom_point(mapping = aes(color = Revenue)) +
  geom_smooth(se = TRUE, alpha = 0.5) +
  ggtitle("Relationship between Exit Rates and Bounce Rates") +
  xlab("Bounce Rates") + ylab("Exit Rates") +
  geom_text(mapping = aes(x = 0.15, y = 0.05, label = "Correlation = 0.913"))
# positive linear relationship expected between bounce rates & exit rate

# bounce rate - percentage of visitors who enter the site from that page
# and then leave ("bounce") without triggering any other requests
# to the analytics server during that session

# exit rate - for all pageviews to the page, the percentage that were
# the last in the session

cor_shop = cor(shop[,c(1:10)])
corrplot(cor_shop, method = "number", type = "lower", diag = TRUE, tl.cex =
    0.75)

# Checking multicollinearity for logistic regression
vif(lm(Admin~.,shop))
vif(lm(PageValues~.,shop))
## MISTAKE HERE - in presentation using lm() - should use glm()

vif(glm(Revenue ~., family=binomial, data=shop))

formula <- as.formula(Revenue ~ Admin + Admin_Duration + Info + Info_Duration
    +
                        ProductRelated + ProductRelated_Duration + BounceRates
   +
                          ExitRates + PageValues + SpecialDay + Month + Browser
    +
                          OperatingSystems + Region +
                          TrafficType + VisitorType + Weekend)
fit <-glm(formula, data = shop, family = binomial)
fit.lm <- lm(formula, data = shop, family = binomial)
vif(fit) # aliased coefficients
vif(fit.lm)

#the linearly dependent variables
ld.vars <- attributes(alias(fit)$Complete)$dimnames[[1]]

#remove the linearly dependent variables variables
formula.new <- as.formula(
  paste(
    paste(deparse(formula), collapse=""),
    paste(ld.vars, collapse="-"),
    sep="-"
  )
)

#run model again
formula.new <- as.formula(Revenue ~ Admin + Admin_Duration + Info + Info_
    Duration +
                          ProductRelated + ProductRelated_Duration +
    BounceRates +
                          ExitRates + PageValues + SpecialDay + Month + Browser
      + Region +
                          TrafficType + VisitorType + Weekend)
fit.new <-glm(formula.new, data = shop, family = binomial)
vif(glm(Revenue ~.-OperatingSystems, data = shop, family = binomial))
```

```
vif_final = as.matrix(vif(glm(Revenue ~.-OperatingSystems, data = shop,
    family = binomial)))

# EDA Focusing on Revenue
##Histograms
H1 <- ggplot(shop, aes(Administrative)) + geom_histogram(aes(fill = Revenue),
  color="black", binwidth = round(sqrt(max(shop$Administrative)))) +
  ggtitle("Distribution of Administrative") +
  labs(y = "Frequency", x = "# of Pages Visited About Account Management") +
  theme_light() + scale_fill_brewer(palette="Accent")
H2 <- ggplot(shop, aes(Administrative_Duration)) + geom_histogram(aes(fill =
    Revenue),
  color="black", binwidth = round(sqrt(max(shop$Administrative_Duration)))) +
  ggtitle("Distribution of Administrative Duration") +
  labs(y = "Frequency", x = "Total Duration of Time Spent on Page About
    Account Management (sec)") +
  theme_light() + scale_fill_brewer(palette="Accent")
H3 <- ggplot(shop, aes(Informational)) + geom_histogram(aes(fill = Revenue),
  color="black", binwidth = round(sqrt(max(shop$Informational)))) +
  ggtitle("Distribution of Informational") +
  labs(y = "Frequency", x = "# of Pages Visited About Info on Web site") +
  theme_light() + scale_fill_brewer(palette="Accent")
H4 <- ggplot(shop, aes(Informational_Duration)) + geom_histogram(aes(fill =
    Revenue),
  color="black", binwidth = round(sqrt(max(shop$Informational_Duration)))) +
  ggtitle("Distribution of Informational Duration") +
  labs(y = "Frequency", x = "Total Time Spent on Page About Info on Web site
    (sec)") +
  theme_light() + scale_fill_brewer(palette="Accent")
H5 <- ggplot(shop, aes(ProductRelated)) + geom_histogram(aes(fill = Revenue),
  color="black",binwidth = round(sqrt(max(shop$ProductRelated)))) +
  ggtitle("Distribution of Product Related Pages") +
  labs(y = "Frequency", x = "# of Pages Visited About Products") +
  theme_light() + scale_fill_brewer(palette="Accent")
H6 <- ggplot(shop, aes(ProductRelated_Duration)) + geom_histogram(aes(fill =
    Revenue),
  color="black",binwidth = 1000) +
  ggtitle("Distribution of Product Related Duration") +
  labs(y = "Frequency", x = "Total Time Spent on Page About Products") +
  theme_light() + scale_fill_brewer(palette="Accent")
H7 <- ggplot(shop, aes(BounceRates)) + geom_histogram(aes(fill = Revenue),
  color="black") +
  ggtitle("Distribution of Bounce Rates") +
  labs(y = "Frequency", x = "Average Bounce Rate Value of Pages by Visitor")
    +
  theme_light() + scale_fill_brewer(palette="Accent")
H8 <- ggplot(shop, aes(ExitRates)) + geom_histogram(aes(fill = Revenue),
  color="black") + ggtitle("Distribution of Exit Rates") +
  labs(y = "Frequency", x = "Average Exit Rate Value of Pages by Visitor") +
  theme_light() + scale_fill_brewer(palette="Accent")
H9 <- ggplot(shop, aes(PageValues)) + geom_histogram(aes(fill = Revenue),
  color="black", binwidth = 20) + ggtitle("Distribution of Page Values") +
  labs(y = "Frequency", x = "Average Page Value of Pages by Visitor") +
  theme_light() + scale_fill_brewer(palette="Accent")
H10 <- ggplot(shop, aes(SpecialDay)) + geom_histogram(aes(fill = Revenue),
  color="black", binwidth = 0.25) +
  ggtitle("Distribution of Special Day") +
  labs(y = "Frequency", x = "Closeness of the Site Visiting Time To A Special
    Day") +
  theme_light() + scale_fill_brewer(palette="Accent")
```

```
ggarrange(H1,H2,H3,H4,H5,H6, labels = c("H1","H2","H3","H4","H5","H6"),ncol =
    3, nrow = 2)
ggarrange(H7, H8, H9, H10, labels = c("H7","H8","H9","H10"),ncol = 2, nrow =
   2)
# BarPlot
B1 <- ggplot(shop, aes(Month)) + geom_bar(aes(fill = Revenue))+
  scale_fill_brewer(palette="Spectral") +
  ggtitle("Revenue By Each Month") + theme_grey(base_size = 8)
B2 <- ggplot(shop, aes(OperatingSystems)) + geom_bar(aes(fill = Revenue))+
  ggtitle("Revenue By Operating System of Visitor") +
  labs(y = "Count", x = "Operating System (by Number)") +
  scale_fill_brewer(palette="Spectral") + theme_grey(base_size = 8)
B3 <- ggplot(shop, aes(Browser)) + geom_bar(aes(fill = Revenue))+
  ggtitle("Revenue By Browser of Visitor") +
  labs(y = "Count", x = "Browser (by Number)") +
  scale_fill_brewer(palette="Spectral") + theme_grey(base_size = 8)
B4 <- ggplot(shop, aes(Region)) + geom_bar(aes(fill = Revenue))+
  ggtitle("Revenue By Geographic Region of Visitor") +
  labs(y = "Count", x = "Region (by Number)") +
  scale_fill_brewer(palette="Spectral") + theme_grey(base_size = 8)
B5 <- ggplot(shop, aes(TrafficType)) + geom_bar(aes(fill = Revenue))+
  ggtitle("Revenue By Traffic Source of Visitor") +
  labs(y = "Count", x = "Traffic Source By Which Visitor Arrived at Website (
    by Number)") +
  scale_fill_brewer(palette="Spectral") + theme(axis.text.x = element_text(
    face = "bold",
    size = 6)) + theme_grey(base_size = 8)
B6 <- ggplot(shop, aes(VisitorType)) + geom_bar(aes(fill = Revenue))+
  ggtitle("Revenue By Visitor Type") +
  labs(y = "Count", x = "Type of Visitor") +
  scale_fill_brewer(palette="Spectral") + theme_grey(base_size = 8)
B7 <- ggplot(shop, aes(Weekend)) + geom_bar(aes(fill = Revenue))+
  ggtitle("Weekend Revenue vs. Weekday Revenue") +
  labs(y = "Count", x = "Weekend or Not (TRUE or FALSE)") +
  scale_fill_brewer(palette="Spectral") + theme_grey(base_size = 8)

B = ggarrange(B2,B3,B4,B5,B6, labels = c('OS','B','GR','T','V'), ncol = 3,
    nrow = 2)
annotate_figure(B, top = text_grob("Visualizing Revenue by Visitor
    Information", color = "red", face = "bold", size = 14))

ggplot(shop, aes(Revenue)) +  geom_bar(aes(fill = Revenue)) +
  geom_text(stat = "count", aes(label = after_stat(count)), size = 3, vjust =
     -1) +
  ggtitle("Revenue Count") +
  labs(y = "Count", x = "Revenue or No Revenue (TRUE or FALSE)") +
  scale_fill_brewer(palette="Spectral") + theme_grey(base_size = 8)

shop_cat <- shop[11:ncol(shop)]
print(freq(shop_cat))
#Trend line for revenue status based on months and trend line for visitor
    type based on months
options(repr.plot.width = 8, repr.plot.height = 5)

trend <- data.frame(table(shop$Month, shop$Revenue))
names(trend) <- c("Months", "Revenue", "Frequency")
ggplot(data = trend, mapping = aes(x = Months, y = Frequency)) +
  geom_line(mapping = aes(color = Revenue, group = Revenue), lwd = 1) +
  geom_point(mapping = aes(color = Revenue, group = Revenue, size = 0.1),
             show.legend = FALSE) + theme_light() +
```

```
  scale_y_continuous(breaks = seq(from = 0, to = 4000, by = 500)) +
  ggtitle("Trend line for revenue status based on months") +
  xlab("Months") + ylab("Visitors")

trend <- data.frame(table(shop$VisitorType, shop$Month))
names(trend) <- c("VisitorType", "Month", "Frequency")
ggplot(data = trend, mapping = aes(x = Month, y = Frequency)) +
  geom_line(mapping = aes(color = VisitorType, group = VisitorType),
  lwd = 1) + geom_point(mapping = aes(color = VisitorType,
  group = VisitorType, size = 0.1), show.legend = FALSE) +
  theme_light() + scale_y_continuous(breaks = seq(from = 0, to = 4000,
  by = 500)) + ggtitle("Trend line for visitor type based on months") +
  xlab("Months") + ylab("Visitors")

month_table <- table(shop$Month, shop$Revenue)
month_tab <- as.data.frame(prop.table(month_table, 2))
colnames(month_tab) <-  c("Month", "Revenue", "perc")

ggplot(data = month_tab, aes(x = Month, y = perc, fill = Revenue)) +
  geom_bar(stat = 'identity', position = 'dodge', alpha = 2/3) +
  xlab("Month")+
  ylab("Percent")+
  scale_fill_brewer(palette="Set1")

## PCA
pca = prcomp(shop[,1:10], center=T,scale. = T)
pca
names(pca)
pca$center
pca$scale

pr.var=pca$sdev^2 #variance explained by each principal component
pve=pr.var/sum(pr.var) #proportion of variance explained by each principal
   component
#pr.var divided by total variance explained by all principal components
pve
#34% of variance explained by first component
#16.75% of variance explained by 2nd component (50.76% explained)
#10.71% of variance explained by 3rd component (61.47% explained)
# so on...
# about 80% of variance explained by 5 principal components
plot(pve, main = "Proportion of Variance of the Principal Components", xlab="
   Principal Component", ylab="Proportion of Variance Explained ",type='b')


shop.label = shop[,ncol(shop)]
shop.pca = prcomp(shop[,1:10], center=TRUE, scale. = TRUE)
names(shop.pca)
head(shop.pca$x)
shop.pca$rotation
plot(shop.pca,type="l",main = "Variance of Principal Components")

source("ggbiplot.R")
g = ggbiplot(pca, obs.scale = 1, var.scale = 1,
             groups = shop.label, ellipse = TRUE,
             circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
g
```

```
# K_MEANS TRIAL
# Data Preparation for Clustering
shop$OperatingSystems <- factor(shop$OperatingSystems, order = TRUE, levels =
    c(6,3,7,1,5,2,4,8))
shop$Browser <- factor(shop$Browser, order = TRUE, levels = c
   (9,3,6,7,1,2,8,11,4,5,10,13,12))
shop$Region <- factor(shop$Region, order = TRUE, levels = c
   (8,6,3,4,7,1,5,2,9))
shop$TrafficType <- factor(shop$TrafficType, order = TRUE, levels = c
   (12,15,17,18,13,19,3,9,1,6,4,14,11,10,5,2,20,8,7,16))

shop$Month <- factor(shop$Month, order = TRUE, levels =c('Feb', 'Mar', 'May',
    'June','Jul', 'Aug', 'Sep','Oct', 'Nov','Dec'))
shop$Month_Numeric <-mapvalues(shop$Month, from = c('Feb', 'Mar', 'May', '
   June','Jul', 'Aug', 'Sep','Oct', 'Nov','Dec'), to = c
   (1,2,3,4,5,6,7,8,9,10))


shop$VisitorType <- factor(shop$VisitorType, order = TRUE, levels = c('
   Returning_Visitor', 'Other', 'New_Visitor'))
shop$VisitorType_Numeric <-mapvalues(shop$VisitorType, from = c("Returning_
   Visitor", "Other", "New_Visitor"), to = c(1,2,3))

# Binary
shop <- shop %>%
  mutate(Weekend_binary = ifelse(Weekend == "FALSE",0,1))

# K-means Clustering
sd.shop = scale(shop[,c(1:10)])
shop.clust = cbind(sd.shop,shop[,-c(1:11,16:18)])
km.out=kmeans(shop.clust, centers = 2, iter.max = 100)
km.out$size
km.out$centers
km.out$betweenss
km.out$totss

## Whithin clusters sum of squares
km.out$betweenss / km.out$totss

t1 <- table(km.out$cluster, shop$Revenue)
t1

# What value of k?
fviz_nbclust(x = shop.clust,FUNcluster = kmeans, method = 'wss')

# it seems around 3,4 or 5 clusters would be best using the elbow method

# K-means Clustering with k = 3
km.out3=kmeans(shop.clust, centers = 3, nstart=25)
km.out3$size
km.out3$centers
km.out3$betweenss
km.out3$totss

# Visualization: With numeric data only
km.num3=kmeans(sd.shop, centers = 3, nstart=25)
km.num4=kmeans(sd.shop, centers = 4, nstart=25)
km.num5=kmeans(sd.shop, centers = 5, nstart=25)
p3 <- fviz_cluster(km.num3, geom = "point",  data = sd.shop) + ggtitle("k = 3
   ")
p4 <- fviz_cluster(km.num4, geom = "point",  data = sd.shop) + ggtitle("k = 4
```

```
    ")
p5 <- fviz_cluster(km.num5, geom = "point",  data = sd.shop) + ggtitle("k = 5
    ")

grid.arrange(p3, p4, p5, nrow = 2)

## Within clusters sum of squares
km.out3$betweenss / km.out$totss

t3 <- table(km.out3$cluster, shop$Revenue)
t3

precision_kmeans1<- t3[1,1]/(sum(t3[1,]))
recall_kmeans1<- t3[1,1]/(sum(t3[,1]))

## Precision
precision_kmeans1
recall_kmeans1

F1_kmeans1<- 2*precision_kmeans1*recall_kmeans1/(precision_kmeans1+recall_
    kmeans1)
F1_kmeans1

## K_MEDOIDS (BETTER FOR MIXED DATA TYPES - categorical, numerical)
k_med_clust <- pam(x = shop.clust, k = 2)
## Size of our clusters
k_med_clust$id.med
## Centers of our clusters (medians)
k_med_clust$mediods
## Objective Function
k_med_clust$objective
## Summary of our cluster
k_med_clust$clusinfo
t1b <- table(k_med_clust$clustering, shop$Revenue)
t1b

# What value of k?
gower_dist <- daisy(shop.clust,
                    metric = "gower")
gower_dist <- daisy(shop,
                    metric = "gower")
summary(gower_dist)
gower_mat <- as.matrix(gower_dist)

# Output most similar pair

shop.clust[
  which(gower_mat == min(gower_mat[gower_mat != min(gower_mat)]),
        arr.ind = TRUE)[1, ], ]
shop[which(gower_mat == min(gower_mat[gower_mat != min(gower_mat)]),
        arr.ind = TRUE)[1, ], ]

# Calculate silhouette width for many k using PAM

sil_width <- c(NA)

for(i in 2:10){

  pam_fit <- pam(gower_dist,
                 diss = TRUE,
                 k = i)
```

```
  sil_width[i] <- pam_fit$silinfo$avg.width

}

# Plot sihouette width (higher is better)

plot(1:10, sil_width, main = "Optimal Number of Clusters by Sihouette Width",
     xlab = "Number of clusters",
     ylab = "Silhouette Width")
lines(1:10, sil_width)
# best number of clusters is 2

# Implement k-medoids clustering
pam_fit <- pam(gower_dist, diss = TRUE, k = 2)

pam_results <- shop.clust %>%
  mutate(cluster = pam_fit$clustering) %>%
  group_by(cluster) %>%
  do(the_summary = summary(.))

pam_results <- shop %>%
  mutate(cluster = pam_fit$clustering) %>%
  group_by(cluster) %>%
  do(the_summary = summary(.))

pam_results$the_summary

shop[pam_fit$medoids, ]

# Visualization
tsne_obj <- Rtsne(gower_dist, is_distance = TRUE)

tsne_data <- tsne_obj$Y %>%
  data.frame() %>%
  setNames(c("X", "Y")) %>%
  mutate(cluster = factor(pam_fit$clustering),
         name = shop$Revenue)

ggplot(aes(x = X, y = Y), data = tsne_data) +
  geom_point(aes(color = cluster))

## Size of our clusters
pam_fit$id.med
## Centers of our clusters (medians)
pam_fit$mediods
## Objective Function
pam_fit$objective
## Summary of our cluster
pam_fit$clusinfo
t1b <- table(pam_fit$clustering, shop$Revenue)
t1b

presicion_kmed<- t1b[1,1]/(sum(t1b[1,]))
recall_kmed<- t1b[1,1]/(sum(t1b[,1]))
## Precision
presicion_kmed

recall_kmed

F1_kmed<- 2*presicion_kmed*recall_kmed/(presicion_kmed+recall_kmed)
```

```
F1_kmed

# Too many levels for Browser and TrafficType Variables
summary(shop$Browser)
summary(shop$TrafficType)

levels(shop$Browser)
shop$Browser <- as.character(shop$Browser)
shop$Browser[shop$Browser == "3" | shop$Browser == "4" |
                  shop$Browser == "5"  | shop$Browser == "6" |
                  shop$Browser == "7" | shop$Browser == "8" |
                  shop$Browser == "9" |shop$Browser == "10" |
                shop$Browser == "11" | shop$Browser == "12" |
                shop$Browser == "13"] <- "1"

shop$TrafficType <- as.character(shop$TrafficType)
shop$TrafficType[shop$TrafficType == "6" |
                   shop$TrafficType == "7" | shop$TrafficType == "8" |
                   shop$TrafficType == "9" |shop$TrafficType == "10" |
                   shop$TrafficType == "11" | shop$TrafficType == "12" |
                   shop$TrafficType == "13" | shop$TrafficType == "14" |
                   shop$TrafficType == "15" | shop$TrafficType == "16" |
                   shop$TrafficType == "17" | shop$TrafficType == "18" |
                   shop$TrafficType == "19" | shop$TrafficType == "20"] <- "5
   "
shop$Browser = as.factor(shop$Browser)
shop$TrafficType = as.factor(shop$TrafficType)

# Remove variable exit rate due to collinearity with bounce rate
# shop = shop[,-c(8)]

# Test and Train Data for Classification Modelling
# Split into test and train data
index = 1:nrow(shop)
set.seed(2020)
train_index=sample(index, 0.75*nrow(shop))
trainset = shop[train_index,]
testset = shop[-train_index,]
xtrain = trainset[,-ncol(trainset)]
ytrain = trainset[,ncol(trainset)]

# Decision Tree
model_train_shop <- rpart(Revenue ~., data = trainset, method = "class")
rpart.plot(model_train_shop)

# Cross validation
set.seed(2020)
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3)
rpart.cv <- train(Revenue ~.,
                  data = trainset,
                  method = "rpart",
                  trControl = caret.control,
                  tuneLength = 15)
cat(paste("\nCross validation standard deviation:",
          sd(rpart.cv$resample$Accuracy), "\n", sep = " "))
rpart.cv$bestTune
rpart.best <- rpart.cv$finalModel
rpart.plot(rpart.best)
preds <- predict(rpart.cv, testset, type = "raw")
```

```
mean(preds==testset$Revenue)
shop_tree = confusionMatrix(preds, testset$Revenue)
shop_tree
shop_tree_acc = shop_tree$overall[['Accuracy']]
shop_tree_acc
CI.tree.low = shop_tree$overall[['AccuracyLower']]
CI.tree.high = shop_tree$overall[['AccuracyUpper']]
CI.tree.acc = c(CI.tree.low, CI.tree.high)
CI.tree.acc
tree.precision = shop_tree$byClass[['Pos Pred Value']]
tree.recall = shop_tree$byClass[['Sensitivity']]
F_tree = 2*((tree.precision*tree.recall) / (tree.precision+tree.recall))

# Logistic Regression
# Cross validation
set.seed(2020)
# caret.control <- trainControl(method = "repeatedcv",
#                               number = 10,
#                               repeats = 3)
logreg.cv <- train(Revenue ~.,
                   data = trainset,
                   method = "glm",
                   family = "binomial",
                   trControl = caret.control,
                   tuneLength = 15)
cat(paste("\nCross validation standard deviation:",
          sd(logreg.cv$resample$Accuracy), "\n", sep = " "))
logreg.best <- logreg.cv$finalModel
logreg.best.sum <- summary(logreg.best)
preds.lg <- predict(logreg.cv, testset, type = "raw")
# lgreg.shop <- glm(Revenue ~., family=binomial, data=trainset)
# summary(lgreg.shop)
# lgreg.shop.probs <- predict(lgreg.shop, testset, type="response")
# contrasts(shop$Revenue)
# log_reg_preds <- ifelse(lgreg.shop.probs > 0.5, "TRUE", "FALSE")
mean(preds.lg==testset$Revenue)
# cm_LG = confusionMatrix(as.factor(log_reg_preds),testset$Revenue)
# cm_LG
# cm_LG_acc = cm_LG$overall[['Accuracy']]
# cm_LG_acc
# CI.LG.low = cm_LG$overall[['AccuracyLower']]
# CI.LG.high = cm_LG$overall[['AccuracyUpper']]
# CI.LG.acc = c(CI.LG.low, CI.LG.high)
# CI.LG.acc

cm_LG = confusionMatrix(preds.lg,testset$Revenue)
cm_LG
cm_LG_acc = cm_LG$overall[['Accuracy']]
cm_LG_acc
CI.LG.low = cm_LG$overall[['AccuracyLower']]
CI.LG.high = cm_LG$overall[['AccuracyUpper']]
CI.LG.acc = c(CI.LG.low, CI.LG.high)
CI.LG.acc
LG.precision = cm_LG$byClass[['Pos Pred Value']]
LG.recall = cm_LG$byClass[['Sensitivity']]
F_LG = 2*((LG.precision*LG.recall) / (LG.precision+LG.recall))

# Random Forests
m = round(sqrt(ncol(trainset)))
# set.seed(2020)
# tune.rf = tuneRF(x = trainset[,-ncol(trainset)], y = trainset$Revenue,
```

```
   mtryStart = m, ntreeTry = 50)
#
# summary(tune.rf)
# print(tune.rf)
# rf.tuneresults = print(tune.rf)
# best.mtry = rf.tuneresults[which.min(rf.tuneresults[,2]),1]

# Cross validation
set.seed(2020)
# caret.control <- trainControl(method = "repeatedcv",
#                               number = 10,
#                               repeats = 3)
t_grid <- expand.grid(.mtry=m)
rf.cv <- train(x = xtrain, y = ytrain,
                  method = 'rf', metric = "Accuracy",
                  trControl = caret.control,
               tuneGrid= t_grid)
cat(paste("\nCross validation standard deviation:",
          sd(rf.cv$resample$Accuracy), "\n", sep = " "))
rf.best <- rf.cv$finalModel
rf.pred <- predict(rf.cv, testset, type = "raw")
# set.seed(2020)
# rf.shop = randomForest(Revenue~., data=trainset,mtry=best.mtry,ntree=50,
   importance=TRUE)
# rf.pred = predict(rf.shop, newdata = testset, type="class")
cm_rf = confusionMatrix(rf.pred, testset$Revenue)
cm_rf
cm_rf_acc = cm_rf$overall[['Accuracy']]
cm_rf_acc
CI.rf.low = cm_rf$overall[['AccuracyLower']]
CI.rf.high = cm_rf$overall[['AccuracyUpper']]
CI.rf.acc = c(CI.rf.low, CI.rf.high)
CI.rf.acc
rf.precision = cm_rf$byClass[['Pos Pred Value']]
rf.recall = cm_rf$byClass[['Sensitivity']]
F_rf = 2*((rf.precision*rf.recall) / (rf.precision+rf.recall))

# Variable Importance
varImpPlot(rf.best, main = "Variable Importance - Random Forest")

# SVM
# library(e1071)
# set.seed(2020)
# tune.out <- tune(svm, Revenue ~ ., data = trainset, ranges = list(cost = c
   (0.01,
#             0.1, 1, 5, 10, 100, 1000),gamma=c(0.5,1,2,3,4)))
# takes a while to execute

# will use small random portion of data to save time
set.seed(2020)
subset_train_index = sample(1:nrow(trainset), round(0.25*nrow(trainset)))
subset_train = trainset[subset_train_index,]

#cross-validation
set.seed(2020)
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3)

svm.cv <- train(Revenue ~ ., data = subset_train,
                method = 'svmRadial',
```

```
                trControl = caret.control,
                tuneGrid = expand.grid(sigma = c(0.25,0.5,1,2),
                                       C = c(0.1 ,1 ,10 ,100)))
# warnings () cannot scale data

cat(paste("\nCross validation standard deviation:",
          sd(svm.cv$resample$Accuracy), "\n", sep = " "))
svm.cv.best <- svm.cv$finalModel

sigma = svm.cv$bestTune[[1]]
C = svm.cv$bestTune[[2]]

svm.pred <- predict(svm.cv.best, testset, type = "raw")
final_svmfit = svm(Revenue ~ ., data = trainset, kernel="radial", cost = C,
   gamma = sigma)
svmfit_pred = predict(final_svmfit,newdata=testset)

cm_SVM = confusionMatrix(svmfit_pred, testset$Revenue)
cm_SVM
cm_SVM_acc = cm_SVM$overall[['Accuracy']]
cm_SVM_acc
CI.SVM.low = cm_SVM$overall[['AccuracyLower']]
CI.SVM.high = cm_SVM$overall[['AccuracyUpper']]
CI.SVM.acc = c(CI.SVM.low, CI.SVM.high)
CI.SVM.acc
SVM.precision = cm_SVM$byClass[['Pos Pred Value']]
SVM.recall = cm_SVM$byClass[['Sensitivity']]
F_SVM = 2*((SVM.precision*SVM.recall) / (SVM.precision+SVM.recall))
F_SVM

# ROCR curve SVM
svmfit.opt=svm(Revenue~., data=trainset, kernel="radial", cost=C, gamma=sigma
   , decision.values=T)
# On test data
pred_svm = predict(svmfit.opt,testset,decision.values=T)
fitted=attr(pred_svm,"decision.values")
fitted = 1 - fitted  # invert predictions to get right orientation for ROC
   curve

rocplot=function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  plot(perf,...)}
rocplot(fitted,testset["Revenue"],main="ROC for SVM", col = "red")

pr_svm <- prediction(fitted,testset["Revenue"])
#perf_svm <- performance(pr_svm, "tpr", "fpr")
auc_svm <- performance(pr_svm, 'auc')
#plot(perf_svm,main="ROC for SVM", col = "red")
auc_svm <- as.numeric(auc_svm@y.values)
auc_svm

# Neural Network
library(neuralnet)

nnetGrid <-  expand.grid(size = seq(from = 1, to = 10, by = 1),
                         decay = seq(from = 0.1, to = 0.5, by = 0.1))
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3)
nnetFit <- train(Revenue ~ .,
```

```
                  data = trainset,
                  method = "nnet",
                  metric = "Accuracy",
                  trControl = caret.control,
                  tuneGrid = nnetGrid,
                  verbose = FALSE)


cat(paste("\nCross validation standard deviation:",
          sd(nnetFit$resample$Accuracy), "\n", sep = " "))
nn.best <- nnetFit$finalModel

#import the function from Github
library(devtools)
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466
   c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r')

#plot each model
plot.nnet(nn.best, cex.val = 0.5)
wts.in = nn.best$wts
# numeric vector with length equal to the expected given the architecture
struct = nn.best$n
# numeric value of length three indicating network architecture
# (no. nodes for input, hidden, output)

plot.nnet(wts.in,struct=struct)

nn.pred <- predict(nnetFit, testset, type = "raw")

cm_nn = confusionMatrix(nn.pred, testset$Revenue)
cm_nn
cm_nn_acc = cm_nn$overall[['Accuracy']]
cm_nn_acc
CI.NN.low = cm_nn$overall[['AccuracyLower']]
CI.NN.high = cm_nn$overall[['AccuracyUpper']]
CI.NN.acc = c(CI.NN.low, CI.NN.high)
CI.NN.acc
NN.precision = cm_nn$byClass[['Pos Pred Value']]
NN.recall = cm_nn$byClass[['Sensitivity']]
F_NN = 2*((NN.precision*NN.recall) / (NN.precision+NN.recall))
F_NN

# MLP Neural Network (TRIAL)
TG = expand.grid(layer1=2:4,layer2=2:4,layer3=2:4)
mlpMLFit <- train(Revenue ~ ., data = trainset, method = "mlpML",
                  trControl = caret.control, preProcess = c("center", "scale"
   ),
                  tuneGrid=TG)

# ROC Plot for Decision Tree
pd.cv = predict(rpart.cv, testset, type = "prob")
pr.dt = prediction(pd.cv[,2],testset$Revenue)
auc_dt <- performance(pr.dt, 'auc')
auc_dt <- as.numeric(auc_dt@y.values)
auc_dt
rocplot(pd.cv[,2],testset$Revenue,main="ROC for Classification Tree", col = "
   orange")

# ROC Plot for Logistic Regression
lgreg.shop.probs = predict(logreg.cv, testset, type = "prob")
pr_lg <- prediction(lgreg.shop.probs[,2],testset$Revenue)
```

```
#perf <- performance(pr,measure = "tpr", x.measure = "fpr")
#plot(perf, main = "ROC for Logistic Regression", col = "blue")
auc_lg <- performance(pr_lg, 'auc')
auc_lg <- as.numeric(auc_lg@y.values) #0.888132
auc_lg
rocplot(lgreg.shop.probs[,2],testset$Revenue,main="ROC for Logistic
   Regression", col = "blue")

# ROC Plot for Random Forest
prf = predict(rf.cv, testset, type = "prob")
pr.rf <- prediction(prf[,2],testset$Revenue)
auc_rf <- performance(pr.rf, 'auc')
auc_rf <- as.numeric(auc_rf@y.values)
auc_rf
rocplot(prf[,2],testset$Revenue,main="ROC for Random Forest", col = "green")

# ROC for Neural Net
pnn = predict(nnetFit, testset, type = "prob")
pr.nn <- prediction(pnn[,2],testset$Revenue)
# omit neural net package
# prediction() conflict between neuralnet package & caret
auc_nn <- performance(pr.nn, 'auc')
auc_nn <- as.numeric(auc_nn@y.values)
auc_nn
rocplot(pnn[,2],testset$Revenue,main="ROC for Neural Net", col = "purple")

# Full ROC Plot
## Full ROC plot of all classification methods
rocplot(fitted,testset$Revenue,main="ROC for all Methods", col = "red")
rocplot(pd.cv[,2],testset$Revenue, col = "orange", add = "T")
rocplot(lgreg.shop.probs[,2],testset$Revenue, col = "blue", add = "T")
rocplot(prf[,2],testset$Revenue, col = "green", add = "T")
rocplot(pnn[,2],testset$Revenue, col = "purple", add = "T")
abline(coef = c(0,1), lty = "dashed")
legend(0.6, 0.4, c('SVM', 'Dec Tree', 'Log Reg', 'RF', 'NN'), c("red", "
   orange", "blue", "green", "purple"))

# Table Output of Results
Results <- data.frame(Model=c('Support Vector Machines', 'Decision Tree', '
   Logistic Regression', 'Random Forests', 'Neural Net'),
                      F_Score = c(F_SVM, F_tree, F_LG, F_rf, F_NN),
                      Accuracy = c(cm_SVM_acc, shop_tree_acc, cm_LG_acc, cm_
   rf_acc, cm_nn_acc),
                      Lower_Conf_Interval = c(CI.SVM.low, CI.tree.low, CI.LG.
   low, CI.rf.low, CI.NN.low),
                      Higher_Conf_Interval = c(CI.SVM.high, CI.tree.high, CI.
   LG.high, CI.rf.high, CI.NN.high),
                      Area_Under_Curve = c(auc_svm, auc_dt, auc_lg, auc_rf,
   auc_nn))
Results
Results %>%
  kbl(caption = "Shop Model Results") %>%
  kable_classic(full_width = F, html_font = "Cambria")

a <- ggplot(data = Results, mapping = aes(x = Results[,1], y = Results[,2],
                                  color = Results[,1])) +

  geom_point()  + xlab("Classification Models") + ylab("Accuracy Rate") +

  geom_errorbar(ymin = Results[,3], ymax = Results[,4], width=.2, position=
   position_dodge(0.05)) +
```

```
  ggtitle("Accuracy Rate for Each Model") +

  scale_x_discrete(labels=c("Dec. Tree", "Log Regn", "NN", "Random Forest", "
    SVM")) + ylim(0.86, 0.92)
a$labels$colour <- "Models"
a
```