



RED SOX PROJECT

Team Members

Oxana Brenes-Angulo

Fatma Duran

Jael Wemبالonge

Sefa Kilic

John Olubambi

Vinisha Thakkar

ISOM 837 Data Mining

Fall 2020

Table of Contents

1.Executive Summary	3
2.Problem Statement	3
3. Data Preparation and Understanding	4
4. Data Visualization	7
5. Data Modeling	13
5.1. Data Partition	13
5.2. Data Sampling	13
5.3. Decision Tree Models	15
5.3.1. User Created Maximal Decision Tree	16
5.3.1.1. User Created Interactive Tree Preparation	17
5.3.2. Optimal Decision Tree and Misclassification Rate	20
5.3.2.1. Optimal Tree Preparation	21
5.4. Logistic Regression Models	24
5.4.1. Logistic Regression	25
5.4.2. Stepwise Regression	27
5.5. Neural Network Model	30
6.1. Model Implementation	35
6.2. Variables of Importance	36
7. Findings and Conclusions	36
7. 1. Virtual Ticket	37
7.2. Adjusting Ticket Price	38
7.3. Transportation	38
7.4. Promotion	39
8. Challenges	39
8.1. Understanding Data Definitions	39
8.2. Complex Data Structure	40
8.3. Merging Data	40
8.4. Data filtration	40
8.5. SAS Coding	41
8.6. Data Modeling	41
9. References	42
Appendix I. SAS Enterprise Model	43
Appendix II. User Created Maximal Tree	44
Appendix III. Optimal Tree	45

1. Executive Summary

This report covers the case of declining revenues of Red Sox and the analysis carried out to uncover insights that could help us in understanding the reasons behind those issues and possibly aid the team at Red Sox with our analysis and recommendations. In this report, we have elaborated in detail about the processes like understanding the business context, cleaning the raw data and prepping it so it can be easily worked on, data visualization, data modeling and model implementation. Lastly, we speak about our findings and conclusions and provide recommendations to improve the ticket sales and revenue for Fenway Park.

Through this analysis, we have uncovered some interesting details related to customer age, ticket type and ticket trends relating to opponent teams. We have found that people aged 50 or older usually buy Seasonal tickets and are also usually the age group that missed out on most games. By digging deeper into the games against each opponent, it was clear that unpopular opponents like Tigers, Astros and Rockies were usually missed out on by a good percentage of people who had already bought tickets for those games. Another interesting factor is the distance. It would seem obvious that people coming from states outside Massachusetts will have a higher probability of missing out on games because of the distance. But contrary to our belief, there is a higher percentage of people who are MA residents and miss out on Red Sox games the most. We have explored this and other topics in much more detail below.

2. Problem Statement

The team at Boston Red Sox noticed that the ticket sales at Fenway park in 2018 were down by about 6 percent compared to the same time in 2017. And this has been an ongoing pattern since a few years. A six percent loss in ticket sales can translate to millions of dollars in ticket revenue for the Boston Red Sox. And the loss in ticket sales will also in turn impact lost sales for concessions at Fenway park. We are given data for ticketing purchase information of Boston Red Sox, who are buying tickets at which section of the park, for what game, at what price, and where are they from.

We would like to work with the data given so we can not only understand our current customer and their behavior better but also formulate a strategy by focusing our attention on the specific problem areas that we find out with our analysis.

3. Data Preparation and Understanding

In the beginning, the huge dataset and the Axiom dataset were provided by the client. There were 16 unary, 37 binary, 27 nominal, 16 interval and more than 5 million observations in the overall dataset. It was necessary to clean the dataset to move forward with the analysis. After our data cleaning and understanding part, the final dataset was better and easier to run the analysis. In the final dataset, there was one binary target variable, Attendance, which was created by SAS Code by considering the attendance of people to the games. Additionally, there were 4 binary, 9 nominal, 4 interval variables and less than 2 million observations.

To reach the final dataset, the data preparation process was crucial. In the data preparation process, sampling, merging, variable selection, filtering, data replacement and coding new variables were utilized. It was an iterative process except for sampling. All the tables were merged and there were some of them rejected which were not going to be used through the analysis. RS_Ticket tables including RS_Ticket_2018_03_04, 2018_05_06, 2018_07_08, 2018_09_10, 2019_03_04, 2019_05_06, 2019_07_08 and 2019_09_10 was used to merge under Ticket.

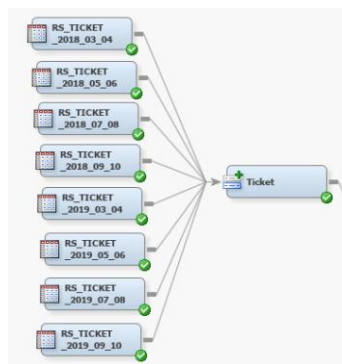


Figure 3.1.1. Merge representation of Ticket.

Also, the ticket scan tables were merged under the ticket scan. The representation is below.

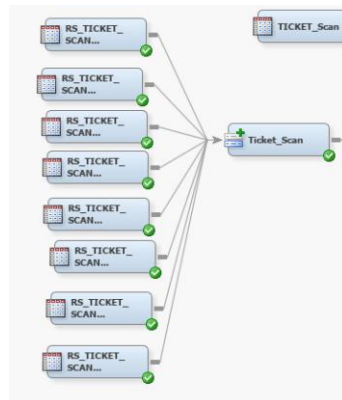


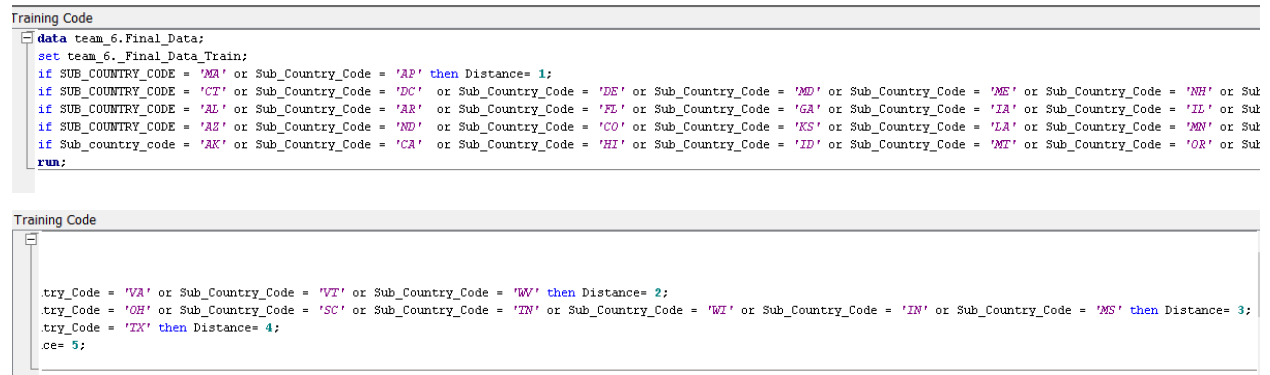
Figure 3.1.2. Merge representation of Ticket Scan.

To make the merge, account ID was needed to be rejected in the ticket scan and only the ticket info was used. To run the analysis, since the data was still so big, the sample node needed to be used to get a subset of the dataset. The subset is the half of the data which makes it easy to run the models and do the analysis. The total number of observations was 7,96,563. Filtering mode was utilized to handle missing values and “rejected” was selected for the variables that were not planned to go ahead with the analysis to include only the relevant information in the analysis. Event_cd, Seats, Sport_leisure_dollar and contact_id were some of the variables that were rejected. The other variables rejected are available below.

Name	Role	Level	Report	Order	Drop
AccountId	Input	Interval	No	No	No
AccountTypeCode	Input	Nominal	No	No	No
AccountTypeDesc	Rejected	Nominal	No	No	No
Age_Level	Input	Interval	No	No	No
Att	Target	Binary	No	No	No
Attendance	Rejected	Interval	No	No	No
Avg_Order_Dol	Rejected	Interval	No	No	No
AwayScore	Input	Nominal	No	No	No
AwayTeam	Input	Nominal	No	No	No
BuyerTypeDesc	Rejected	Nominal	No	No	No
Childress	Input	Nominal	No	No	No
CityState	Rejected	Nominal	No	No	No
ContactId	Rejected	Interval	No	No	No
Daylight	Input	Binary	No	No	No
Distance	Input	Nominal	No	No	No
EventCd	Rejected	Nominal	No	No	No
GameDate	Time ID	Interval	No	No	No
HomeScore	Input	Nominal	No	No	No
Offline_Avg_Dol	Rejected	Interval	No	No	No
PriceScaleDesc	Rejected	Nominal	No	No	No
REP_Gender	Input	Binary	No	No	No
REP_Income_Level	Input	Nominal	No	No	No
REP_Marital	Input	Binary	No	No	No
RepPosition	Input	Unary	No	No	No
SUB_COUNTRY_CODE	Rejected	Nominal	No	No	No
SeatRow	Rejected	Nominal	No	No	No
SeatSectionCd	Rejected	Nominal	No	No	No
Seats	Rejected	Nominal	No	No	No
Sport_leisure_Dol	Rejected	Interval	No	No	No
Temperature	Input	Interval	No	No	No
TicketId	Rejected	Interval	No	No	No
Ticket_Spend	Input	Interval	No	No	No
Tickets	Input	Nominal	No	No	No

Figure 3.1.3. Rejected variables

Moreover, the Distance variable is grouped in a way that MA is represented by 1, closer states are represented by 2 and the farther states are represented by 5. It was achieved by the SAS Code including if clauses to determine the states and each of its representations.



The figure consists of two screenshots of a SAS Training Code window. The top screenshot shows the following code:

```
data team_6.Final_Data;
set team_6.Final_Data_Train;
if SUB_COUNTRY_CODE = 'MA' or Sub_Country_Code = 'AP' then Distance= 1;
if SUB_COUNTRY_CODE = 'CT' or Sub_Country_Code = 'DC' or Sub_Country_Code = 'DE' or Sub_Country_Code = 'MD' or Sub_Country_Code = 'MS' or Sub_Country_Code = 'NH' or Sub
if SUB_COUNTRY_CODE = 'AL' or Sub_Country_Code = 'AR' or Sub_Country_Code = 'FL' or Sub_Country_Code = 'GA' or Sub_Country_Code = 'IA' or Sub_Country_Code = 'IL' or Sub
if SUB_COUNTRY_CODE = 'AZ' or Sub_Country_Code = 'ND' or Sub_Country_Code = 'CO' or Sub_Country_Code = 'KS' or Sub_Country_Code = 'LA' or Sub_Country_Code = 'MN' or Sub
if Sub_country_code = 'AK' or Sub_Country_Code = 'CA' or Sub_Country_Code = 'HI' or Sub_Country_Code = 'ID' or Sub_Country_Code = 'MT' or Sub_Country_Code = 'OR' or Sub
run;
```

The bottom screenshot shows the following code:

```
.try_Code = 'VA' or Sub_Country_Code = 'VT' or Sub_Country_Code = 'WV' then Distance= 2;
.try_Code = 'OH' or Sub_Country_Code = 'SC' or Sub_Country_Code = 'TN' or Sub_Country_Code = 'WY' or Sub_Country_Code = 'IN' or Sub_Country_Code = 'MS' then Distance= 3;
.try_Code = 'TX' then Distance= 4;
.ce= 5;
```

Figure 3.1.4. SAS Code representation for the Distance

Also, the attendance variable was created which is the target variable and to understand whether the people attend the games or not. It is important for the overall analysis to understand the core reasons that cause people not to participate in the games. The new column is created and when people have the ticket and attend the game, the value is assigned as 1 and if they have the ticket and they do not attend the game, the value will be assigned as 0. Again, SAS Code is utilized to reach the new binary variable. The SAS Code is:

```
data team_6.em_save_train_Att;
set team_6.em_save_train;
if Scan_Count > 0 then Att=1;
else Att= 0;
Run;
```

The Att in the code represents the attendance variable that is the target variable of the analysis and by doing so, the duplicate variables are handled as well.

Furthermore, the Stat Explore node is facilitated to be sure about the skewness and kurtosis of the variables and which of them needs any transform and impute.

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
Age_Level	INPUT	52.02393	14.03781	1573957	19168	18	52	99	0.012543	0.19516
AwayScore	INPUT	4.456097	2.991367	1593125	0	0	4	14	0.686777	0.187775
Childrens	INPUT	1.089392	0.944491	1593125	0	0	1	7	0.92736	0.85436
HomeScore	INPUT	5.698225	3.600226	1593125	0	0	5	19	1.184975	2.068946
Temperature	INPUT	67.98038	12.86562	1593125	0	34	70	94	-0.36329	-0.439
Ticket_Spend	INPUT	140.6052	171.4498	1593125	0	0	95	10396	7.858288	164.9099
Tickets	INPUT	1.674867	1.163929	1593125	0	1	1	113	11.45538	675.0695
Year_In_Home_Ind	INPUT	11.37517	4.4739	1593125	0	0	14	15	-0.90388	-0.53794

Figure 3.1.5. Stat Explore Node for variables

According to Figure 3.1.5., the HomeScore, Ticket_Spend and Tickets variables need transformation because they are highly skewed. Also, for the Age_Level, there are 19168 missing valued observations. Therefore, the Impute mode is utilized to handle the missing values in the Age_Level variable. Each missing variable creates a new variable and these variables and observations are named as IMP_Age_Level in the results. It was explained in a detailed way under the 5.4. Logistic regression models. Also, for the missing values, the filter node was run. For the Ticket_Spend the logarithmic transformation was utilized to solve the skewness issue. Afterwards, it was named under LG10_Ticket_Spend in the analysis.

4. Data Visualization

Before we worked on data modeling, we wanted to understand the data better by visualizing it. Since our core focus is around trying to understand the Fenway Park customers and their behavior, we wanted to chart their demographics details and see that against their attendance level for Red Sox games at Fenway Park. First, we started by looking at the location where our customers are located and how many people come from each area. We made a Map chart where the color indicates the Distance from Boston and the size indicates the number of people coming from each region. We can see that the highest population comes from MA and after that, the neighboring states of Maine, New Hampshire and so on.

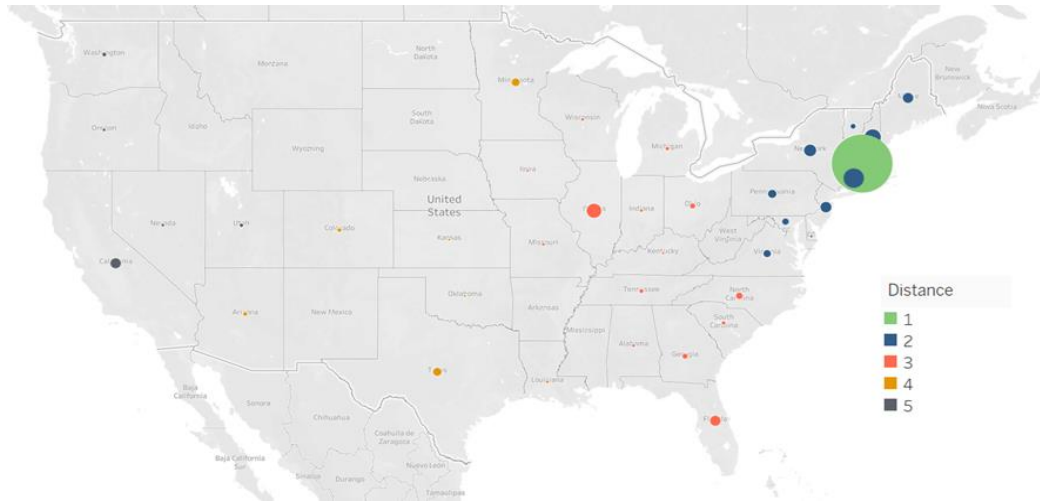


Figure 4.1. Distance and number of people

Next, we wanted to look at the income level of our customers compared with attendance levels. We thought we would see a pattern where high wage earners would be more likely to miss out on games than low wage earners and we wanted to test it out.

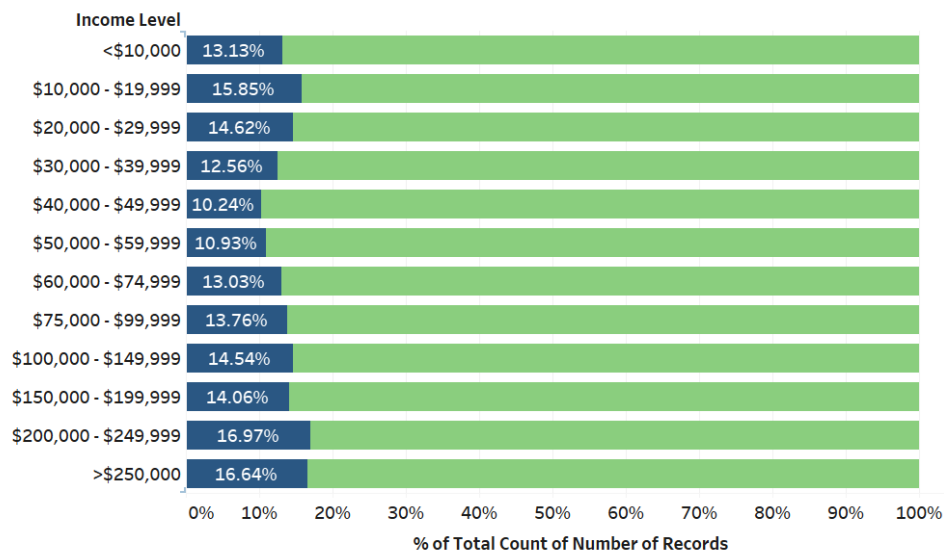


Figure 4.2. Income level by Attendance

The above chart shows the breakdown of income level and the percentage of people within that group who didn't attend the game. We hoped to see a pattern here, but unfortunately there is not much discernible difference between these income groups. Next, we looked at Age by attendance.

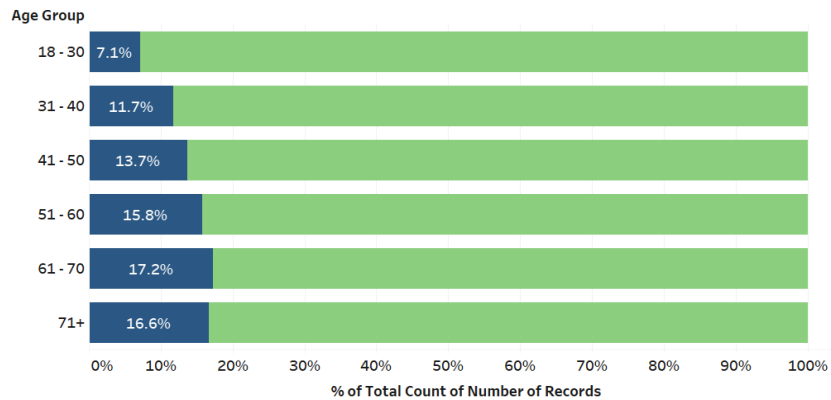


Figure 4.3. Age Group by Attendance

When comparing age by attendance levels, there seems to be a more obvious pattern here. Younger people end up missing the games a lot less than the older population. People 50 years or older generally tend to miss almost 15%+ games that they have purchased tickets for. Next, we wanted to see how does married vs single and the number of kids affected the attendance levels at Fenway Park.

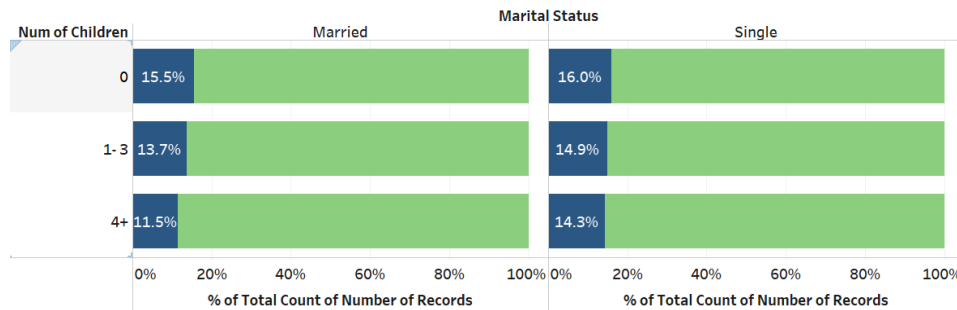


Figure 4.4. Marital Status and Number of Children by Attendance

We can see an obvious pattern here for the Married people bar graph on the left. People who are married usually attend more games if they have children. In fact, if they have more than 3 kids, their likelihood of attending the game increases by almost 2%. Same cannot be said for Single parents. We think this could be because they are too busy juggling multiple things and might not find the time to go to all the games, they purchase tickets for. After looking at the impact of different demographic details on the attendance level, we wanted to check out other factors to dig out interesting insights. First, we checked the attendance level against AccountTypeCode.

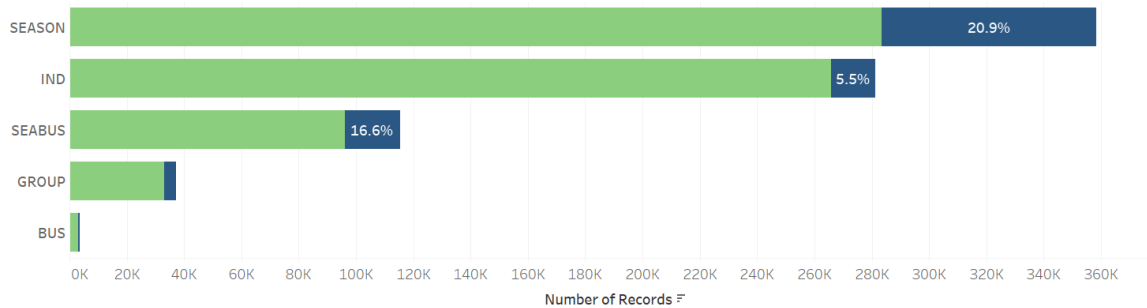


Figure 4.5. AccountTypeCode by Attendance

People who have bought Seasonal tickets miss almost 20% of the games they have paid for, whereas people buying Individual tickets miss out on only 5.5%. This is a huge gap, but it is understandable. Seasonal ticket buyers are not necessarily interested in all the games. Whereas individual ticket buyers specifically buy tickets for games they are interested in going to. We further wanted to look at the AccountTypeCode and dissect it with the Age group to see if people of different ages prefer buying different types of tickets.

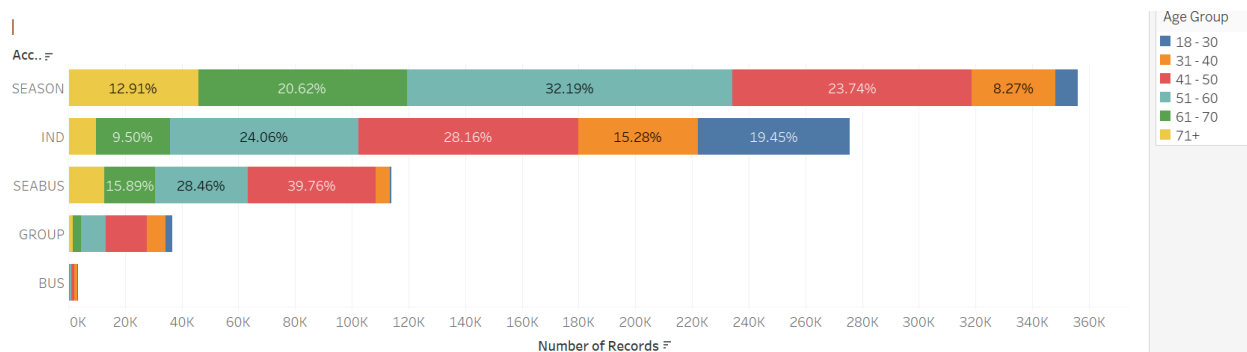


Figure 4.6. AccountTypeCode by Age Group

If we look at the age level 18-30, we can see that 19% of Individual tickets are bought by them but only a small fraction (2%) for Seasonal tickets. On the other hand 51-60, 61-70 and 71+ age groups have bigger percentages of Seasonal tickets compared to Individual tickets. This shows that older people prefer buying tickets for the full season instead of single games. The likely reason could be that older people have higher disposable income and can afford the seasonal tickets prices. Next, we wanted to look at the impact of temperature levels. For this, we first grouped the

temperature levels in five categories - Freezing, Cold, Cool, Warm and Hot. Then we plotted a chart for temperature against attendance levels.

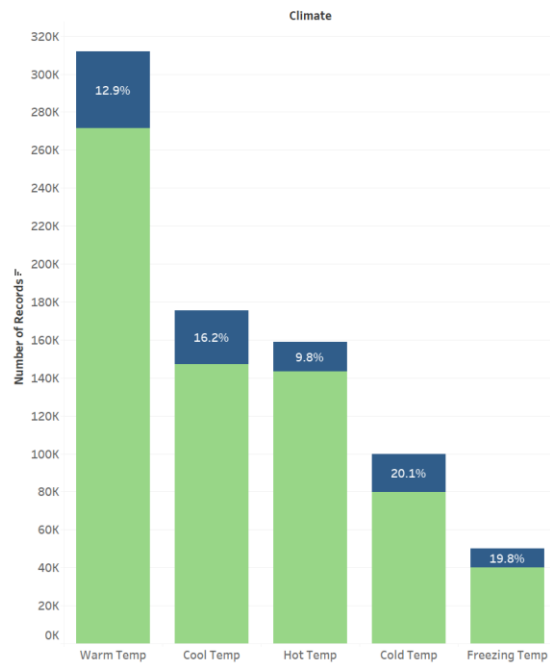


Figure 4.7. Number of transactions and Attendance by Temperature

Like we expected games during freezing temperatures are missed the most by ticket buyers. Although what is surprising is that only 9.8% people miss out on games on a Hot day. Another thing that we thought might be interesting to look at was attendance by the opponent team. And to add to that we wanted to look at the average ticket price for these games and if that would have any effect on the level of attendance.

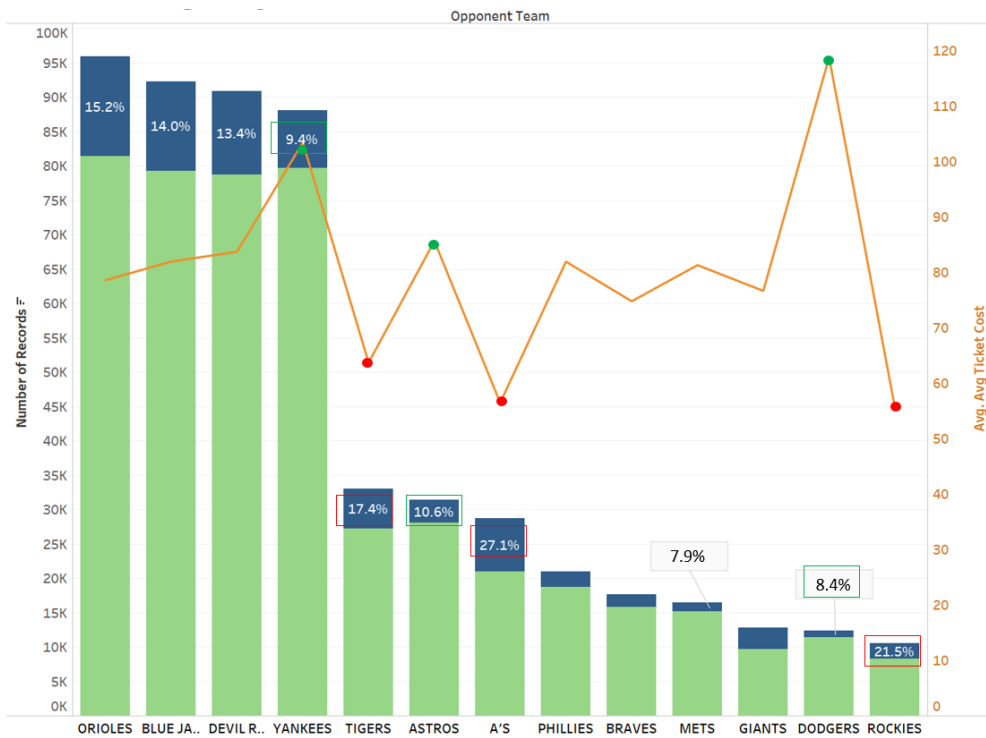


Figure 4.1.7. Number of transactions and Average Ticket Price by Away Team and Attendance

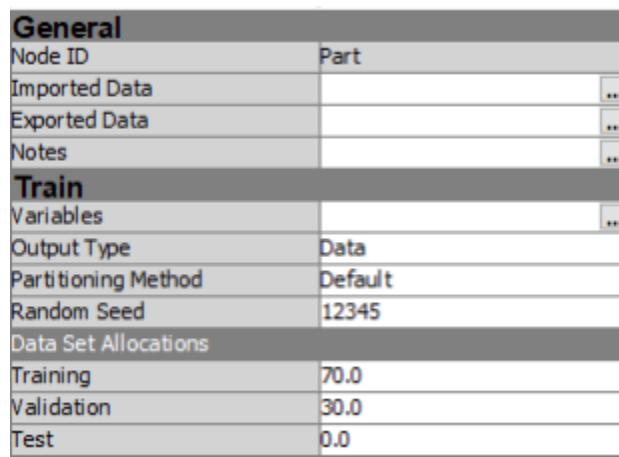
In the above graph, the line chart represents the average price for tickets for a game against each opponent team and the bar graph represents the number of games played against an opponent team and the attendance levels for these games. The highest peak in the line chart is for the Dodgers game with an average ticket price of around \$120. There are only 8% of people who missed out on games of Red Sox vs Dodgers. Similarly, for Yankees and Astros, the prices are comparatively high, and the non-attendance rates are on the lower side. On the other hand, games against Tigers, A's and Rockies which represent some of the lower ticket prices have above average non-attendance rates. It is possible that games against those opponents are not very interesting to people and hence the price is marked very low for them. But it also makes you wonder if people care less for the game when they spend less but make sure to attend a game if they have spent a higher price. Another interesting thing to note, Mets games have a very low percentage of non-attendance, but those games are not very highly priced.

5. Data Modeling

Our data modeling consisted of running and analyzing five different models: Decision Tree (User Created), Decision Tree Optimal, Logistic Regression, Logistic Stepwise Regression, and Neural Network. In the following sections we will describe each model in detail.

5.1. Data Partition

To assess the quality of the model generalization, we partitioned the data source. We used 70% of the dataset for training data, used for preliminary model fitting, and 30% for the validation data, used to prevent a modeling node from overfitting the training data and to compare models. To do that we used a Data Partition Node. In the property panel, for training, we entered 70 and for validation, we entered 30 as shown in Figure 5.1.1.



General	
Node ID	Part
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Output Type	Data
Partitioning Method	Default
Random Seed	12345
Data Set Allocations	
Training	70.0
Validation	30.0
Test	0.0

Figure 5.1.1. Settings for Data Partition Node

5.2. Data Sampling

Sampling is a statistical procedure that is concerned with the selection of the individual observation; it helps us to make statistical inferences about the population (Statistics Solution, 2020). Sampling data is recommended within SAS Enterprise Miner for larger data sets. Due to the size of our dataset, we wanted to be able to extract a sample from our final merged input data source, so that our models fitting time would decrease and be trained more rapidly to predict accurate outcomes.

After allocating 70% of our data to the training set and 30% to the validation set, the sample node was used, and the proper property panel settings were selected to create the sample. The percentage, criterion and level selection properties were adjusted so that it could give us a data set that has all the events and an equal size of a random sample of non-events.

Figure 5.2.1 shows how the property settings for the sample node were adjusted to create a sample data that was used to build our predictive models. The train data was sampled by selecting 50% of the event sample to be used in our model predictions. Since the target variable “Attendance” chosen to build our models was a binary; the level selection property was set to Event so that all events are part of the created sample. For the criterion property, the proportional stratified criterion was chosen since during stratified sampling, the proportion of observations in each stratum is the same in the sample as it is in the population (SAS,(a), 2020).

General	
Node ID	Smpl2
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Output Type	Data
Sample Method	Default
Random Seed	12345
Size	
Type	Percentage
Observations	.
Percentage	50.0
Alpha	0.01
PValue	0.01
Cluster Method	Random
Stratified	
Criterion	Proportional
Ignore Small Strata	No
Minimum Strata Size	5
Level Based Options	
Level Selection	Event
Level Proportion	100.0
Sample Proportion	50.0
Oversampling	
Adjust Frequency	No
Based on Count	No
Exclude Missing Levels	No

Figure 5.2.1. Settings for Sample Node

After adjustments were made in the property panel, the results window for the Sample node after being run displays a single output window. The obtained result in Figure 5.2.2 depicts the number of observations within the data set and the sample that has been created.

Sampling Summary		
Type	Data Set	Number of Observations
DATA	EMWS2.Stat_TRAIN	1593125
SAMPLE	EMWS2.Smpl2_DATA	796563

Figure 5.2.2 Sample Node Summary

Moreover, from the sample node result window, a list of the summary statistics is also created for each data set as shown in Figure 5.2.3. The output displays a list of the summary statistics for each of the data sets that have been created using the sample node. Lastly, we can also notice how the number of observations that were in the original training data, has noticeably changed in the sample data

Summary Statistics for Class Targets (maximum 500 observations printed)					
Data=DATA					
Variable	Numeric Value	Formatted Value	Frequency Count	Percent	Label
Att	0	0	228867	14.3659	
Att	1	1	1364258	85.6341	
Data=SAMPLE					
Variable	Numeric Value	Formatted Value	Frequency Count	Percent	Label
Att	0	0	114434	14.3660	
Att	1	1	682129	85.6340	

Figure 5.2.3. Sample Node Summary Statistics

5.3. Decision Tree Models

Decision tree model involves recursive partitioning of the training data to isolate groups of cases (observations or data points) with identical target values. The splitting is done by recursive partitioning, starting with all the observations, which are represented by the node at the top of the

tree. The algorithm splits this parent node into two or more child nodes in such a way that the responses within each child region are as similar as possible. The splitting process is then repeated for each of the child nodes, and the recursion continues until a stopping criterion is satisfied and the tree is fully built (SAS, (b), 2020).

The Decision Tree model was chosen for our dataset because of its treatment of missing data. Unlike Regressions and Neural Networks models, Imputation was not necessary to build our predictive models since decision trees have built-in ways to handle missing values. The search for a splitting rule uses the missing values of an input. Surrogate rules are available as backup when missing data prohibits the application of a splitting rule. Additionally, this model was ideal for our dataset because the Decision Tree tool ability to build models autonomously (automatically without user input) or interactively enabled us to play with the data and explore various assessment statistics measures to evaluate the performance of the validation data.

For our analysis, the first predictive model was built interactively to create the maximal tree, then pruned to obtain the Optimal Tree using Decision Tree. Building models interactively, gives more control and flexibility to create splitting rules by either manually selecting useful variables input, or by going with the default highest log worth variables for the predictive model. Decision tree models involve recursive partitioning of the training data to isolate groups of cases (observations or data points) with identical target values.

5.3.1. User Created Maximal Decision Tree

The first model was built using the Interactive Tree window under the Decision Tree node. The whole model maximal Tree can be found in Appendix I. All the variables chosen to build this model came from the ACXIOM dataset provided by the client, which contained Red Sox customers demographics and purchasing behaviors data. The model is called User Created Interactive Tree, because variables were selected manually without considering the log worth of the variables to create all the splitting rules. For our analysis, we were interested in the customers' demographics variables to understand their effects on the target variable attendance

5.3.1.1. User Created Interactive Tree Preparation

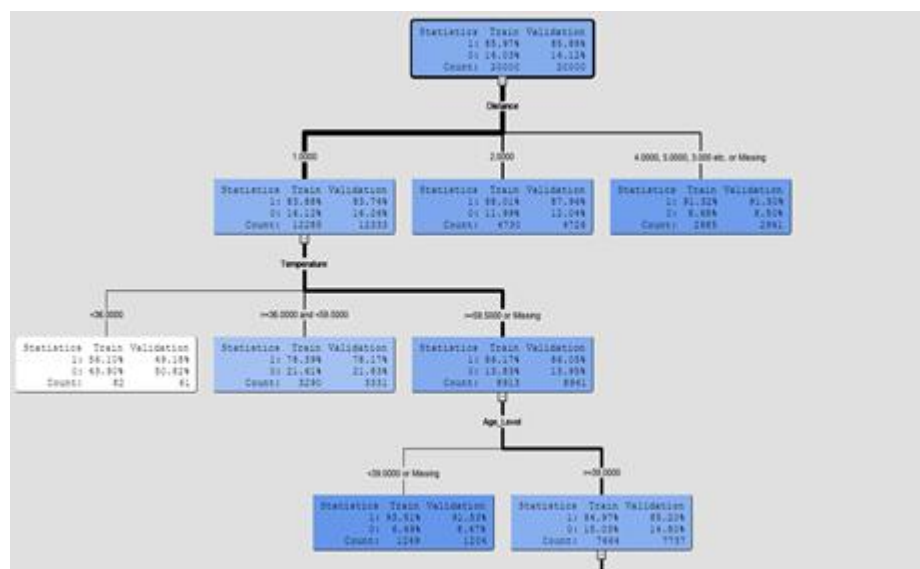
To construct our Interactive Tree model, adjustments were made in the decision tree property panel under the Interactive window as shown in Figure 5.3.1.1 (A), where all the variables of interest (*Distance*, *Temperature*, *AccountTypeCode*, *Tickets*, *AwayTeam*) were selected as shown in Figure 5.3.1.1 (B), then split to create the maximal tree as shown partially in Figure 5.3.1.3 (C). The whole model maximal Tree can be found in Appendix II.

General	
Node ID	Tree2
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Interactive	<input checked="" type="checkbox"/>
Import Tree Model	No
Tree Model Data Set	
Use Frozen Tree	Yes
Use Multiple Targets	No

(A)

Target Variable: Att			
Variable	Variable Description	-Log(p)	Branches
AccountTypeCode	AccountTypeCode	164.5472	3
Temperature	Temperature	61.969	3
Tickets	Tickets	41.5188	3
AwayTeam	AwayTeam	33.3286	3
Distance	Distance	26.6346	3

(B)



(C)

Figure 5.3.1.1 Settings for User Created Interactive Model

In addition to the above settings adjustment, The Use Frozen Tree property was changed from No to Yes, as shown in Figure 5.3.1.2 to assess the Interactive Tree on the validation data since the Frozen Tree property prevents the newly created Interactive tree from being changed by other property settings when the flow is run.

General	
Node ID	Tree2
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Interactive	
Import Tree Model	No
Tree Model Data Set	
Use Frozen Tree	Yes
Use Multiple Targets	No
Splitting Rule	
Interval Target Criterion	ProbF
Nominal Target Criterion	ProbChisq
Ordinal Target Criterion	Entropy
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	3
Maximum Depth	6
Minimum Categorical Size	5
Node	
Leaf Size	5
Number of Rules	5
Number of Surrogate Rule	0
Split Size	.
Split Search	
Use Decisions	No
Use Priors	No
Exhaustive	5000

Figure 5.3.1.2 Settings for Use Frozen Tree property

In Figure 5.3.1.3, the Subtree Assessment Plot Under the assessment statistic of Average Squared Error, confirms the optimality of the 29-leaf tree. The plot indicates that Using average squared error as the assessment measure, resulted in a tree with 29 leaves. Trees can have more leaves either by having more splits, more input variables, or both. We noticed that the performance on the training sample became better as the tree became more complex. But on the other hand, we see that the performance on the validation sample diminishes as model complexity increases. The validation performance shows evidence of model overfitting.

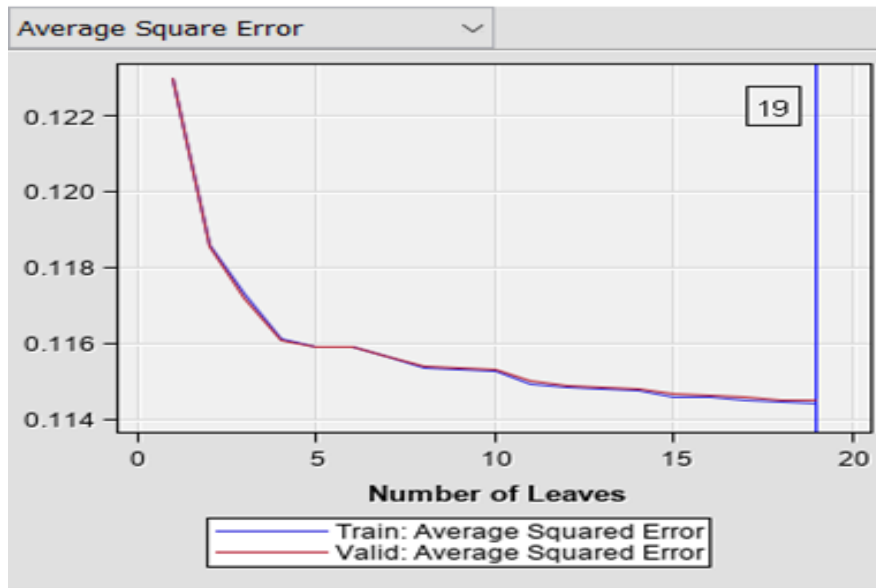


Figure 5.3.1.3 Assessment Plot based of Average Squared Error

For further exploration of validation performance, the assessment statistic was changed to Misclassification Rate as shown in figure 5.3.1.4. With the optimality of the 19-leaf tree confirmed, the plot shows that the model performance in terms of “misclassification” for the training data seems to indicate that the maximal tree is the most preferred for assigning predictions to cases. But the validation performance under Misclassification Rate appears to show evidence of model overfitting as it showed under the Average Squared Error assessment statistic plot.

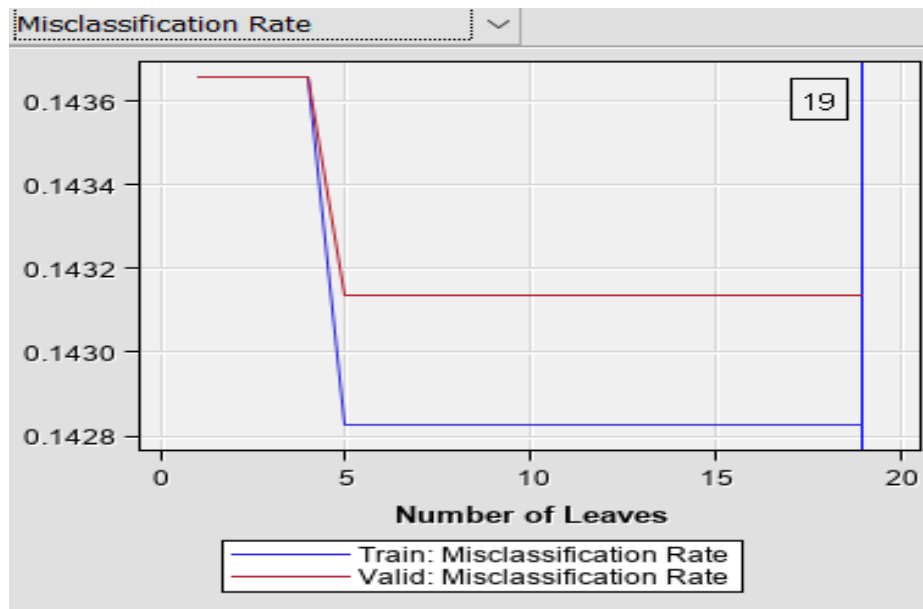


Figure 5.3.1.4 Assessment Plot based of Misclassification Rate

The User Create Maximal tree model performed well only with the train data under both average squared error and misclassification rate assessment measures. As shown in (Table 5.3.1.5), the train set performed well under both assessment measure with lower ASE and MISC rates, compared to the validation set who performed poorly due to evidence of model overfitting, as we can see, the rates obtained for the validation set both 0.143135 and 0.11448 under both assessments, are slightly higher than those obtained in the train. Therefore, further exploration of the validation performance was required.

Table 5.3.1.5 Summary Statistics for User Created Maximal Tree

Statistics	Train	Validation
Misclassification Rate	0.142825	0.143135
Average Squared Error	0.11141	0.11448

5.3.2. Optimal Decision Tree and Misclassification Rate

The second model built was the Optimal Decision Tree. The User Created Maximal tree model although it performed well for the training set, did not perform well for the validation set under both ASE (Average Square Error) and MISC (Misclassification Rate) due to evidence of model

overfitting. Therefore, to get a better model, we needed to “prune” the User Created maximal tree to find the number of leaves that would give us the best result.

5.3.2.1. Optimal Tree Preparation

In SAS Miner, the default method used to prune the maximal tree is Assessment. This simply means that SAS algorithms would choose the best tree in the sequence based on some optimality measure.

To have a better optimal model, pruning our data had to be done correctly using tools contained in the Decision Tree Node. The main tree pruning properties had to be adjusted so that the Decision Tree node could give us a more precise and accurate prediction model. For the Assessment Measure, we stuck to the default “Decision” assessment one as shown in Figure 5.3.2.1 because we wanted SAS Enterprise Miner to give us a tree that is optimized for making the best decisions, by identifying the tree with the lowest misclassification rate on the validation sample.

Property	Value
Exhaustive	5000
Node Sample	20000
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Decision
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validation	No
Number of Subsets	10
Number of Repeats	1
Seed	12345

Figure 5.3.2.1 Setting properties for Tree Pruning

After the node was run, The Plot showed us the model performance under Average Square Error by default as shown in Figure 5.3.2.2. But since we were interested in getting the lowest misclassification rate on the validation sample, we had to choose the correct criterion used to select the optimal tree which is the misclassification rate as shown in Figure 5.3.2.3.

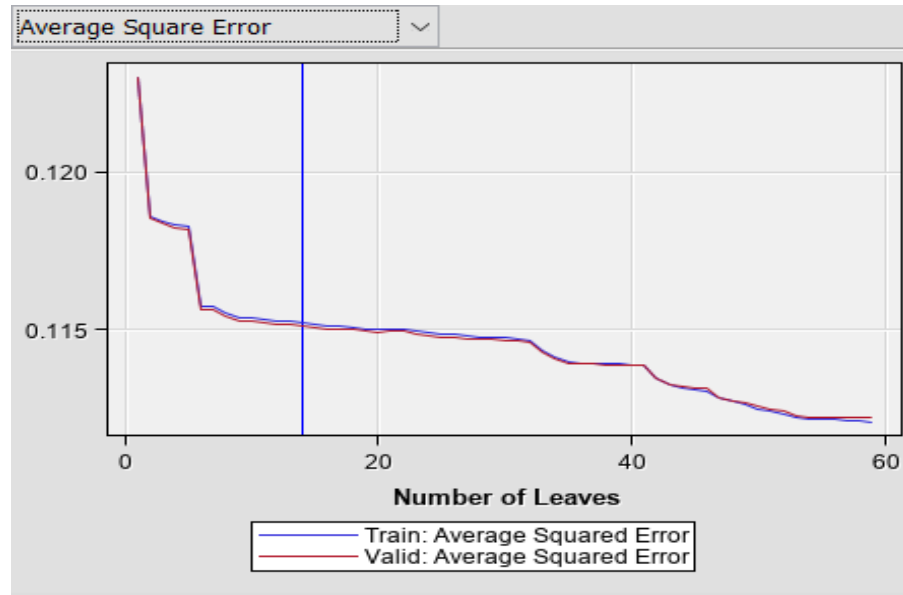


Figure 5.3.2.2 Model Performance under Average Squared Error

After Changing the basis of the plot to Misclassification Rate as shown in Figure 5.3.2.3. The 14-leaf tree is selected as the tree that has the lowest misclassification rate on the validation sample. The blue line represents “the best of the best”. To better understand our model performance, we looked at the blue line to understand and see visually that the 14-leaf tree has a lower misclassification rate for the validation set than any of those best trees with less or more than 14 leaves. It’s the lowest point on the red line known as the validation set. Therefore, the model outcome generated by SAS indicated that the 14-leaf tree is our optimal tree. The optimal tree generated by the Decision Tree node can be viewed in Appendix III.

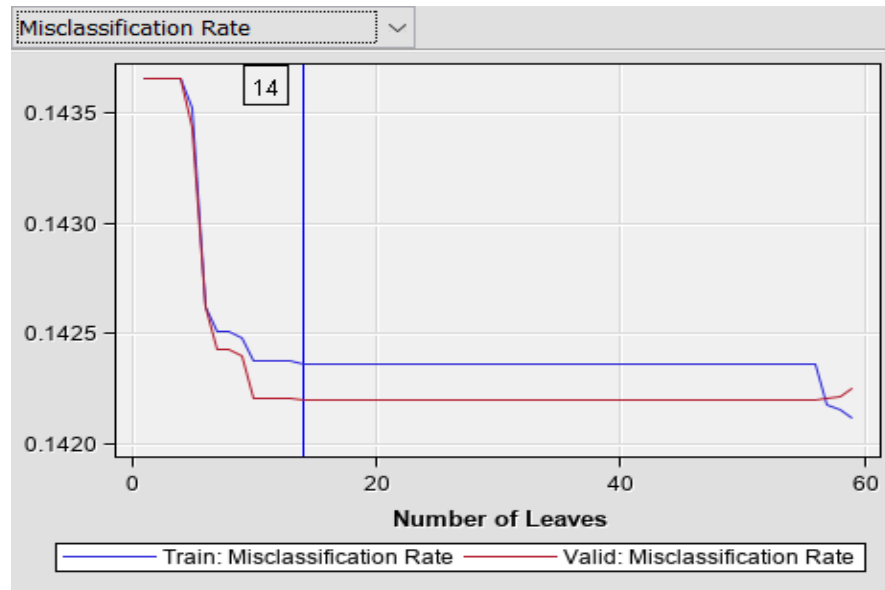


Figure 5.3.2.3 Model Performance under Misclassification Rate

Under the optimal tree model, both the Train and Validation sets performed well as shown in (Table 5.3.2.4.) The Average Square Error for train and validation was 0.115221 and 0.115113, respectively. The misclassification rate for train and validation was 0.142362 and 0.142198, respectively. With the misclassification rate assessment statistics, SAS generated the tree with the lowest misclassification rate, as well as the lowest point on the red line. Under the Maximal Tree as shown in (Table 5.3.1.6), the validation set didn't perform well and generated a slightly higher number compared to the numbers generated by the Optimal Tree

Table 5.3.1.5 Summary Statistics for Optimal Tree

Statistics	Train	Validation
Misclassification Rate	0.142362	0.142198
Average Squared Error	0.115221	0.115113

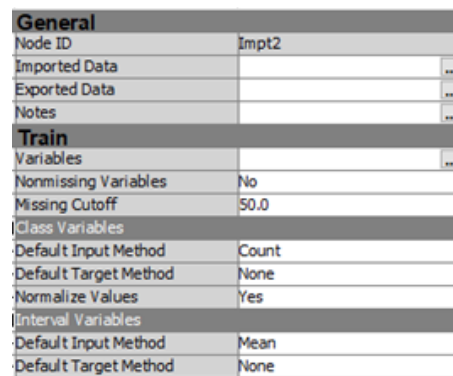
Table 5.3.1.5 Decision Trees Assessment Statistics Comparison

Assessment Statistics	Optimal Tree		Maximal Tree	
	Train	Validation	Train	Validation
Misclassification Rate	0.142362	0.142198	0.142825	0.143135
Average Square Error	0.115221	0.115113	0.11141	0.11448

5.4. Logistic Regression Models

The regression model belongs to the Model category in the SAS data mining process of Sample, Explore, Modify, Model, Assess (SEMMA). Logistic and Stepwise regression models were suitable for our predictive analysis, as these models attempt to predict the binary target will acquire the event of interest as a function of one or more independent inputs (SAS, (c), 2020). Before running any regression model, we needed to perform the following data mining tasks.

- **Imputing the missing values:** To address missing values in the dataset, we started with the impute node. In the property panel, we changed the Default Impute Method to Count for class variables and Mean for interval variables as shown in Figure 5.4.1. With these settings, each input with missing values generates a new input. In our results, new inputs named as IMP_Age_Level and IMP_Distance have missing values replaced by mean and count of nonmissing values and other nonmissing values copied from the original input.

The image shows a screenshot of the 'Imputation Settings' dialog box in SAS. The dialog is divided into several sections: 'General', 'Train', 'Class Variables', and 'Interval Variables'. Under 'General', 'Node ID' is set to 'Impt2'. Under 'Train', 'Variables' is set to 'All'. Under 'Class Variables', 'Default Input Method' is set to 'Count', 'Default Target Method' is set to 'None', and 'Normalize Values' is set to 'Yes'. Under 'Interval Variables', 'Default Input Method' is set to 'Mean' and 'Default Target Method' is set to 'None'.

General	
Node ID	Impt2
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Nonmissing Variables	No
Missing Cutoff	50.0
Class Variables	
Default Input Method	Count
Default Target Method	None
Normalize Values	Yes
Interval Variables	
Default Input Method	Mean
Default Target Method	None

Figure.5.4.1. Imputation Settings

- **Transform Variables:** Regression models are sensitive to extreme or outlying values in the input space (SAS, (c), 2020). In the results, inputs with highly skewed or kurtotic distributions can be selected over inputs that do not have this problem. So, to avoid this problem, we used the transform node before running regression models. In the property panel, we changed Interval Inputs to Log 10 as shown in Figure 5.4.2.

General	
Node ID	Trans
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Formulas	
Interactions	
SAS Code	
Default Methods	
Interval Inputs	Log 10
Interval Targets	None
Class Inputs	None
Class Targets	None
Treat Missing as Level	No
Sample Properties	
Method	First N
Size	Default
Random Seed	12345

Figure.5.4.2 Settings for Transform Variables

In the results panel, we see that IMP_Age_Level, Temperature, Ticket_Spend values standardized and new inputs named as LG10_IMP_Age_Level, LG10_Temperature, and LG10_Ticket_Spend (Figure 5.4.3).

Source	Method	Variable Name	Formula	Number of Levels	Non-Missing	Missing	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
Input	Original	IMP_Age_Level			557593	0	18	99	52.02605	13.94824	0.015046	0.236341
Input	Original	Temperature			557593	0	34	94	67.9976	12.85841	-0.36597	-0.436041
Input	Original	Ticket_Spend			557593	0	0	10396	140.6386	172.3883	8.034011	174.1176
Output	Computed	LG10_IMP_Age_Level	log10(IMP_Age_Level)		557593	0	1.278754	2	1.707491	0.126851	-0.92948	1.097449
Output	Computed	LG10_Temperature	log10(Temperature)		557593	0	1.544058	1.977724	1.83053	0.08732	-0.82509	0.318522
Output	Computed	LG10_Ticket_Spend	log10(Ticket_Spend)		557593	0	0	4.016908	1.97998	0.384997	-0.24305	1.879941

Figure. 5.4.3 Transform Node Results

After performing these data mining tasks, our data set was ready to run regression nodes for our Logistic and Stepwise Regression Models.

5.4.1. Logistic Regression

To run Logistic Regression Model, in the property panel, under Class Targets, we selected Regression Type as Logistic Regression, and under Model Selection, we changed Use Selection Defaults to No as shown in Figure 5.4.1.1.

Variables	
Equation	
Main Effects	Yes
Two-Factor Interactions	No
Polynomial Terms	No
Polynomial Degree	2
User Terms	No
Term Editor	
Class Targets	
Regression Type	Logistic Regression
Link Function	Logit
Model Options	
Suppress Intercept	No
Input Coding	Deviation
Model Selection	
Selection Model	None
Selection Criterion	Default
Use Selection Defaults	No
Selection Options	

Figure.5.4.1.1. Settings for Logistic Regression Node

In Figure 5.4.1.2, we see roles of variables used and not used by the Regression node: The fit model had 16 inputs to predict binary target and rejected 17 inputs. In Figure 5.4.1.2., we see the roles of variables used and not used by the Regression node: The fit model had 16 inputs to predict binary target and rejected 17 inputs. The number of parameters is 119.

Variable Summary		
Role	Measurement Level	Frequency Count
INPUT	BINARY	3
INPUT	INTERVAL	3
INPUT	NOMINAL	9
INPUT	UNARY	1
REJECTED	INTERVAL	8
REJECTED	NOMINAL	9
TARGET	BINARY	1

Figure 5.4.1.2. Logistic Regression Node Results, Variable Summary

Model Information	
Training Data Set	WORK.EM_DMREG.VIEW
DMDB Catalog	WORK.REG2_DMDB
Target Variable	Att
Target Measurement Level	Ordinal
Number of Target Categories	2
Error	MBernoulli
Link Function	Logit
Number of Model Parameters	119
Number of Observations	557593

Figure 5.4.1.3. Logistic Regression Node Results, Model Information

As shown in the Figure 5.4.1.4., all inputs, except REP_Gender are significant in the Regression Model.

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
AccountTypeCode	8	12610.3682	<.0001
AwayScore	14	832.1852	<.0001
AwayTeam	20	3840.3423	<.0001
Childrens	6	183.6110	<.0001
DayNight	1	50.0706	<.0001
HomeScore	17	1489.5487	<.0001
IMP_Distance	4	572.9638	<.0001
LG10_IMP_Age_Level	1	144.3409	<.0001
LG10_Temperature	1	2280.5903	<.0001
LG10_Ticket_Spend	1	1491.9513	<.0001
REP_Gender	1	0.0741	0.7854
REP_Income_Level	11	405.4835	<.0001
REP_Marital	1	122.5943	<.0001
RoofPosition	0	0.0000	.
Tickets	17	865.2956	<.0001
Year_In_Home_Ind	15	376.0922	<.0001

Figure 5.4.1.4. Logistic Regression Results, Type 3 Analysis of Effects

The Average Square Error for train and validation was 0.11407 and 0.114137, respectively. The misclassification rate for train and validation was 0.143657 and 0.143692, respectively (Table 5.4.1.1). Depending on the results, we can say the model is a good model.

Table 5.4.1.1. Summary Statistics for Logistic Regression

Statistics	Train	Validation
Average Square Error	0.11407	0.114137
Misclassification Rate	0.143657	0.143692

5.4.2. Stepwise Regression

To run the Stepwise Regression Model, in the property panel, under Class Targets, we selected Regression Type as Logistic Regression, and under Model Selection, we selected Selection Model as Stepwise as shown in Figure 5.4.2.1.

Variables	
Equation	
Main Effects	Yes
Two-Factor Interactions	No
Polynomial Terms	No
Polynomial Degree	2
User Terms	No
Term Editor	
Class Targets	
Regression Type	Logistic Regression
Link Function	Logit
Model Options	
Suppress Intercept	No
Input Coding	Deviation
Model Selection	
Selection Model	Stepwise
Selection Criterion	Default
Use Selection Defaults	Yes

Figure.5.4.2.1. Settings for Stepwise Regression Node

The Average Square Error for train and validation was 0.11407 and 0.337842, respectively. The misclassification rate for train and validation was 0.143664 and 0.143696, respectively (Table 5.4.2.1). Depending on the results, the model was selected as last first.

Table 5.4.2.1. Summary Statistics for Stepwise Regression

Statistics	Train	Validation
Average Square Error	0.11407	0.337842
Misclassification Rate	0.143664	0.143696

The iteration plot in Figure 5.4.2.2. shows Average Square Error (Train and Validation) from the model that is selected in each step of the stepwise process. Model fit statistics for Train and Validation are slightly different and the smallest average square error occurs in the last step, Step 14. The model from Step 14 should provide a less biased one.

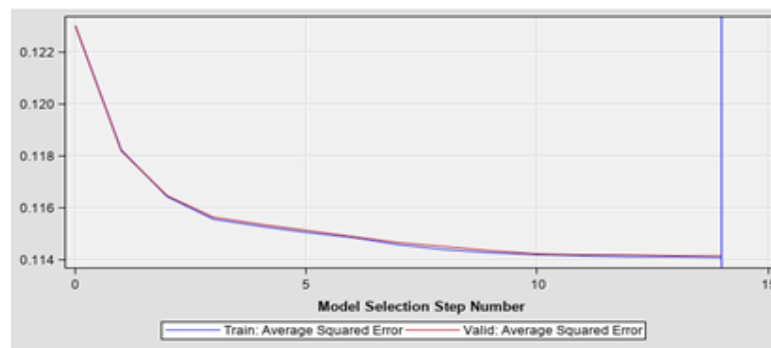


Figure.5.4.2.2. Iteration Plot based on Average Square Error for Stepwise Regression

The iteration plot in Figure Figure.5.4.2.3. shows that the model has 14 steps and the selected model is the model trained in the last step. As shown in Figure.5.4.2.4., all inputs had a chi-square p-value below 0.05. With the validation data, we see overfitting, that's why this model is not good to consider to use.

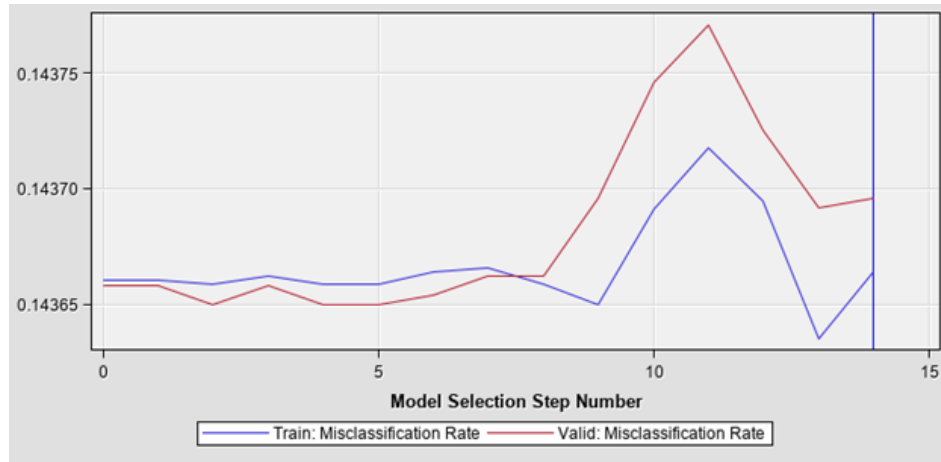


Figure.5.4.2.3. Iteration Plot based on Misclassification Rate for Stepwise Regression

Type 3 Analysis of Effects			
Effect	DF	Wald	
		Chi-Square	Pr > ChiSq
AccountTypeCode	8	13046.0411	<.0001
AwayScore	14	832.2792	<.0001
AwayTeam	20	3840.2824	<.0001
Childrens	6	183.5343	<.0001
DayNight	1	50.0447	<.0001
HomeScore	17	1489.5431	<.0001
IMP_Distance	4	574.6368	<.0001
LG10_IMP_Age_Level	1	144.3536	<.0001
LG10_Temperature	1	2280.5263	<.0001
LG10_Ticket_Spend	1	1493.2029	<.0001
REP_Income_Level	11	405.5729	<.0001
REP_Marital	1	123.3901	<.0001
Tickets	17	865.6906	<.0001
Year_In_Home_Ind	15	376.0317	<.0001

Figure.5.4.2.4. Stepwise Regression Model Results, Type 3 Analysis of Effects

5.5. Neural Network Model

The Neural Network model passes information between its layers so a specific category or predicted value's input can be mapped (SAS, (c),2020). The model was ideal for our data set because neural networks have the capability to be implemented quickly even in enormous data sets.

Our Neural Network Node took advantage of the transformation, replacement, and imputation nodes used to prepare the data for the Logistic and Stepwise Regressions. The following settings were used:

- In the property panel, we selected Optimization. Then, in the optimization window, we changed Enable to No, as shown in Figure 5.5.1. (A).
- In the same Optimization window, we changed the maximum iterations to 200, as shown in Figure 5.5.1 (B). This was a “back and forth” process as we had to try different values that fit our data set. For example, when we run the model with the default 50 iterations, we obtained (in the result window) that more than 50 iterations were needed. Then, we proceeded to run the model with 100 iterations, and we got as well that more than 100 iterations were needed. Finally, 200 iterations were optimal for our model.

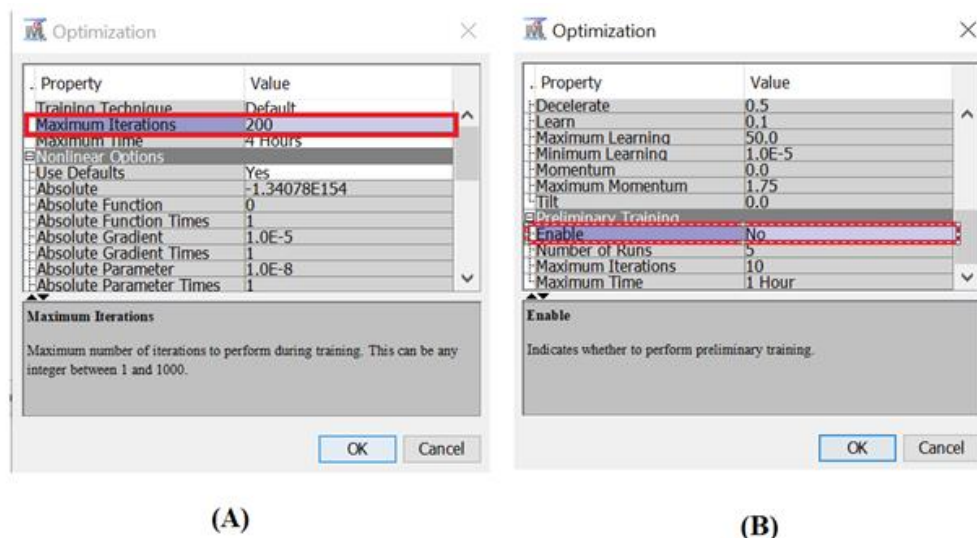


Figure 5.5.1. Optimization Settings for the Neural Network Model.

The Neural Network had a good model performance on both average squared error and misclassification rate scales. The average squared error for train and validation was 0.111297 and 0.111415, respectively. The misclassification rate for train and validation was 0.142419 and 0.142323, respectively (Table 5.5.1.). The slightly higher number in the validation square error compared to the train, is an indication of possible data fitting. Thus, for the further model comparison analysis, we decided to use the misclassification rate.

Table 5.5.1. Summary Statistics for Neural Network

Statistics	Train	Validation
Average Square Error	0.111297	0.111415
Misclassification Rate	0.142419	0.142323

The iteration plot in Figure 5.5.2. shows the average square error versus the optimal iteration. As shown in Figure 2, our data preparation helped us to reduce the number of inputs in the data set. The reduction in inputs, helped the neural network to reduce the number of modeling weights and thus improve model performance. The optimal number of iterations was achieved near iteration 129 as indicated by the vertical blue line.

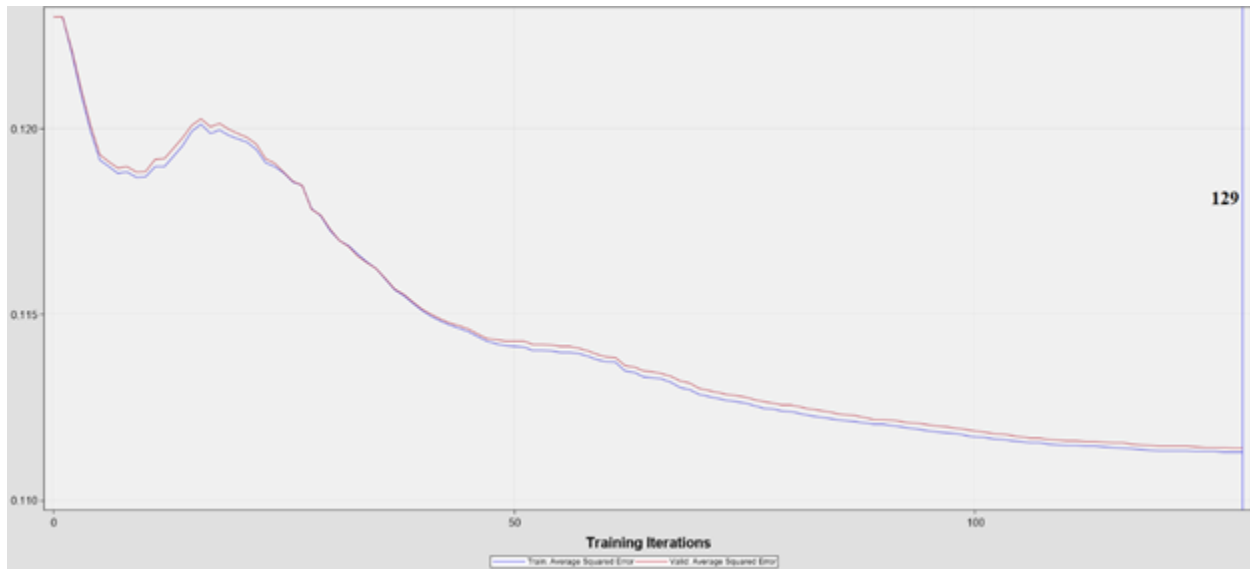


Figure 5.5.2. Iteration Plot based on average square error for Neural Network.

The iteration plot in Figure 5.5.3. shows the misclassification rate versus the optimal iteration. As shown in Figure 3, there was an unusual peak at iteration 19. This could have been caused by the modeling weights. However, the optimal performance of the model was reached at iteration 129 indicated by the vertical blue line. This result was achieved in the average square error as well.



Figure 5.5.3. Iteration Plot based on misclassification rate for Neural Network.

6. Model Evaluation

The model comparison node is a tool used when several models need to be compared for decision prediction. The node uses different statistics to rate model performance. For example, it can use average square error, misclassification rate, Schwarz's Bayesian, Lomogorov-Smirnov, among others (SAS (d), 2020). For the present analysis, the model comparison node assessment setting was set to misclassification rate.

Table 6.1. summarizes the results from the model comparison. Overall, all the models had a similar accuracy of 85%. The Optimal Decision Tree was the model that performed out the best by having the lowest validation misclassification rate of 0.142198, and accuracy of 85.78%, and a proportion of false positives and negatives of 13.99% and 41.34%, respectively. The

misclassification validation rate was also lower than the misclassification train rate, indicating there were no overfitting issues.

Table 6.1. Summary statistics for the Model Comparison.

Model	Misclassification Rate (Train)	Misclassification Rate (Validation)	Accuracy	Proportion False Positives	Proportion False Negatives
★ Decision Tree (Optimal)	0.142362	0.142198	85.78%	13.99%	41.31%
Neural Network	0.142419	0.142323	85.77%	3.43%	66.65%
Decision Tree (User Created)	0.14366	0.143658	85.63%	14.37%	0.00%
Logistic Regression	0.143657	0.143692	85.63%	14.35%	52.90%
Stepwise Logistic Regression	0.143664	0.143696	85.63%	14.35%	53.24%

The Decision Tree (User Created) had a misclassification validation rate of 0.142323, the accuracy of 85.77%, and a respective proportion of false positives and negatives of 3.43% and 66.65%. Additionally, Logistic regression had a misclassification validation rate of 0.143692 and accuracy of 85.63%, and a respective proportion of false positives and negatives of 14.35% and 52.90%. In the case of Stepwise Logistic Regression, the misclassification validation rate was 0.143696, the accuracy of 85.63%, and the proportion of false positives and negatives of 14.45% and 53.24%, respectively.

Regarding the model assessment, the ROC curves in Figure 6.1. show that all the models were having similar behavior. However, the Optimal Decision Tree was the one having a higher area under the curve which means that the model is better as it gets closer to the value of 1.

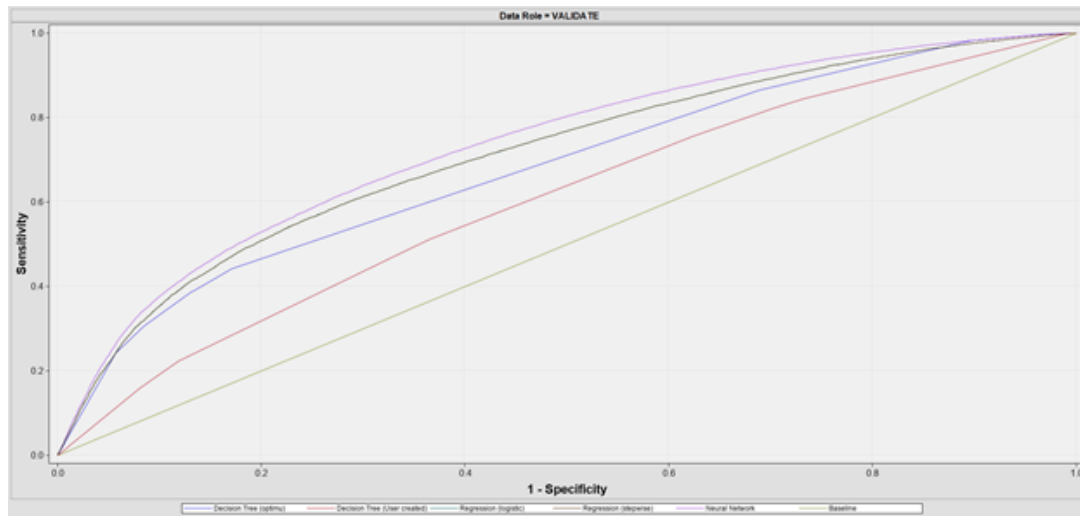


Figure 6.1. ROC Chart for all the Models.

Moreover, the cumulative lift chart in Figure 6.2., shows the plotted cumulative actual probabilities of customers that would not attend the game. As it can be seen in the chart the Optimal Decision Tree had the highest cumulative lift. However, none of the models had a cumulative lift higher than 1.50. This means that none of the models performed more than 50% better than having no model.

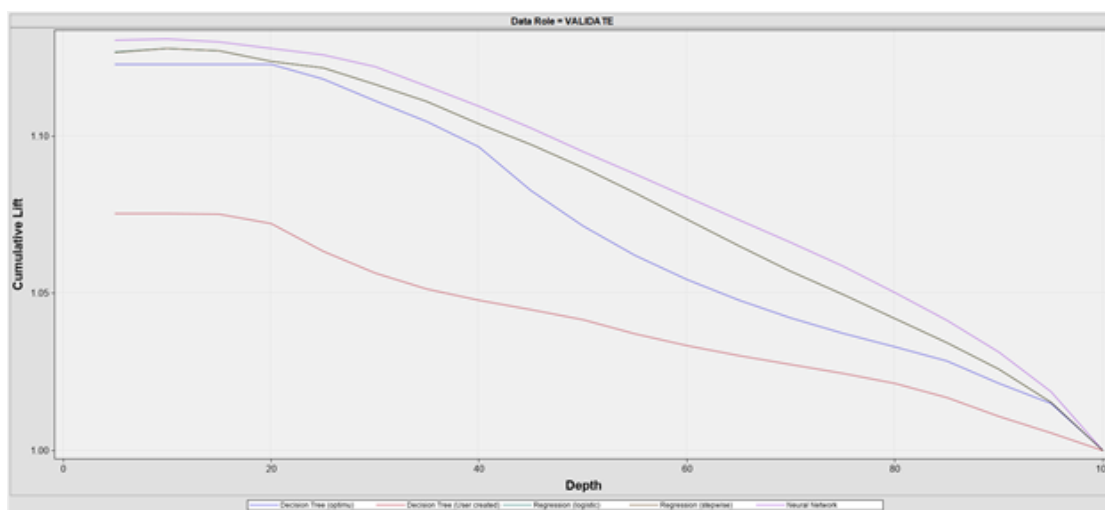


Figure 6.2. Cumulative Lift Chart for all the Models.

6.1. Model Implementation

We used the Score Node to generate predicted target values on our selected model Optimal Decision Tree. The predictions from the Scoring node are similar to the Training data and help determine the accuracy of the model.

In order to use the Score Node, we had to add first the data source with the role of the score. Then we made the following changes in the property panel in the decisions section: 1) we set to use default inverse weights and 2) we set decision weights to maximize, as shown in Figure 6.1.1.

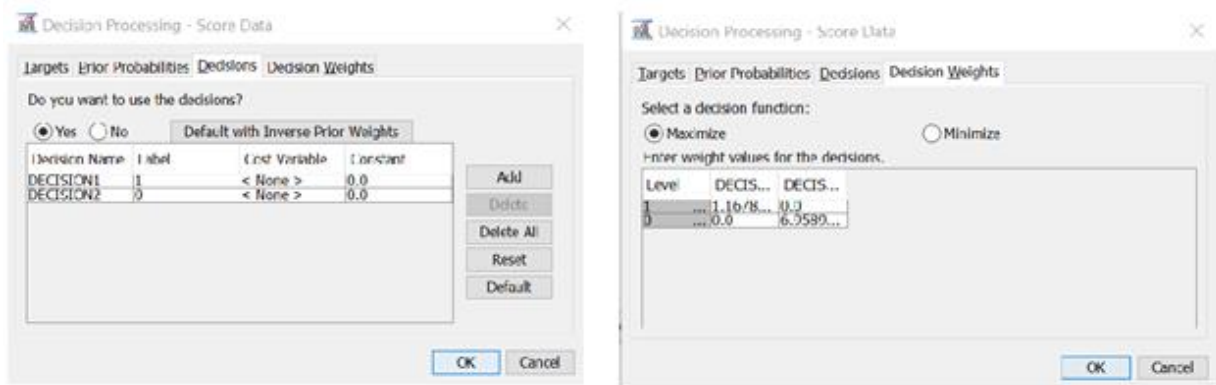
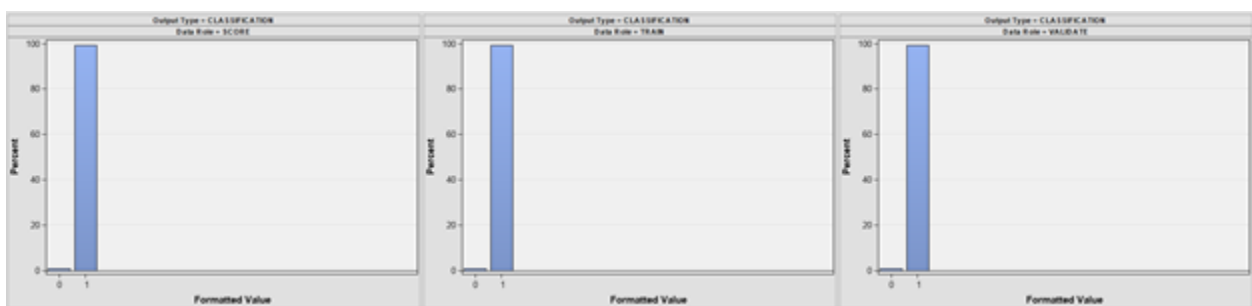


Figure 6.1.1. Decision panel for the Score data.

The results from the Score node showed that the predicted values are much closer to 1 than 0. Thus, we can be certain of the accuracy of the prediction. Also, the results were consistent for



the prediction with Score, Train, and Validation, as shown in Figure 6.1.2.

Figure 6.1.2. Predicted Values from the Score Node.

6.2. Variables of Importance

Based on the above-mentioned results, we selected the Optimal Decision Tree as our model to answer our first research question: What are the factors that affect attendance in Fenway Park? We obtained that the variables with higher effects were Ticket Spend with validation for attendance = 0 of 8.81% and attendance = 1 of 91.19%. Followed by Age level, Acc. Type Code, Temperature, Away Team, and Distance. The validation for each variable is summarized in Table 6.3.1. The whole Optimal Decision Tree model can be found in Appendix III.

Table 6.2.1. Summary statistics for the Variables of Importance for Optimal Decision Tree.

Variable	Validation		Main Levels
	Attendance=0	Attendance=1	
Ticket Spend	8.81%	91.19%	<35.5
Age Level	9.65%	90.35%	>=53
Acc. Type Code	19.87%	80.13%	SEABUS, Season, INTRL
Temperature	27.66%	72.34%	<59
Away Team	42.78%	52.22%	Rangers, Indians, Royals
Distance	57.85%	42.15%	1, 2

According to table 6.3.1, the biggest percentages of non-attendance can be found in age levels equal or higher to 53 years old. Also, account types like Seabus, Seasons, and International have the higher non-attendance. In terms of Away Team, the games with less attendance are those where Red Sox play against Rangers, Indians, and Royals. Lastly, people that live near the Boston and Massachusetts area are the ones that tend to miss more games with a validation percentage of 57.85%.

7. Findings and Conclusions

With all the analysis done, the first interesting find is that individuals who are less than 50 years of age attend more games on average than people who are above 50. The rationality for this could be that people who are less than 50 have younger kids whom they bring to ball games. These individuals want to pass on the joy they have for the sport to their kids with the hope they fall in love with the game. People who are above 50 attend the games less and they mostly buy seasonal

tickets. The rationality for this could be that people who are above 50 are more inclined to buy the whole season because it saves them money and they can afford it.

We found that when the temperature is above 60 degrees, people attend the games more. This is normal and makes sense as New England winter is known to be very harsh. Most people prefer to watch the games when the weather is better as baseball is a family event. Also, since a lot of people bring their kids to games, they want to avoid getting their kids sick. Additionally, when the temperature is bad and the game is unpopular, the attendance is poor. This can be seen in section 4 of our report. As the temperature drops, the likelihood of people attending is low. But, if the game is played with a rival like the Mets or Yankees, even though the temperature is poor the turnout percentage is higher.

Lastly, we found people to be more likely to attend games played in Fenway when they are out of state. There is a higher chance of people who live in Boston to miss games. We believe the reason for this is that people who bought tickets and are coming from the outer state are more emotionally invested in the games. Also, it could be the tourist effect. Fenway Park is a historical landmark. People who come to Boston make sure they visit the park. Also, those who come out of state are supporting their team and they want to ensure their presence is felt during games.

When brainstorming our recommendations, we wanted to create an action plan for Red Sox. We considered the situation currently happening. We aimed for our recommendation to be used during and post the current pandemic. Below are four recommendations that solve the problems presented to us by our client. Plus, they are supported by the analysis provided in this report.

7. 1. Virtual Ticket

The idea of the virtual ticket is to create a streaming platform that provides customers with an experience as if they were in the stadium. This ticket will be the same price as charged if they were coming into Fenway. The price will range from a basic to a diamond ticket. The more the customer pays the better the package. Customers who pay more will get exclusive items. For instance, VIP customers will be given snacks and drinks which are sold in the stadium. They will be delivered alongside a signed t-shirt. They get to pick the snacks and drinks to be delivered.

Those who purchase the virtual ticket get an exclusive viewing of the players practicing before the start of the game. We aim to use the virtual ticket to eliminate low ticket sales during low-temperature days. We believe it removes uncontrollable elements as customers get to watch the games in the comfort of their homes. The last incentive is to raffle home run game balls. For instance, if the Red Sox gets three home runs in a game, customers who purchase virtual tickets have a chance to win the ball. The ball will also be signed by the player who scores the home run.

7.2. Adjusting Ticket Price

As shown in the last visualization diagram in section 4, we noticed that some key games are not priced correctly. The example that will be used for this recommendation is looking focused on Dodgers and Mets when they are at Red sox. We noticed the percentage of people who do not attend when we play the Mets is 7.8% and for Dodgers is 8.4%. There is a 6% difference and the likelihood of them not attending when it comes to Mets is lower than Dodgers. But when we look at the average ticket price, we see that Dodgers is more expensive than the Mets. Dodgers tickets cost \$118 while Mets cost \$81. This is a \$37 difference which is significant. It currently looks like money is being left on the table which is not being utilized. We believe if the Mets game price is increased, this will boost revenue. Mets are a New York team and according to our geographical map provided in section 4, there are a lot of people who attend games played in Fenway Park that come from New York.

7.3. Transportation

According to our findings, we see that people who come outside of Boston purchase and attend more. We understand that there is a current incentive to reduce the number of people who drive into Fenway. We know the traffic in Kenmore square gets chaotic especially on game days. This is why we are recommending that a shuttle bus be provided around areas like Downtown Crossing and Back Bay train stations. We believe a partnership with local parking garages in this area will reduce the traffic in Kenmore square. Our vision is that there will be the discouragement of people to park in the Kenmore area by increasing the price for parking and blocking street parking except they live in Kenmore. Customers are encouraged to use the free shuttle that will be provided from the key train station to the park. Also, customers have the option of using the train. From living in Boston, we know that during game days the MBTA is free. This is used to encourage

people to walk to the stadium instead of parking their cars close to the stadium. Also, from the analysis done, we found that people who have four or more kids attend the game more. It can be stressful for parents to bring their kids to every game as they may be worried about their safety and this will reduce attendance over time.

7.4. Promotion

The theme of our analysis has been the game of baseball as a family event. From the description Professor Hom painted for us, we got to understand how the game has been an American tradition. MLB was founded in 1869 in Ohio which makes it the first sport invented on American soil. It is the oldest sport being played to date. It has been a game passed down from generation to generation. Fenway Park on its own is a legacy for the city of Boston with nine world series, and the stadium becoming a historical landmark. We believe messages like this should be communicated to the younger generation. We anticipate for them to have emotional connections to stories like this which will pick their interest and encourage them to attend games. The audience for this message will be people less than 30. We found that the chairman of Red Sox created initiatives to bring people more to the games by reducing the amount of a ticket for college students. We hope incentives like this can be communicated alongside the creation of emotional promotional stories.

8. Challenges

8.1. Understanding Data Definitions

Before performing any manner of analysis, We had to understand the client's information. Our client's data was not easy to analyze. For instance, if we took the scenario of determining the likelihood of a potential customer being approved for a credit card, the information we will use to determine their approval odds are given. It is easy to determine the outcome of customer approval odds. We know that a bad credit score will mean rejection and a strong credit score means approval for the credit card. In our case, we had to come up with a predictor target variable to help support or recommendations and findings. By doing this we got to understand the issues our client faced, and we began brainstorming ideas for a solution.

8.2. Complex Data Structure

With the unstructured data we had, after we were able to make sense of the data, the next step was to see how the tables related and what the proper usage of the tables will be. With the help of the Visio diagram provided by the client, we saw a link by using the primary and foreign keys of each table. Tables like the ticket scan and the ticket had multiple tables that were segmented to information for months and years. We had to first rationalize the best method of combining the information before merging it with the Axiom and account table. As mentioned in the data preparation section, we explored each table in a graphical format to get a better understanding of variables we did not need and those where filtration or more information needed to be provided. We also kept a close connection with Professor Hom to ensure better understanding. Our client was a great resource especially when we got confused and needed more clarity.

8.3. Merging Data

After we felt we understood the data better, we began merging the tables. This was a tedious process as we had a limited understanding of how the merge worked. In the beginning, we created a merge which turned out not to be accurate. There was a need to reject some variables and reject one of the primary keys in either the ticket or scan table. Both tables had an Eventcd which affected the structure of the data. Duplicates were created because the Ticket scan table contained extra information that was not in the ticket table. Knowing this, we focused on the information that was in the Ticket table and used the ticket Scan table to understand the Scan time of people coming to Fenway Park.

8.4. Data filtration

We then began filtering our data. As mentioned in section 3, there was a need to filter the data. Due to the merge, our final table had blank roles that were not needed. We use the filter node to delete blank pieces of information. The aim of this was to get data that was complete and error-free. We replaced some information that had errors in their spelling. This was a tedious task as we had to explore the final merge and then go into the replacement node to manually correct the information. We believed that by doing this our data was going to be collapsed and it would be accurate when it came to performing modeling.

8.5. SAS Coding

One of the new skills we learned from this project was SAS Coding. We had a brief understanding of how to use the SAS code node, but when performing our analysis, we knew it was going to be valuable. We created binary information and grouped information to help cluster and create a target variable. As mentioned in previous sections, we created attendance as our target variable after we understood the data better. This helped in getting a model we could trust and understand. The major challenge was learning how to structure the codes. We had to find the best variable to use to create the logic of our code.

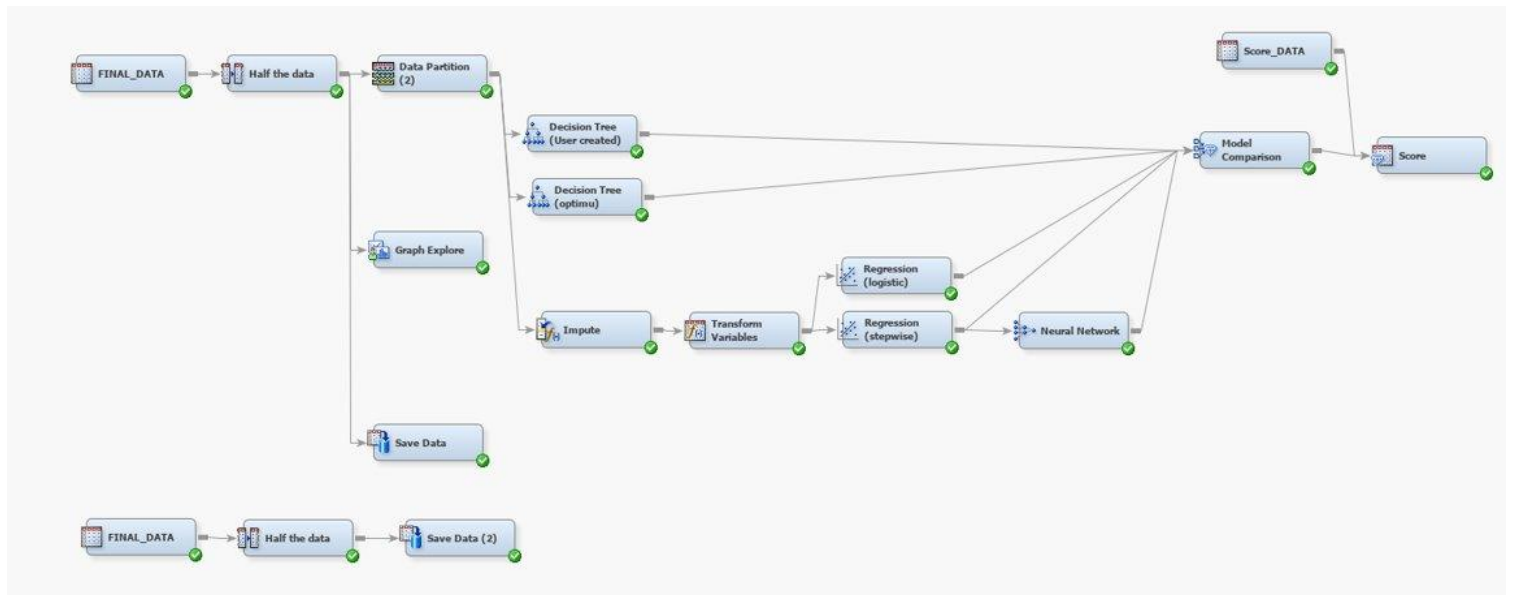
8.6. Data Modeling

Modeling was the last stage of our analysis. We had taken the time to make sure the data was correct. It then became time to run our models to answer our research question. One challenge we had was finding the best modeling tool for our analysis. We found the decision tree to be the best modeling tool. We began using the tree created by the decision tree node, but we soon found out the information it provides did not make much sense. We opted to create the decision tree by ignoring the highest log worth. By doing this, we created a tree that supported our research theory. Using a model comparison helped with comparing models and finding the best.

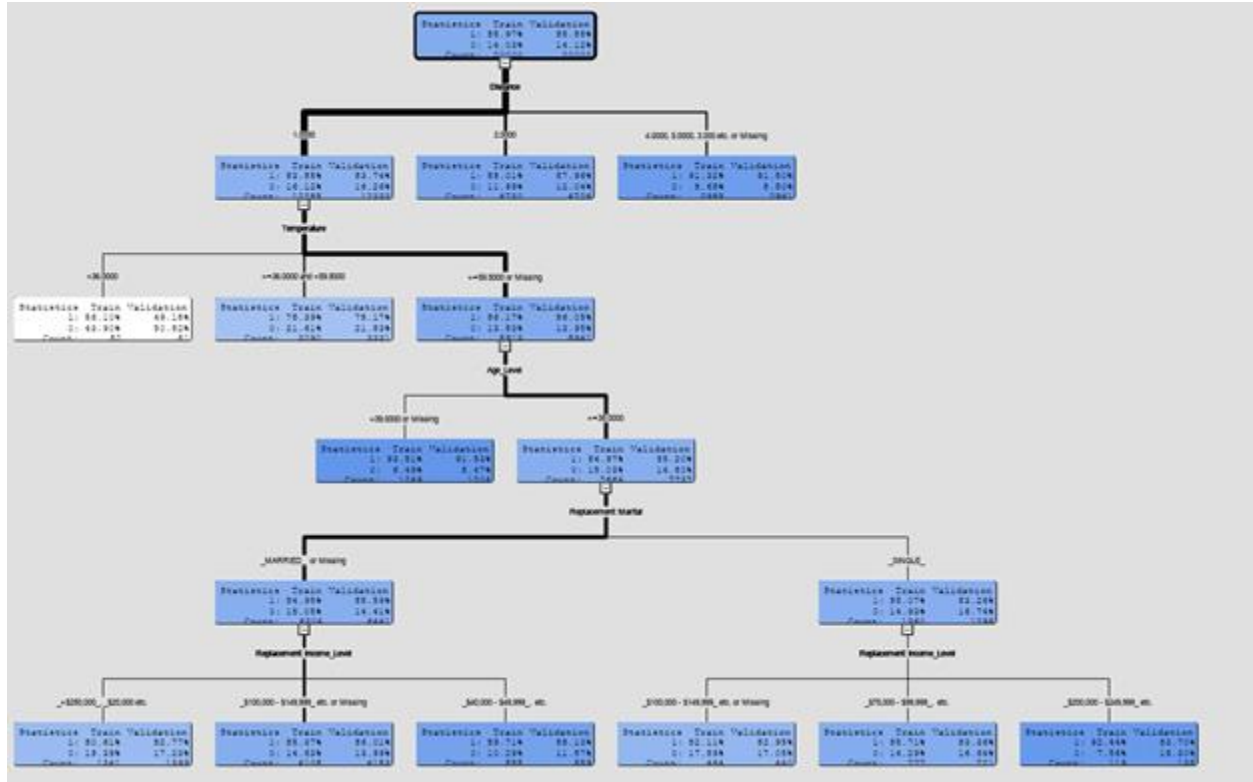
9. References

1. SAS (a). (2020). Sample Node. Retrieved December 13, 2020 from:
<https://documentation.sas.com/?docsetId=emref&docsetTarget=p15ebk1ysuwhqln11brhbopvmpj7.htm&docsetVersion=15.1&locale=en>
2. SAS (b). (2020). Decision Tree Node. Retrieved December 13, 2020 from:
<https://documentation.sas.com/?docsetId=emref&docsetTarget=n0cx4ud03paymdn1kargegadueml.htm&docsetVersion=15.1&locale=en>
3. Statistics Solutions (2020), Sampling. Retrieved December 12, 2020 from
<https://www.statisticssolutions.com/sample-size-calculation-and-sample-size-justification/sampling/>
4. SAS (c). (2020). Neural Network Model. Retrieved from
<https://documentation.sas.com/?cdcId=fcmrcdc&cdcVersion=14.2&docsetId=fcmrug&docsetTarget=n13kenqv6wmw1fn1jb5zr4kaepwu.htm&locale=en>
5. SAS (d). (2020). Model Comparison Node. Retrieved from
<https://documentation.sas.com/?docsetId=emref&docsetTarget=p01jgc9rmzsg37n1lfncp67t0unm.htm&docsetVersion=14.3&locale=en>
6. SAS, (e). (2020). Regression Node. Retrieved December 13, 2020, from
<https://documentation.sas.com/?docsetId=emref>

Appendix I. SAS Enterprise Model



Appendix II. User Created Maximal Tree



Appendix III. Optimal Tree

