Assignment 3: Support Vector Machines Using SAS Enterprise Miner

Daanish Ahmed

DATA 640 9041

Fall 2017

dsahmed2334@yahoo.com

Professor Sounak Chakraborty

UMUC

November 5, 2017

Introduction

Support Vector Machines (SVMs) are some of the newest predictive models developed for classification. These models involve using a hyperplane to separate the classes as best as possible while maximizing the margin between the closest data points of each class (Knode, 2016a). For datasets that are not linearly separable, a kernel function is used to transform the data into a higher dimensional space to allow for accurate classification (Kane, 2015). These models function best on binary targets, and some of the most useful kernels include linear, polynomial, radial basis function, and sigmoid. SVMs work well on datasets with many inputs and generalize well to new cases, but they are less effective when using large datasets and can be computationally demanding (Knode, 2016a). In this assignment, I will build five SVM models using SAS Enterprise Miner on a dataset containing vehicular accident information. These models will classify accidents based on whether any fatalities occurred. My goal is to accurately identify cases of fatal accidents to help prevent deaths in the future. I intend to build one model for each of the four kernel functions mentioned. I will also use appropriate variable transformations for these models to minimize their misclassification rates. To explore the impact of preprocessing, I will create another linear model that does not use any data preparation nodes (see Figure 1 in Appendix). Finally, I will compare the models to see which one is the most effective at classifying accident fatalities.

This dataset contains 42,183 observations and 24 variables—all of which are numeric. The variable descriptions can be found in the attached data file "Accidents.xls." The target variable is fatalities, which is a binary variable with the values of 0 (no fatalities) and 1 (fatalities). Some of the inputs include speed limit, work zone, interstate highway, pedestrians involved, surface conditions, vehicles involved, and number of injuries (see Figure 3 in Appendix). The dataset contains 3 interval variables, 7 nominal variables, and 14 binary variables. The interval variables

include the number of injuries, speed limit, and vehicles involved. Nominal variables include maximum injury severity, traffic conditions, and surface conditions. Binary variables include weather conditions, pedestrians involved, and work zone. According to the descriptive statistics (see Figure 3 in Appendix), there are no missing values in any of the variables.

Figure 3 also reveals that some of the variables are skewed. The interval inputs with the highest skewness are number of injuries (2.74) and vehicles involved (1.87). Examining the dataset reveals that the target variable is skewed as well, since there are only 466 accidents that involved fatalities. Furthermore, some of the inputs have outliers. For instance, number of injuries and vehicles involved have values more than three standard deviations away from the mean. By examining the data (see Figure 2 in Appendix), I also found that "interstate highway" contains errors. This variable should only have the values 0 (no interstate) and 1 (interstate). However, a few observations have a value of 9. These issues will be resolved during data preparation. Finally, I found that the target variable (fatalities) is strongly correlated with maximum injury severity. All cases with a maximum severity of 2 (fatal) also have a fatality value of 1 (fatalities occurred), and vice versa. To solve this issue, I set the maximum injury severity to "rejected."

Data Preparation

This dataset was originally in .xls format, so I imported it into SAS Enterprise Miner using the File Import node and converted it into a .sas7bdat file using the Save Data node. I then used the StatExplore node to create a bar graph showing each input variable's worth (see Figure 4 in Appendix). According to this image, the most significant variables are the number of injuries, injury crash, property damage crash, pedestrians or cyclists involved, and head-on collisions.

Likewise, the least important variables are vehicles towed, work zone, traffic way, and whether the accident was on a weekday. One advantage of using SVMs over other classification models is that SVMs can handle a much higher number of input variables (Ray, 2017). Because of this, I will not focus on variable selection like I had done for my regression models.

The next step is to handle missing values, outliers, and errors in the data. As mentioned earlier, there are no missing values in my dataset. However, variables such as "number of injuries" and "vehicles involved" contain values more than three standard deviations away from the mean (see Figure 3 in Appendix). To address this, I used the Replacement node to replace all outliers. I set the interval variable default limits method to "standard deviations from the mean" and used the default cutoff value of 3.0 to replace all values outside of this range. I also handled inputs with errors using the Replacement node. I set the class variable unknown levels replacement to "mode," and I used the replacement editor to find variables with incorrect entries. I found that only "interstate highway" contained errors, so I set the error's replacement value to "_UNKNOWN_" to replace these incorrect entries with the mode.

Next, I sampled the dataset using the Sample node. Data sampling is useful for SVM models since it reduces the amount of data and therefore lowers the training time. Furthermore, using a weighted sample will reduce the extreme skewness of my target variable by balancing the number of 0's and 1's (Knode, 2016b). I created weighted samples by using the "level based" sampling criterion and focusing on the rarest level to increase the proportion of fatal accidents. I set the rarest level to take up 40% of all cases, which produced a more balanced sample with 1165 observations and 466 fatal accidents. This helps the SVM algorithm to identify true positives more easily. I will apply this sampling method to my first four models using a different random seed for each model. Doing this will ensure that each model contains different cases but with the same

proportion of accident cases. This will eliminate any bias that would occur from using the same data in every model. One problem with data sampling is that it greatly reduces the amount of data used in the models, making the models less representative of the entire dataset. To address this issue, I will create a linear model that excludes all sampling and preprocessing steps aside from data partition. I will compare it to my other models to see how sampling affects accuracy. I will also use this model to determine if preprocessing improved my SVM's accuracy.

From here, I used the Data Partition node to designate 50% of the data to the training set, 30% to the validation set, and 20% to the test set. Each model uses the same percentages for data partition, but with different random seeds to prevent them from having identical samples. Next, I implemented two types of transformations on my models: "best" transformation and maximum normal. According to SAS Enterprise Miner Reference Help, both methods involve using multiple transformation types on each input variable. However, "best" transform selects the transformation with the strongest impact on the target, while max normal selects the method that maximizes each variable's normality and minimizes skewness. For the linear and polynomial models, I used the "best" transformation for interval inputs. For the radial basis function and sigmoid models, I selected the "maximum normal" interval input. In both cases, I used "dummy indicators" for class inputs to handle variables with spiked distributions—such as number of injuries (see Figure 5 in Appendix). I will compare these models to see how transformations affect accuracy.

Model Development

Now that data preparation is finished, I will construct my five SVM models. The first four models each involve one of the following kernel functions: linear, polynomial, radial basis

function, and sigmoid. According to SAS Enterprise Miner's reference help page, the linear kernel performs the transformation $K(u,v) = u^Tv$, while the polynomial kernel uses the function $K(u,v) = (u^Tv + 1)^p$ where p is the polynomial order. The help page describes the radial basis function as a Gaussian function K(u,v) = exp[-p (u - v)2] where p is the user-specified kernel scale parameter. Finally, the source describes the sigmoid kernel as the transformation $K(u,v) = \tanh(p*(u^Tv) + q)$ with a kernel scale parameter p and a kernel location parameter q.

Each of my models was built using the HP SVM node. The linear model used "interior point" as its optimization method, with its kernel type set to "linear." The next three models used the "active set" optimization method, with their kernels set to "polynomial," "radial basis function," and "sigmoid." I experimented with different active set parameters to maximize each model's accuracy. Due to the results, I left the polynomial model with its default degree of 2 and gave the radial basis function model a parameter of 2.0. For the sigmoid model, I kept the default parameters because the results did not improve by changing them. My fifth model involves a linear kernel with the same properties as my previous linear model. However, it does not have any preprocessing except for a Data Partition node, which uses the same percentages as the other models but with a unique random seed. As stated earlier, this model will check whether the results are improved using sampling, transformations, and preprocessing. One key advantage of this model is that it retains all 42,183 observations—making it a better representation of my dataset when compared to the weighted samples that each contain only 1165 instances.

I will now describe each of the important results in this analysis. The lift curve shows the cumulative lift for the training and validation data (see Figure 6 in Appendix). An ideal model will have training and validation curves that match almost perfectly. In the fit statistics window, the most important measures are the misclassification rate and the number of wrong classifications

(see Figure 7 in Appendix). These figures reflect the model's overall accuracy, and an optimal model will have very low values for these two measures. Other useful measures can be found in the classification matrix and fit statistics tables (see Figure 8 in Appendix). The classification matrix shows a breakdown of the predicted values compared to the actual values. In an ideal model, most of the predictions (both 1's and 0's) will match the observed values. The fit statistics table displays the accuracy, error rate, sensitivity (number of true positives), and specificity (number of true negatives). Having a high accuracy and low error is important, but we also want to identify the highest number of fatal accidents as possible. Thus, sensitivity will be given more importance than specificity in this analysis. In the results section of this paper, I will evaluate the pros and cons of each model by assessing the statistics mentioned. Unfortunately, the SVM output does not provide any information about which variables are the most effective predictors. SVM is a black box algorithm that is difficult to interpret (Kane, 2015), which is a disadvantage when compared to models such as regression that are easier to understand.

Results

I will now explore the results of my analysis to determine which model is the most effective at classifying fatal accidents. First, I will analyze the output of my first SVM with a linear kernel. According to the cumulative lift curve, the training lift matches the validation lift almost perfectly (see Figure 6 in Appendix). This eliminates the possibility of overfitting and suggests that the training and validation results are extremely close. By examining the fit statistics, we find that the misclassification rate and number of wrong classifications are 0 for the training, validation, and test sets (see Figure 7 in Appendix). This model therefore has a 100% accuracy rate on the sample data. Furthermore, the classification matrix shows that the linear kernel accurately classified every

instance in the training and validation sets (see Figure 8 in Appendix). This figure also shows that the accuracy, sensitivity, and specificity are 100% while the error is 0%. Altogether, the linear model appears perfect not only in terms of overall accuracy, but also in its ability to identify fatal accidents. These results suggest that the sample data used for this model is linearly separable. However, this does not confirm whether the entire dataset is linearly separable.

Next, I examined the results from my polynomial model with a degree of 2. This model's cumulative lift curve appears roughly identical to that of my first model. Its misclassification rate and number of wrong predictions are also 0 for the training, validation, and test sets (see Figure 9 in Appendix). Likewise, its classification matrix and fit statistics tables show that the model has perfect classification accuracy and has successfully identified all true positives and negatives without any errors (see Figure 10 in Appendix). Though this model has all the advantages of the linear model, I would currently recommend using the linear model because it is the simplest to use computationally. These first two models both use the "best" transformation—and thus it is possible that the transformation type significantly affects the accuracy. I will check if this is true by examining the next two models, which use the "max normal" transformation.

I now look at the output from my radial basis function SVM. While experimenting with different parameters, I found that using the default RBF parameter of 1.0 produces a very weak model on the validation data. Its accuracy decreases from 100% in the training set to 80.6% in the validation set, while the number of true positives decreases from 100% to 51.4% (see Figure 13 in Appendix). As such, this model experiences overfitting when using a parameter near 1.0. However, I found that using a parameter of 2.0 will produce a model with the same accuracy as my first two models. This model now has a lift curve that is identical to those of the previous models, and its misclassification rate and number of wrong classifications are 0 for the training,

validation, and test sets (see Figure 11 in Appendix). Additionally, the classification matrix and fit statistics show that the model contains no errors and accurately classified all 1's and 0's in the sample data (see Figure 12 in Appendix). These results suggest that changing the parameter has a much greater impact on the accuracy than choosing a different transformation type. Since all of my models use a different random seed, it is possible that my first three models will perform well on the entire dataset due to their perfect accuracy on random samples.

By analyzing the sigmoid model, I found that its results differ drastically from the first three models. The cumulative lift curve has a different shape than those of the previous models, and its validation lift is consistently lower than the training lift (see Figure 16 in Appendix). The fit statistics reveal that this model is the most inaccurate by far—having a misclassification rate of 39.9% for the training data, 40.1% for the validation data, and 40.2% for the test data (see Figure 14 in Appendix). The classification matrix shows that the model failed to classify any values as 1's—instead it classified all values as 0's (see Figure 15 in Appendix). This figure shows that the model has an accuracy of 60.1% on the training set and 59.9% on the validation set. Although the model did not identify any false positives, it did identify hundreds of false negatives since every value was classified as a 0. As mentioned earlier, experimenting with different parameters did little to improve the results of this model. Because of these findings, I would not recommend using a sigmoid kernel for this dataset. The sigmoid kernel is not a good fit for separating the data in the current sample, and it seems unlikely to be effective when using the entire dataset.

Finally, I will address the results of my linear model that did not use any data preparation. This model's cumulative lift curve has a different shape from the previous graphs, but its training and validation curves match almost perfectly (see Figure 19 in Appendix). This suggests that the training and validation results are relatively close. The fit statistics indicate that the model

misclassified 2 cases in the training set, 5 in the validation set, and 3 in the test set (see Figure 17 in Appendix). But since each partition contains around 10,000 to 20,000 cases, the overall misclassification rate is extremely small. The classification matrix and fit statistics tables reveal that the model identified all true negatives and has an accuracy of 99.99% for the training set and 99.96% for the validation set (see Figure 18 in Appendix). The model identified 99.1% of true positives in the training set and 96.4% in the validation set. These results are very impressive, especially since the target is extremely skewed in the original dataset. This model did not use any sampling, preprocessing, or transformations. And while the results may not be 100% perfect, they are still extremely close. Performing data preparation and sampling on a linear model will eliminate errors, but using the unmodified dataset still yields highly accurate results. Due to the linear kernel's effectiveness, it seems that the entire dataset is almost linearly separable. But since its accuracy is less than 100%, it does contain a very small amount of noise.

Conclusion

In this analysis, I built five SVM models to classify accidents based on whether fatalities occurred. I experimented with different kernels, parameters, samples, and transformations to create the most accurate model possible. Through this analysis, I found that using different transformations does not affect the accuracy as much as changing the kernel's parameters. The most accurate models were built using linear, polynomial, and radial basis function kernels—with the latter two models both using parameter values of 2. The sigmoid kernel was the only kernel which failed to classify my target regardless of the parameters used. This model classified every value as a 0, even though it used a balanced data sample. As a result, the sigmoid kernel is not effective at transforming the values in my dataset, and I would not recommend using it. Of the

three most accurate kernels, I would recommend using the linear kernel. Although all three kernels can have perfect accuracy with the right model tuning, the linear kernel is the easiest to use computationally and does not require any parameter configuration.

My last model excluded all preprocessing and sampling nodes, and it was used to determine if data preparation improves the linear model's accuracy. Based on my findings, preprocessing does indeed eliminate errors and allow the model to achieve perfect accuracy. However, it is likely that sampling contributed more heavily than any other preprocessing step. By using the entire unaltered dataset, the linear model only misclassified 10 accidents out of 42,183—and thus any random sample of 1100 cases is likely to contain linearly separable data. Future analysis could involve using preprocessing steps without sampling to see if these steps have any impact on accuracy. My results indicate that the selected dataset is almost linearly separable, but it does contain a very small amount of noise. Overall, I would recommend using the sampled linear model instead of the non-sampled model. This sampled model is more accurate, and it is not always necessary to use the entire dataset to get a good understanding of the data. I may also suggest using larger sample sizes to see if the models will still have 100% accuracy.

The biggest shortcoming of this analysis is the black box nature of SVM models, which prevents us from determining which input variables are the most significant. One solution is to use different models such as logistic regression to find important predictors and their weights. These models can be compared to my SVMs to see which algorithm is better for my dataset. Another topic for future analysis would be to create polynomial, radial basis function, and sigmoid models without sampling to see how these kernels perform on the entire dataset. It is likely that one of these kernels can handle the noise in the data better than the linear kernel. If so, then I would recommend using that model instead to classify future accidents.

References

Kane, D. (Performer) (2015, January 26). *Data science part IX: Support Vector Machine* [Web]. Retrieved October 24, 2017, from https://www.youtube.com/watch?v=fMWjhQ2UcNs

Knode, S. (2016a, August 25). *Support Vector Machine Models*. Lecture presented at UMUC. Retrieved October 3, 2017.

Knode, S. (2016b, October 11). *Adjusting for Skewed Target Distribution*. Lecture presented at UMUC. Retrieved October 14, 2017.

Knode, S. (2016c, October 27). *SVM with SAS Enterprise Miner*. Lecture presented at UMUC. Retrieved October 3, 2017.

Ray, S. (2017, September 13). Understanding Support Vector Machine algorithm from examples (along with code). Retrieved October 23, 2017, from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Appendix

Relevant SAS Enterprise Miner Output Images

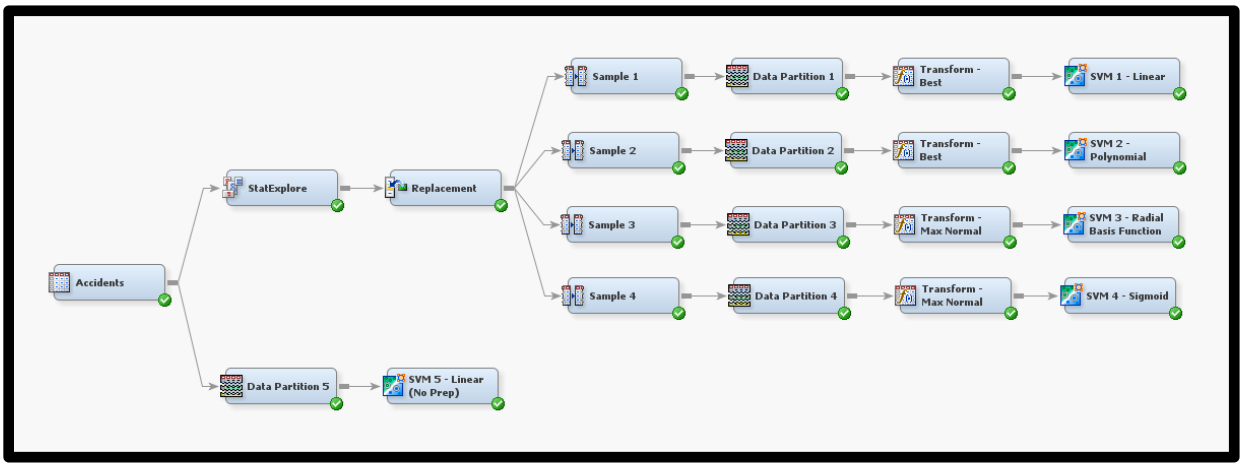

Figure 1. Process Flow Diagram for Accidents SVM Models.

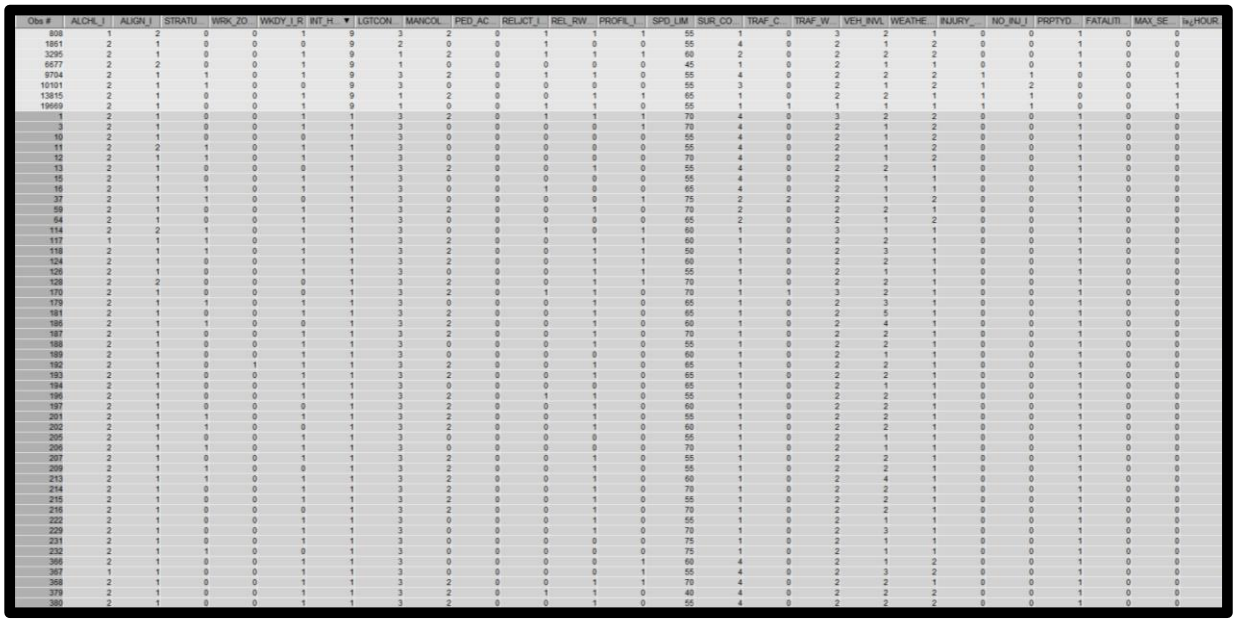

Figure 2. Accidents Dataset with Errors in "INT_HWY" Variable.

Name	Role	Level	Type	Number of Levels	Percent Missing	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
ALCHL_I	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
ALIGN_I	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
FATALITIES	**Target**	**Binary**	Numeric	2	0	.	.	.	.	.	.
INJURY_CRASH	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
INT_HWY	**Input**	**Nominal**	Numeric	3	0	.	.	.	.	.	.
LGTCON_I_R	**Input**	**Nominal**	Numeric	3	0	.	.	.	.	.	.
MANCOL_I_R	**Input**	**Nominal**	Numeric	3	0	.	.	.	.	.	.
MAX_SEV_IR	**Input**	**Nominal**	Numeric	3	0	.	.	.	.	.	.
NO_INJ_I	**Input**	**Interval**	Numeric	.	0	0	31	0.778702	1.035169	2.736153	27.19016
PED_ACC_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
PROFIL_I_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
PRPTYDMG_CRASH	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
RELJCT_I_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
REL_RWY_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
SPD_LIM	**Input**	**Interval**	Numeric	.	0	5	75	43.54787	12.9484	0.327279	-0.71394
STRATUM_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
SUR_COND	**Input**	**Nominal**	Numeric	5	0	.	.	.	.	.	.
TRAF_CON_R	**Input**	**Nominal**	Numeric	3	0	.	.	.	.	.	.
TRAF_WAY	**Input**	**Nominal**	Numeric	3	0	.	.	.	.	.	.
VAR1	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
VEH_INVL	**Input**	**Interval**	Numeric	.	0	1	23	1.816964	0.684843	1.865724	27.16742
WEATHER_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
WKDY_I_R	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.
WRK_ZONE	**Input**	**Binary**	Numeric	2	0	.	.	.	.	.	.

Figure 3. Descriptive Statistics of Accidents Dataset.

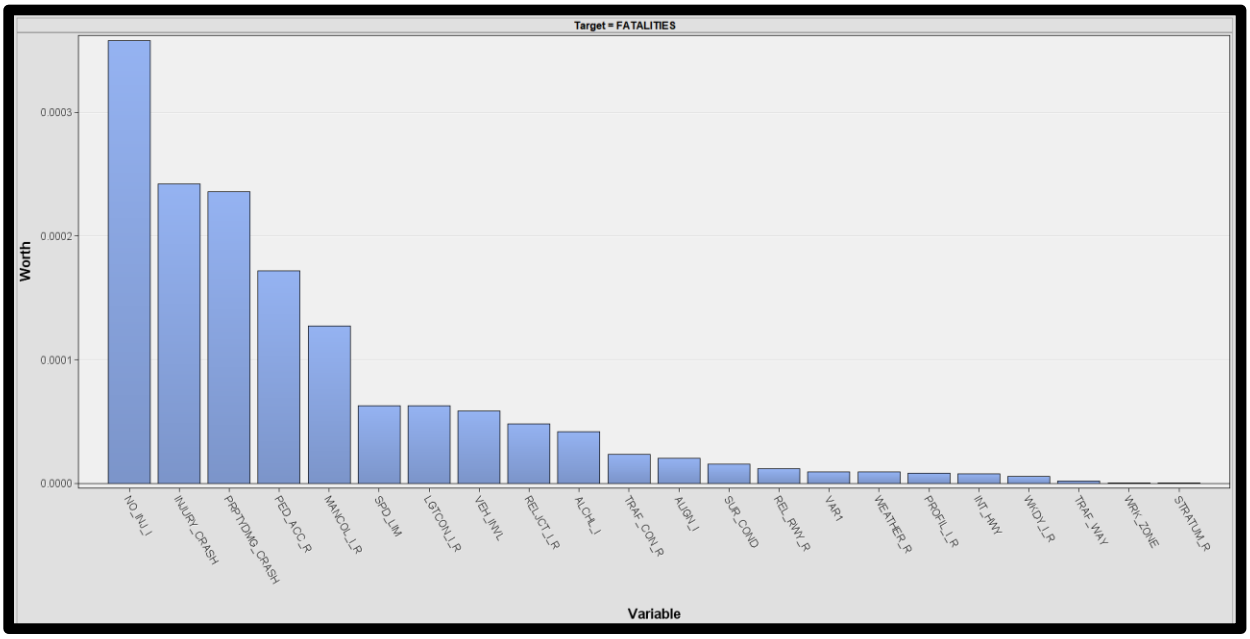

Figure 4. StatExplore Plot of Input Variable Worth.

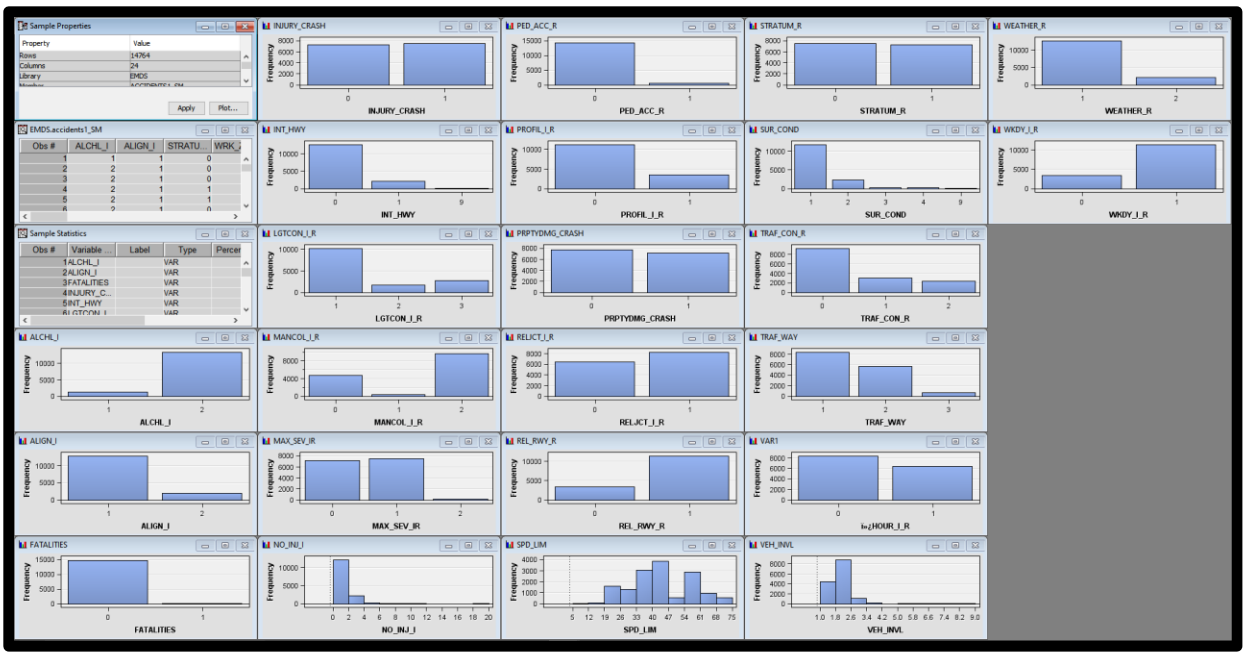

Figure 5. Initial Distributions of Variables in Accidents Dataset.

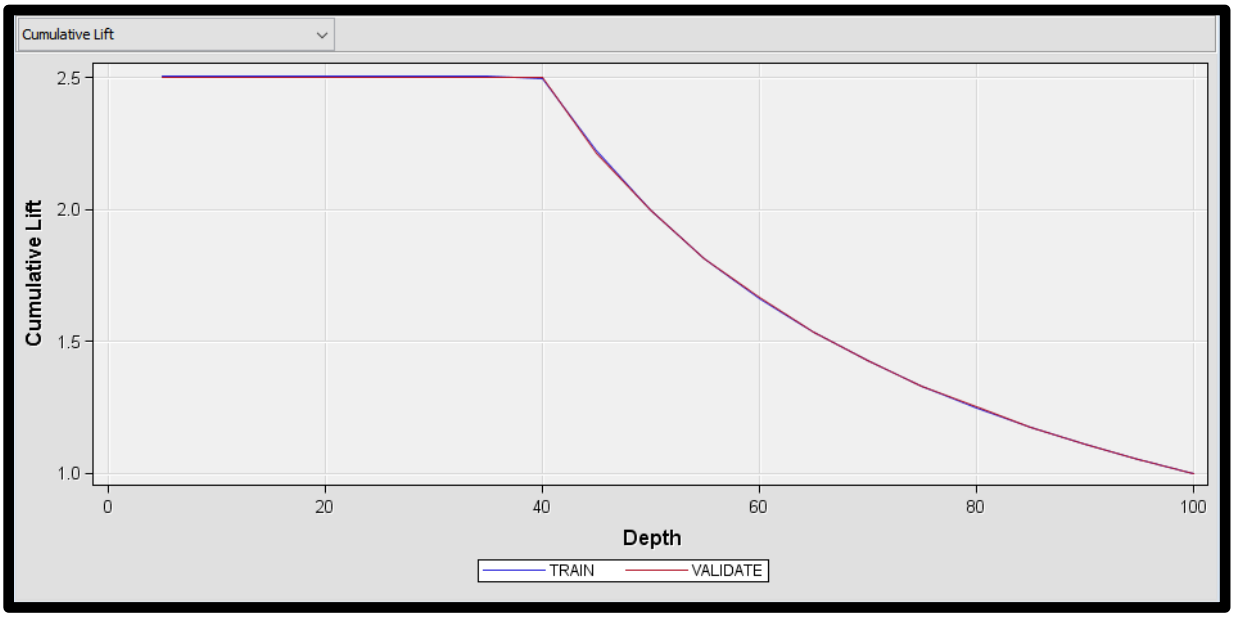

Figure 6. Cumulative Lift for Linear Kernel SVM Model.

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
REP_FATALITIES	Replacement: FATAL...	_ASE_	Average Squared Error	4.391E-6	4.388E-6	4.402E-6
REP_FATALITIES	Replacement: FATAL...	_DIV_	Divisor for ASE	1164	700	466
REP_FATALITIES	Replacement: FATAL...	_MAX_	Maximum Absolute Error	0.002534	0.003801	0.002529
REP_FATALITIES	Replacement: FATAL...	_NOBS_	Sum of Frequencies	582	350	233
REP_FATALITIES	Replacement: FATAL...	_RASE_	Root Average Squared Error	0.002095	0.002095	0.002098
REP_FATALITIES	Replacement: FATAL...	_SSE_	Sum of Squared Errors	0.005111	0.003072	0.002052
REP_FATALITIES	Replacement: FATAL...	_DISF_	Frequency of Classified Cases	582	350	233
REP_FATALITIES	Replacement: FATAL...	_MISC_	Misclassification Rate	0	0	0
REP_FATALITIES	Replacement: FATAL...	_WRONG_	Number of Wrong Classifications	0	0	0

Figure 7. Fit Statistics for Linear Kernel SVM Model.

Classification Matrix

	Training Prediction			Validation Prediction		
Observed	1	0	Total	1	0	Total
1	232	0	232	140	0	140
0	0	350	350	0	210	210
Total	232	350	582	140	210	350

Fit Statistics

Statistic	Training	Validation
Accuracy	1.0000	1.0000
Error	0.0000	0.0000
Sensitivity	1.0000	1.0000
Specificity	1.0000	1.0000

Figure 8. Classification Matrix and More Fit Statistics for Linear Model.

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
REP_FATALITIES	Replacement: FATALI...	_ASE_	Average Squared Error	0.002035	0.00243	0.00246
REP_FATALITIES	Replacement: FATALI...	_DIV_	Divisor for ASE	1166	698	466
REP_FATALITIES	Replacement: FATALI...	_MAX_	Maximum Absolute Err...	0.065859	0.116448	0.113472
REP_FATALITIES	Replacement: FATALI...	_NOBS_	Sum of Frequencies	583	349	233
REP_FATALITIES	Replacement: FATALI...	_RASE_	Root Average Squared...	0.045106	0.049294	0.049603
REP_FATALITIES	Replacement: FATALI...	_SSE_	Sum of Squared Errors	2.372281	1.696051	1.146578
REP_FATALITIES	Replacement: FATALI...	_DISF_	Frequency of Classifie...	583	349	233
REP_FATALITIES	Replacement: FATALI...	_MISC_	Misclassification Rate	0	0	0
REP_FATALITIES	Replacement: FATALI...	_WRONG_	Number of Wrong Cla...	0	0	0

Figure 9. Fit Statistics for Polynomial Kernel SVM Model.

Classification Matrix

	Training Prediction			Validation Prediction		
Observed	1	0	Total	1	0	Total
1	233	0	233	140	0	140
0	0	350	350	0	209	209
Total	233	350	583	140	209	349

Fit Statistics

Statistic	Training	Validation
Accuracy	1.0000	1.0000
Error	0.0000	0.0000
Sensitivity	1.0000	1.0000
Specificity	1.0000	1.0000

Figure 10. Classification Matrix and More Fit Statistics for Polynomial Model.

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
REP_FATALITIES	Replacement: FATALI...	_ASE_	Average Squared Error	0.010462	0.030094	0.03082
REP_FATALITIES	Replacement: FATALI...	_DIV_	Divisor for ASE	1164	700	466
REP_FATALITIES	Replacement: FATALI...	_MAX_	Maximum Absolute Err...	0.123223	0.472351	0.367051
REP_FATALITIES	Replacement: FATALI...	_NOBS_	Sum of Frequencies	582	350	233
REP_FATALITIES	Replacement: FATALI...	_RASE_	Root Average Squared...	0.102282	0.173477	0.175556
REP_FATALITIES	Replacement: FATALI...	_SSE_	Sum of Squared Errors	12.17738	21.06603	14.36214
REP_FATALITIES	Replacement: FATALI...	_DISF_	Frequency of Classifie...	582	350	233
REP_FATALITIES	Replacement: FATALI...	_MISC_	Misclassification Rate	0	0	0
REP_FATALITIES	Replacement: FATALI...	_WRONG_	Number of Wrong Cla...	0	0	0

Figure 11. Fit Statistics for Radial Basis Function SVM Model with RBF Parameter 2.0.

Classification Matrix

	Training Prediction			Validation Prediction		
Observed	1	0	Total	1	0	Total
1	232	0	232	140	0	140
0	0	350	350	0	210	210
Total	232	350	582	140	210	350

Fit Statistics

Statistic	Training	Validation
Accuracy	1.0000	1.0000
Error	0.0000	0.0000
Sensitivity	1.0000	1.0000
Specificity	1.0000	1.0000

Figure 12. Classification Matrix and More Fit Statistics for RBF Model with Parameter 2.0.

Fit Statistics

Statistic	Training	Validation
Accuracy	1.0000	0.8057
Error	0.0000	0.1943
Sensitivity	1.0000	0.5143
Specificity	1.0000	1.0000

Figure 13. Fit Statistics for Radial Basis Function SVM Model with RBF Parameter 1.0.

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
REP_FATALITIES	Replacement: FATALI...	_ASE_	Average Squared Error	0.398625	0.401146	0.401709
REP_FATALITIES	Replacement: FATALI...	_DIV_	Divisor for ASE	1164	698	468
REP_FATALITIES	Replacement: FATALI...	_MAX_	Maximum Absolute Err...	1	1	1
REP_FATALITIES	Replacement: FATALI...	_NOBS_	Sum of Frequencies	582	349	234
REP_FATALITIES	Replacement: FATALI...	_RASE_	Root Average Squared...	0.631368	0.633361	0.633805
REP_FATALITIES	Replacement: FATALI...	_SSE_	Sum of Squared Errors	464	280	188
REP_FATALITIES	Replacement: FATALI...	_DISF_	Frequency of Classifie...	582	349	234
REP_FATALITIES	Replacement: FATALI...	_MISC_	Misclassification Rate	0.398625	0.401146	0.401709
REP_FATALITIES	Replacement: FATALI...	_WRONG_	Number of Wrong Cla...	232	140	94

Figure 14. Fit Statistics for Sigmoid Kernel SVM Model.

Classification Matrix

	Training Prediction			Validation Prediction		
Observed	1	0	Total	1	0	Total
1	0	232	232	0	140	140
0	0	350	350	0	209	209
Total	0	582	582	0	349	349

Fit Statistics

Statistic	Training	Validation
Accuracy	0.6014	0.5989
Error	0.3986	0.4011
Sensitivity	0.0000	0.0000
Specificity	1.0000	1.0000

Figure 15. Classification Matrix and More Fit Statistics for Sigmoid Model.

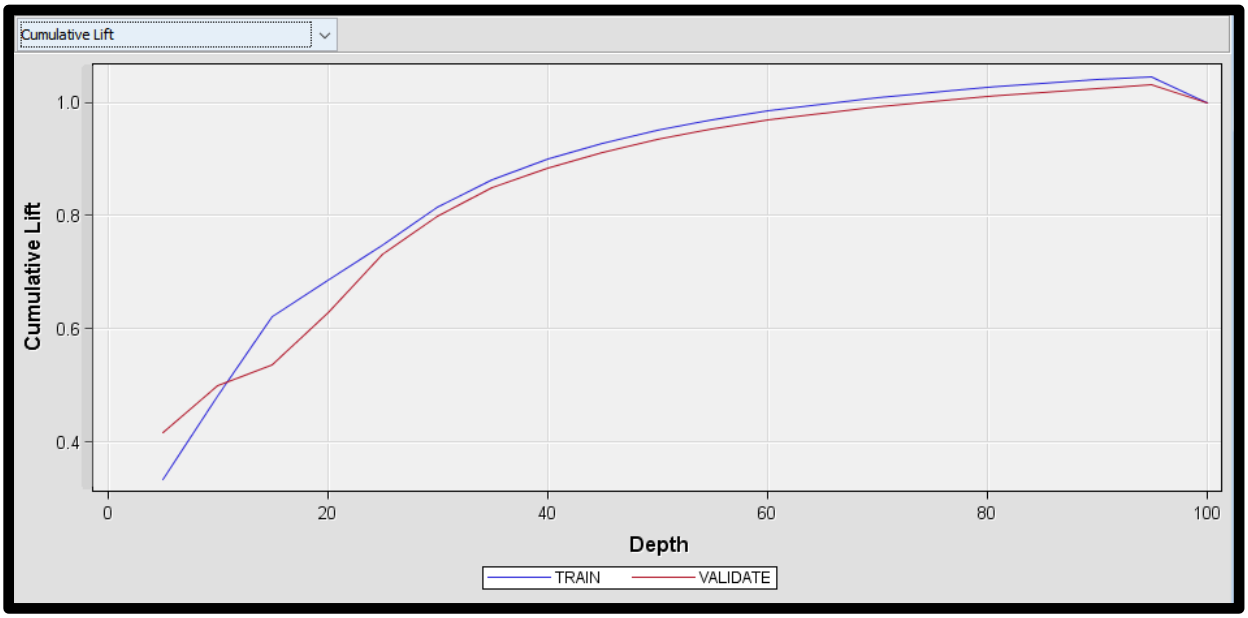

Figure 16. Cumulative Lift for Sigmoid Kernel SVM Model.

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
FATALITIES		_ASE_	Average Squared Error	0.003468	0.003415	0.003454
FATALITIES		_DIV_	Divisor for ASE	42182	25308	16876
FATALITIES		_MAX_	Maximum Absolute Err...	0.503195	0.511526	0.50958
FATALITIES		_NOBS_	Sum of Frequencies	21091	12654	8438
FATALITIES		_RASE_	Root Average Squared...	0.058887	0.058434	0.058773
FATALITIES		_SSE_	Sum of Squared Errors	146.2748	86.41509	58.29481
FATALITIES		_DISF_	Frequency of Classifie...	21091	12654	8438
FATALITIES		_MISC_	Misclassification Rate	9.483E-5	.0003951	.0003555
FATALITIES		_WRONG_	Number of Wrong Cla...	2	5	3

Figure 17. Fit Statistics for Linear Kernel SVM Model with no Preprocessing.

Classification Matrix

	Training Prediction			Validation Prediction		
Observed	1	0	Total	1	0	Total
1	231	2	233	134	5	139
0	0	20858	20858	0	12515	12515
Total	231	20860	21091	134	12520	12654

Fit Statistics

Statistic	Training	Validation
Accuracy	0.9999	0.9996
Error	0.0001	0.0004
Sensitivity	0.9914	0.9640
Specificity	1.0000	1.0000

Figure 18. Classification Matrix and More Fit Statistics for Model with no Preprocessing.

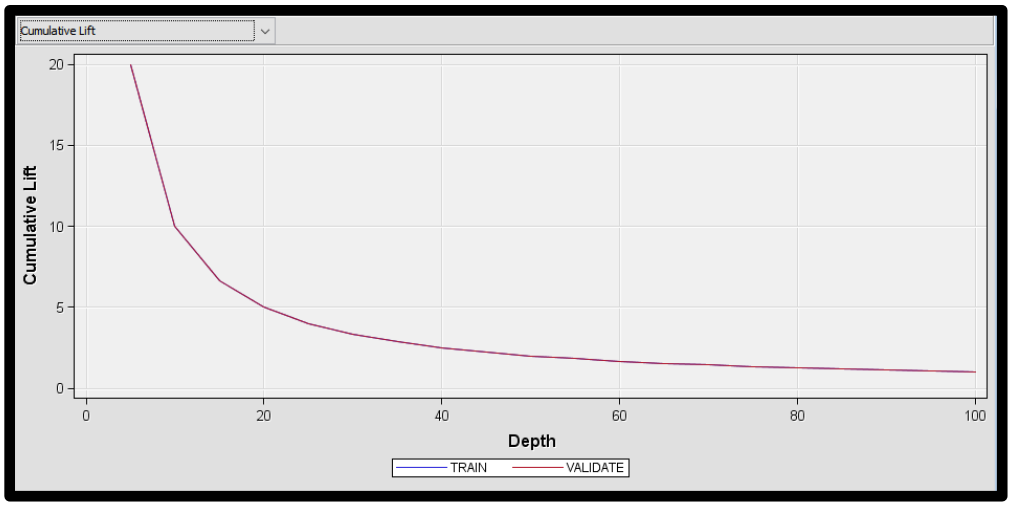

Figure 19. Cumulative Lift for Linear Kernel SVM Model with no Preprocessing.