BA 706 - Applied Analytic Modelling

Predicting Bank Application Fraud

Group 5

Rakeen Ahmed - 301307050

Faiza Zahin - 301318801

Raghav Gupta - 301272406

Drashti Lakhani - 301241918

Table of Contents

Introduction and Objective

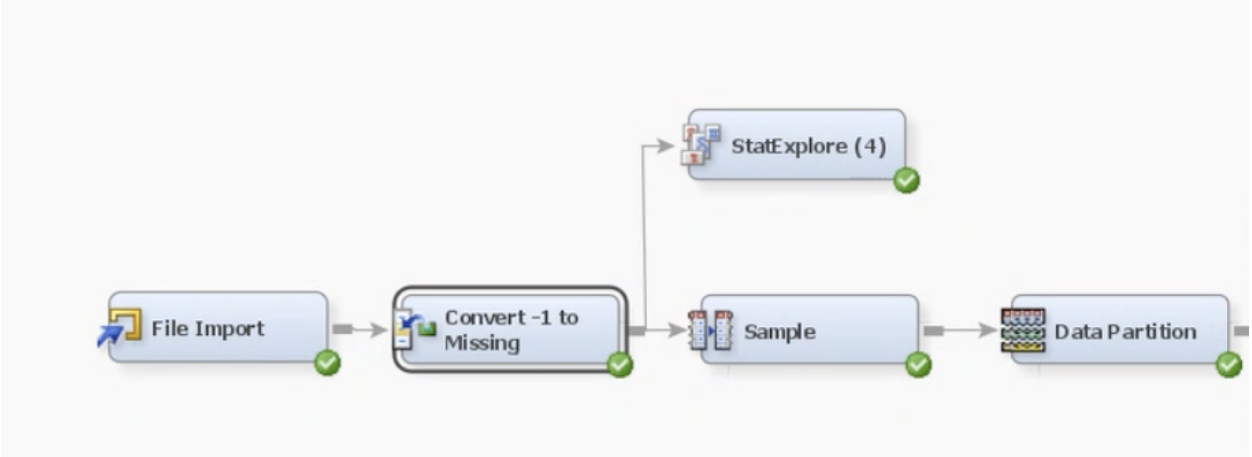
New Account Fraud is a major problem in the banking industry, and is one of the most common types of bank account fraud, accounting for 23% of all bank account frauds. It involves the creation of a new bank account using false or stolen personal information by the fraudster, which is then onboarded by the bank as a legitimate account. The account can then be used for various fraudulent activities such as money laundering, illegal transactions, credit card fraud, etc. For our project, the dataset obtained from Kaggle.com contains 1 million instances of synthetic bank account opening applications with 31 variables and a binary label indicating whether they were deemed fraudulent.

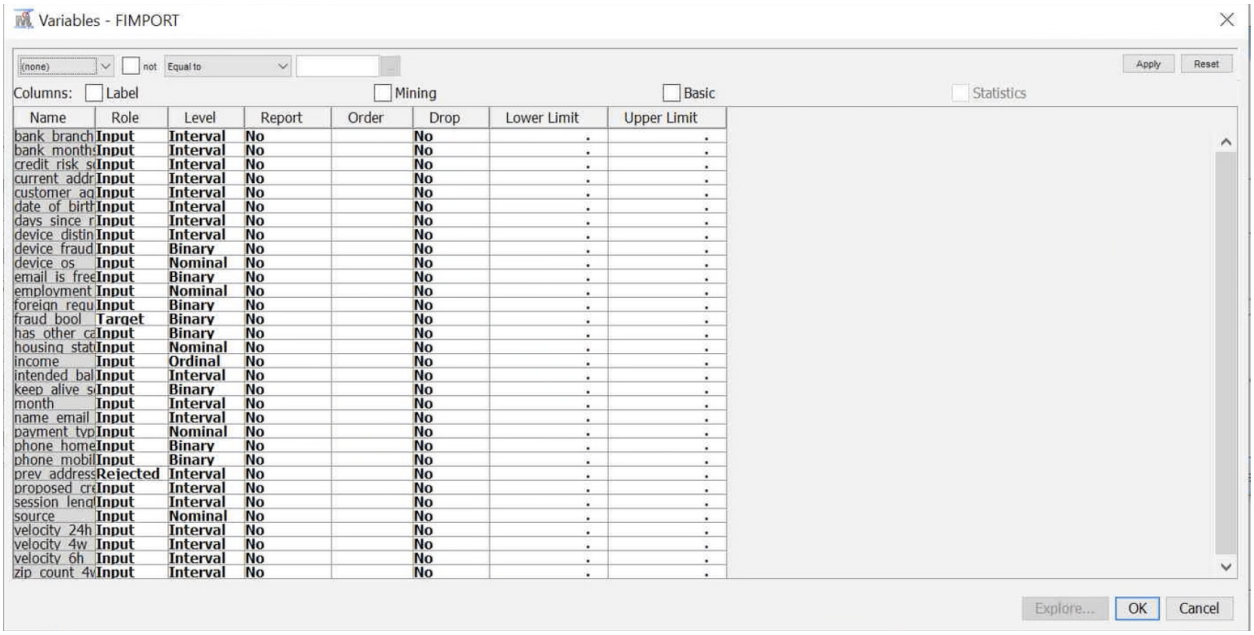
The objective of our project is to understand the key features that can predict fraudulent account applications from the dataset, and train machine learning models that can accurately predict fraudulent applications so that such applications can be flagged and investigated before they are approved by the bank. For this project, we will be training three types of models - Decision Trees, Logistic Regressions, and Neural Networks. The performance criteria for evaluating the accuracy of models will be Average Squared Error.

Data Setup and Exploration

Procedure

Kaggle Dataset->SAS Enterprise Miner ->File Import Node-> Import .csv file from H: Drive





Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
bank_branch	Input	Interval	No		No	.	.
bank_months	Input	Interval	No		No	.	.
credit_risk_s	Input	Interval	No		No	.	.
current_addr	Input	Interval	No		No	.	.
customer_ag	Input	Interval	No		No	.	.
date_of_birth	Input	Interval	No		No	.	.
days_since_r	Input	Interval	No		No	.	.
device_distin	Input	Interval	No		No	.	.
device_fraud	Input	Binary	No		No	.	.
device_os	Input	Nominal	No		No	.	.
email_is_free	Input	Binary	No		No	.	.
employment	Input	Nominal	No		No	.	.
foreign_requ	Input	Binary	No		No	.	.
fraud_bool	Target	Binary	No		No	.	.
has_other_ca	Input	Binary	No		No	.	.
housing_stat	Input	Nominal	No		No	.	.
income	Input	Ordinal	No		No	.	.
intended_bal	Input	Interval	No		No	.	.
keep_alive_s	Input	Binary	No		No	.	.
month	Input	Interval	No		No	.	.
name_email	Input	Interval	No		No	.	.
payment_typ	Input	Nominal	No		No	.	.
phone_home	Input	Binary	No		No	.	.
phone_mobil	Input	Binary	No		No	.	.
prev_address	Rejected	Interval	No		No	.	.
proposed_cr	Input	Interval	No		No	.	.
session_lengt	Input	Interval	No		No	.	.
source	Input	Nominal	No		No	.	.
velocity_24h	Input	Interval	No		No	.	.
velocity_4w	Input	Interval	No		No	.	.
velocity_6h	Input	Interval	No		No	.	.
zip_count_4w	Input	Interval	No		No	.	.

Variables discussion

Target variable

We have chosen **fraud_bool** as our target variable as we are predicting bank fraud cases. It is of Binary level i.e.1 & 0, where 1 implies fraud.

Rejected variable

We have chosen **prev_address_months_count** as our rejected variable because all the missing values in this variable have been modified as -1 instead of 0. It is our redundant variable.

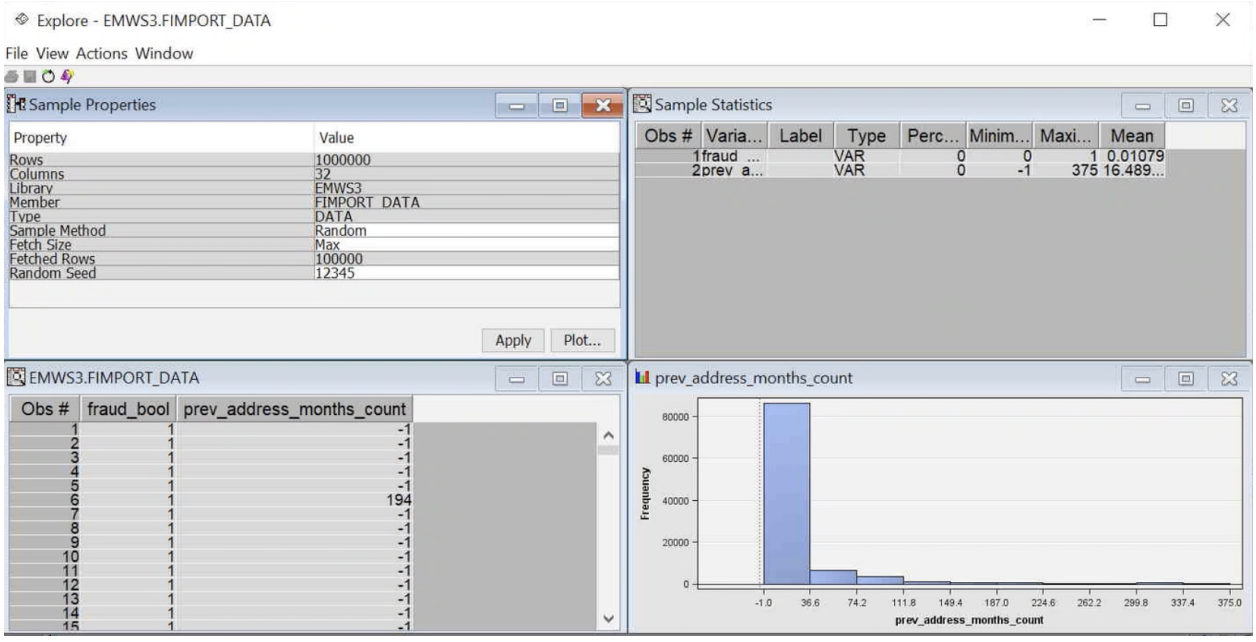
Binary variable

For our model, we have chosen some of our variables as binary such as device_fraud_count, email_is_free, phone_home_valid, phone_mobile_valid, has_other_cards, foreign_request, keep_alive_session.

Missing Data

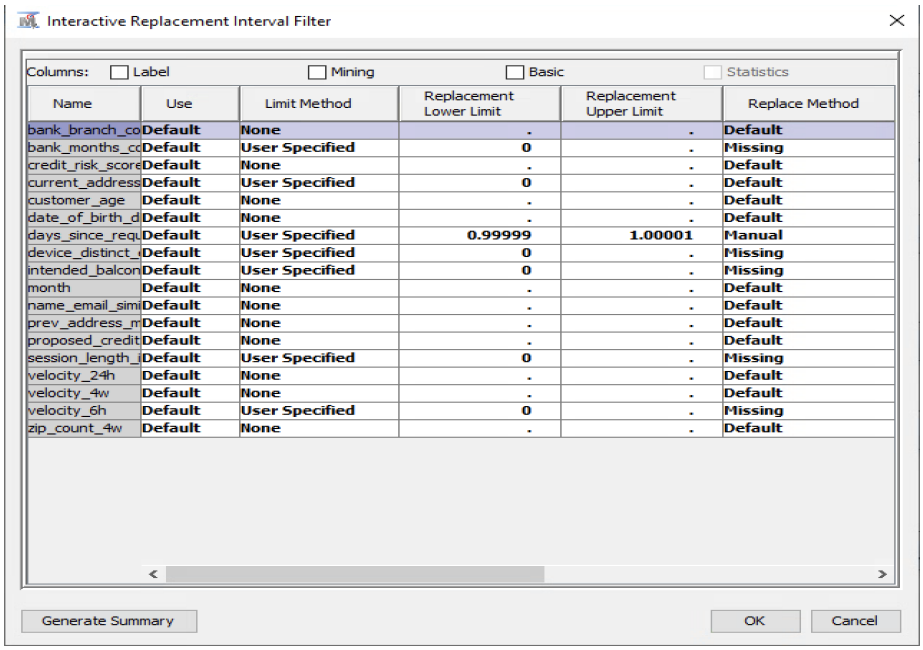
The dataset chosen from Kaggle had some disclaimers. One of which was, all missing values across variables have been modified as -1 instead of 0. Complications arose from this status quo as SAS Enterprise Miner needed to recognize -1 as genuinely missing. Besides that, -1 would have affected all models.

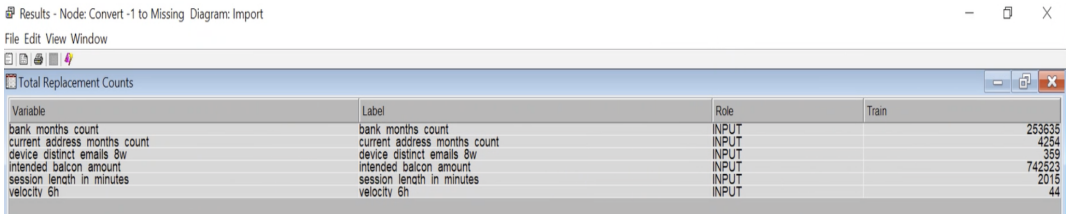
The screenshot below is illustrating one of the variables, prev_address_months_count. In the histogram we can see, that -1 has the highest frequency.



Therefore, as step 2 of our project, we added a Replacement node, to identify -1 as 0. In short, place the missings. The screenshot below shows all the successfully replaced values. For example- bank_months_count had 253635 rows replaced.

Since the values for days_since_request variables were concentrated mostly around 0 to 1, we created a flag for this variable using the replacement node. We set a lower limit of 0.99999 and an upper limit of 1.00001.





Skewed Data

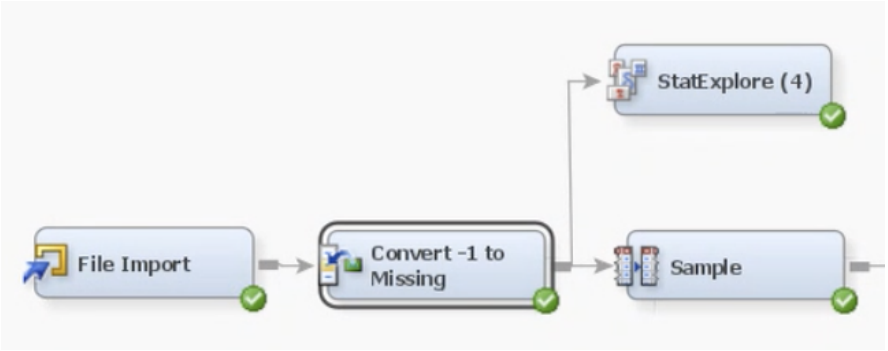
While also exploring the dataset, preliminary perusal showed skewness in multiple variables. The cut-off for skew for this project has been set at -1 to 1.

Upon further inquiry, all the statistics for the interval inputs were brought to light. As per the table below, the variables days_since_request, bank_branch_count_8w, session_length_in_minutes, device_distinct, etc. are heavily skewed. For now, we have only treated days_since_request using the flag in the replacement node. But, more will be done in the latter parts of the project.

Interval Variables

Data Role	Target	Target Level	Variable	Skewness ▼
TRAIN	fraud_bool	1	days_since_request	9.296595
TRAIN	fraud_bool	0	days_since_request	9.278118
TRAIN	fraud_bool	1	bank_branch_count_8w	3.443373
TRAIN	fraud_bool	0	REP_session_length_in_...	3.309741
TRAIN	fraud_bool	0	REP_device_distinct_em...	3.145775
TRAIN	fraud_bool	1	REP_session_length_in_...	3.06599
TRAIN	fraud_bool	0	bank_branch_count_8w	2.740934
TRAIN	fraud_bool	1	REP_device_distinct_em...	1.720496
TRAIN	fraud_bool	0	zip_count_4w	1.458261
TRAIN	fraud_bool	0	REP_current_address_m...	1.392139
TRAIN	fraud_bool	1	zip_count_4w	1.318249
TRAIN	fraud_bool	0	proposed_credit_limit	1.312176
TRAIN	fraud_bool	0	REP_intended_balcon_a...	1.302182
TRAIN	fraud_bool	1	REP_current_address_m...	1.166059
TRAIN	fraud_bool	1	REP_intended_balcon_a...	1.077586
TRAIN	fraud_bool	1	date_of_birth_distinct_em...	0.966699
TRAIN	fraud_bool	0	date_of_birth_distinct_em...	0.702392
TRAIN	fraud_bool	1	REP_velocity_6h	0.605518
TRAIN	fraud_bool	0	REP_velocity_6h	0.562333
TRAIN	fraud_bool	1	name_email_similarity	0.500577
TRAIN	fraud_bool	0	customer_age	0.478931
TRAIN	fraud_bool	1	proposed_credit_limit	0.383434
TRAIN	fraud_bool	0	velocity_24h	0.331332
TRAIN	fraud_bool	1	velocity_24h	0.298774
TRAIN	fraud_bool	0	credit_risk_score	0.29163
TRAIN	fraud_bool	1	customer_age	0.185965
TRAIN	fraud_bool	0	month	0.114383
TRAIN	fraud_bool	1	velocity_4w	0.089334
TRAIN	fraud_bool	0	REP_bank_months_count	0.04184
TRAIN	fraud_bool	0	name_email_similarity	0.038477
TRAIN	fraud_bool	1	credit_risk_score	-0.02536
TRAIN	fraud_bool	0	velocity_4w	-0.06165
TRAIN	fraud_bool	1	month	-0.07406
TRAIN	fraud_bool	1	REP_bank_months_count	-0.27931

StatExplore



The screenshot below depicts the results of all the variables post the replacement node via a StatExplore node.

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
REP_bank_months_count	INPUT	14.86262	11.52785	746365	253635	1	15	32	0.039016	-1.62086
REP_current_address_months_count	INPUT	86.59212	88.40241	1000000	0	0	52	428	1.387191	1.35724
REP_days_since_request	INPUT	0.081734	0.273959	1000000	0	0	0	1.000005	3.053498	7.323862
REP_device_distinct_emails_8w	INPUT	1.019037	0.1767	999641	359	0	1	2	3.126065	27.9827
REP_intended_balcon_amount	INPUT	36.5825	23.23689	257477	742523	0.000054	32.43325	112.9569	1.301721	1.904417
REP_session_length_in_minutes	INPUT	7.562193	8.032021	997985	2015	0.000872	5.122822	85.89914	3.308576	14.97626
REP_velocity_6h	INPUT	5665.549	3009.207	999956	44	0.651202	5319.873	16715.57	0.562857	0.003057
bank_branch_count_8w	INPUT	184.3618	459.6253	1000000	0	0	9	2385	2.747161	6.502921
credit_risk_score	INPUT	130.9896	69.68181	1000000	0	-170	122	389	0.295895	0.068087
customer_age	INPUT	33.68908	12.0258	1000000	0	10	30	90	0.478079	-0.1152
date_of_birth_distinct_emails_4w	INPUT	9.503544	5.033792	1000000	0	0	9	39	0.70325	0.436449
month	INPUT	3.288674	2.209994	1000000	0	0	3	7	0.112396	-1.12833
name_email_similarity	INPUT	0.493694	0.289125	1000000	0	1.43E-6	0.492152	0.999999	0.042839	-1.28028
proposed_credit_limit	INPUT	515.851	487.5599	1000000	0	190	200	2100	1.30141	0.168839
velocity_24h	INPUT	4769.782	1479.213	1000000	0	1300.307	4749.919	9506.897	0.331134	-0.37365
velocity_4w	INPUT	4856.324	919.8439	1000000	0	2825.748	4913.436	6994.764	-0.06012	-0.35963
zip_count_4w	INPUT	1572.692	1005.375	1000000	0	1	1263	6700	1.456657	2.139983

The first few variables starting with the prefix REP, refer to the ones which have been modified in the previous step, the replacement node. We used the node to replace missing values and flag values. For example, REP_device_distinct_emails_8w has 359 missing values in the dataset and 999641 non-missings. Similarly, all the variables which have undergone replacement have their missing and non-missing listed in the 3rd and 4th columns with the REP prefixes.

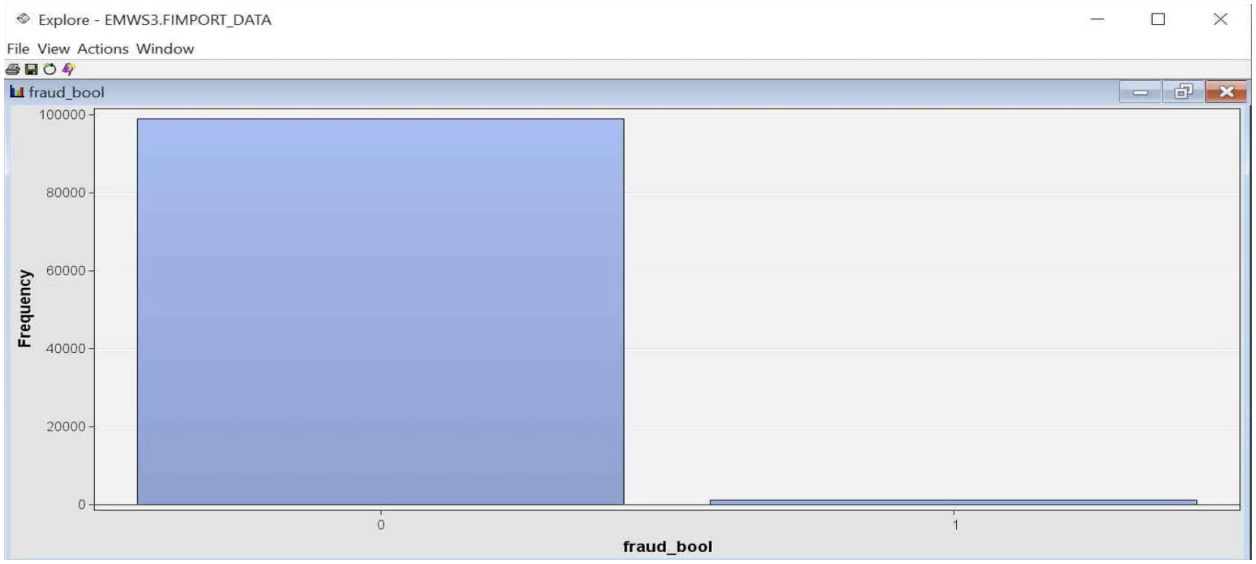
We can refer to the means and standard deviations of all the inputs from the first 2 columns. For instance, the average credit risk score for all the clients in the bank is around 130.98, with a standard deviation of 69.6 bps on both the positive and negative scales.

Minimum, Median, and Maximum values give an overall view of the data. The minimum or youngest customer is 10 years old, the median is 30 years old and the maximum is 90 years old at this bank. Lastly, we can also view the skew for each input in this output panel too.

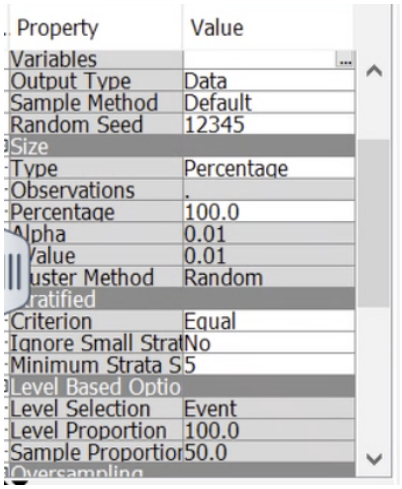
Data Oversampling

The dataset we are working with has Bank Account Fraud data points. Fraud is usually a rare event that is denoted by binary variables 0 and 1. In our dataset, the percentage of fraud is 1%, which is extremely low for data testing.

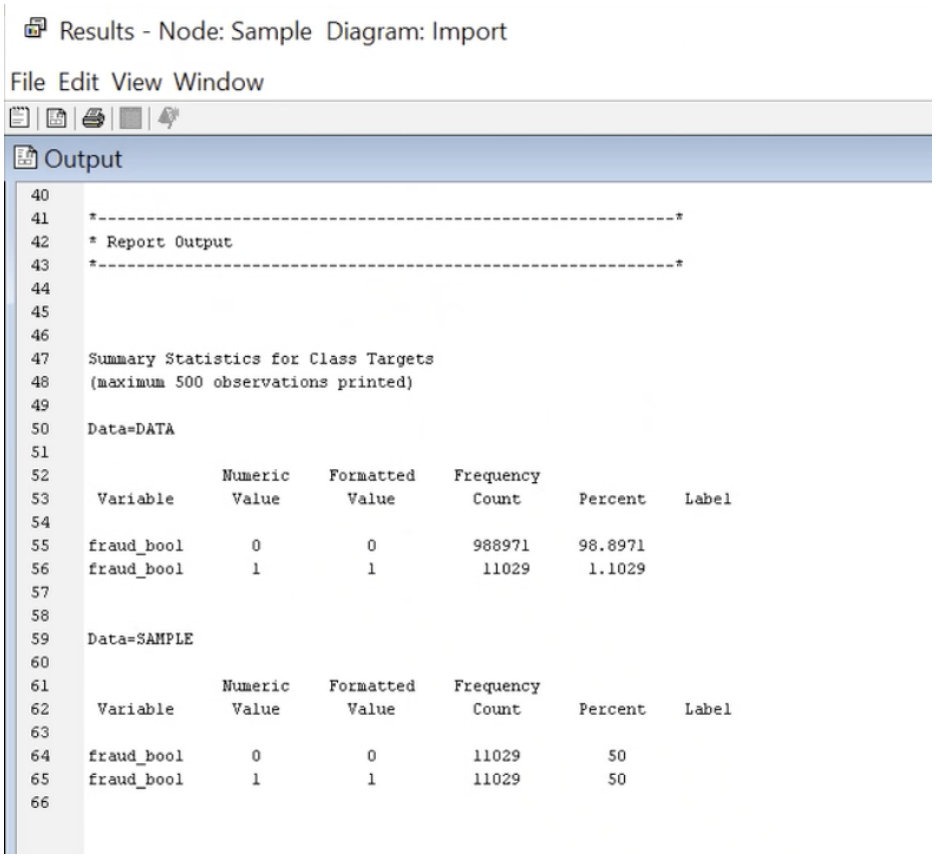
As depicted in the screenshot below, the accounts of ‘No Fraud’ severely outweigh the ‘Fraud’ events.



To bring a balance to the data, we are oversampling fraud events. We decided to keep a 50:50 ratio of 0 vs. 1. Using the Sample node in SAS, we adjusted the percentage by 100% and equaled it in the stratified criterion. The properties panel screenshot is below:



The screenshot below refers to the post-run results on the ‘Sample’ node. The initial dataset had almost 99% of non-fraud events. Whereas, after the successful run of the Sample node, the new percentages are 50:50 for fraud_bool(0 vs. 1).

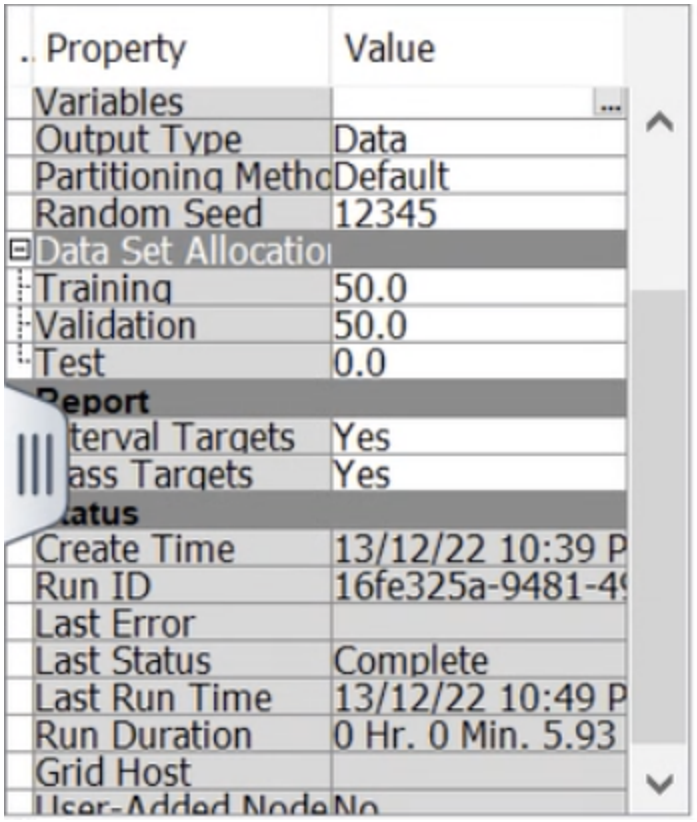



```
*------------------------------------------------------------*
* Report Output
*------------------------------------------------------------*



Summary Statistics for Class Targets
(maximum 500 observations printed)

Data=DATA

               Numeric    Formatted    Frequency
 Variable       Value       Value        Count      Percent    Label

fraud_bool        0           0          988971     98.8971
fraud_bool        1           1           11029      1.1029


Data=SAMPLE

               Numeric    Formatted    Frequency
 Variable       Value       Value        Count      Percent    Label

fraud_bool        0           0          11029         50
fraud_bool        1           1          11029         50
```

Data Partitioning: 50:50

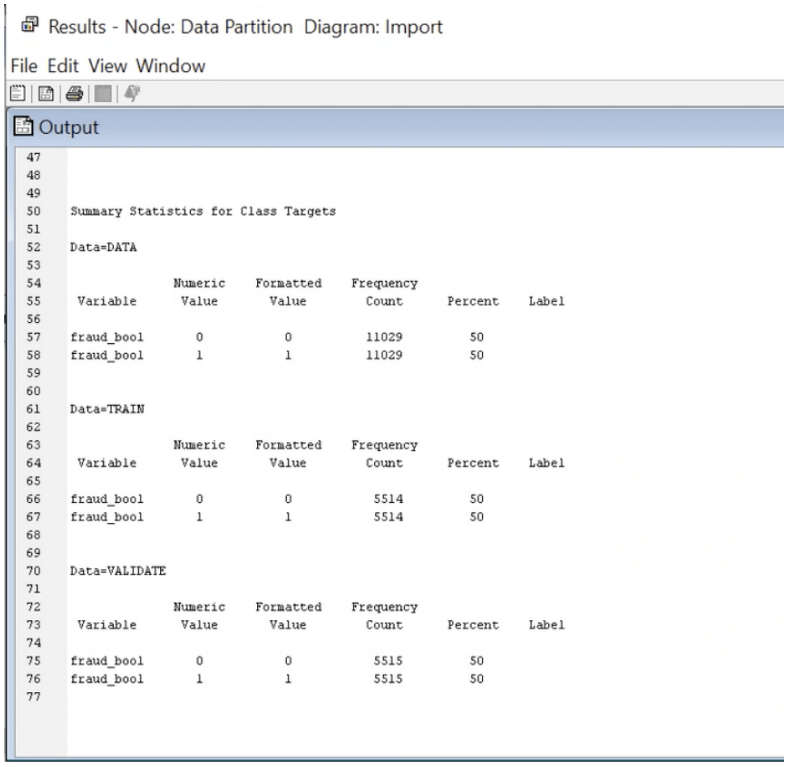
Data partition is a procedure for best model prediction. We have split our data into two parts i.e., a 50:50 ratio for training and validation. Training data is used to fit each model and the validation model is a random sample that is used for model selection.

For data partition, we drag the data partition node from the sample tab and connect it to our data set, and as depicted in our screenshot we changed the properties of training and validation data set allocation to 50% in both.



Property	Value
Variables	...
Output Type	Data
Partitioning Metho	Default
Random Seed	12345
Data Set Allocatio	
Training	50.0
Validation	50.0
Test	0.0
eport	
terval Targets	Yes
ass Targets	Yes
atus	
Create Time	13/12/22 10:39 P
Run ID	16fe325a-9481-4
Last Error	
Last Status	Complete
Last Run Time	13/12/22 10:49 P
Run Duration	0 Hr. 0 Min. 5.93
Grid Host	
User-Added Node	No

After making the necessary changes we ran our data partition node and viewed the results. As per the results that can be seen in our screenshot, training data has been allocated 50:50 to 0 vs 1 and their frequency count is 5514 for each. Validation data has also been allocated 50:50 and their frequency count is 5515 for each.


Results - Node: Data Partition Diagram: Import

File Edit View Window

Output

```
47
48
49
50  Summary Statistics for Class Targets
51
52  Data=DATA
53
54                Numeric    Formatted    Frequency
55   Variable      Value       Value        Count      Percent    Label
56
57  fraud_bool        0           0          11029         50
58  fraud_bool        1           1          11029         50
59
60
61  Data=TRAIN
62
63                Numeric    Formatted    Frequency
64   Variable      Value       Value        Count      Percent    Label
65
66  fraud_bool        0           0           5514         50
67  fraud_bool        1           1           5514         50
68
69
70  Data=VALIDATE
71
72                Numeric    Formatted    Frequency
73   Variable      Value       Value        Count      Percent    Label
74
75  fraud_bool        0           0           5515         50
76  fraud_bool        1           1           5515         50
77
```

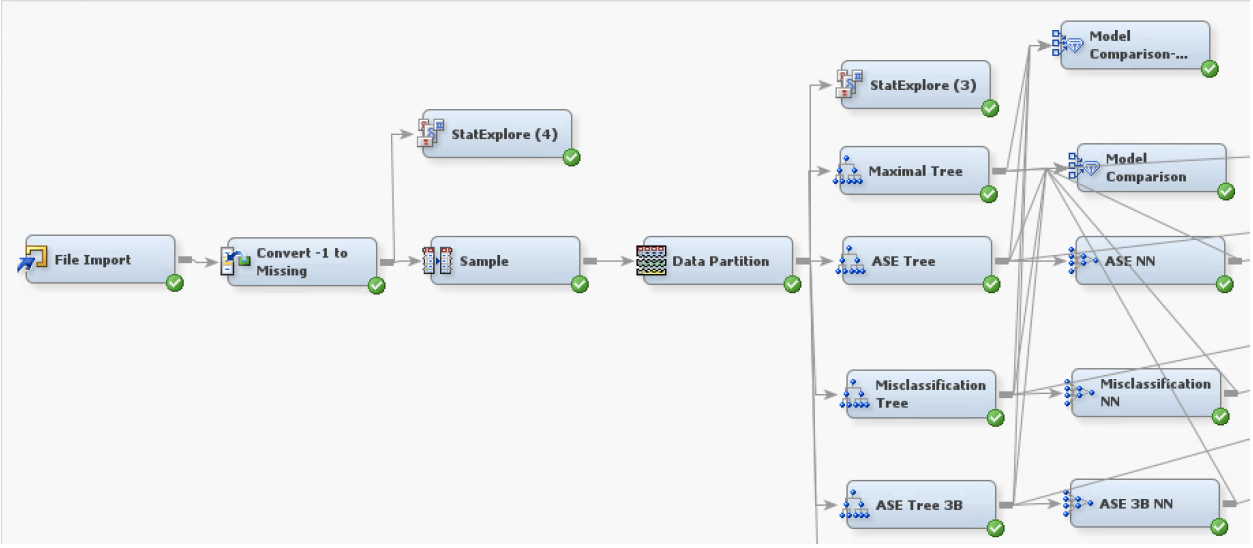
Decision Trees

After partitioning our data, we continue with the data analysis and one of the most effective methods for predictive modeling is decision trees. A split search strategy is used to choose the inputs, and it eliminates any variables with p-values less than 0.7. Pruning makes decision trees less complex by limiting the variables in the final tree to those with p values greater than or equal to 1. The Root Node is the first split, while the Leaf Nodes are the last splits.

We have implemented four different decision trees for this project:

- Maximal Tree
- ASE Tree
- Misclassification Tree
- ASE 3B Tree

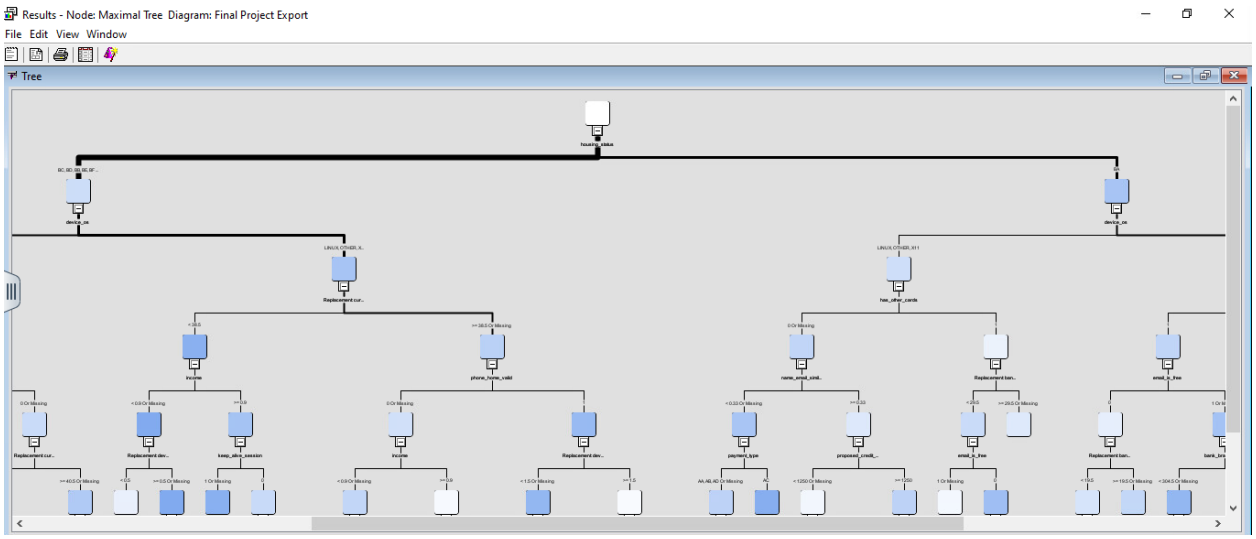
The screenshot of the Decision Trees is shown below.



Maximal Tree

Out of four different trees, we performed the Maximal Tree as our first decision tree. This tree is the largest statistically. This model has 55 leaves.

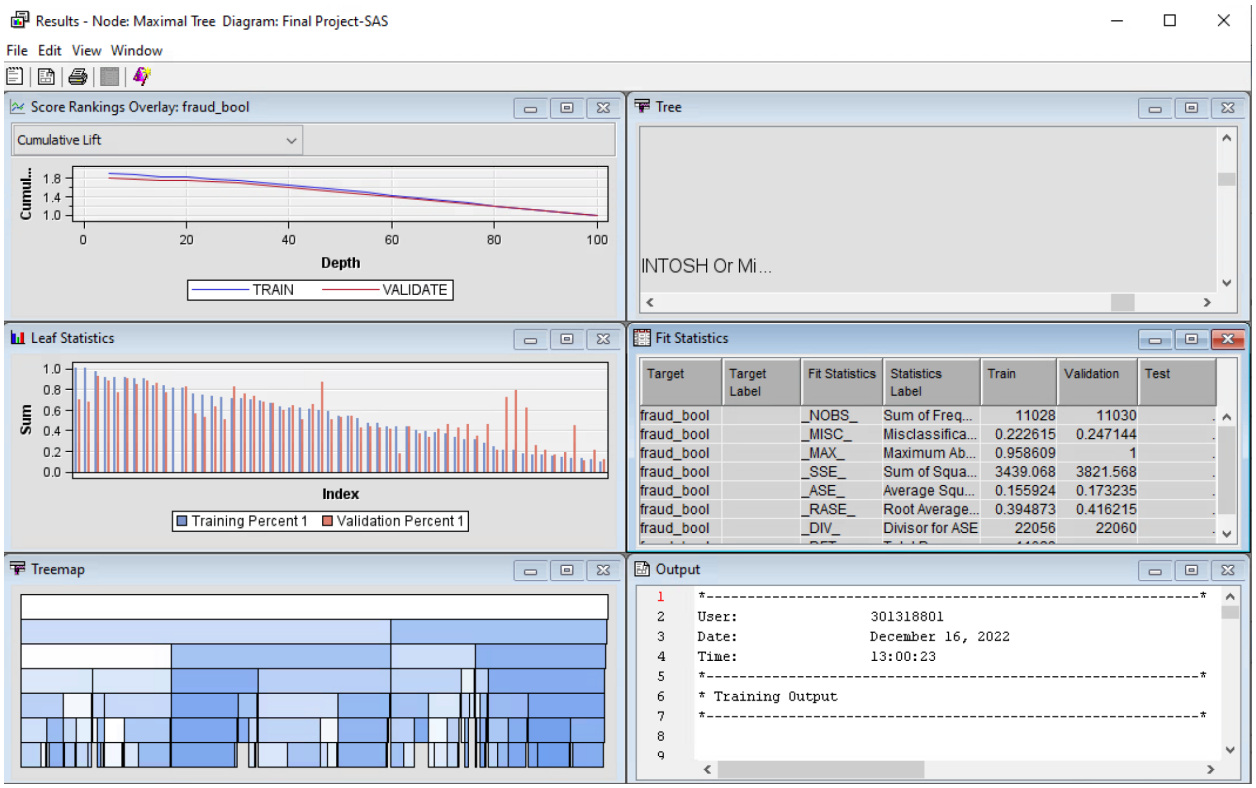
The root node is split using ‘housing_status’, followed by ‘device_os’. The 3rd splitting variable has changed with respect to each of the branches either to ‘has_other_cards’ or ‘replacement: current_address’. Screenshot below.

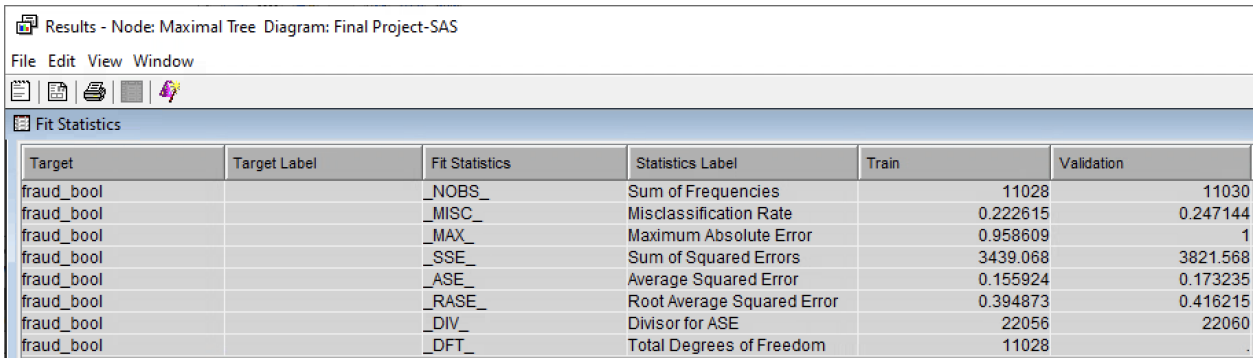


From the variables split, we see that more than 60% of the count has swayed to a housing_status besides BA. BA has a fraudulent validation rate of 77.27% compared to non-BA where fraudulent validation is 33.78%. Following BA, those with MAC, WINDOWS have the

higher fraud validation rate of 85.55%. The 3rd split on this has_other cards, and those who have shown 0 or missing cards have a validation rate of 87.37%.

Having mentioned one area of the maximal tree, the ASE derived from the maximal tree was **0.173235** which is the highest among all the trees. The misclassification rate was 0.222615 for the maximal tree. The screenshot below shows the result of maximal tree.

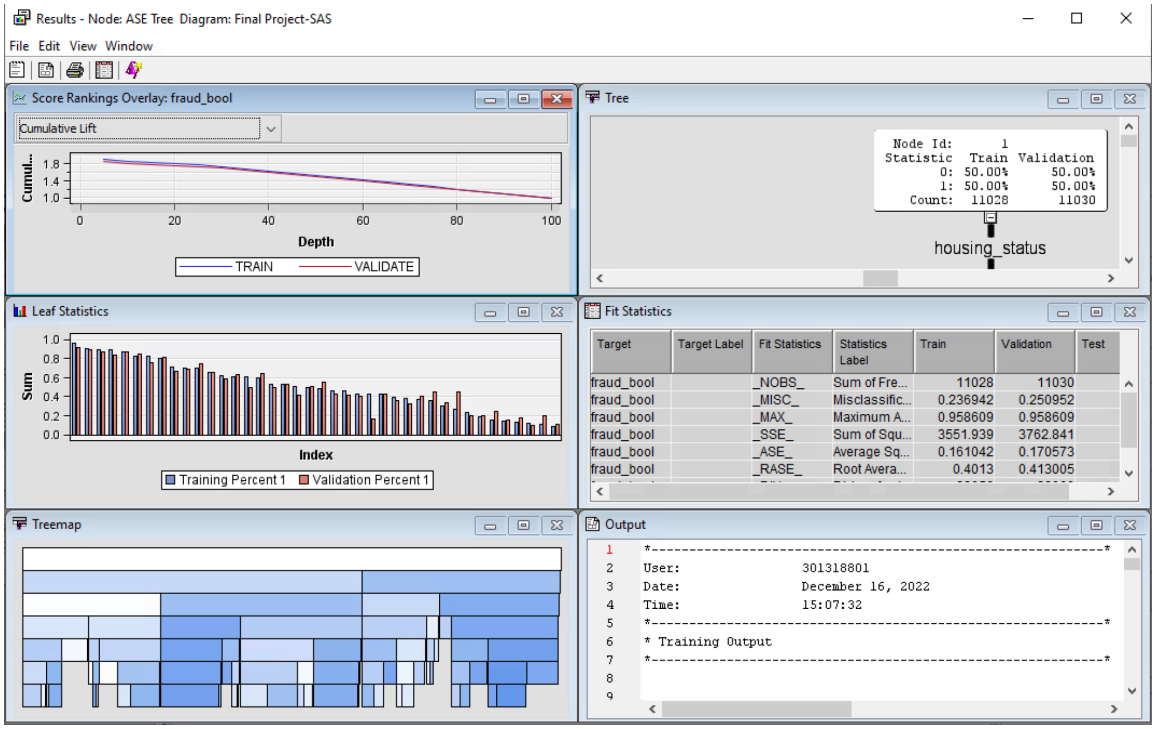


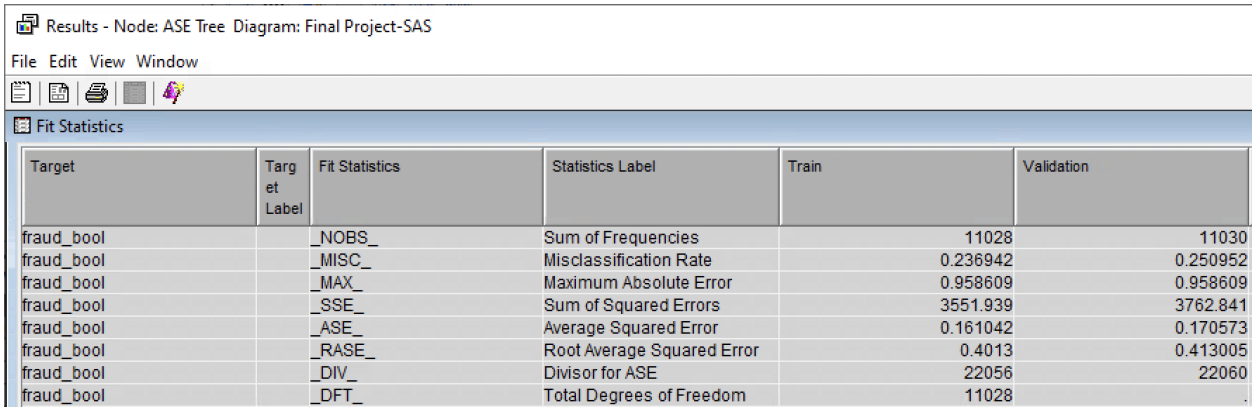


Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_MISC_	Misclassification Rate	0.222615	0.247144
fraud_bool		_MAX_	Maximum Absolute Error	0.958609	1
fraud_bool		_SSE_	Sum of Squared Errors	3439.068	3821.568
fraud_bool		_ASE_	Average Squared Error	0.155924	0.173235
fraud_bool		_RASE_	Root Average Squared Error	0.394873	0.416215
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_DFT_	Total Degrees of Freedom	11028	

ASE Tree

As expected the first 3 splits and validation rates remain the same, as optimal trees are produced by pruning branches from the bottom. We can refer to the tree map in the picture below. It is less dense than maximal. ASE tree contains 40 leaves which are lower than the maximal tree.
Given the reduction in the number of leaves, ASE has pruned the tree to its best. The ASE obtained from the ASE tree was **0.170573**, slightly lower than the Maximal Tree.



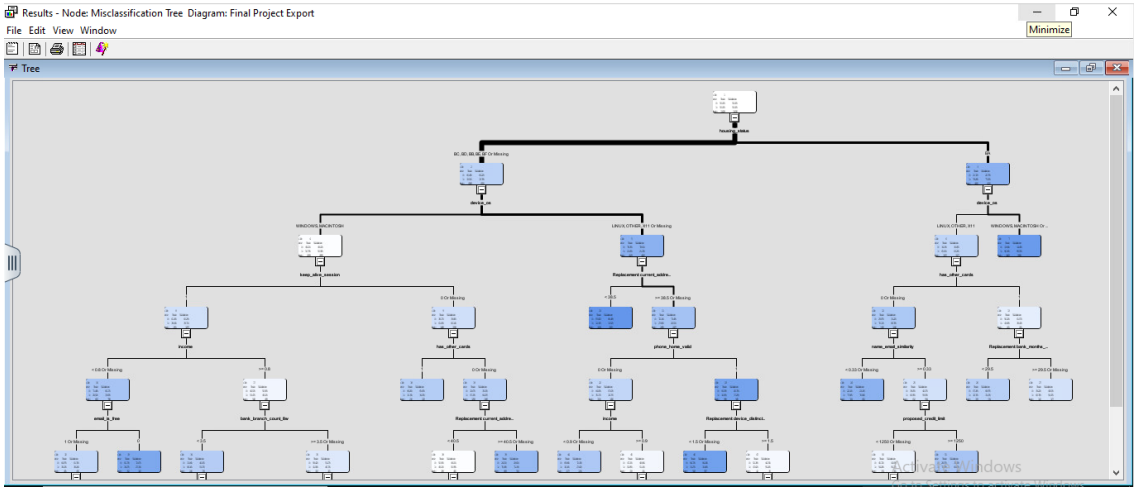


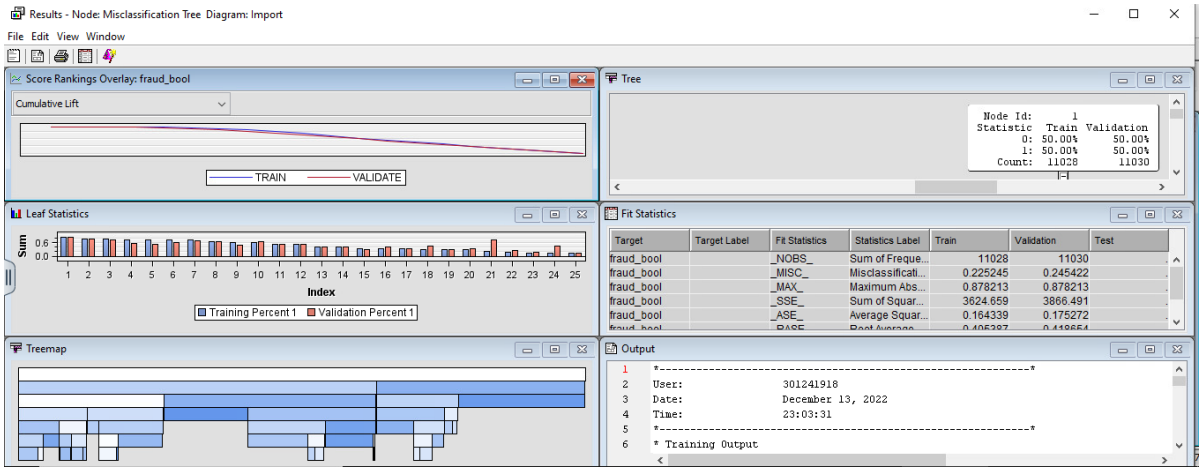
Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_MISC_	Misclassification Rate	0.236942	0.250952
fraud_bool		_MAX_	Maximum Absolute Error	0.958609	0.958609
fraud_bool		_SSE_	Sum of Squared Errors	3551.939	3762.841
fraud_bool		_ASE_	Average Squared Error	0.161042	0.170573
fraud_bool		_RASE_	Root Average Squared Error	0.4013	0.413005
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_DFT_	Total Degrees of Freedom	11028	

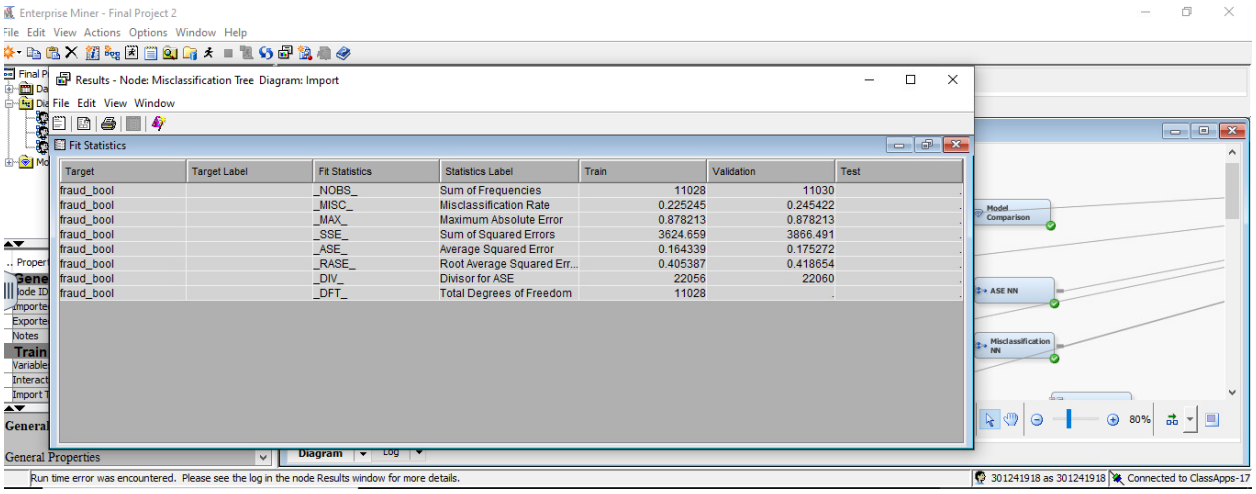
Misclassification Tree:

This model contains 25 leaves altogether, which is much fewer than the preceding decision trees when compared to their total number of leaves. However, the misclassification tree's ASE is the highest of all the trees at **0.175272**. A screenshot of the maximal tree's outcome is shown below.

Even though pruning is an efficient way to reduce error rates, it can also do the opposite. Such is this tree, where the tree has been pruned to an extent that the error rates are rising. So far, this is the worst decision tree model.



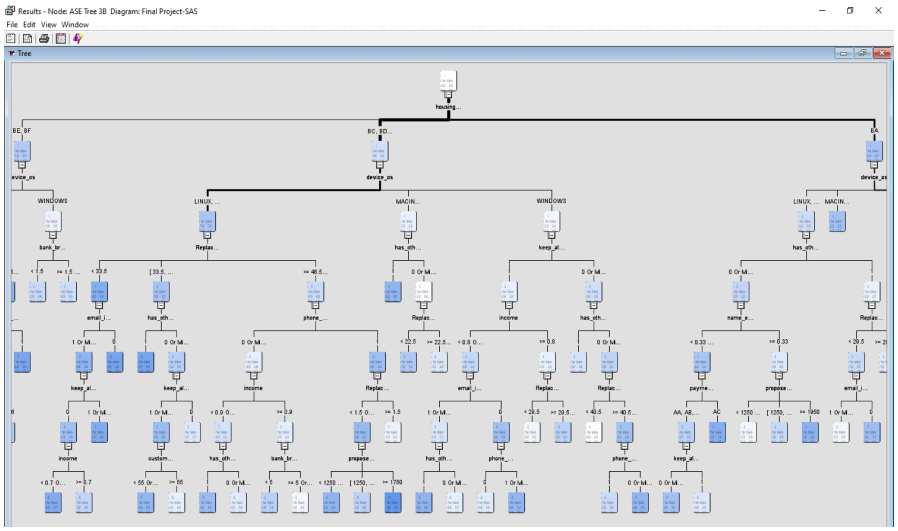




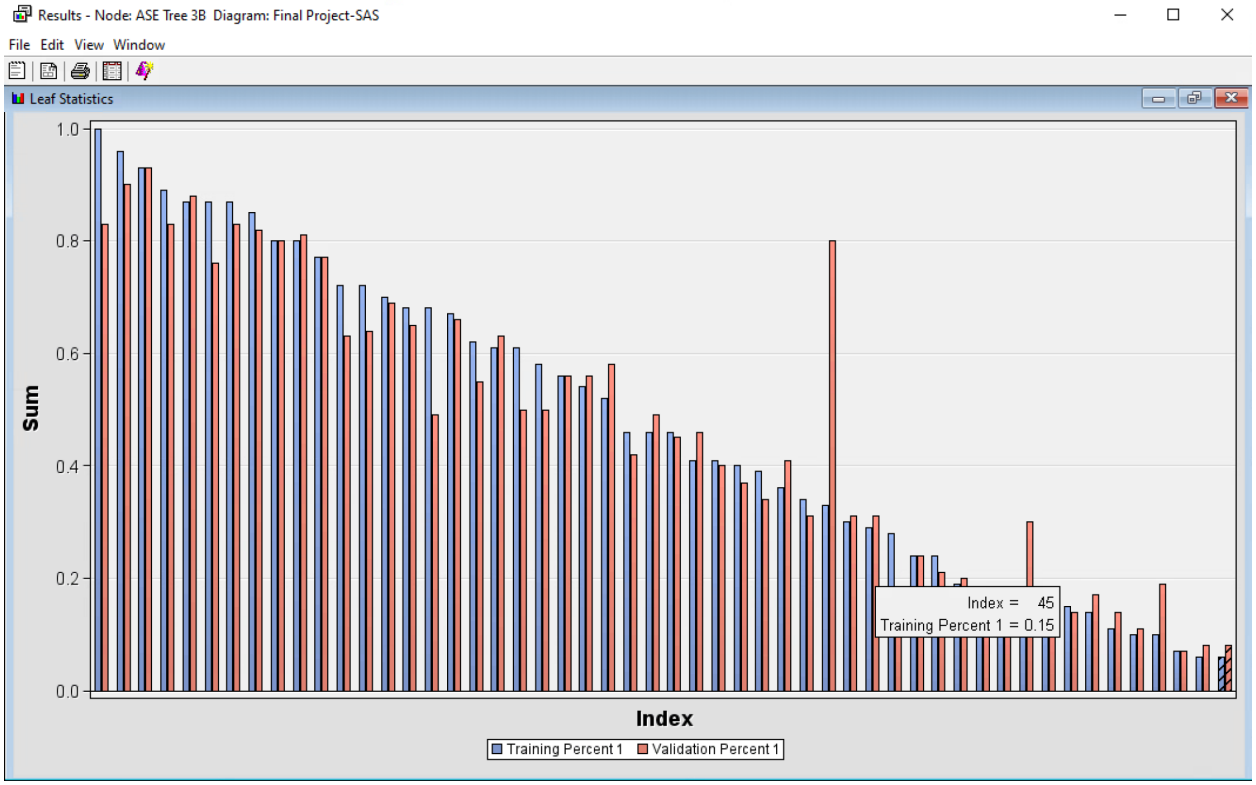
ASE 3-Branch Tree

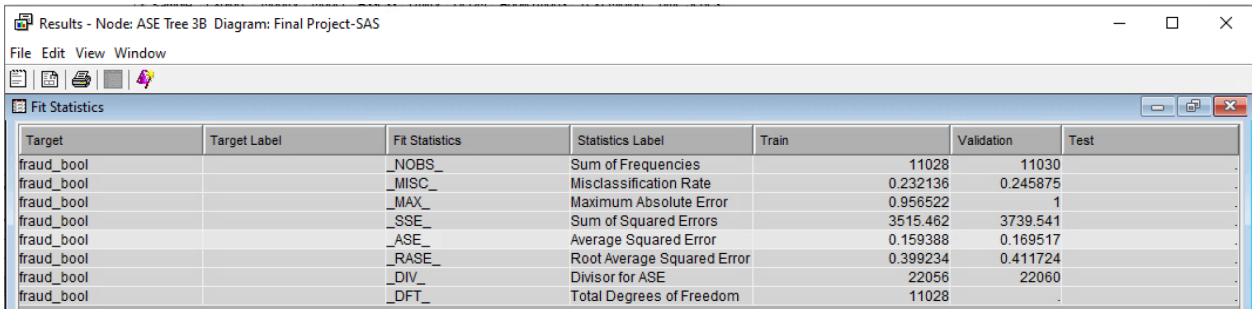
After exploring the 3 different trees, a 3-branch tree was deemed fit. However, due to the default function of SAS Enterprise Miner, we were getting 2 branches as main splits from the 'Root Node'.

While deciding on the model to apply a 3-branch on, ASE 2-Branch Decision Tree proved best. The ASE derived from ASE Tree (2B) was 0.170573 which is the lowest among all the trees. So we created a 4th tree using ASE Tree (2B) as the base, only with 3 branches this time. Screenshot:



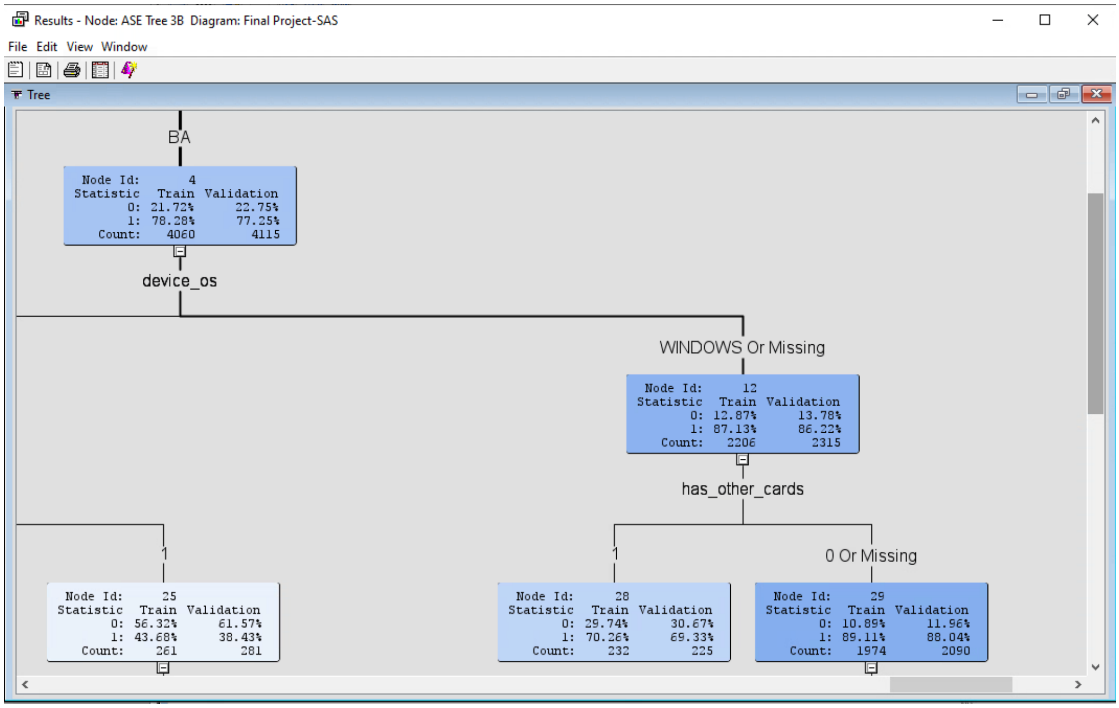
The number of leaves for this 3-branch tree is 52 and the ASE is 0.169517 which is the best so far and has been the expected result.





Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030	
fraud_bool		_MISC_	Misclassification Rate	0.232136	0.245875	
fraud_bool		_MAX_	Maximum Absolute Error	0.956522	1	
fraud_bool		_SSE_	Sum of Squared Errors	3515.462	3739.541	
fraud_bool		_ASE_	Average Squared Error	0.159388	0.169517	
fraud_bool		_RASE_	Root Average Squared Error	0.399234	0.411724	
fraud_bool		_DIV_	Divisor for ASE	22056	22060	
fraud_bool		_DFT_	Total Degrees of Freedom	11028		

The splits on this tree give more insight than the trees above due to its 3 branch property. In comparison to the maximal tree BA fraudulent validation rate remains at 77.27%. However, we start to see changes in the next splits. Previously, MAC, WINDOWS & Missing split on BA derived an 85.55% fraudulent validation rate, now it has been split into 2 groups. Those using WINDOWS or Missing devices have a fraudulent validation rate of 86.22%. has_other _cards which are denoted as 0 or Missing have a fraudulent validation rate of 88.04% compared to maximal tree's 87.37%.

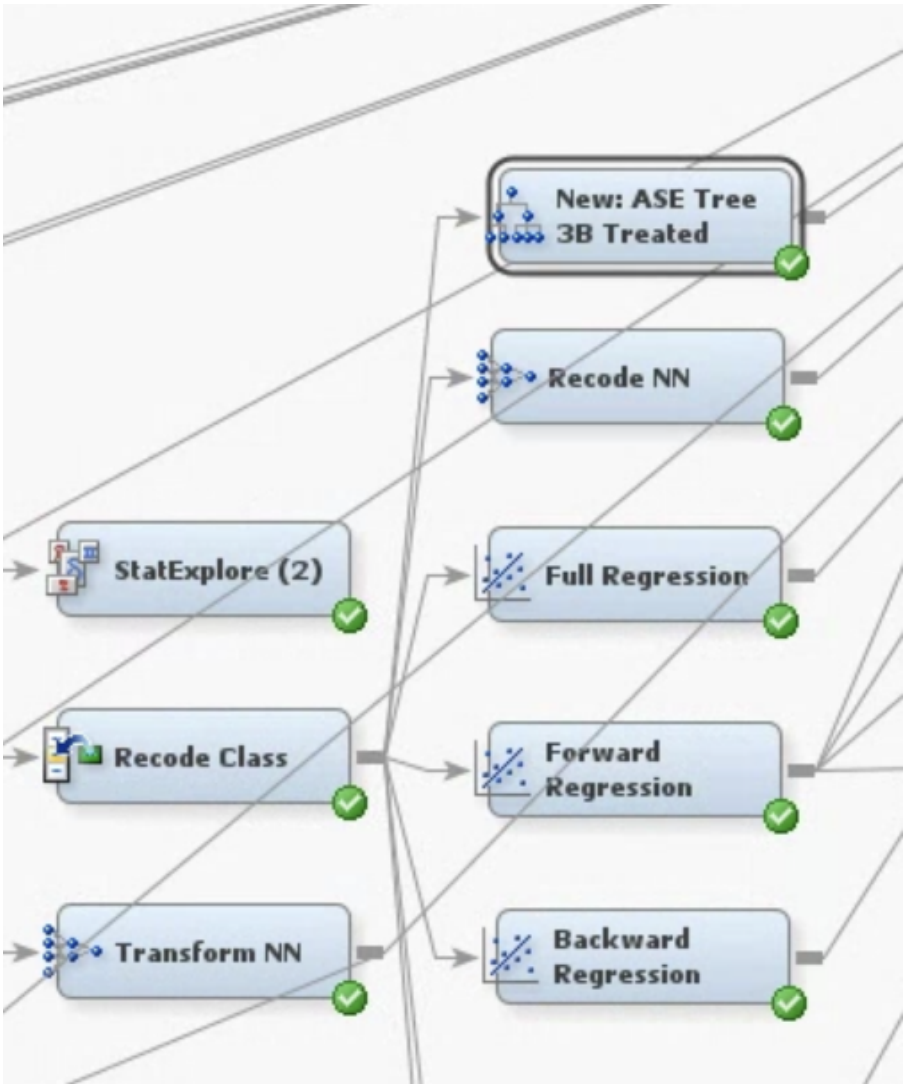


However, we further fine-tune our Trees with Neural Network nodes which will be covered in the Neural Network Section.

ASE Tree 3B Treated

The 3 branch ASE was the best tree model developed so far. However, all the tree models were developed before treating the data (imputing missing variables, adjusting skews, etc). and we wanted to see whether the same tree built from the treated data would be more accurate.

We attached a Decision Tree node after the Recode Class node (the final step of our data manipulation) with the same settings as the ASE Tree 3B developed earlier. Screenshot below:



However, this tree performed significantly worse, with an ASE of 0.169793 and a misclassification rate of 0.247325.

Model Comparison: Trees

Since we have created a few decision trees, we attached a model comparison node to all the trees. This node gives a concise snapshot of all the relevant statistics. In short, ASE with 3 branches is the Best Optimal Tree with 0.169517, followed by ASE 3-branch Tree with 0.169793. The Maximal Tree places 3rd with 0.173235, and the least reliable tree is Misclassification Tree with 0.175272. Screenshot below:

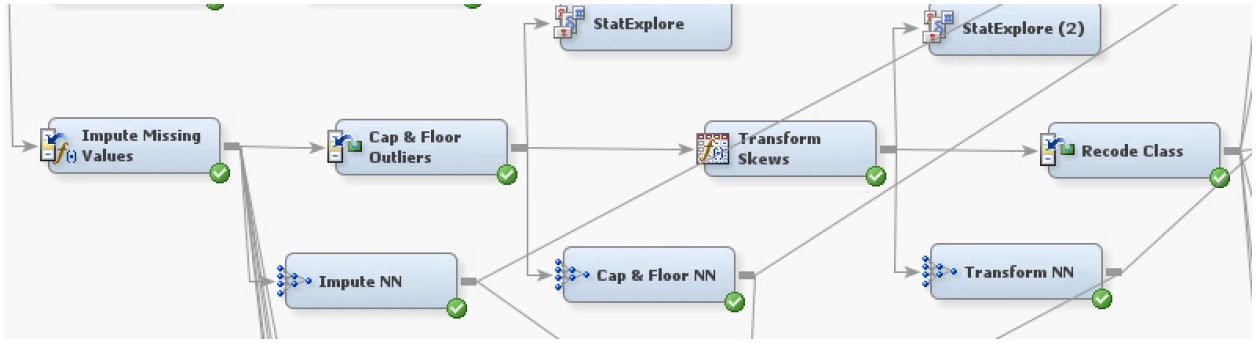
Results - Node: Model Comparison-Trees Diagram: Final Project-SAS

File Edit View Window

Fit Statistics

Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Valid: Average Squared Error ▲	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Sum of Frequencies	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error
	Tree4	Tree4	ASE Tree 3B	fraud_bool	0.169517		0.245875	11028	0.232136	0.956522	3515.462	0.159388
	Tree2	Tree2	ASE Tree	fraud_bool	0.170573		0.250952	11028	0.236942	0.958609	3551.939	0.161042
	Tree	Tree	Maximal Tree	fraud_bool	0.173235		0.247144	11028	0.222615	0.958609	3439.068	0.155924
Y	Tree3	Tree3	Misclassification Tree	fraud_bool	0.175272		0.245422	11028	0.225245	0.878213	3624.659	0.164339

Data Manipulation



We have followed the following processes to refine our data:

1. Impute Missing Values-After partitioning the data 50:50, we were still left with significant missing values. When it comes to regression, we could have left the missings untreated, but we preferred to work with a treated dataset. As per the screenshot below, we had up to 80% data missing in some cases.

Explore - EMWS12.Part_TRAIN

File View Actions Window

Sample Statistics

Obs #	Variable ...	Label	Type	Percent Missing	Minimum	Maximum	Mean	Number o...	Mode Per...	Mode
1	device_os		CLASS	0	.	.	.5		42.11099	WINDOWS
2	employmen...		CLASS	0	.	.	.7		76.3239	CA
3	housing_st...		CLASS	0	.	.	.7		36.81538	BA
4	payment_ty...		CLASS	0	.	.	.5		37.16902	AB
5	source		CLASS	0	.	.	.2		99.20203	INTERNET
6	REP_bank...	Replac...	VAR	31.18426	1	31	15.88431.		.	
7	REP_curre...	Replac...	VAR	0	0	392	101.2478.		.	
8	REP_days_...	Replac...	VAR	0	0	1	0.093671.		.	
9	REP_devic...	Replac...	VAR	0.036271	0	2	1.052885.		.	
10	REP_inten...	Replac...	VAR	80.83968	0.001803	112.2091	37.72258.		.	
11	REP_sessi...	Replac...	VAR	0.199492	0.05028	82.03582	7.931759.		.	
12	REP_veloci...	Replac...	VAR	0.009068	42.6942	16471.5	5400.038.		.	
13	_dataobs_	Observ...	VAR	0	1	999815	476241.3.		.	
14	bank_branc...		VAR	0	0	2251	167.3659.		.	

The customizations we used for the impute node are given below:

Property	Value
Random Seed	12345
Tuning Parameters	
Tree Imputation	
Score	
Hide Original Variables	Yes
Indicator Variables	
Type	Unique
Source	Imputed Variables
Role	Input
Report	
Validation and Test Data	No
Distribution of Missing	No
Status	
Create Time	16/12/22 10:11 AM
Run ID	b09587ef-4d71-4351-8b5a
Last Error	
Last Status	Complete
Last Run Time	16/12/22 3:15 PM
Run Duration	0 Hr. 0 Min. 6.77 Sec.
Grid Host	

After running the impute node, new variations of inputs were created with the prefix IMP short for impute. From the picture below, M_REP_banks_months_count had 31.184% missing, which is now 0% as per the new imputed version(IMP_REP_bank_months_count). The results are similar for both training and validating data.

Explore - EMWS12.Impt_TRAIN

File View Actions Window

Sample Statistics

Obs #	Variable Name	Label	Type	Percent Missing
1	device_os		CLASS	0
2	employment_status		CLASS	0
3	housing_status		CLASS	0
4	payment_type		CLASS	0
5	source		CLASS	0
6	IMP_REP_bank_months_count	Imputed: Replacement: bank_months_count	VAR	0
7	IMP_REP_device_distinct_emails_8	Imputed: Replacement: device_distinct_emai...	VAR	0
8	IMP_REP_session_length_in_minute	Imputed: Replacement: session_length_in_...	VAR	0
9	IMP_REP_velocity_6h	Imputed: Replacement: velocity_6h	VAR	0
10	M_REP_bank_months_count	Imputation Indicator for REP_bank_months_...	VAR	0
11	M_REP_device_distinct_emails_8	Imputation Indicator for REP_device_distinct...	VAR	0
12	M_REP_session_length_in_minute	Imputation Indicator for REP_session_length...	VAR	0
13	M_REP_velocity_6h	Imputation Indicator for REP_velocity_6h	VAR	0
14	REP_bank_months_count	Replacement: bank_months_count	VAR	31.18426
15	REP_current_address_months_count	Replacement: current_address_months_cou...	VAR	0
16	REP_days_since_request	Replacement: days_since_request	VAR	0
17	REP_device_distinct_emails_8w	Replacement: device_distinct_emails_8w	VAR	0.036271
18	REP_intended_balcon_amount	Replacement: intended_balcon_amount	VAR	80.83968
19	REP_session_length_in_minutes	Replacement: session_length_in_minutes	VAR	0.199492
20	REP_velocity_6h	Replacement: velocity_6h	VAR	0.009068

2. Cap and Floor-Having treated missings, we needed to adjust the outliers in the dataset. We added a replacement node to cap and floor the extreme values.

Results referred to below after running Cap & Floor:

Results - Node: Cap & Floor Outliers Diagram: Final Project-SAS

File Edit View Window

Total Replacement Counts

Variable	Label	Role	Train	Validation
IMP_REP_bank_months_count	Imputed: Replacement: bank_mont...	INPUT	0	0
IMP_REP_device_distinct_emails_8	Imputed: Replacement: device_disti...	INPUT	787	750
IMP_REP_session_length_in_minute	Imputed: Replacement: session_le...	INPUT	305	297
IMP_REP_velocity_6h	Imputed: Replacement: velocity_6h	INPUT	58	89
REP_current_address_months_co...	Replacement: current_address_mo...	INPUT	165	136
REP_days_since_request	Replacement: days_since_request	INPUT	1033	932
bank_branch_count_8w	bank_branch_count_8w	INPUT	473	469
credit_risk_score	credit_risk_score	INPUT	13	7
customer_age	customer_age	INPUT	45	41
date_of_birth_distinct_emails_4w	date_of_birth_distinct_emails_4w	INPUT	79	67
month	month	INPUT	0	0
name_email_similarity	name_email_similarity	INPUT	0	0
proposed_credit_limit	proposed_credit_limit	INPUT	0	0
velocity_24h	velocity_24h	INPUT	16	20
velocity_4w	velocity_4w	INPUT	0	0
zip_count_4w	zip_count_4w	INPUT	146	170

There have been multiple replacements in the overall dataset. For instance, customer age had 45 replacements in train data and 41 in validation data. The previous maximum age was 90 years old (refer to StatExplore in Data Exploration). Now the upper limit is 76.097 (screenshot below):

Results - Node: Cap & Floor Outliers Diagram: Final Project-SA

File Edit View Window

Interval Variables

Variable	Replace Variable	Upper Limit
IMP_REP_bank_m...	REP_IMP_REP_ba...	45.06553
IMP_REP_device_...	REP_IMP_REP_de...	1.838485
IMP_REP_session...	REP_IMP_REP_se...	34.6557
IMP_REP_velocity_...	REP_IMP_REP_vel...	14202.18
REP_current_addr...	REP_REP_current...	368.0167
REP_days_since_r...	REP_REP_days_s...	0.96782
bank_branch_coun...	REP_bank_branch...	1518.128
credit_risk_score	REP_credit_risk_s...	394.0411
customer_age	REP_customer_age	76.09712
date_of_birth_disti...	REP_date_of_birth...	23.64016
month	REP_month	10.21407
name_email_simil...	REP_name_email...	1.331466
proposed_credit_li...	REP_proposed_cr...	2446.652
velocity_24h	REP_velocity_24h	9044.762
velocity_4w	REP_velocity_4w	7649.599
zip_count_4w	REP_zip_count_4w	4592.446

The following screenshot is a consolidated list of all the upper and lower limits for each variable. The range between the limits is quite vast in terms of magnitude. There are chances of skews sustaining.

Results - Node: Cap & Floor Outliers Diagram: Final Project-SAS

File Edit View Window

Output

Limits and Replacement Values for Interval Variables

Variable	Replace Variable	Lower limit	Lower Replacement Value	Upper Limit	Upper Replacement Value
IMP_REP_bank_months_count	REP_IMP_REP_bank_months_count	-13.30	-13.30	45.07	45.07
IMP_REP_device_distinct_emails_8	REP_IMP_REP_device_distinct_emai	0.27	0.27	1.84	1.84
IMP_REP_session_length_in_minute	REP_IMP_REP_session_length_in_mi	-18.79	-18.79	34.66	34.66
IMP_REP_velocity_6h	REP_IMP_REP_velocity_6h	-3402.10	-3402.10	14202.18	14202.18
REP_current_address_months_count	REP_REP_current_address_months_c	-165.52	-165.52	368.02	368.02
REP_days_since_request	REP_REP_days_since_request	-0.78	-0.78	0.97	0.97
bank_branch_count_8w	REP_bank_branch_count_8w	-1183.40	-1183.40	1518.13	1518.13
credit_risk_score	REP_credit_risk_score	-86.28	-86.28	394.04	394.04
customer_age	REP_customer_age	-1.78	-1.78	76.10	76.10
date_of_birth_distinct_emails_4w	REP_date_of_birth_distinct_email	-6.69	-6.69	23.64	23.64
month	REP_month	-3.35	-3.35	10.21	10.21
name_email_similarity	REP_name_email_similarity	-0.45	-0.45	1.33	1.33
proposed_credit_limit	REP_proposed_credit_limit	-1106.51	-1106.51	2446.65	2446.65
velocity_24h	REP_velocity_24h	323.81	323.81	9044.76	9044.76
velocity_4w	REP_velocity_4w	1954.60	1954.60	7649.60	7649.60
zip_count_4w	REP_zip_count_4w	-1394.12	-1394.12	4592.45	4592.45

3. Transform Skews-The initial skews for the inputs were as high as 9 whereas it should be between -1 to 1.

The skews below show the before of transformation. Most of the variables are skewed positively. Three of the highly skewed variables are:

- REP_IMP_REP device_distinct_email- 3.9.
- REP_bank_branch_count_8w-3.15
- REP_REP_days_since_request -2.95

Results - Node: StatExplore Diagram: Final Project-SAS

File Edit View Window

Interval Variables

Data Role	Target	Target Level	Variable	Skewness ▼
TRAIN	fraud_bool	0	REP_IMP_REP_device_distinct_emai	3.905741
TRAIN	fraud_bool	1	REP_bank_branch_count_8w	3.150615
TRAIN	fraud_bool	0	REP_REP_days_since_request	2.954947
TRAIN	fraud_bool	1	REP_REP_days_since_request	2.642093
TRAIN	fraud_bool	0	REP_bank_branch_count_8w	2.390073
TRAIN	fraud_bool	1	REP_IMP_REP_session_length_in_mi	2.140994
TRAIN	fraud_bool	0	REP_IMP_REP_session_length_in_mi	2.118179
TRAIN	fraud_bool	1	REP_IMP_REP_device_distinct_emai	1.985692
TRAIN	fraud_bool	0	REP_REP_current_address_months_c	1.391076
TRAIN	fraud_bool	0	REP_proposed_credit_limit	1.326893
TRAIN	fraud_bool	0	REP_zip_count_4w	1.275582
TRAIN	fraud_bool	1	REP_REP_current_address_months_c	1.163739
TRAIN	fraud_bool	1	REP_zip_count_4w	1.146333

We edited the variables with the highest skews with a log transformation. In the variables edit panel, we opened the interval variables and chose ‘log’ instead of ‘default’ to minimize skew. We changed 6 variables. The variables are given below:

Variables - Trans

(none) not Equal to

Columns: Label

Name	Method
REP_days_since_request	Log
REP_zip_count_4w	Log
REP_proposed_credit_limit	Log
REP_IMP_REP_session_length_in_mi	Log
REP_bank_branch_count_8w	Log
REP_REP_current_address_months_c	Log

After transforming variables, the skews are as follows:

Data Role=TRAIN

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness
LOG_REP_IMP_REP_session_length_i	INPUT	1.885558	0.698889	11028	0	0.049057	1.813917	3.573909	0.418206
LOG_REP_REP_current_address_mont	INPUT	4.129014	1.18554	11028	0	0	4.317488	5.910842	-1.01391
LOG_REP_bank_branch_count_8w	INPUT	2.294334	2.165631	11028	0	0	2.079442	7.325891	1.050047
LOG_REP_proposed_credit_limit	INPUT	6.105275	0.894113	11028	0	5.252273	5.303305	7.650169	0.426677
LOG_REP_zip_count_4w	INPUT	7.194527	0.614542	11028	0	2.639057	7.173192	8.432386	-0.33139
REP_IMP_REP_bank_months_count	INPUT	15.88431	9.727075	11028	0	1	15.88431	31	-0.10056
REP_IMP_REP_device_distinct_emai	INPUT	1.045324	0.21584	11028	0	0.267284	1	1.838485	2.641262
REP_IMP_REP_velocity_6h	INPUT	5396.566	2922.976	11028	0	42.6942	5081.81	14202.18	0.549797
REP_REP_days_since_request	INPUT	0.090656	0.282007	11028	0	0	0	0.96782	2.789475
REP_credit_risk_score	INPUT	153.9016	79.9823	11028	0	-86.283	147	378	0.212643
REP_customer_age	INPUT	37.1404	12.92266	11028	0	10	40	76.09712	0.322286
REP_date_of_birth_distinct_email	INPUT	8.458159	4.98683	11028	0	0	8	23.64016	0.675764
REP_month	INPUT	3.432989	2.260361	11028	0	0	3	7	0.008786
REP_name_email_similarity	INPUT	0.440276	0.297063	11028	0	0.000132	0.398713	0.999985	0.275152
REP_velocity_24h	INPUT	4684.161	1453.113	11028	0	1328.41	4694.3	9044.762	0.315318
REP_velocity_4w	INPUT	4802.102	949.1657	11028	0	2863.783	4860.331	6889.978	0.019002

Most of the skews have reduced and come inside the acceptable range of -1 to 1. There are only 2 variables where skews still persist, device_distinct_email and days_since_request. Both are around 2 which is still an improvement over the pre-transform state. After careful consideration, we have decided to leave these 2 variables as is. This decision was solely made to retain the natural state of the data. We could have manipulated the data to reach a desired range between -1 to 1, however, doing so we would have tweaked the data well beyond its original comprehension. Manipulation helps eliminate noise in the data, bring structure, and derive clearer relationships among variable. On the other hand, we are letting go of some of the originality which may or may not be crucial to understand the causal relationship with the dependent variable. In this case, we already had performed multiple modifications and aligned most variables within our desired threshold, any more modification would have caused more harm than benefit.

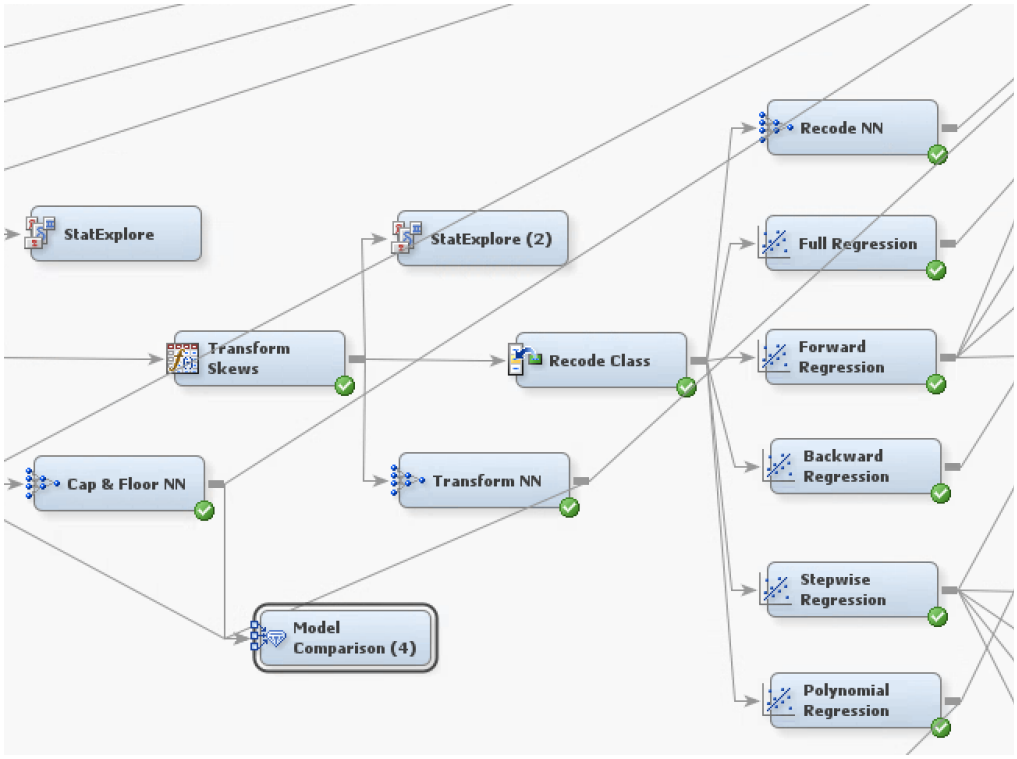
4. Recode Class Variables-All the transformations done so far mostly impacted interval variables. In the case of class variables, we wanted to recode some class variables. We were given limited options in the dataset. The only options which made sense were:
    - Employment status
    - Housing status
    - Payment type
    - Income

The first 3 options on paper seem feasible, however, the data dictionary did not suffice. Very little clarity was provided on the acronyms, hence, we did not have a basis to group the classes. Income was naturally the only variable we decided to recode. Classes ranged from 0.1 to 0.9. We divided the data into 3 classes and took the mean for each class and denoted the class with the mean value. For example, 0.1, 0.2, and 0.3 all were classed as 0.2. Due to SAS limitations, we could not assign the degree of income in terms of ‘High’, ‘Med’, and ‘Low’, though it would have been ideal.

Replacement Editor-WORK.OUTCLASS

Variable	Formatted Value	Replacement Value
housing_status	BE	
housing_status	BD	
housing_status	BF	
housing_status	BG	
housing_status	_UNKNOWN_	_DEFAULT_
income	0.9	**0.8**
income	0.8	**0.8**
income	0.1	**0.2**
income	0.6	**0.5**
income	0.7	**0.8**
income	0.4	**.5**
income	0.2	**0.2**
income	0.5	**0.5**
income	0.3	**0.2**
income	_UNKNOWN_	_DEFAULT_

Regressions



For our model, we have chosen logistic regression for the analysis.
Logistic regression uses previous observations from a data set to predict a binary outcome, such as yes or no. By examining the correlation between one or more already present independent variables, a logistic regression model forecasts a dependent data variable.

Logistic Regression Prediction Formula

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 \cdot x_1 + \hat{w}_2 \cdot x_2$$

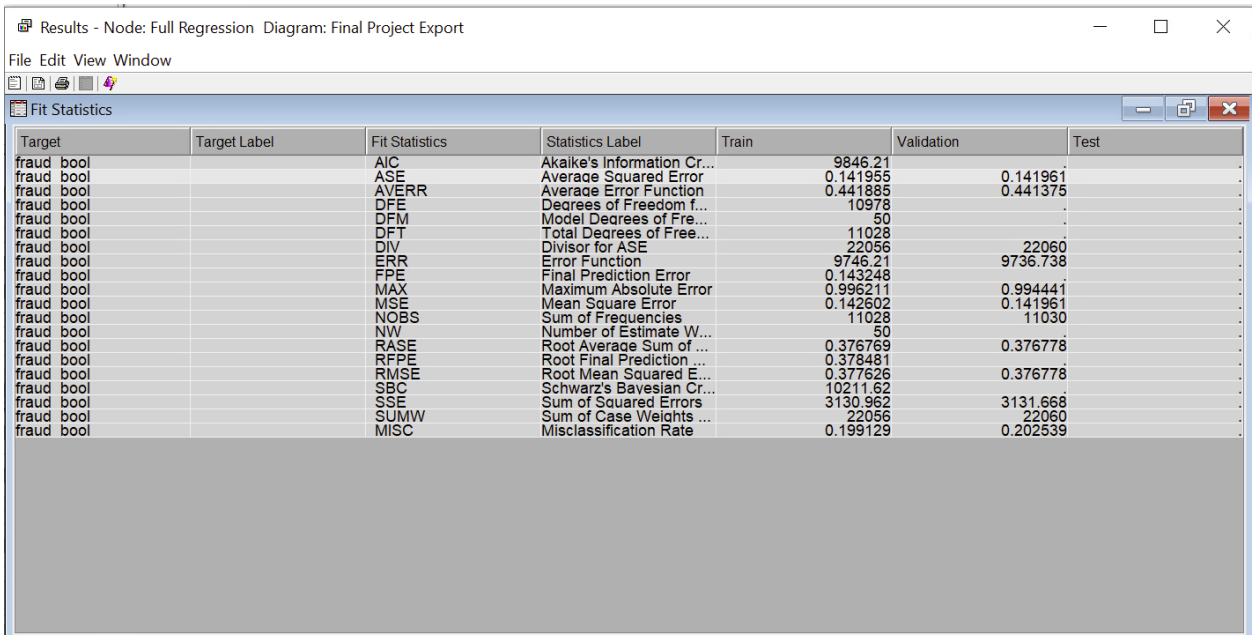
logit scores

We have used 4 types of regression i.e.,

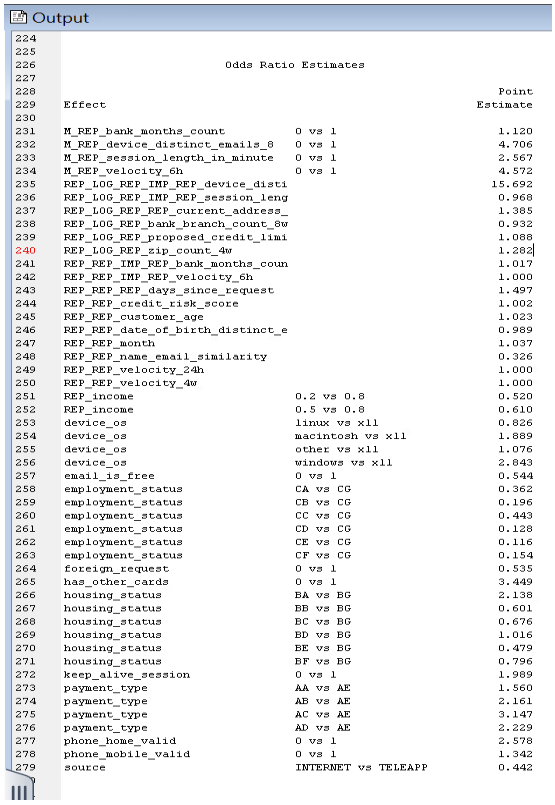
- Full Regression
- Forward Regression
- Backward Regression
- Stepwise Regression
- Polynomial Regression

Full Regression

We first conducted a full regression of our model. As per the screenshot it can be depicted that the ASE of full regression is **0.141961**.



Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		AIC	Akaike's Information Cr...	9846.21		.
fraud bool		ASE	Average Squared Error	0.141955	0.141961	.
fraud bool		AVERR	Average Error Function	0.441885	0.441375	.
fraud bool		DFE	Degrees of Freedom f...	10978	.	.
fraud bool		DFM	Model Degrees of Fre...	50	.	.
fraud bool		DFT	Total Degrees of Free...	11028	.	.
fraud bool		DIV	Divisor for ASE	22056	22060	.
fraud bool		ERR	Error Function	9746.21	9736.738	.
fraud bool		FPE	Final Prediction Error	0.143248	.	.
fraud bool		MAX	Maximum Absolute Error	0.996211	0.994441	.
fraud bool		MSE	Mean Square Error	0.142602	0.141961	.
fraud bool		NOBS	Sum of Frequencies	11028	11030	.
fraud bool		NW	Number of Estimate W...	50	.	.
fraud bool		RASE	Root Average Sum of ...	0.376769	0.376778	.
fraud bool		RFPE	Root Final Prediction ...	0.378481	.	.
fraud bool		RMSE	Root Mean Squared E...	0.377626	0.376778	.
fraud bool		SBC	Schwarz's Bayesian Cr...	10211.62	.	.
fraud bool		SSE	Sum of Squared Errors	3130.962	3131.668	.
fraud bool		SUMW	Sum of Case Weights ...	22056	22060	.
fraud bool		MISC	Misclassification Rate	0.199129	0.202539	.


Output

Odds Ratio Estimates

Effect		Point Estimate
M_REP_bank_months_count	0 vs 1	1.120
M_REP_device_distinct_emails_8	0 vs 1	4.706
M_REP_session_length_in_minute	0 vs 1	2.567
M_REP_velocity_6h	0 vs 1	4.572
REP_LOG_REP_IMP_REP_device_disti		15.692
REP_LOG_REP_IMP_REP_session_leng		0.968
REP_LOG_REP_REP_current_address_		1.385
REP_LOG_REP_bank_branch_count_8w		0.932
REP_LOG_REP_proposed_credit_limi		1.088
REP_LOG_REP_zip_count_4w		1.282
REP_REP_IMP_REP_bank_months_coun		1.017
REP_REP_IMP_REP_velocity_6h		1.000
REP_REP_REP_days_since_request		1.497
REP_REP_credit_risk_score		1.002
REP_REP_customer_age		1.023
REP_REP_date_of_birth_distinct_e		0.989
REP_REP_month		1.037
REP_REP_name_email_similarity		0.326
REP_REP_velocity_24h		1.000
REP_REP_velocity_4w		1.000
REP_income	0.2 vs 0.8	0.520
REP_income	0.5 vs 0.8	0.610
device_os	linux vs x11	0.826
device_os	macintosh vs x11	1.889
device_os	other vs x11	1.076
device_os	windows vs x11	2.843
email_is_free	0 vs 1	0.544
employment_status	CA vs CG	0.362
employment_status	CB vs CG	0.196
employment_status	CC vs CG	0.443
employment_status	CD vs CG	0.128
employment_status	CE vs CG	0.116
employment_status	CF vs CG	0.154
foreign_request	0 vs 1	0.535
has_other_cards	0 vs 1	3.449
housing_status	BA vs BG	2.138
housing_status	BB vs BG	0.601
housing_status	BC vs BG	0.676
housing_status	BD vs BG	1.016
housing_status	BE vs BG	0.479
housing_status	BF vs BG	0.796
keep_alive_session	0 vs 1	1.989
payment_type	AA vs AE	1.560
payment_type	AB vs AE	2.161
payment_type	AC vs AE	3.147
payment_type	AD vs AE	2.229
phone_home_valid	0 vs 1	2.578
phone_mobile_valid	0 vs 1	1.342
source	INTERNET vs TELEAPP	0.442

As per the odds ratio, for every one unit increase in the M_REP_bank_months_count variable, the odds of bank application fraud increase by approximately 12.0%. Customers with M_REP_device_distinct_emails_0 have approximately 4.7 times higher odds of being involved in fraud compared to customers without this characteristic. Customers with higher M_REP_velocity_6h have significantly higher odds of being associated with bank fraud. Specifically, for each unit increase in M_REP_velocity_6h, the odds of bank fraud increase by approximately 4.6 times. Customers with REP_LOG_REP_IMP_REP_device_distinct_emails are approximately 15.7 times more likely to be involved in bank fraud compared to customers without this characteristic. Comparing all log transformed variables, REP_LOG_REP_IMP_REP_device_distinct_emails is more related to application fraud. Customers with device_os of x11 are 88% more likely to be involved in fraud as compared to those with macintosh. Customers with housing_status of BG the odds of bank fraud increase by approximately 2.1 times as compared with customers with housing_status of BA.

Forward Regression

For forward regression, we changed the model selection to forward and the selection criteria are Validation error.

Property	Value
General	
Node ID	Reg3
Imported Data	
Exported Data	
Notes	
Train	
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	Forward
Selection Criterion	Validation Error
Use Selection Defaults	Yes
Selection Options	
Optimization Options	
Technique	Default
Default Optimization	Yes
Max Iterations	0
Max Function Calls	0
Maximum Time	1 Hour

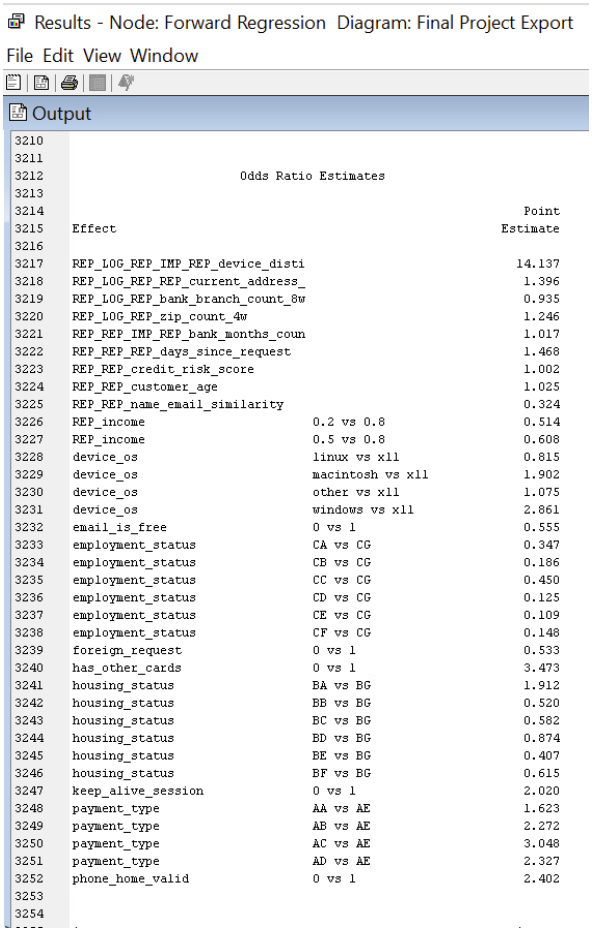
As per the forward regression model, our ASE is **0.141946** which is slightly better than full regression.

Results - Node: Forward Regression Diagram: Final Project Export

File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		AIC	Akaike's Information Cr...	9862.33	.	.
fraud bool		ASE	Average Squared Error	0.142634	0.141946	.
fraud bool		AVERR	Average Error Function	0.443794	0.441087	.
fraud bool		DFE	Degrees of Freedom f...	10991	.	.
fraud bool		DFM	Model Degrees of Fre...	37	.	.
fraud bool		DFT	Total Degrees of Free...	11028	.	.
fraud bool		DIV	Divisor for ASE	22056	22060	.
fraud bool		ERR	Error Function	9788.33	9730.371	.
fraud bool		FPE	Final Prediction Error	0.143594	.	.
fraud bool		MAX	Maximum Absolute Error	0.995982	0.992461	.
fraud bool		MSE	Mean Square Error	0.143114	0.141946	.
fraud bool		NOBS	Sum of Frequencies	11028	11030	.
fraud bool		NW	Number of Estimate W...	37	.	.
fraud bool		RASE	Root Average Sum of ...	0.377669	0.376757	.
fraud bool		RFPE	Root Final Prediction ...	0.378938	.	.
fraud bool		RMSE	Root Mean Squared E...	0.378304	0.376757	.
fraud bool		SBC	Schwarz's Bayesian Cr...	10132.73	.	.
fraud bool		SSE	Sum of Squared Errors	3145.937	3131.325	.
fraud bool		SUMW	Sum of Case Weights ...	22056	22060	.
fraud bool		MISC	Misclassification Rate	0.201306	0.202629	.



Results - Node: Forward Regression Diagram: Final Project Export

File Edit View Window

Output

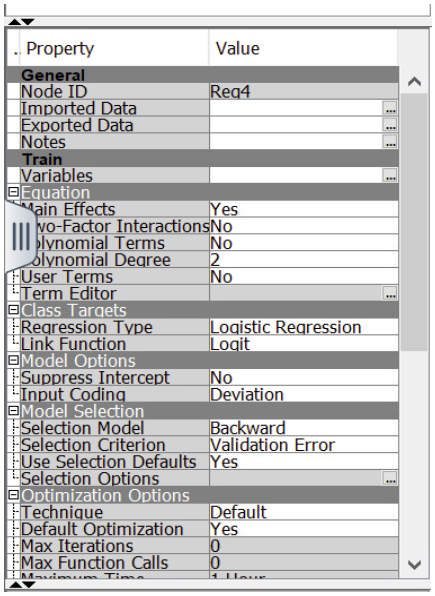
Odds Ratio Estimates

Effect		Point Estimate
REP_LOG_REP_IMP_REP_device_disti		14.137
REP_LOG_REP_REP_current_address_		1.396
REP_LOG_REP_bank_branch_count_8w		0.935
REP_LOG_REP_zip_count_4w		1.246
REP_REP_IMP_REP_bank_months_coun		1.017
REP_REP_REP_days_since_request		1.468
REP_REP_credit_risk_score		1.002
REP_REP_customer_age		1.025
REP_REP_name_email_similarity		0.324
REP_income	0.2 vs 0.8	0.514
REP_income	0.5 vs 0.8	0.608
device_os	linux vs x11	0.815
device_os	macintosh vs x11	1.902
device_os	other vs x11	1.075
device_os	windows vs x11	2.861
email_is_free	0 vs 1	0.555
employment_status	CA vs CG	0.347
employment_status	CB vs CG	0.186
employment_status	CC vs CG	0.450
employment_status	CD vs CG	0.125
employment_status	CE vs CG	0.109
employment_status	CF vs CG	0.148
foreign_request	0 vs 1	0.533
has_other_cards	0 vs 1	3.473
housing_status	BA vs BG	1.912
housing_status	BB vs BG	0.520
housing_status	BC vs BG	0.582
housing_status	BD vs BG	0.874
housing_status	BE vs BG	0.407
housing_status	BF vs BG	0.615
keep_alive_session	0 vs 1	2.020
payment_type	AA vs AE	1.623
payment_type	AB vs AE	2.272
payment_type	AC vs AE	3.048
payment_type	AD vs AE	2.327
phone_home_valid	0 vs 1	2.402

Customers with REP_LOG_REP_IMP_REP_device_disti have approximately 14.1 times higher odds of being involved in bank fraud compared to customers with a lower count of distinct devices. Our missing variable which has been treated after impute, REP_REP_IMP_REP_bank_months_coun for each unit increase in it, the odds of bank fraud increases by approximately 1.6%. Customers with REP_LOG_REP_REP_current_address have approximately 1.4 times higher odds of being associated with bank application fraud compared to customers without a current address. Customers with device_os of x11 are 90% more likely to be involved in fraud as compared to those with macintosh. Customers with housing_status of BG the odds of bank fraud increase by approximately 90% more likely to be involved in fraud as compared with customers with housing_status of BA. payment_type AE is 3 times more related to fraud as compared with AC. Customers with foreign_request the odds of bank fraud decreased by 46.7% as compared to those without foreign_request.

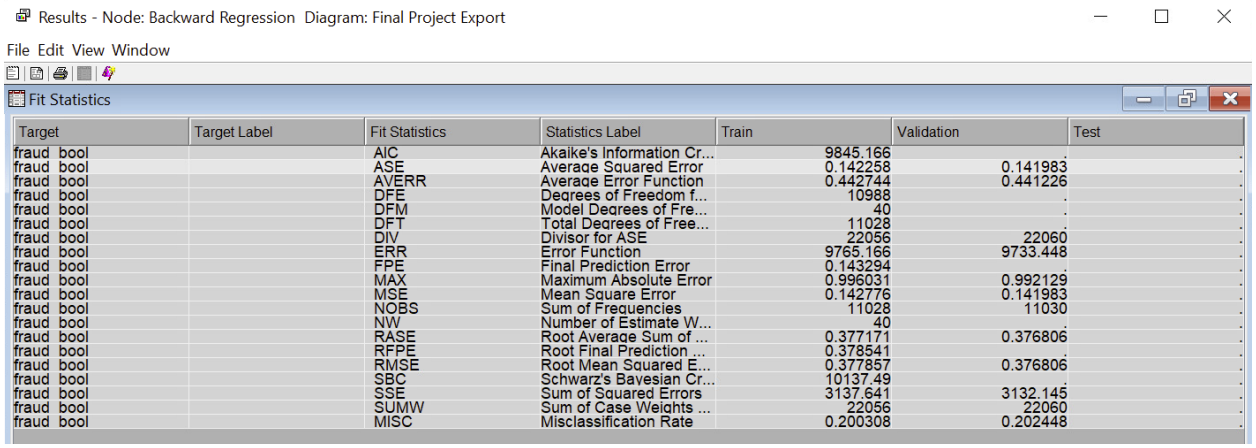
Backward Regression

For backward regression we changed the model selection to backward and the selection criteria is Validation error.



Property	Value
General	
Node ID	Reg4
Imported Data	
Exported Data	
Notes	
Train	
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	Backward
Selection Criterion	Validation Error
Use Selection Defaults	Yes
Selection Options	
Optimization Options	
Technique	Default
Default Optimization	Yes
Max Iterations	0
Max Function Calls	0

As per backward regression model, our ASE is **0.141983** which is worse than full and forward regression.



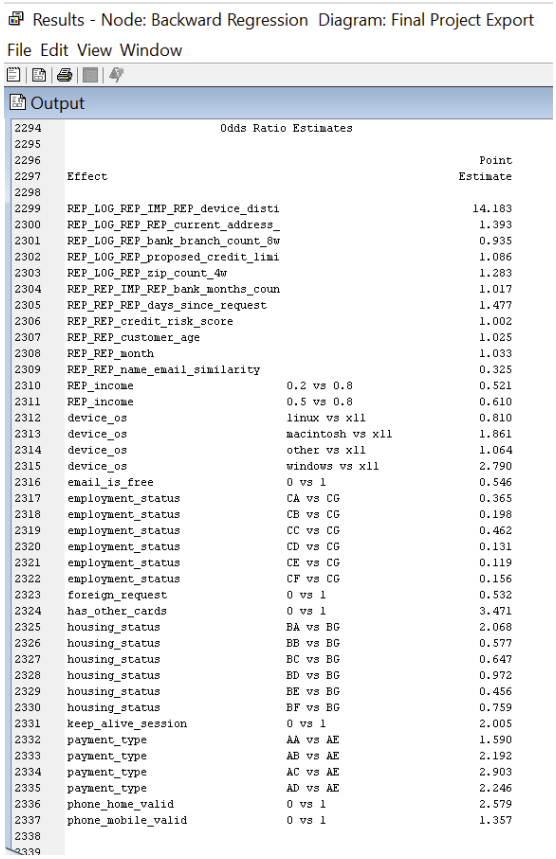
Results - Node: Backward Regression Diagram: Final Project Export

File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		AIC	Akaike's Information Cr...	9845.166	.	.
fraud bool		ASE	Average Squared Error	0.142258	0.141983	.
fraud bool		AVERR	Average Error Function	0.442744	0.441226	.
fraud bool		DFE	Degrees of Freedom f...	10988	.	.
fraud bool		DFM	Model Degrees of Fre...	40	.	.
fraud bool		DFT	Total Degrees of Free...	11028	.	.
fraud bool		DIV	Divisor for ASE	22056	22060	.
fraud bool		ERR	Error Function	9765.166	9733.448	.
fraud bool		FPE	Final Prediction Error	0.143294	.	.
fraud bool		MAX	Maximum Absolute Error	0.996031	0.992129	.
fraud bool		MSE	Mean Square Error	0.142776	0.141983	.
fraud bool		NOBS	Sum of Frequencies	11028	11030	.
fraud bool		NW	Number of Estimate W...	40	.	.
fraud bool		RASE	Root Average Sum of ...	0.377171	0.376806	.
fraud bool		RFPE	Root Final Prediction ...	0.378541	.	.
fraud bool		RMSE	Root Mean Squared E...	0.377857	0.376806	.
fraud bool		SBC	Schwarz's Bayesian Cr...	10137.49	.	.
fraud bool		SSE	Sum of Squared Errors	3137.641	3132.145	.
fraud bool		SUMW	Sum of Case Weights ...	22056	22060	.
fraud bool		MISC	Misclassification Rate	0.200308	0.202448	.

Customers with REP_LOG_REP_IMP_REP_device_disti have approximately 14.2 times higher odds of being involved in bank fraud compared to customers with a lower count of distinct devices. Our missing variable which has been treated after impute, REP_REP_IMP_REP_bank_months_coun for each unit increase in it, the odds of bank fraud increases by approximately 1.7%. Customers with REP_LOG_REP_REP_current_address have approximately 1.4 times higher odds of being associated with bank application fraud compared to customers without a current address. Customers with device_os of x11 are 86% more likely to be involved in fraud as compared to those with macintosh. Customers with housing_status of BG the odds of being involved in bank fraud is 2 times more as compared with customers with housing_status of BA. payment_type AE is 2.9 times more related to fraud as compared with AC. Customers with foreign_request the odds of bank fraud decreased by 46.7% as compared to those without foreign_request. If a customer has_other_cards then they are 3.47 times more likely to be involved in application fraud as compared with those who has not.



Results - Node: Backward Regression Diagram: Final Project Export

File Edit View Window

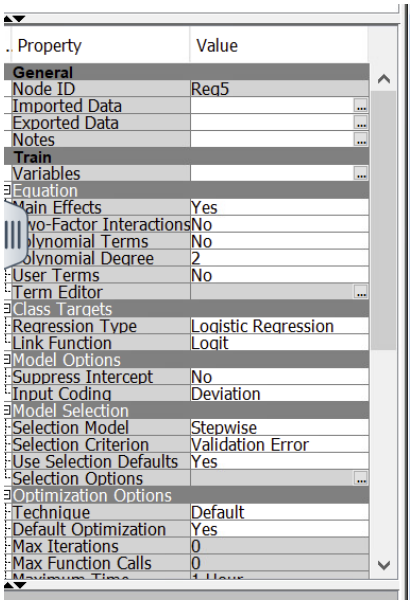
Output

Odds Ratio Estimates

Effect		Point Estimate
REP_LOG_REP_IMP_REP_device_disti		14.183
REP_LOG_REP_REP_current_address_		1.393
REP_LOG_REP_bank_branch_count_8w		0.935
REP_LOG_REP_proposed_credit_limi		1.086
REP_LOG_REP_zip_count_4w		1.283
REP_REP_IMP_REP_bank_months_coun		1.017
REP_REP_REP_days_since_request		1.477
REP_REP_credit_risk_score		1.002
REP_REP_customer_age		1.025
REP_REP_month		1.033
REP_REP_name_email_similarity		0.325
REP_income	0.2 vs 0.8	0.521
REP_income	0.5 vs 0.8	0.610
device_os	linux vs x11	0.810
device_os	macintosh vs x11	1.861
device_os	other vs x11	1.064
device_os	windows vs x11	2.790
email_is_free	0 vs 1	0.546
employment_status	CA vs CG	0.365
employment_status	CB vs CG	0.198
employment_status	CC vs CG	0.462
employment_status	CD vs CG	0.131
employment_status	CE vs CG	0.119
employment_status	CF vs CG	0.156
foreign_request	0 vs 1	0.532
has_other_cards	0 vs 1	3.471
housing_status	BA vs BG	2.068
housing_status	BB vs BG	0.577
housing_status	BC vs BG	0.647
housing_status	BD vs BG	0.972
housing_status	BE vs BG	0.456
housing_status	BF vs BG	0.759
keep_alive_session	0 vs 1	2.005
payment_type	AA vs AE	1.590
payment_type	AB vs AE	2.192
payment_type	AC vs AE	2.903
payment_type	AD vs AE	2.246
phone_home_valid	0 vs 1	2.579
phone_mobile_valid	0 vs 1	1.357

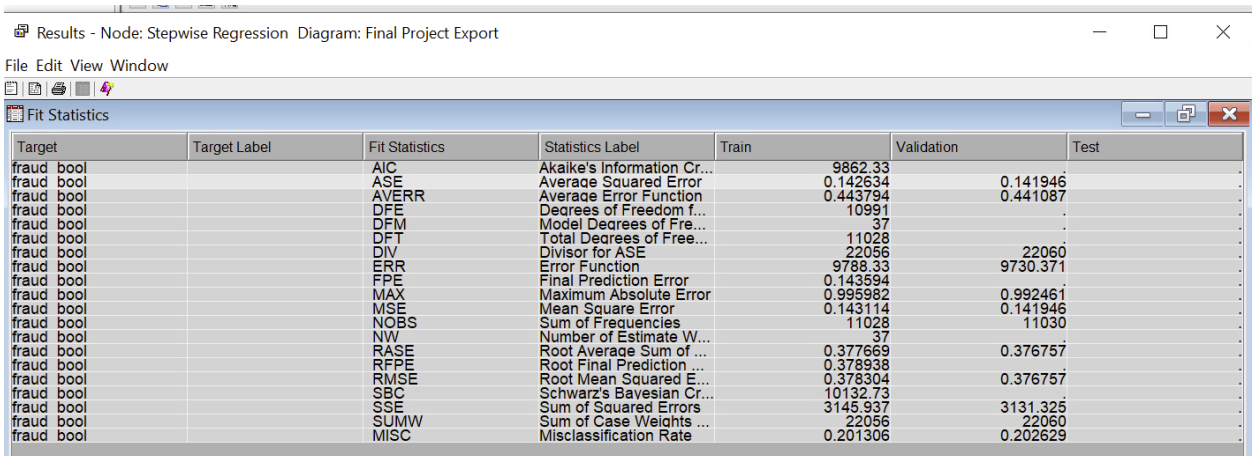
Stepwise regression

For stepwise regression, we changed the model selection to stepwise and the selection criteria is Validation error.



Property	Value
General	
Node ID	Reg5
Imported Data	
Exported Data	
Notes	
Train	
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	Validation Error
Use Selection Defaults	Yes
Selection Options	
Optimization Options	
Technique	Default
Default Optimization	Yes
Max Iterations	0
Max Function Calls	0

As per stepwise regression model our ASE is **0.141946** which is the same as forward regression.

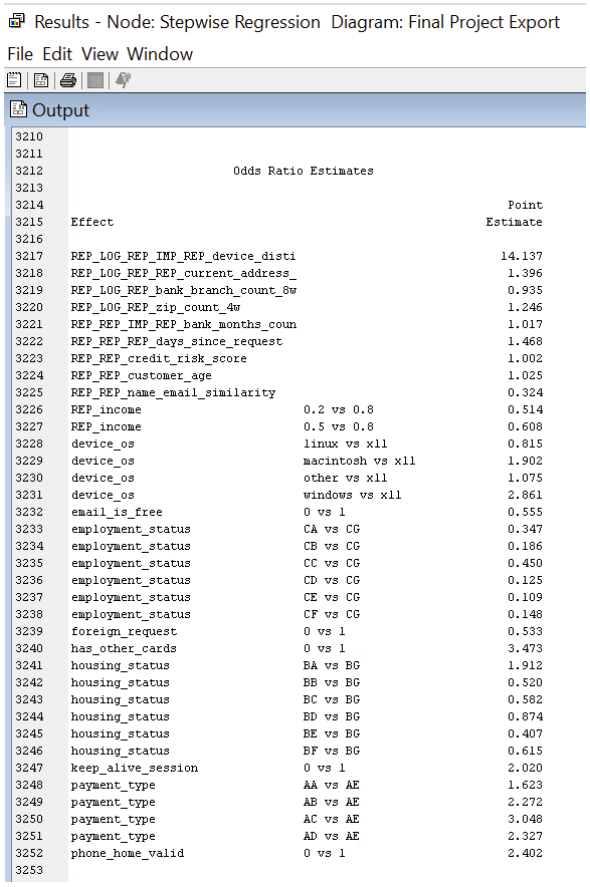


Results - Node: Stepwise Regression Diagram: Final Project Export

File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		AIC	Akaike's Information Cr...	9862.33	.	.
fraud bool		ASE	Average Squared Error	0.142634	0.141946	.
fraud bool		AVERR	Average Error Function	0.443794	0.441087	.
fraud bool		DFE	Degrees of Freedom f...	10991	.	.
fraud bool		DFM	Model Degrees of Fre...	37	.	.
fraud bool		DFT	Total Degrees of Free...	11028	.	.
fraud bool		DIV	Divisor for ASE	22056	22060	.
fraud bool		ERR	Error Function	9788.33	9730.371	.
fraud bool		FPE	Final Prediction Error	0.143594	.	.
fraud bool		MAX	Maximum Absolute Error	0.995982	0.992461	.
fraud bool		MSE	Mean Square Error	0.143114	0.141946	.
fraud bool		NOBS	Sum of Frequencies	11028	11030	.
fraud bool		NW	Number of Estimate W...	37	.	.
fraud bool		RASE	Root Average Sum of ...	0.377669	0.376757	.
fraud bool		RFPE	Root Final Prediction ...	0.378938	.	.
fraud bool		RMSE	Root Mean Squared E...	0.378304	0.376757	.
fraud bool		SBC	Schwarz's Bayesian Cr...	10132.73	.	.
fraud bool		SSE	Sum of Squared Errors	3145.937	3131.325	.
fraud bool		SUMW	Sum of Case Weights ...	22056	22060	.
fraud bool		MISC	Misclassification Rate	0.201306	0.202629	.



Results - Node: Stepwise Regression Diagram: Final Project Export

File Edit View Window

Output

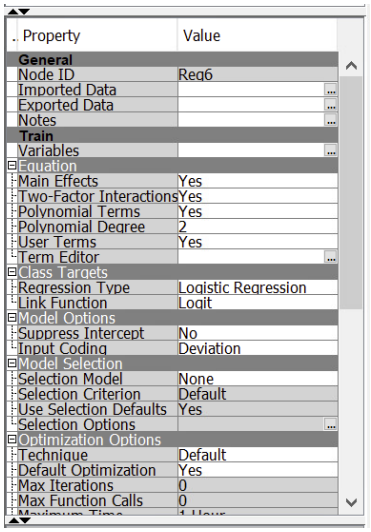
Odds Ratio Estimates

Effect		Point Estimate
REP_LOG_REP_IMP_REP_device_disti		14.137
REP_LOG_REP_REP_current_address_		1.396
REP_LOG_REP_bank_branch_count_8w		0.935
REP_LOG_REP_zip_count_4w		1.246
REP_REP_IMP_REP_bank_months_coun		1.017
REP_REP_REP_days_since_request		1.468
REP_REP_credit_risk_score		1.002
REP_REP_customer_age		1.025
REP_REP_name_email_similarity		0.324
REP_income	0.2 vs 0.8	0.514
REP_income	0.5 vs 0.8	0.608
device_os	linux vs x11	0.815
device_os	macintosh vs x11	1.902
device_os	other vs x11	1.075
device_os	windows vs x11	2.861
email_is_free	0 vs 1	0.555
employment_status	CA vs CG	0.347
employment_status	CB vs CG	0.186
employment_status	CC vs CG	0.450
employment_status	CD vs CG	0.125
employment_status	CE vs CG	0.109
employment_status	CF vs CG	0.148
foreign_request	0 vs 1	0.533
has_other_cards	0 vs 1	3.473
housing_status	BA vs BG	1.912
housing_status	BB vs BG	0.520
housing_status	BC vs BG	0.582
housing_status	BD vs BG	0.874
housing_status	BE vs BG	0.407
housing_status	BF vs BG	0.615
keep_alive_session	0 vs 1	2.020
payment_type	AA vs AE	1.623
payment_type	AB vs AE	2.272
payment_type	AC vs AE	3.048
payment_type	AD vs AE	2.327
phone_home_valid	0 vs 1	2.402

Customers with REP_LOG_REP_IMP_REP_device_disti have approximately 14.1 times higher odds of being involved in bank fraud compared to customers with a lower count of distinct devices. Our missing variable which has been treated after impute, REP_REP_IMP_REP_bank_months_coun for each unit increase in it, the odds of bank fraud increases by approximately 1.6%. Customers with REP_LOG_REP_REP_current_address have approximately 1.4 times higher odds of being associated with bank application fraud compared to customers without a current address. For every one unit increase in the bank branch count, the odds of bank fraud decrease by approximately 6.5%. . Customers with device_os of x11 are 90% more likely to be involved in fraud as compared to those with macintosh. Customers with housing_status of BG the odds of bank fraud increase by approximately 90% more likely to be involved in fraud as compared with customers with housing_status of BA. payment_type AE is 3 times more related to fraud as compared with AC. Customers with foreign_request the odds of bank fraud decreased by 46.7% as compared to those without foreign_request. Customers with employment_status of CG the odds of bank fraud decrease by approximately 55% as compared with customers with employment_status of CC.

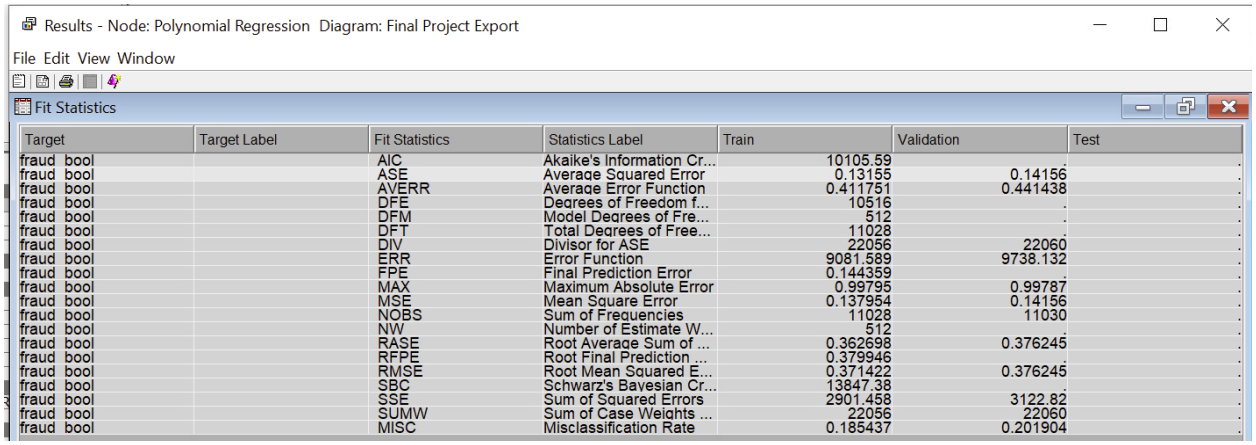
Polynomial regression

For stepwise regression we didn't change any model selection and the selection criteria but instead we made changes in the equation tab.



Property	Value
General	
Node ID	Reg6
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Equation	
Main Effects	Yes
Two-Factor Interactions	Yes
Polynomial Terms	Yes
Polynomial Degree	2
User Terms	Yes
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	Yes
Selection Options	
Optimization Options	
Technique	Default
Default Optimization	Yes
Max Iterations	0
Max Function Calls	0

As per polynomial regression model, our ASE is **0.14156** which is best amongst all the regression models.

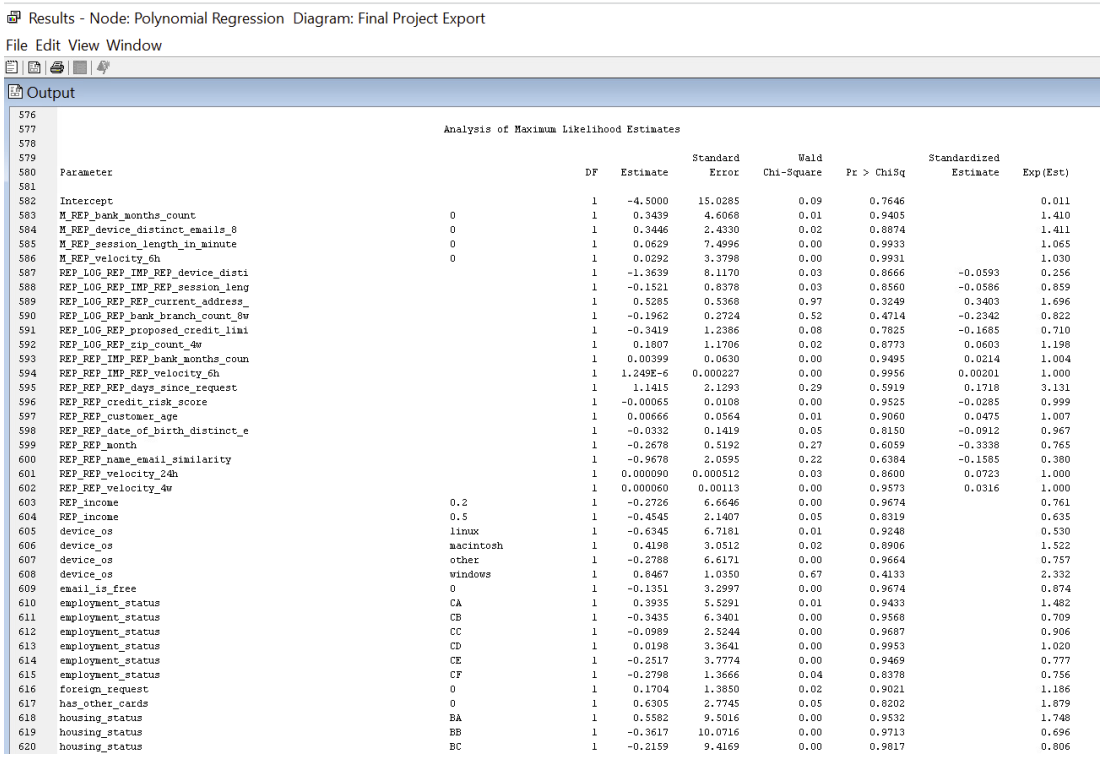


Results - Node: Polynomial Regression Diagram: Final Project Export

File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		AIC	Akaike's Information Cr...	10105.59	.	.
fraud bool		ASE	Average Squared Error	0.13155	0.14156	.
fraud bool		AVERR	Average Error Function	0.411751	0.441438	.
fraud bool		DFE	Degrees of Freedom f...	10516	.	.
fraud bool		DFM	Model Degrees of Fre...	512	.	.
fraud bool		DFT	Total Degrees of Free...	11028	.	.
fraud bool		DIV	Divisor for ASE	22056	22060	.
fraud bool		ERR	Error Function	9081.589	9738.132	.
fraud bool		FPE	Final Prediction Error	0.144359	.	.
fraud bool		MAX	Maximum Absolute Error	0.99795	0.99787	.
fraud bool		MSE	Mean Square Error	0.137954	0.14156	.
fraud bool		NOBS	Sum of Frequencies	11028	11030	.
fraud bool		NW	Number of Estimate W...	512	.	.
fraud bool		RASE	Root Average Sum of ...	0.362698	0.376245	.
fraud bool		RFPE	Root Final Prediction ...	0.379946	.	.
fraud bool		RMSE	Root Mean Squared E...	0.371422	0.376245	.
fraud bool		SBC	Schwarz's Bayesian Cr...	13847.38	.	.
fraud bool		SSE	Sum of Squared Errors	2901.458	3122.82	.
fraud bool		SUMW	Sum of Case Weights ...	22056	22060	.
fraud bool		MISC	Misclassification Rate	0.185437	0.201904	.



Results - Node: Polynomial Regression Diagram: Final Project Export

File Edit View Window

Output

Analysis of Maximum Likelihood Estimates

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Standardized Estimate	Exp(Est)
Intercept		1	-4.5000	15.0285	0.09	0.7646		0.011
M_REP_bank_months_count	0	1	0.3439	4.6068	0.01	0.9405		1.410
M_REP_device_distinct_emails_8	0	1	0.3446	2.4330	0.02	0.8874		1.411
M_REP_session_length_in_minute	0	1	0.0629	7.4996	0.00	0.9933		1.065
M_REP_velocity_6h	0	1	0.0292	3.3798	0.00	0.9931		1.030
REP_LOG_REP_IMP_REP_device_disti		1	-1.3639	8.1170	0.03	0.8666	-0.0593	0.256
REP_LOG_REP_IMP_REP_session_leng		1	-0.1521	0.8378	0.03	0.8560	-0.0586	0.859
REP_LOG_REP_REP_current_address_		1	0.5285	0.5368	0.97	0.3249	0.3403	1.696
REP_LOG_REP_bank_branch_count_8w		1	-0.1962	0.2724	0.52	0.4714	-0.2342	0.822
REP_LOG_REP_proposed_credit_limi		1	-0.3419	1.2386	0.08	0.7825	-0.1685	0.710
REP_LOG_REP_zip_count_4w		1	0.1807	1.1706	0.02	0.8773	0.0603	1.198
REP_REP_IMP_REP_bank_months_coun		1	0.00399	0.0630	0.00	0.9495	0.0214	1.004
REP_REP_IMP_REP_velocity_6h		1	1.249E-6	0.000227	0.00	0.9956	0.00201	1.000
REP_REP_REP_days_since_request		1	1.1415	2.1293	0.29	0.5919	0.1718	3.131
REP_REP_credit_risk_score		1	-0.00065	0.0108	0.00	0.9525	-0.0285	0.999
REP_REP_customer_age		1	0.00666	0.0564	0.01	0.9060	0.0475	1.007
REP_REP_date_of_birth_distinct_e		1	-0.0332	0.1419	0.05	0.8150	-0.0912	0.967
REP_REP_month		1	-0.2678	0.5192	0.27	0.6059	-0.3338	0.765
REP_REP_name_email_similarity		1	-0.9678	2.0595	0.22	0.6384	-0.1585	0.380
REP_REP_velocity_24h		1	0.000090	0.000512	0.03	0.8600	0.0723	1.000
REP_REP_velocity_4w		1	0.000060	0.00113	0.00	0.9573	0.0316	1.000
REP_income	0.2	1	-0.2726	6.6646	0.00	0.9674		0.761
REP_income	0.5	1	-0.4545	2.1407	0.05	0.8319		0.635
device_os	linux	1	-0.6345	6.7181	0.01	0.9248		0.530
device_os	macintosh	1	0.4198	3.0512	0.02	0.8906		1.522
device_os	other	1	-0.2788	6.6171	0.00	0.9664		0.757
device_os	windows	1	0.8467	1.0350	0.67	0.4133		2.332
email_is_free	0	1	-0.1351	3.2997	0.00	0.9674		0.874
employment_status	CA	1	0.3935	5.5291	0.01	0.9433		1.482
employment_status	CB	1	-0.3435	6.3401	0.00	0.9568		0.709
employment_status	CC	1	-0.0989	2.5244	0.00	0.9687		0.906
employment_status	CD	1	0.0198	3.3641	0.00	0.9953		1.020
employment_status	CE	1	-0.2517	3.7774	0.00	0.9469		0.777
employment_status	CF	1	-0.2798	1.3666	0.04	0.8378		0.756
foreign_request	0	1	0.1704	1.3850	0.02	0.9021		1.186
has_other_cards	0	1	0.6305	2.7745	0.05	0.8202		1.879
housing_status	BA	1	0.5582	9.5016	0.00	0.9532		1.748
housing_status	BB	1	-0.3617	10.0716	0.00	0.9713		0.696
housing_status	BC	1	-0.2159	9.4169	0.00	0.9817		0.806

Unlike other regressions, polynomial regression is a bit harder to comprehend. As for the results, the odd estimates may not be the best parameter to associate correlations among the dependent and independent variables. We will be prioritizing ASE in this case.

Neural Networks

A neural network is a collection of linked input-output variables, where each link has a certain weight that affects the result. The input variables for neural networks are linear combinations of nonlinear functions. This methodology is strong as well as very generic for both regression and classification, and it has been proven to be the most effective machine learning technique for a variety of issues.

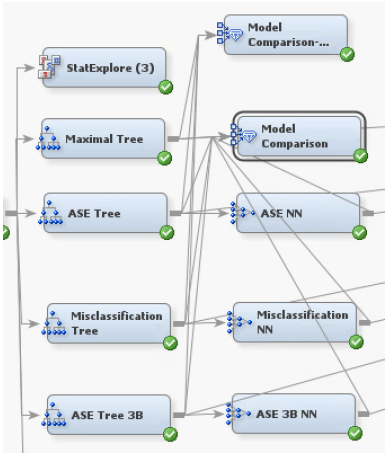
We attached neural nodes to 5 sections:

- Decision Trees NN
- Forward Regression NN
- Polynomial Regression NN
- Data Manipulation NN
- Additional NN

Decision Trees Neural Networks

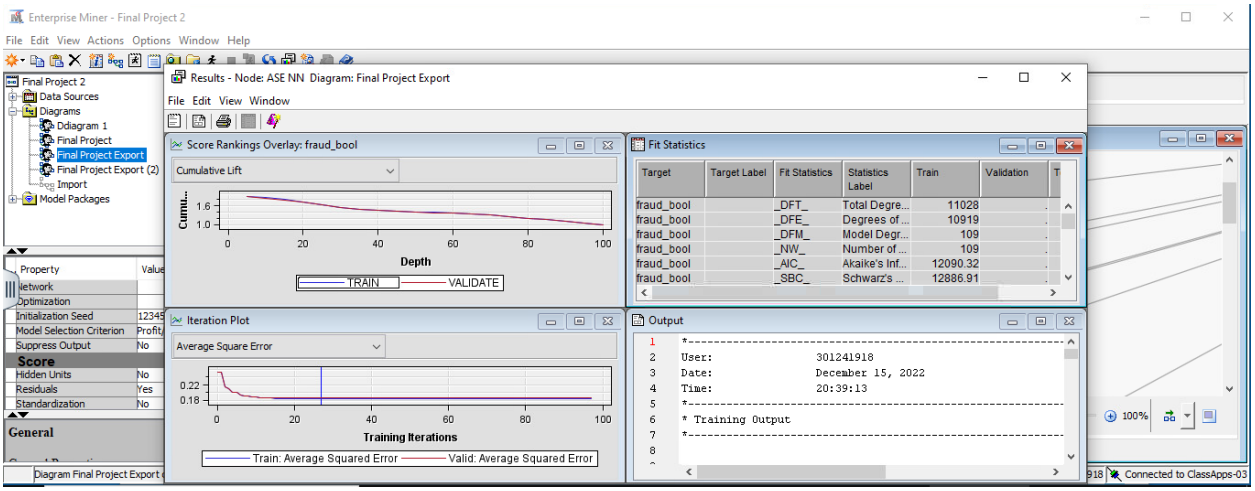
We attached Neural Nodes to our optimal trees to experiment if the error rates get better or not. We kept the number of iterations at 100, and then turned off any preliminary training. Furthermore, we kept the number of hidden units at the default setting which is three.

Screenshot of our NNs attached to optimal trees is given below:

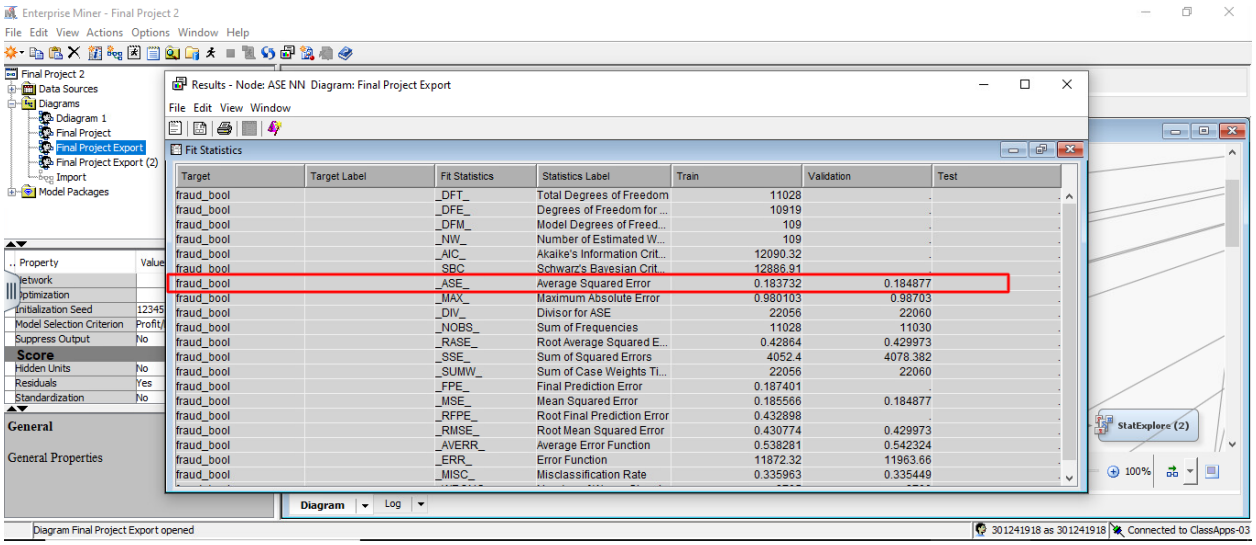


ASE Neural Network

The below screenshot shows the results we derived from the ASE neural network node. As per the iteration plot, 27 iterations is the best cut-off point as per average squared error metric.

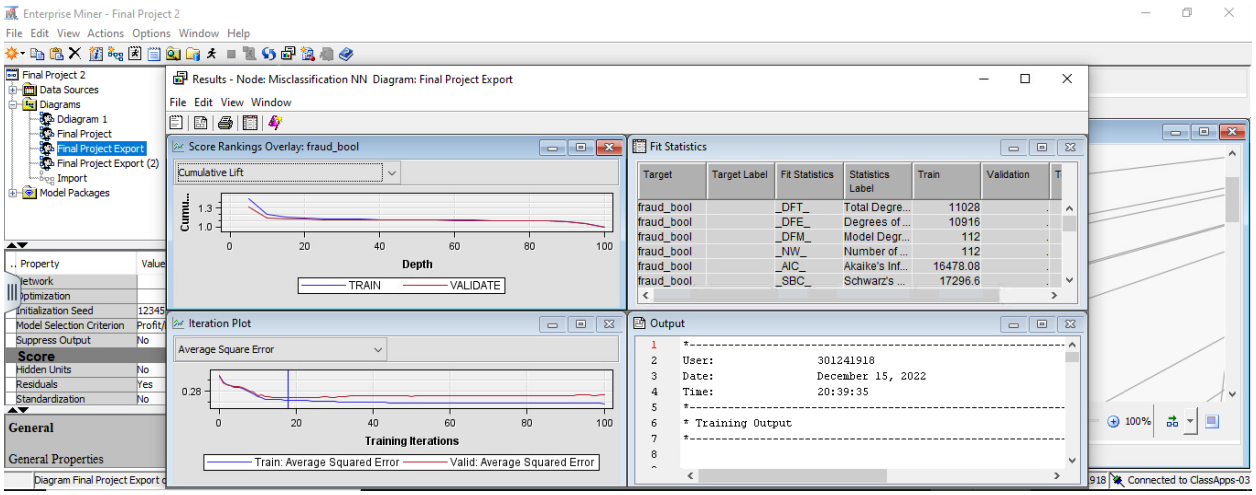


The average square error for ASE Neural network is 0.184877 which is worse than ASE tree with 0.170573 error rate. Hence, this Neural node did not add to ASE tree's efficiency.

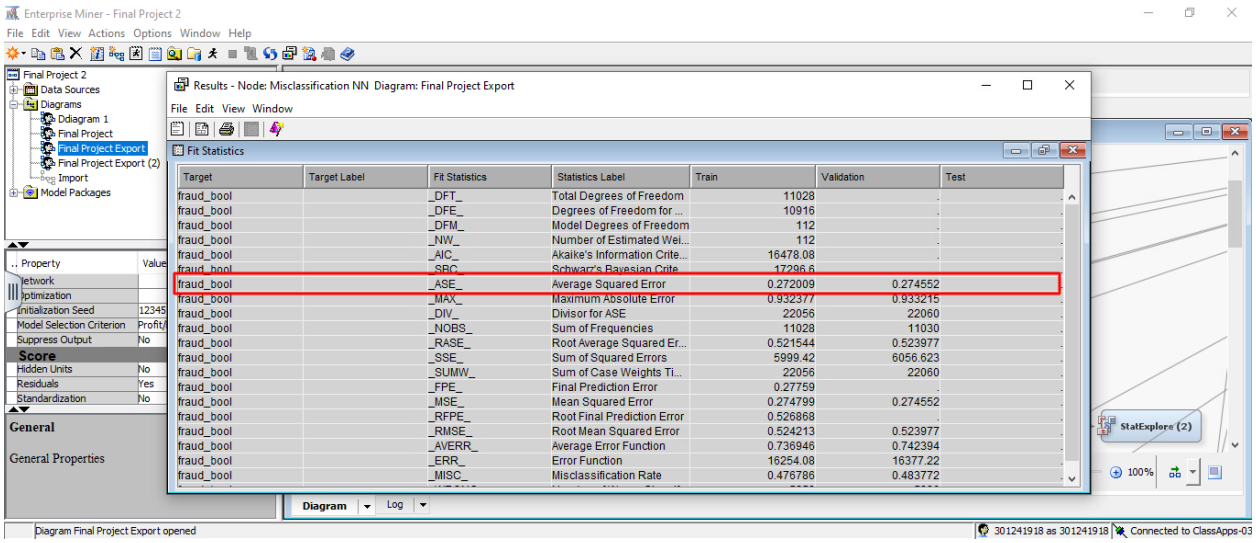


Misclassification Neural Network

The below screenshot shows the results we derived from the Misclassification neural network node. Unlike, ASE NN , we achieved an iteration cut-off at 18 as per average squared error metric.

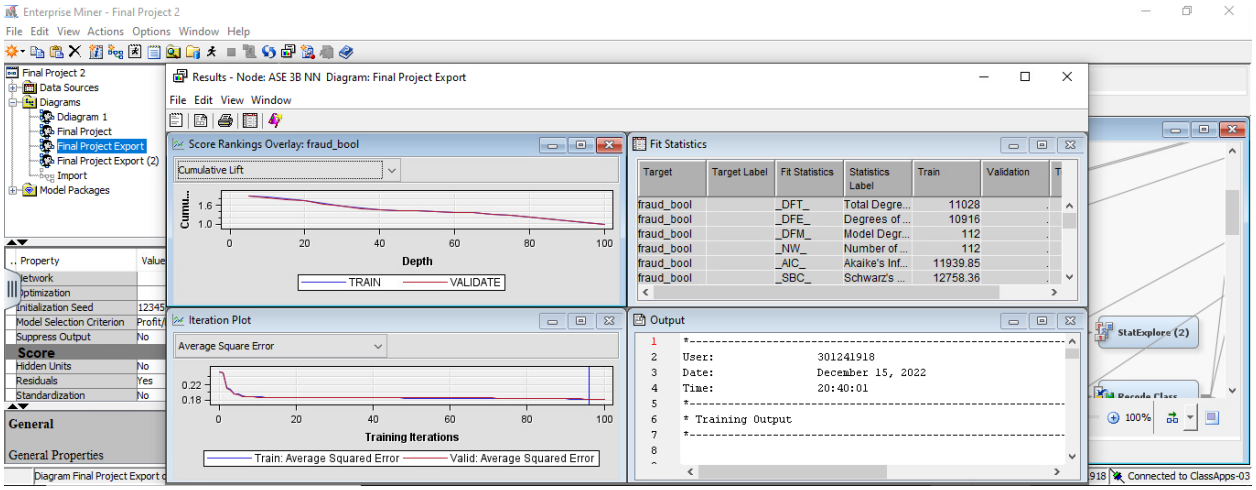


The average square error for Misclassification Neural network is 0.274552 which is highest among all three networks which is much worse than Misclassification Tree at 0.175272 error rate.

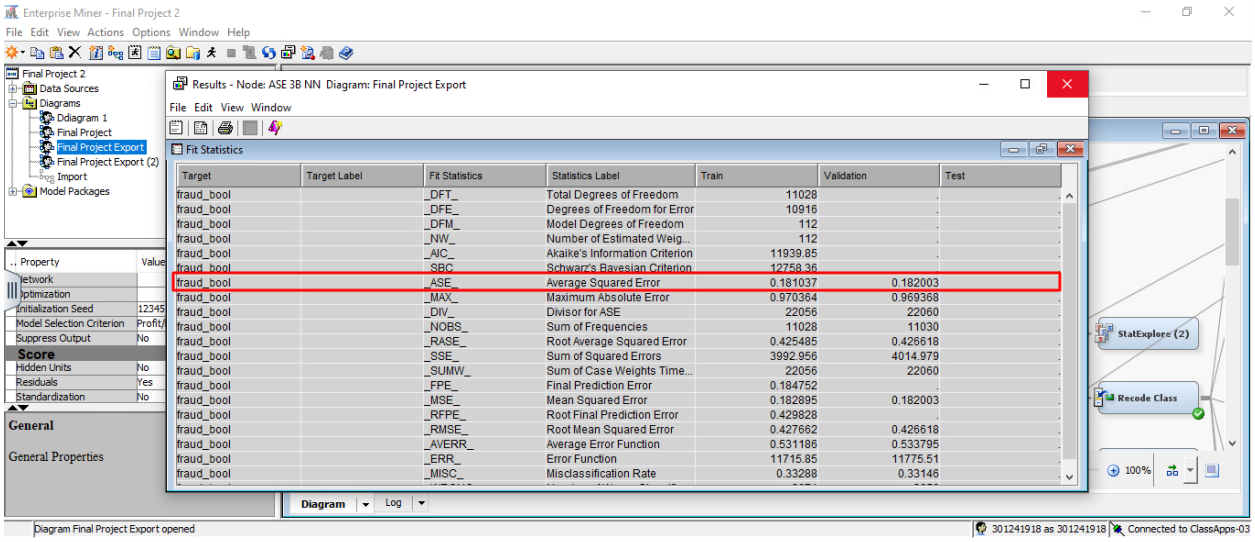


ASE 3B Neural Network

The number of suggested iterations for this model is an astonishing 96. This is the highest so far for NNs.



However, the average square error for ASE 3B Neural network is 0.182003 which is lowest among all three nodes. In comparison to ASE 3B Tree, it is much higher, approximately by 0.01.

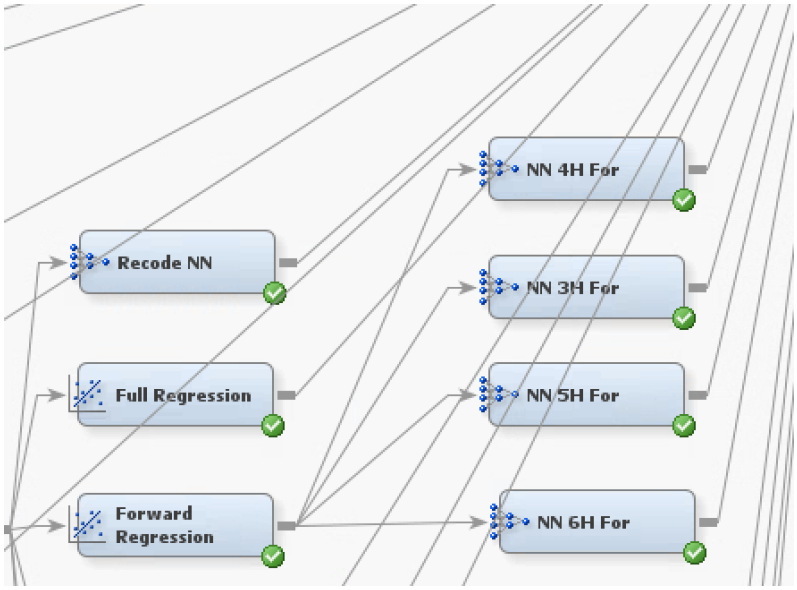


Summary: Decision Tree NN

Model	ASE NN	Misclassification NN	ASE 3Branch NN
Average Squared Error	0.184877	0.274552	0.182003
Misclassification Rate	0.335449	0.483772	0.33146

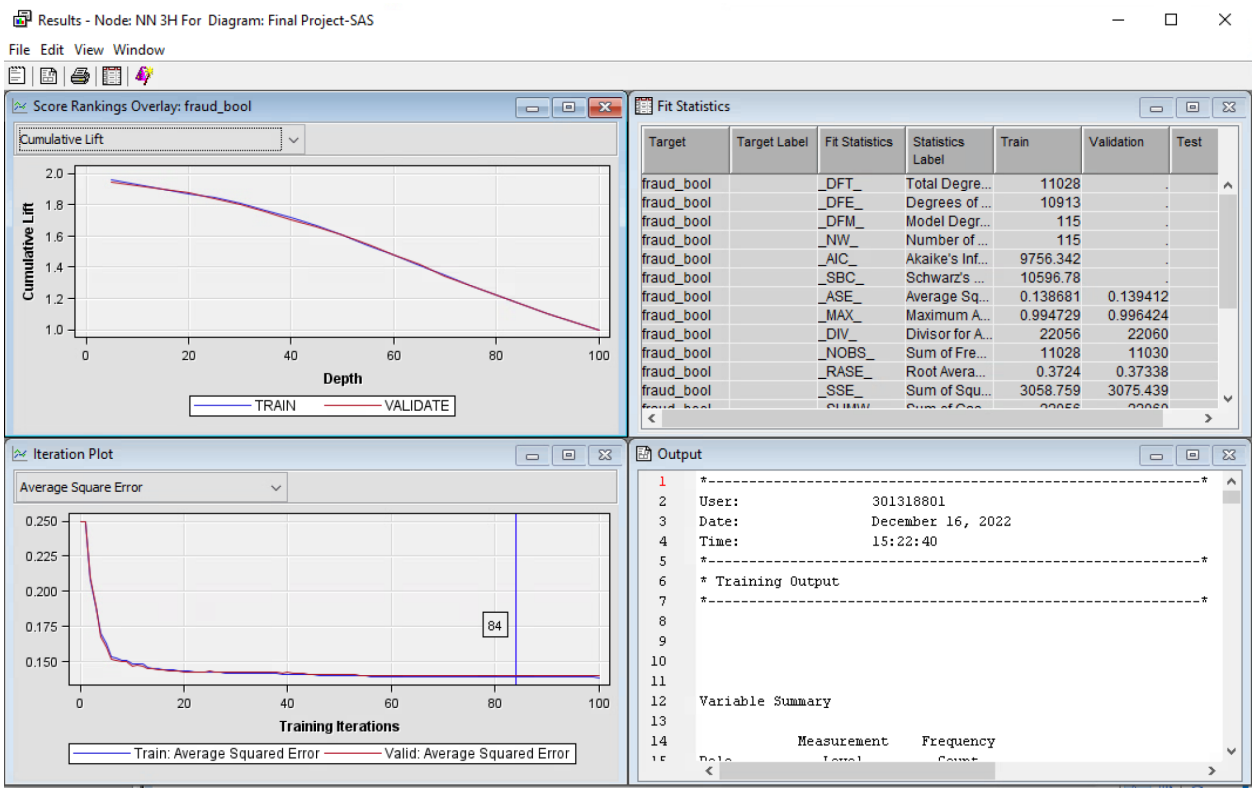
Forward Regression Neural Networks

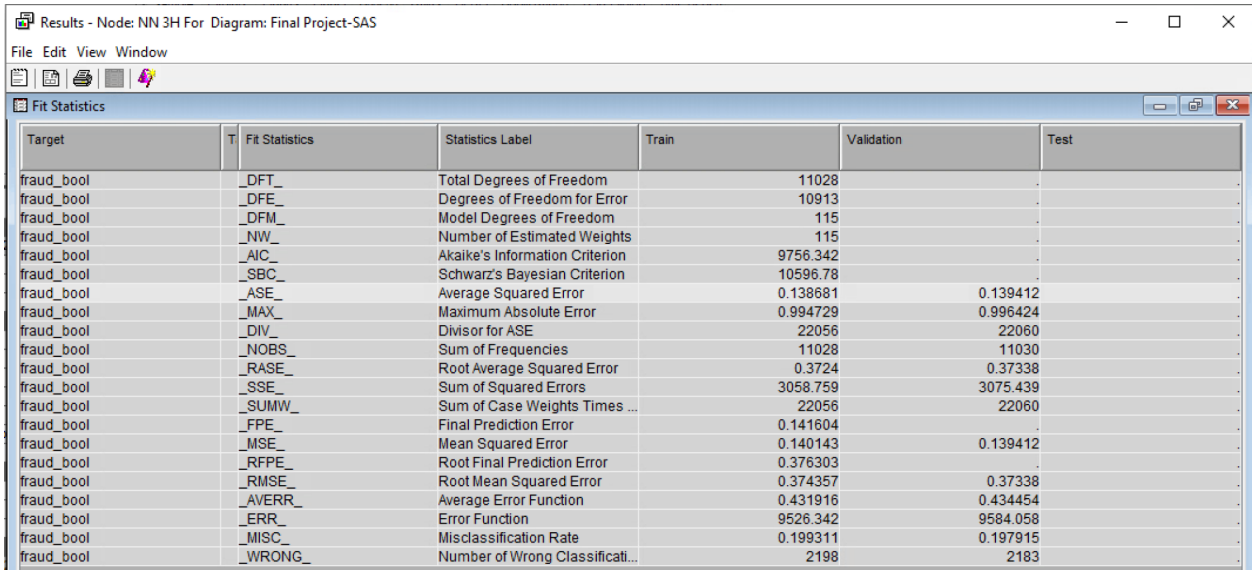
Here we connect the neural network with the forward regression with various hidden units and iterations. This has been done as one of the last steps in our project. We attached NNs with multiple hidden units to find increasing or decresing efficiency. We started with the default setting of 3 and went upwards. We experimented till the point efficiency started faltering.



3 Hidden Unit Neural Network (100 iterations)

The average square error of a neural network with 3 hidden unit and 100 iterations is 0.139412 with cut-off iterations at 84. Compared to Forward Regression, the error rate has improved from 0.141936

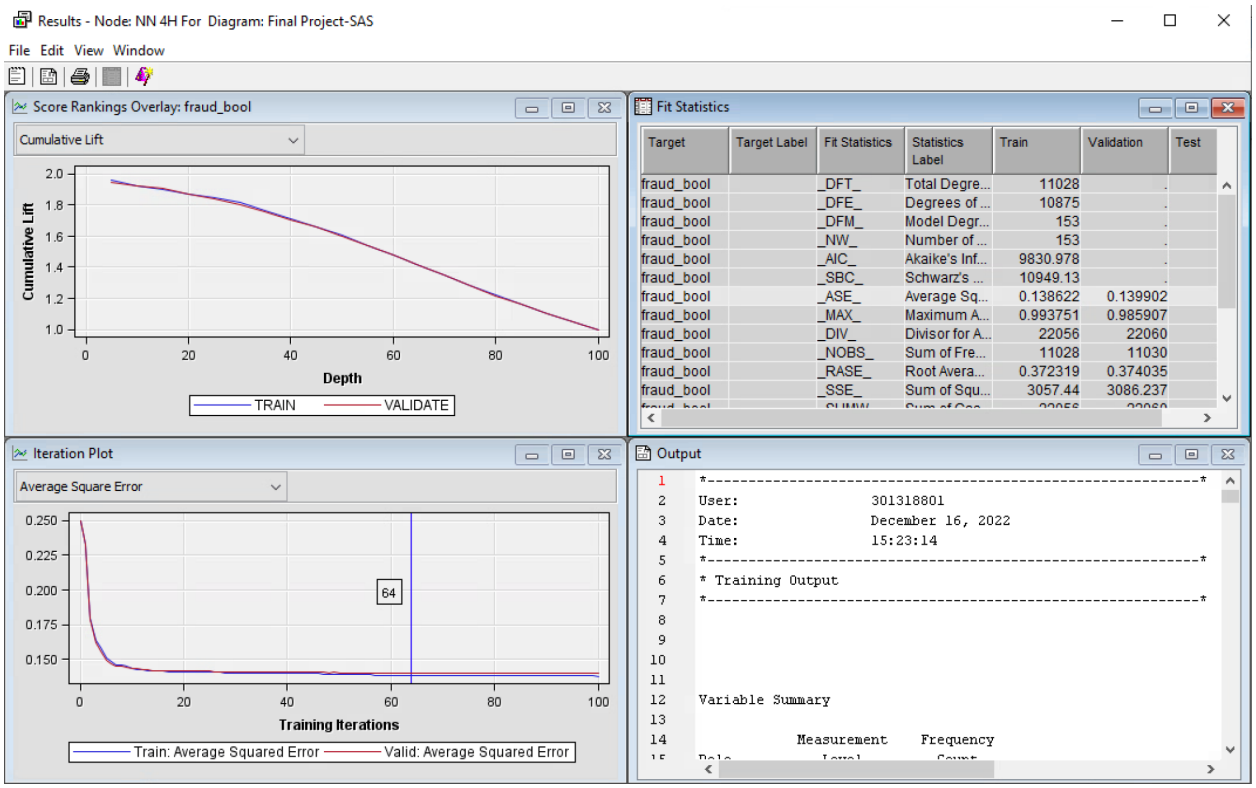




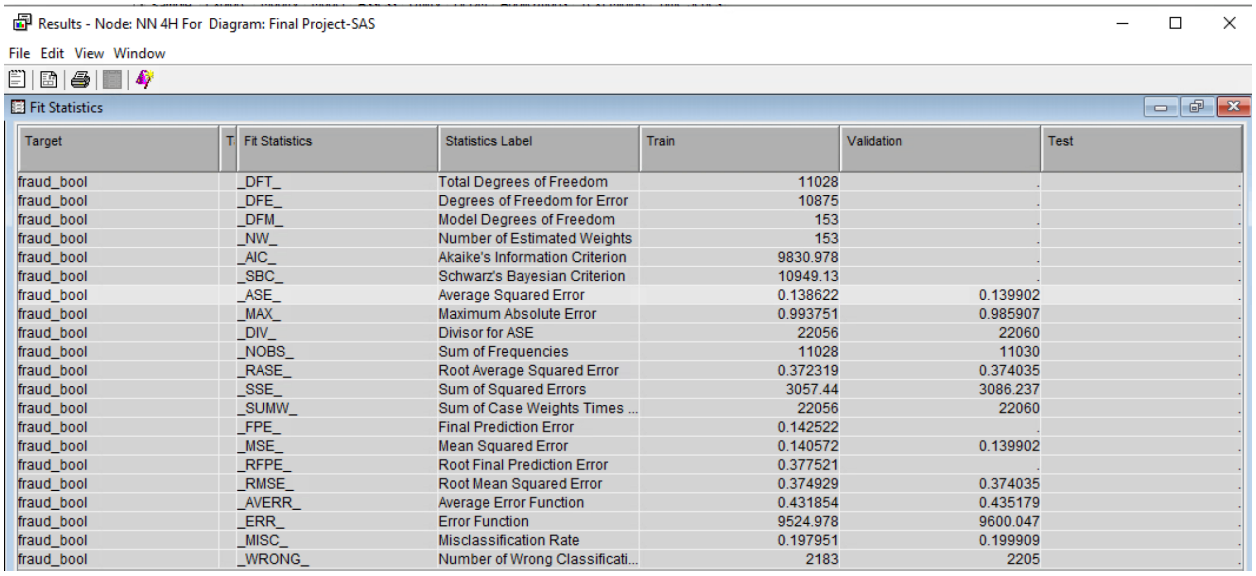
Target	T	Fit Statistics	Statistics Label	Train	Validation	Test
fraud_bool		_DFT_	Total Degrees of Freedom	11028	.	.
fraud_bool		_DFE_	Degrees of Freedom for Error	10913	.	.
fraud_bool		_DFM_	Model Degrees of Freedom	115	.	.
fraud_bool		_NW_	Number of Estimated Weights	115	.	.
fraud_bool		_AIC_	Akaike's Information Criterion	9756.342	.	.
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	10596.78	.	.
fraud_bool		_ASE_	Average Squared Error	0.138681	0.139412	.
fraud_bool		_MAX_	Maximum Absolute Error	0.994729	0.996424	.
fraud_bool		_DIV_	Divisor for ASE	22056	22060	.
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030	.
fraud_bool		_RASE_	Root Average Squared Error	0.3724	0.37338	.
fraud_bool		_SSE_	Sum of Squared Errors	3058.759	3075.439	.
fraud_bool		_SUMW_	Sum of Case Weights Times ...	22056	22060	.
fraud_bool		_FPE_	Final Prediction Error	0.141604	.	.
fraud_bool		_MSE_	Mean Squared Error	0.140143	0.139412	.
fraud_bool		_RFPE_	Root Final Prediction Error	0.376303	.	.
fraud_bool		_RMSE_	Root Mean Squared Error	0.374357	0.37338	.
fraud_bool		_AVERR_	Average Error Function	0.431916	0.434454	.
fraud_bool		_ERR_	Error Function	9526.342	9584.058	.
fraud_bool		_MISC_	Misclassification Rate	0.199311	0.197915	.
fraud_bool		_WRONG_	Number of Wrong Classificati...	2198	2183	.

4 Hidden Unit Neural Network (100 iterations)

With the hypothesis of improving reliability we kept on increasing hidden units, we approached 4 hidden units. The average square error of a neural network with 4 hidden unit and 100 iterations is 0.139902 which is slightly lower than 3 hidden unit neural network..



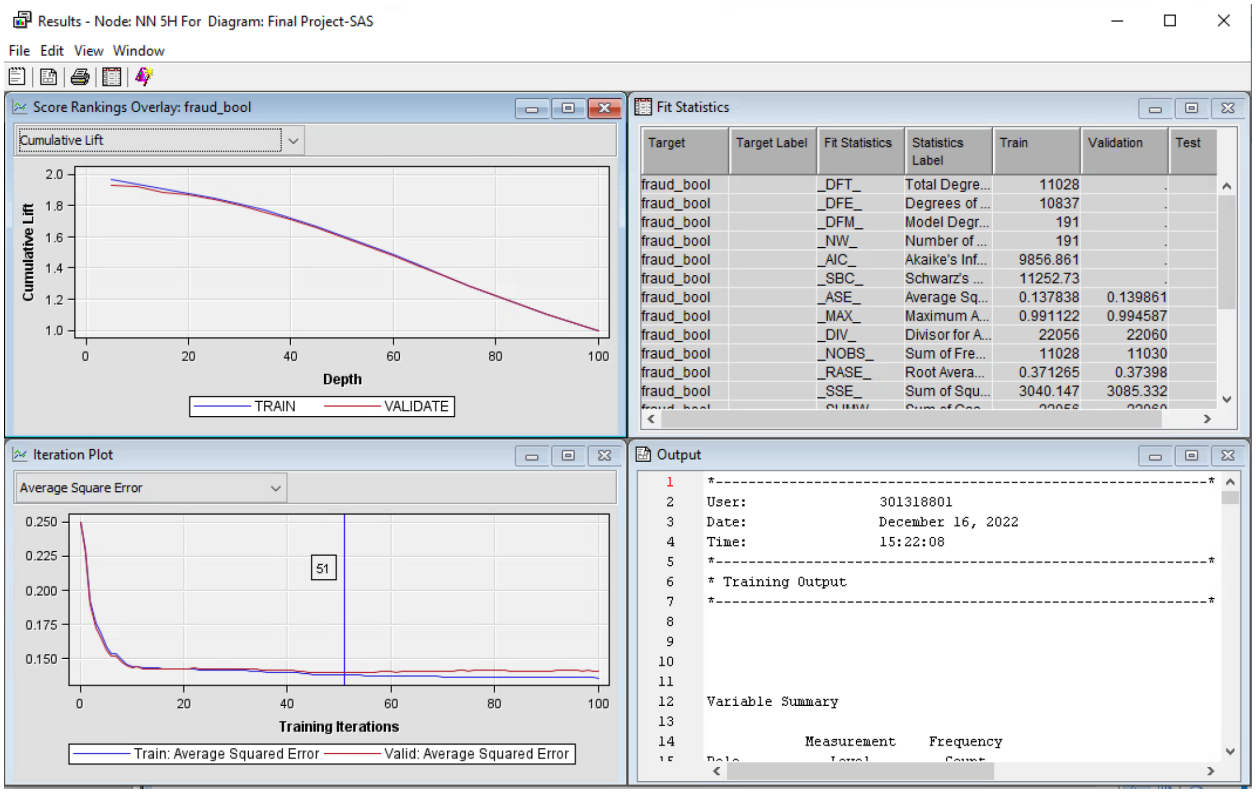
Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud_bool		_DFT_	Total Degre...	11028	.	
fraud_bool		_DFE_	Degrees of ...	10875	.	
fraud_bool		_DFM_	Model Degr...	153	.	
fraud_bool		_NW_	Number of ...	153	.	
fraud_bool		_AIC_	Akaike's Inf...	9830.978	.	
fraud_bool		_SBC_	Schwarz's ...	10949.13	.	
fraud_bool		_ASE_	Average Sq...	0.138622	0.139902	
fraud_bool		_MAX_	Maximum A...	0.993751	0.985907	
fraud_bool		_DIV_	Divisor for A...	22056	22060	
fraud_bool		_NOBS_	Sum of Fre...	11028	11030	
fraud_bool		_RASE_	Root Avera...	0.372319	0.374035	
fraud_bool		_SSE_	Sum of Squ...	3057.44	3086.237	



Target	T	Fit Statistics	Statistics Label	Train	Validation	Test
fraud_bool		_DFT_	Total Degrees of Freedom	11028	.	.
fraud_bool		_DFE_	Degrees of Freedom for Error	10875	.	.
fraud_bool		_DFM_	Model Degrees of Freedom	153	.	.
fraud_bool		_NW_	Number of Estimated Weights	153	.	.
fraud_bool		_AIC_	Akaike's Information Criterion	9830.978	.	.
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	10949.13	.	.
fraud_bool		_ASE_	Average Squared Error	0.138622	0.139902	.
fraud_bool		_MAX_	Maximum Absolute Error	0.993751	0.985907	.
fraud_bool		_DIV_	Divisor for ASE	22056	22060	.
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030	.
fraud_bool		_RASE_	Root Average Squared Error	0.372319	0.374035	.
fraud_bool		_SSE_	Sum of Squared Errors	3057.44	3086.237	.
fraud_bool		_SUMW_	Sum of Case Weights Times ...	22056	22060	.
fraud_bool		_FPE_	Final Prediction Error	0.142522	.	.
fraud_bool		_MSE_	Mean Squared Error	0.140572	0.139902	.
fraud_bool		_RFPE_	Root Final Prediction Error	0.377521	.	.
fraud_bool		_RMSE_	Root Mean Squared Error	0.374929	0.374035	.
fraud_bool		_AVERR_	Average Error Function	0.431854	0.435179	.
fraud_bool		_ERR_	Error Function	9524.978	9600.047	.
fraud_bool		_MISC_	Misclassification Rate	0.197951	0.199909	.
fraud_bool		_WRONG_	Number of Wrong Classificati...	2183	2205	.

5 Hidden Unit Neural Network (100 iterations)

Following the trend of increasing error rates, we went ahead with 5 hidden units. The average square error of a neural network with 5 hidden unit and 100 iterations is 0.139861 which is lowest so far. The sequence of improving rates keeps going on.



Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud_bool		_DFT_	Total Degre...	11028	.	
fraud_bool		_DFE_	Degrees of ...	10837	.	
fraud_bool		_DFM_	Model Degr...	191	.	
fraud_bool		_NW_	Number of ...	191	.	
fraud_bool		_AIC_	Akaike's Inf...	9856.861	.	
fraud_bool		_SBC_	Schwarz's ...	11252.73	.	
fraud_bool		_ASE_	Average Sq...	0.137838	0.139861	
fraud_bool		_MAX_	Maximum A...	0.991122	0.994587	
fraud_bool		_DIV_	Divisor for A...	22056	22060	
fraud_bool		_NOBS_	Sum of Fre...	11028	11030	
fraud_bool		_RASE_	Root Avera...	0.371265	0.37398	
fraud_bool		_SSE_	Sum of Squ...	3040.147	3085.332	

Results - Node: NN 5H For Diagram: Final Project-SAS

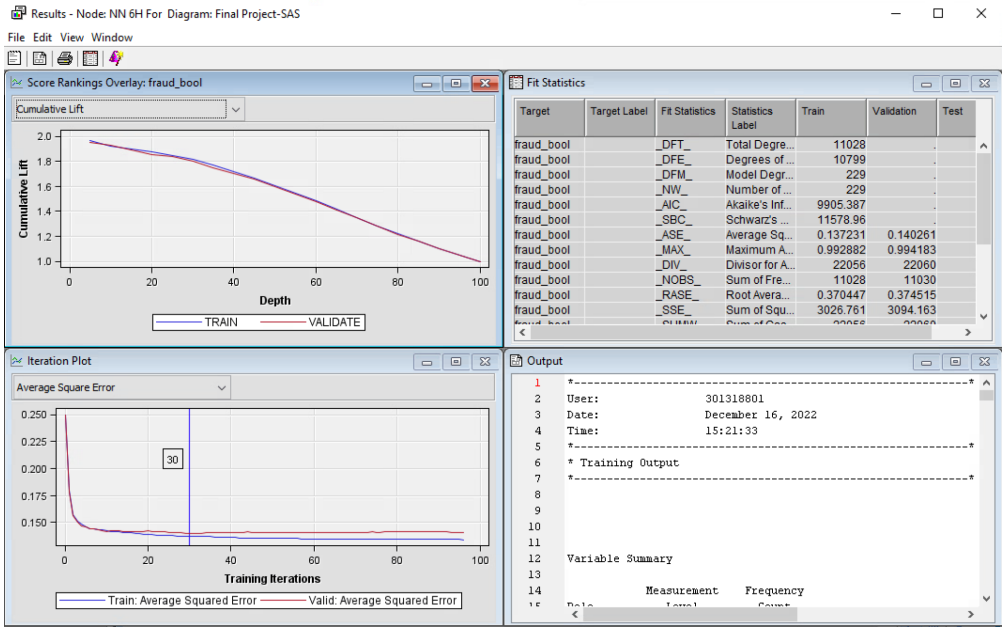
File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_DFT_	Total Degrees of Freedom	11028	.
fraud_bool		_DFE_	Degrees of Freedom for Error	10837	.
fraud_bool		_DFM_	Model Degrees of Freedom	191	.
fraud_bool		_NW_	Number of Estimated Weights	191	.
fraud_bool		_AIC_	Akaike's Information Criterion	9856.861	.
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	11252.73	.
fraud_bool		_ASE_	Average Squared Error	0.137838	0.139861
fraud_bool		_MAX_	Maximum Absolute Error	0.991122	0.994587
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_RASE_	Root Average Squared Error	0.371265	0.37398
fraud_bool		_SSE_	Sum of Squared Errors	3040.147	3085.332
fraud_bool		_SUMW_	Sum of Case Weights Time...	22056	22060
fraud_bool		_FPE_	Final Prediction Error	0.142696	.
fraud_bool		_MSE_	Mean Squared Error	0.140267	0.139861
fraud_bool		_RFPE_	Root Final Prediction Error	0.377752	.
fraud_bool		_RMSE_	Root Mean Squared Error	0.374522	0.37398
fraud_bool		_AVERR_	Average Error Function	0.429582	0.435588
fraud_bool		_ERR_	Error Function	9474.861	9609.066
fraud_bool		_MISC_	Misclassification Rate	0.197497	0.2
fraud_bool		_WRONG_	Number of Wrong Classifica...	2178	2206

6 Hidden Unit Neural Network (100 iterations)

The average square error of a neural network with 6 hidden unit and 100 iterations is 0.140261 which is increasing from the prior hidden unit models. Hence, we stopped here in terms of experimenting with hidden units.



Results - Node: NN 6H For Diagram: Final Project-SAS

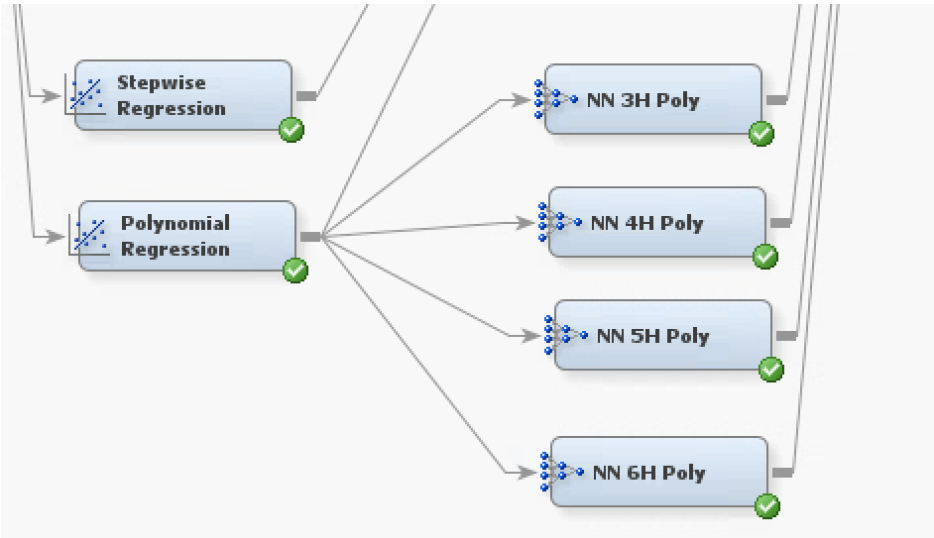
File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_DFT_	Total Degrees of Freedom	11028	.
fraud_bool		_DFE_	Degrees of Freedom for Error	10799	.
fraud_bool		_DFM_	Model Degrees of Freedom	229	.
fraud_bool		_NW_	Number of Estimated Weights	229	.
fraud_bool		_AIC_	Akaike's Information Criterion	9905.387	.
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	11578.96	.
fraud_bool		_ASE_	Average Squared Error	0.137231	0.140261
fraud_bool		_MAX_	Maximum Absolute Error	0.992882	0.994183
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_RASE_	Root Average Squared Error	0.370447	0.374515
fraud_bool		_SSE_	Sum of Squared Errors	3026.761	3094.163
fraud_bool		_SUMW_	Sum of Case Weights Times...	22056	22060
fraud_bool		_FPE_	Final Prediction Error	0.143051	.
fraud_bool		_MSE_	Mean Squared Error	0.140141	0.140261
fraud_bool		_RFPE_	Root Final Prediction Error	0.378221	.
fraud_bool		_RMSE_	Root Mean Squared Error	0.374354	0.374515
fraud_bool		_AVERR_	Average Error Function	0.428336	0.436441
fraud_bool		_ERR_	Error Function	9447.387	9627.894
fraud_bool		_MISC_	Misclassification Rate	0.199129	0.199819
fraud_bool		_WRONG_	Number of Wrong Classificat...	2196	2204

Polynomial Regression Neural Networks

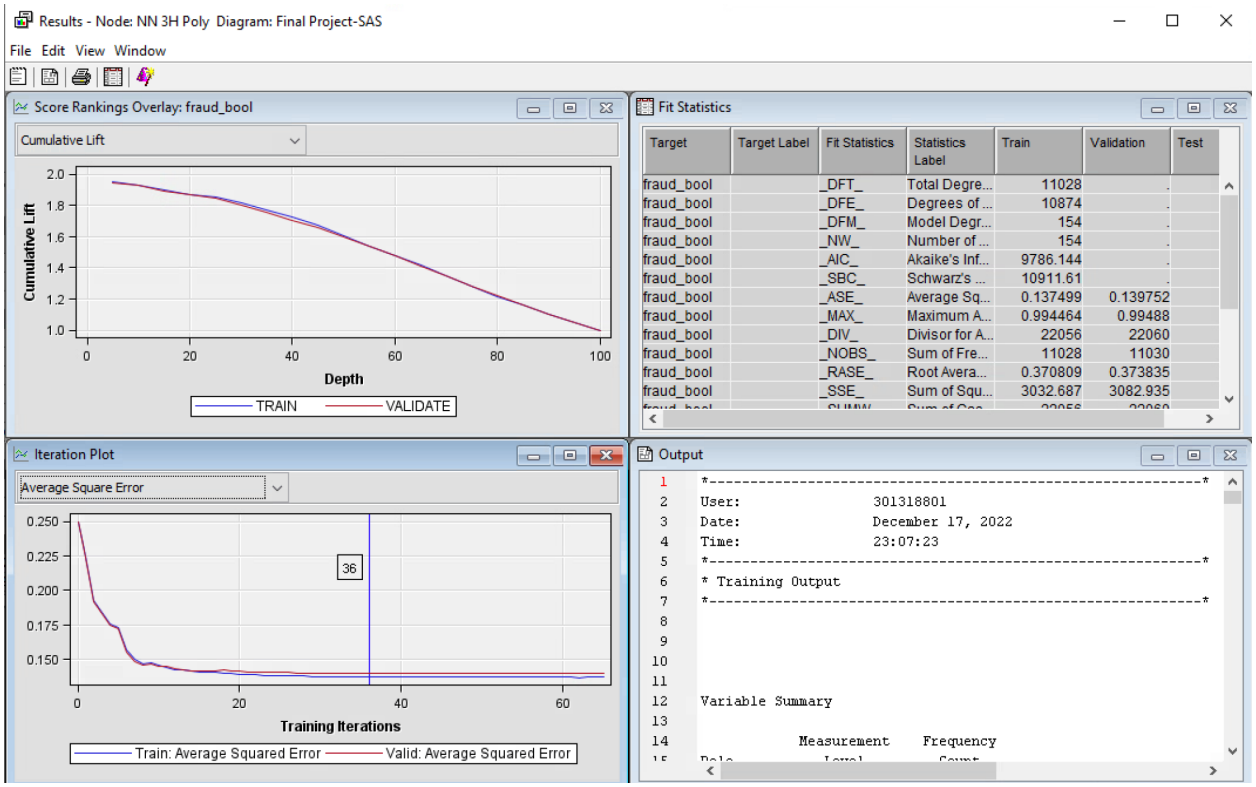
We did a polynomial regression to cater to the skews which were persistent in our project despite all the changes made through the replacement and transform nodes. Last step taken was changing 6 variables to their log format instead of default. After which all changes were made to class variables.



In the case of experimentation with hidden units, we used the same rationale as before. Exercise testing as long as efficiency is being achieved. In short, we used upto 6 hidden units and stopped there due to increasing error rates. Screenshots are shared below for each specification.

3 Hidden Unit Neural Network (100 iterations)

The ASE rate for default 3 hidden units was 0.139752. The number of iterations suggested were 36. This error rate is lower than the polynomial regression rate of 0.141826.



Results - Node: NN 3H Poly Diagram: Final Project-SAS

File Edit View Window

Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_DFT_	Total Degrees of Freedom	11028	.
fraud_bool		_DFE_	Degrees of Freedom for Error	10874	.
fraud_bool		_DFM_	Model Degrees of Freedom	154	.
fraud_bool		_NW_	Number of Estimated Weights	154	.
fraud_bool		_AIC_	Akaike's Information Criterion	9786.144	.
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	10911.61	.
fraud_bool		_ASE_	Average Squared Error	0.137499	0.139752
fraud_bool		_MAX_	Maximum Absolute Error	0.994464	0.99488
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_RASE_	Root Average Squared Error	0.370809	0.373835
fraud_bool		_SSE_	Sum of Squared Errors	3032.687	3082.935
fraud_bool		_SUMW_	Sum of Case Weights Time...	22056	22060
fraud_bool		_FPE_	Final Prediction Error	0.141394	.
fraud_bool		_MSE_	Mean Squared Error	0.139447	0.139752
fraud_bool		_RFPE_	Root Final Prediction Error	0.376024	.
fraud_bool		_RMSE_	Root Mean Squared Error	0.373426	0.373835
fraud_bool		_AVERR_	Average Error Function	0.429731	0.435211
fraud_bool		_ERR_	Error Function	9478.144	9600.748
fraud_bool		_MISC_	Misclassification Rate	0.195774	0.199365
fraud_bool		_WRONG_	Number of Wrong Classifica...	2159	2199

4 Hidden Unit Neural Network (100 iterations)

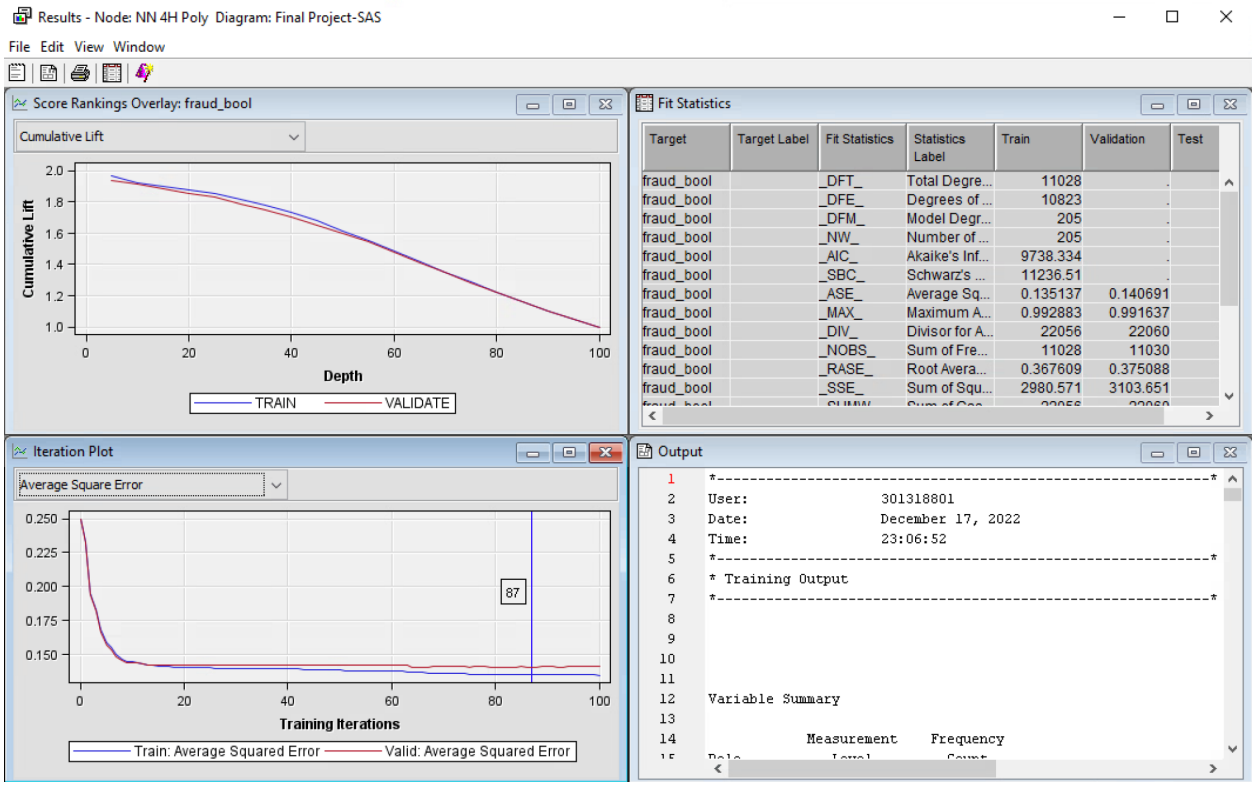
The average square error of a neural network with 4 hidden unit and 100 iterations is 0.140691.

Results - Node: NN 4H Poly Diagram: Final Project-SAS

File Edit View Window

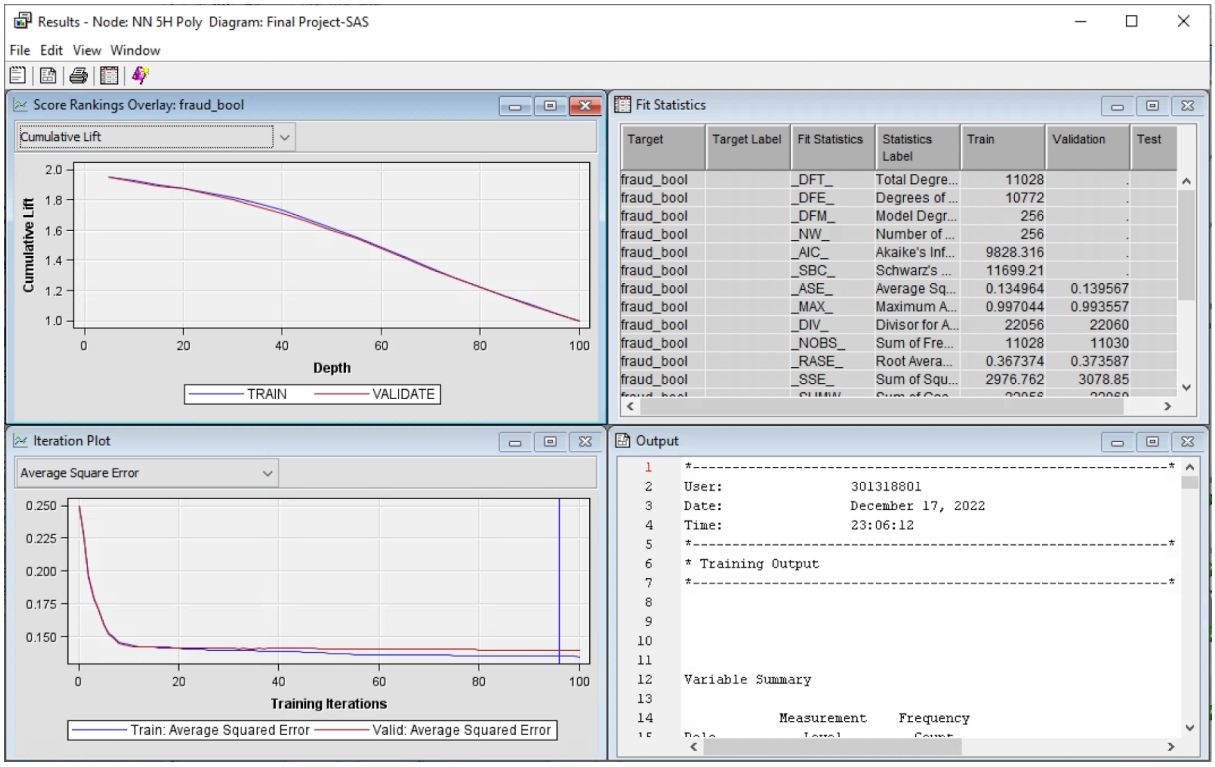
Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_DFT_	Total Degrees of Freedom	11028	.
fraud_bool		_DFE_	Degrees of Freedom for Error	10823	.
fraud_bool		_DFM_	Model Degrees of Freedom	205	.
fraud_bool		_NW_	Number of Estimated Weigh...	205	.
fraud_bool		_AIC_	Akaike's Information Criterion	9738.334	.
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	11236.51	.
fraud_bool		_ASE_	Average Squared Error	0.135137	0.140691
fraud_bool		_MAX_	Maximum Absolute Error	0.992883	0.991637
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_RASE_	Root Average Squared Error	0.367609	0.375088
fraud_bool		_SSE_	Sum of Squared Errors	2980.571	3103.651
fraud_bool		_SUMW_	Sum of Case Weights Time...	22056	22060
fraud_bool		_FPE_	Final Prediction Error	0.140256	.
fraud_bool		_MSE_	Mean Squared Error	0.137696	0.140691
fraud_bool		_RFPE_	Root Final Prediction Error	0.374507	.
fraud_bool		_RMSE_	Root Mean Squared Error	0.371074	0.375088
fraud_bool		_AVERR_	Average Error Function	0.422939	0.437438
fraud_bool		_ERR_	Error Function	9328.334	9649.882
fraud_bool		_MISC_	Misclassification Rate	0.193689	0.201179
fraud_bool		_WRONG_	Number of Wrong Classifica...	2136	2219



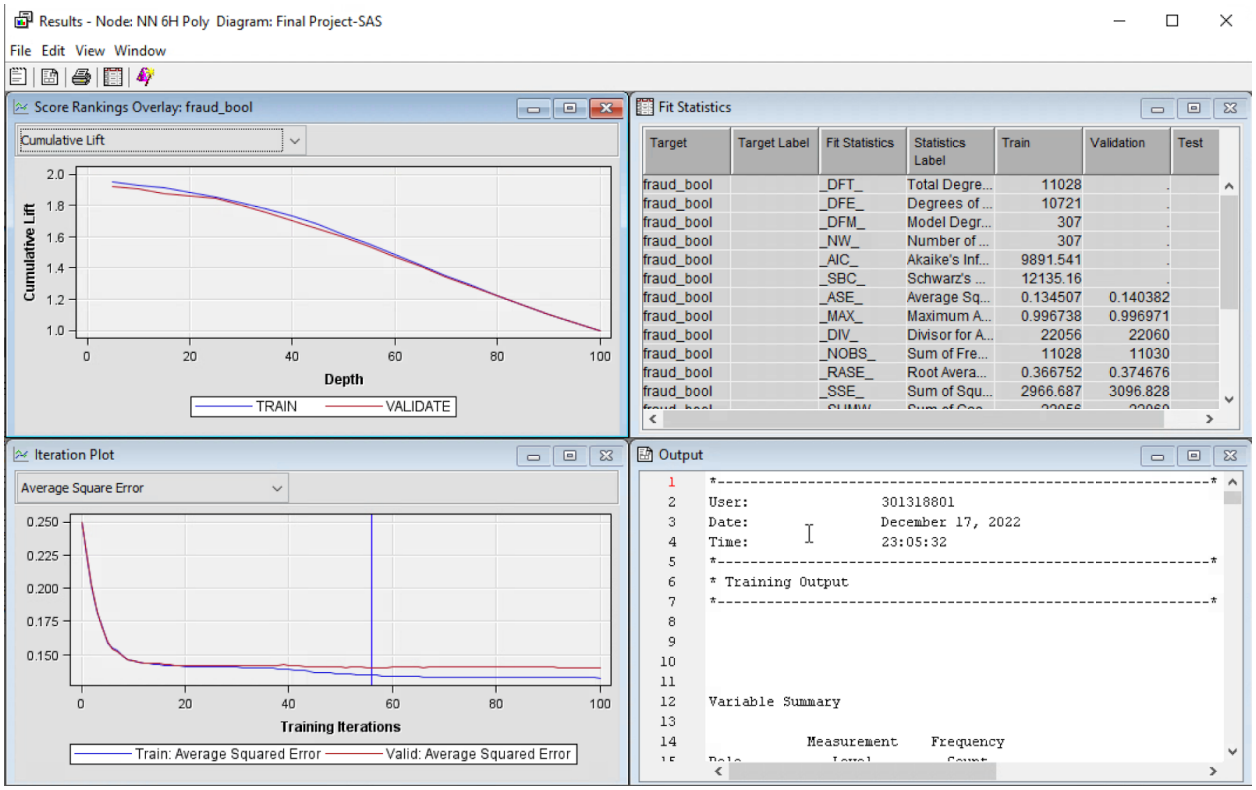
5 Hidden Unit Neural Network (100 iterations)

The average square error of a neural network with 5 hidden unit and 100 iterations is 0.139567



6 Hidden Unit Neural Network (100 iterations)

The average square error of a neural network with 6 hidden unit and 100 iterations is 0.140382



Results - Node: NN 6H Poly Diagram: Final Project-SAS

File Edit View Window

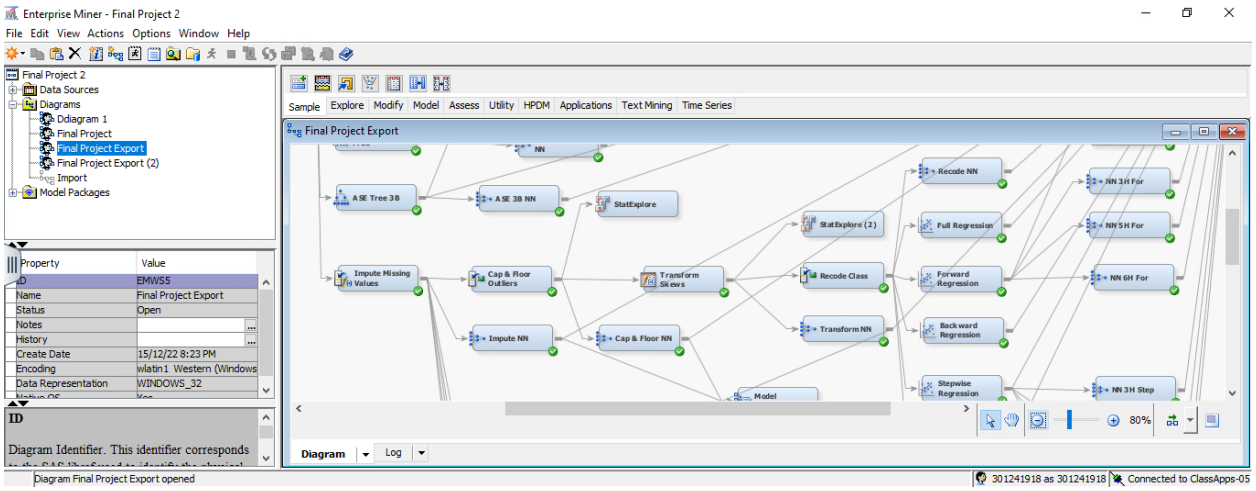
Fit Statistics

Target	Fit Statistics	Statistics Label	Train	Validation
fraud_bool	_DFT_	Total Degrees of Freedom	11028	.
fraud_bool	_DFE_	Degrees of Freedom for Error	10721	.
fraud_bool	_DFM_	Model Degrees of Freedom	307	.
fraud_bool	_NW_	Number of Estimated Weights	307	.
fraud_bool	_AIC_	Akaike's Information Criterion	9891.541	.
fraud_bool	_SBC_	Schwarz's Bayesian Criterion	12135.16	.
fraud_bool	_ASE_	Average Squared Error	0.134507	0.140382
fraud_bool	_MAX_	Maximum Absolute Error	0.996738	0.996971
fraud_bool	_DIV_	Divisor for ASE	22056	22060
fraud_bool	_NOBS_	Sum of Frequencies	11028	11030
fraud_bool	_RASE_	Root Average Squared Error	0.366752	0.374676
fraud_bool	_SSE_	Sum of Squared Errors	2966.687	3096.828
fraud_bool	_SUMW_	Sum of Case Weights Times ...	22056	22060
fraud_bool	_FPE_	Final Prediction Error	0.14221	.
fraud_bool	_MSE_	Mean Squared Error	0.138359	0.140382
fraud_bool	_RFPE_	Root Final Prediction Error	0.377108	.
fraud_bool	_RMSE_	Root Mean Squared Error	0.371966	0.374676
fraud_bool	_AVERR_	Average Error Function	0.420636	0.437163
fraud_bool	_ERR_	Error Function	9277.541	9643.808
fraud_bool	_MISC_	Misclassification Rate	0.193779	0.200907
fraud_bool	_WRONG_	Number of Wrong Classificati...	2137	2216

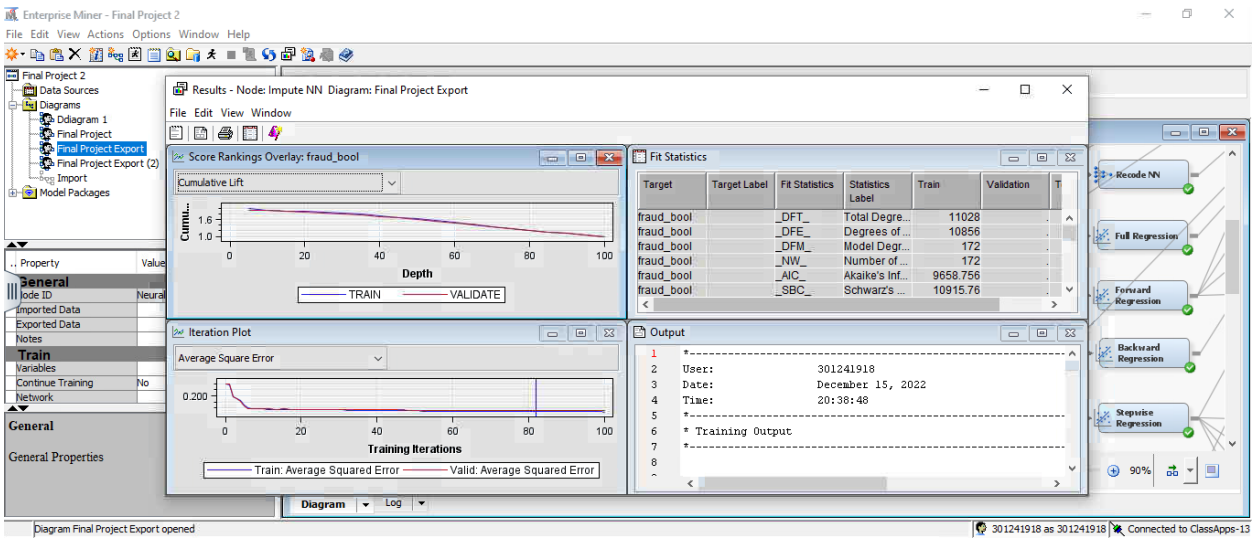
Data Modification Neural Network

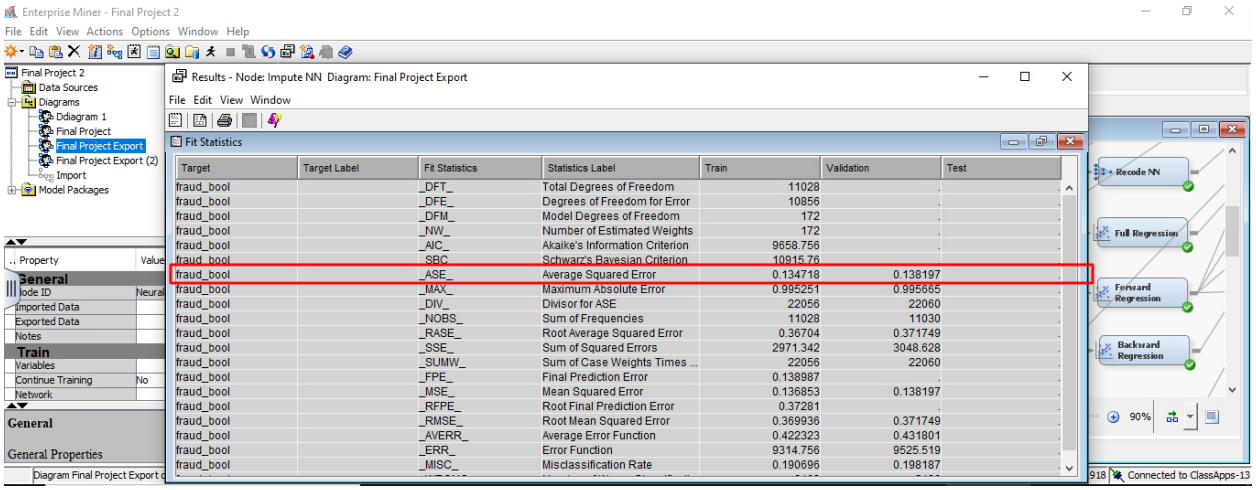
Impute Neural Network

After imputing missing values in our dataset, we attach a Neural Network node to see if imputing the missing values increases the accuracy of our models.



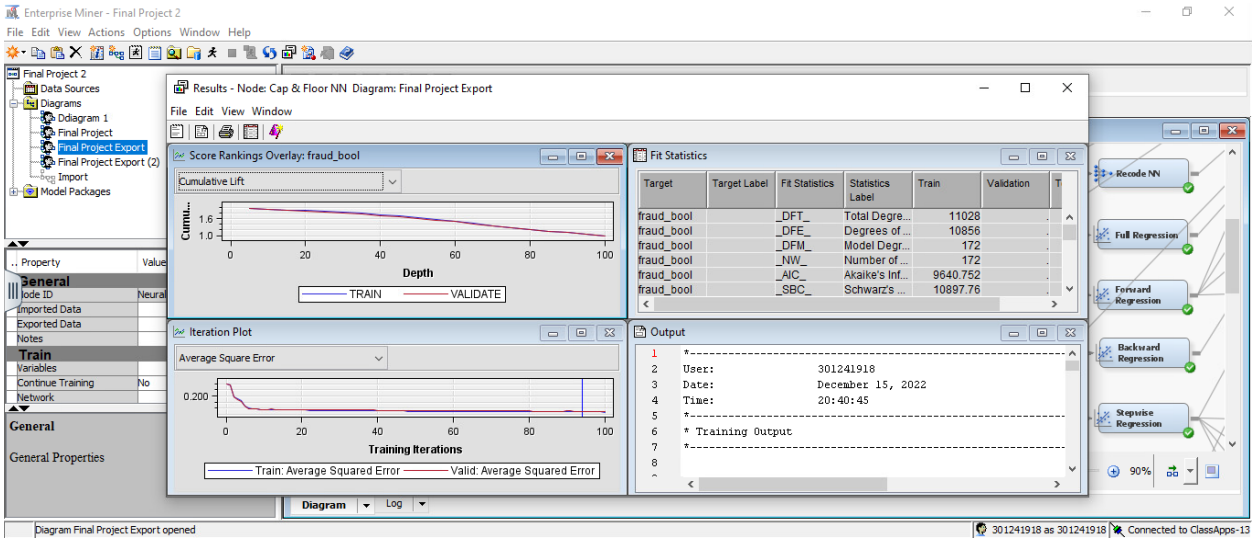
The validation average square error of the impute NN node is 0.138197 and the number of iterations suggested from this model is 82. The ASE we derived from this node is by far one of the best models so far. The closest model that achieved an error rate close to this was the ASE 3 branch model with an ASE of 0.169517. So the accuracy is significantly better.



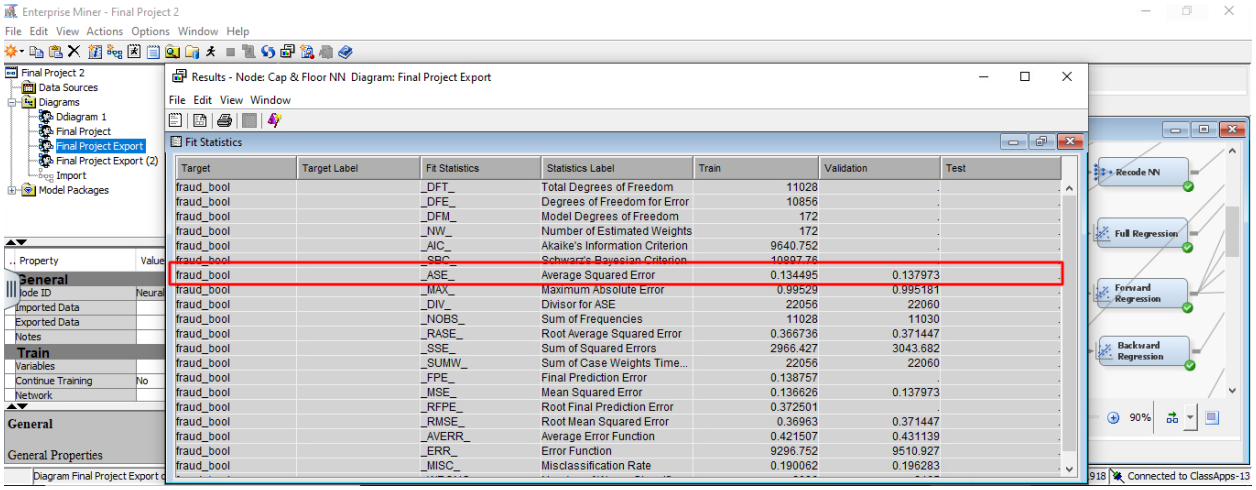


Cap and Floor Neural Network

After imputing all the missing variables, we added a replacement node to adjust the outliers of the dataset. Cap and Floor suggest the range of values that will be capped or floored by this node. Having run this node, we connected the neural network and below is the screenshot of the results panel.

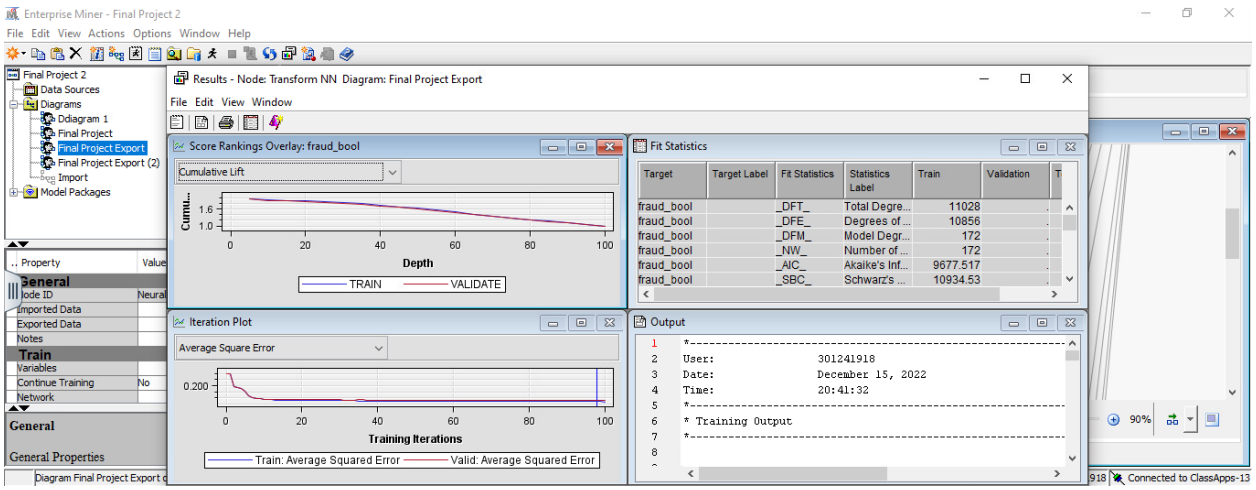


The average square error of the Cap & Floor NN node is 0.137973 and number of iterations is 94. This model has beat the previous Impute NN node by a few decimals only.

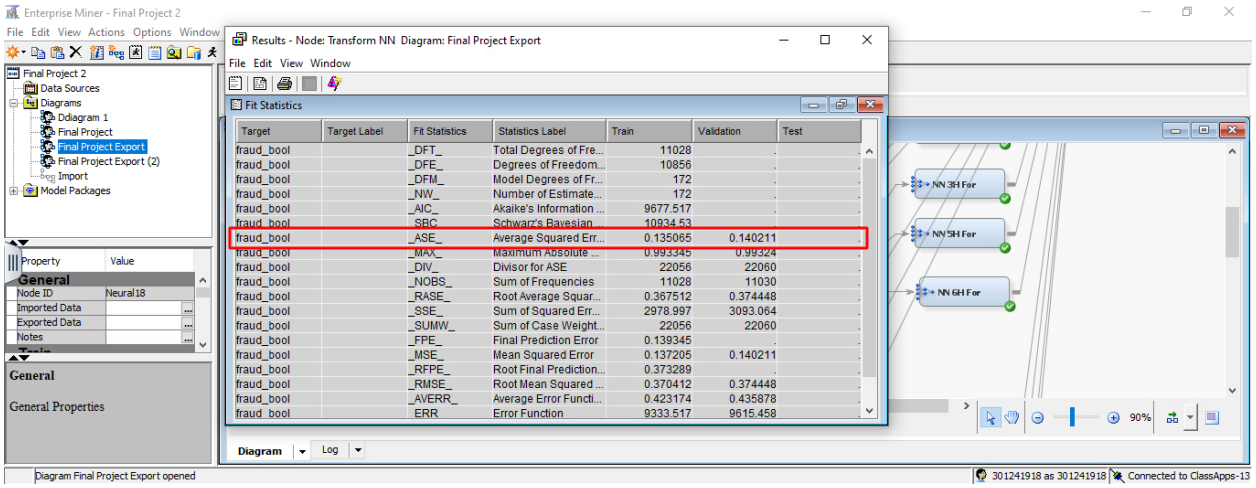


Transform Neural Network

After applying log transformation to the skews in our dataset, we connect the neural network with the Transform Skews node and below is the screenshot.

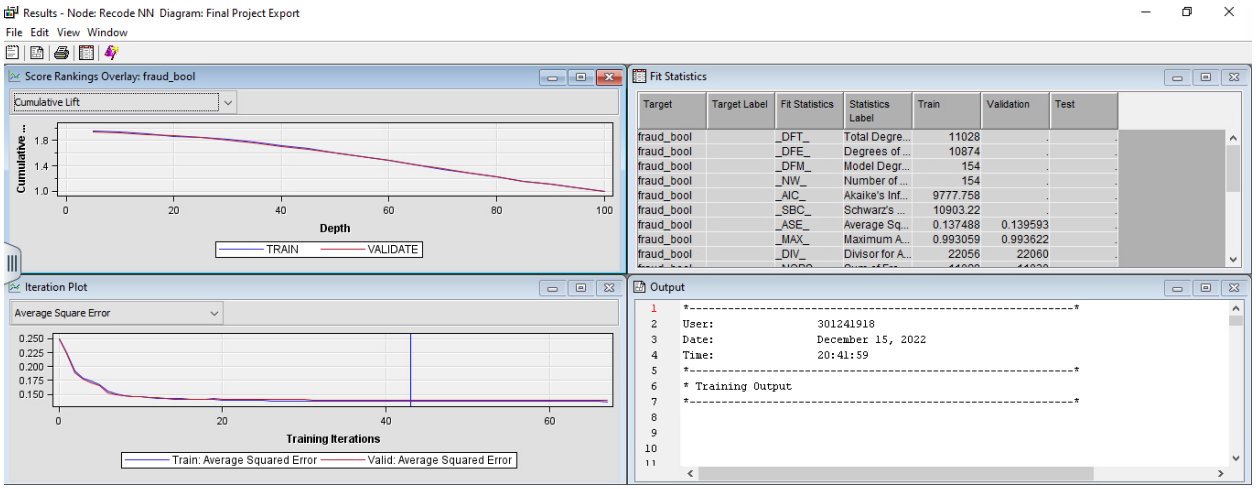


The average square error of the Transform NN node is 0.140211. This however has lower accuracy than the Cap and Impute NN and Impute NN.

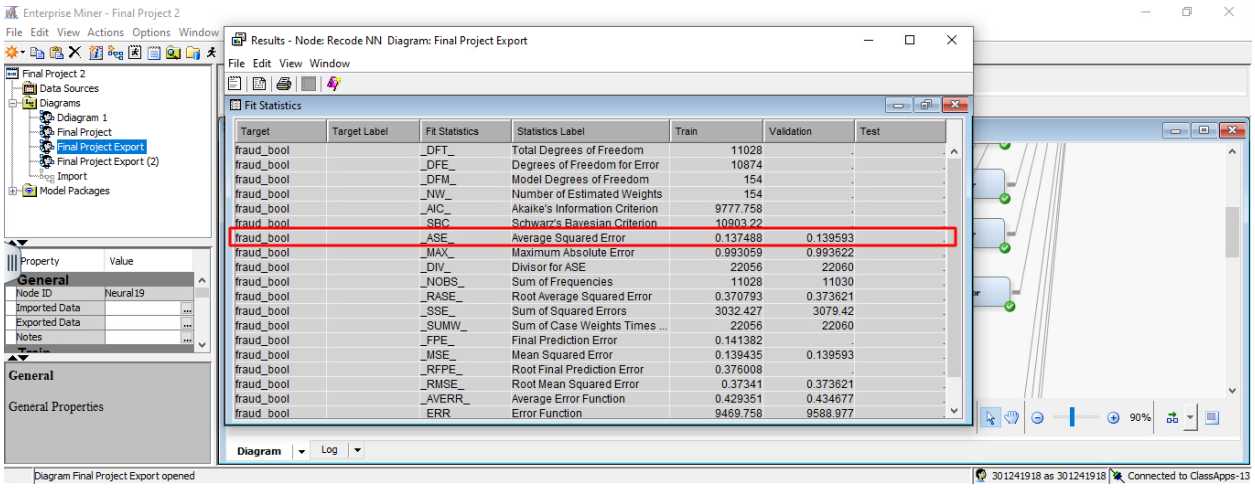


Recode Class Neural Network

We connect the neural network with the Record Class node as the step of last data manipulation. Screenshot referred below:

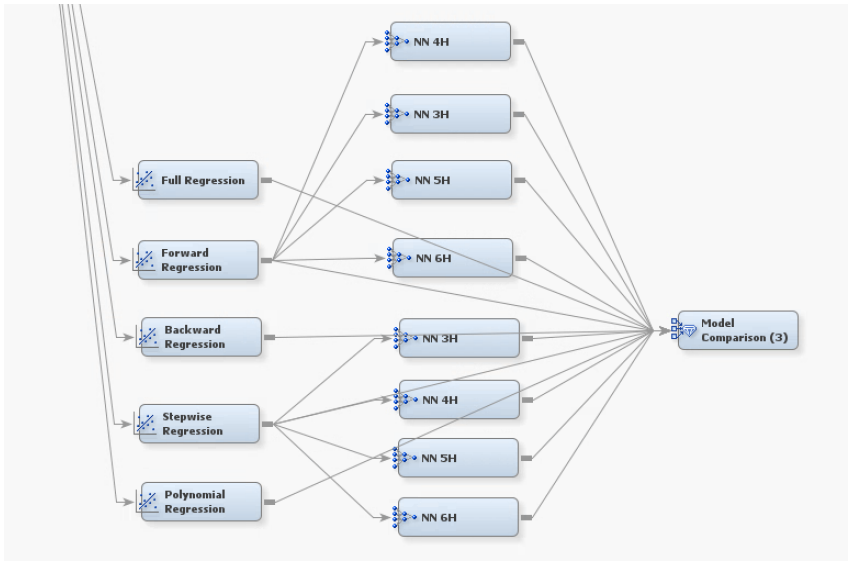


The average square error of the Recode NN node is 0.139593. As it seems, from the data manipulation section Cap and Floor NN and Impute NN are the best models so far.



Other Neural Networks

Besides these neural networks, we also worked on a few more NNs which were extrapolated before any data modification in the interval and class variables. Snapshot below:

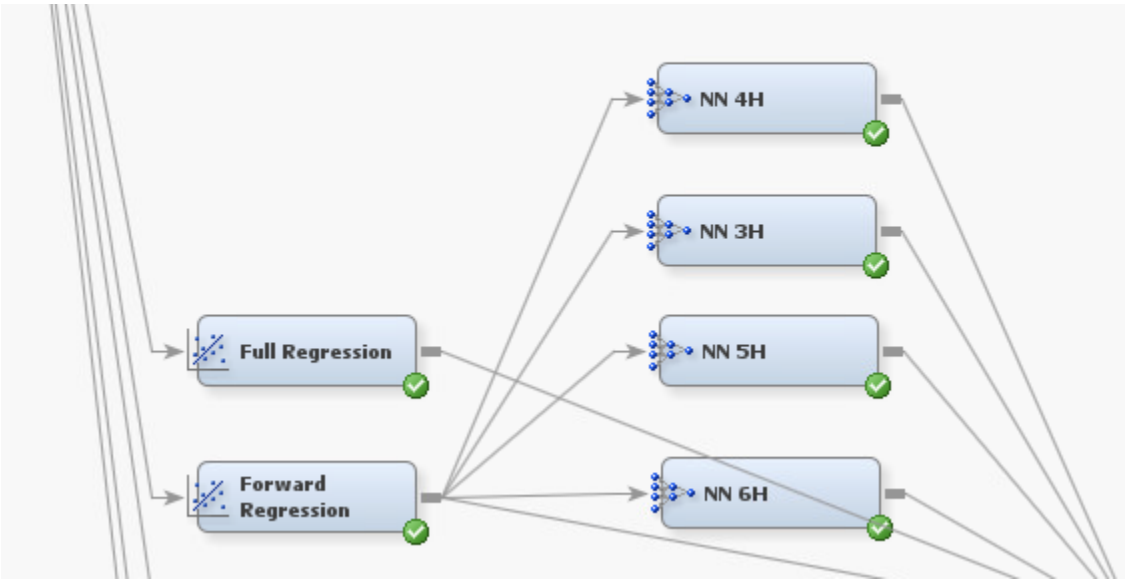


*** However, post consultation we decided to work with regressions which were derived from our treated data. Though the untreated data gave us better ASE in general across different model types, after careful consideration we proceeded with data which were more fit.

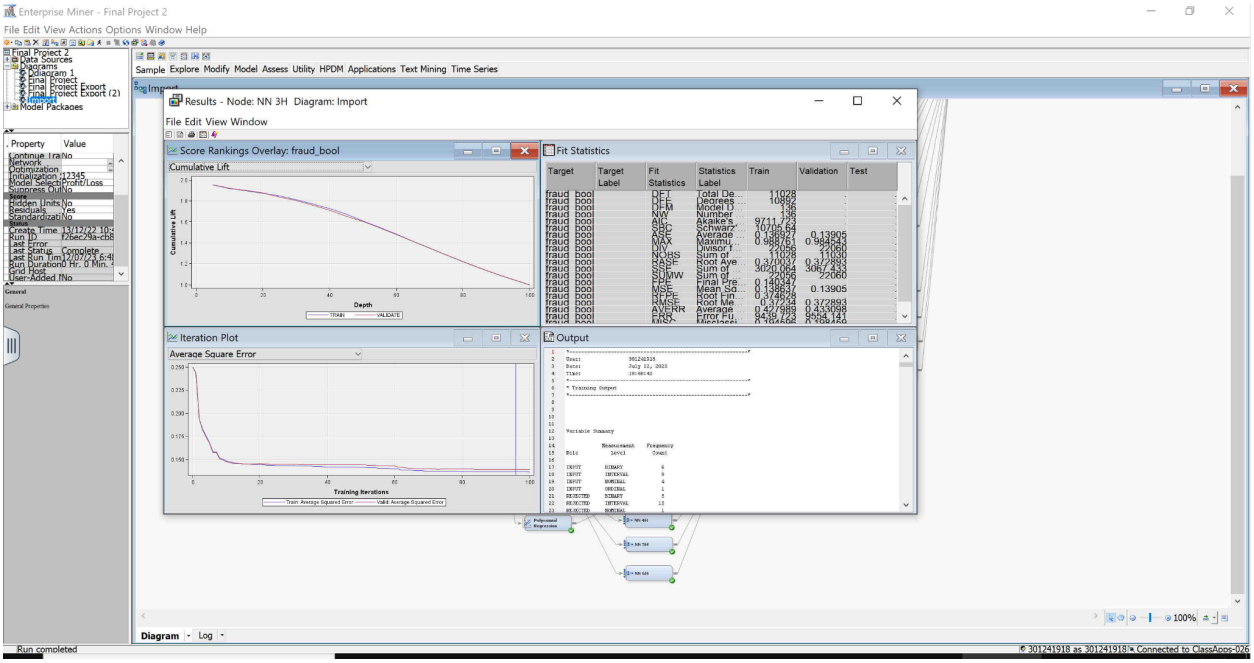
Forward Regression Neural Network:

In our project, we combined a neural network with forward regression and rigorously investigated the effectiveness of changing the number of hidden units and iterations. To determine the ideal setup, we gradually tested with 3 default units at first. We determined the

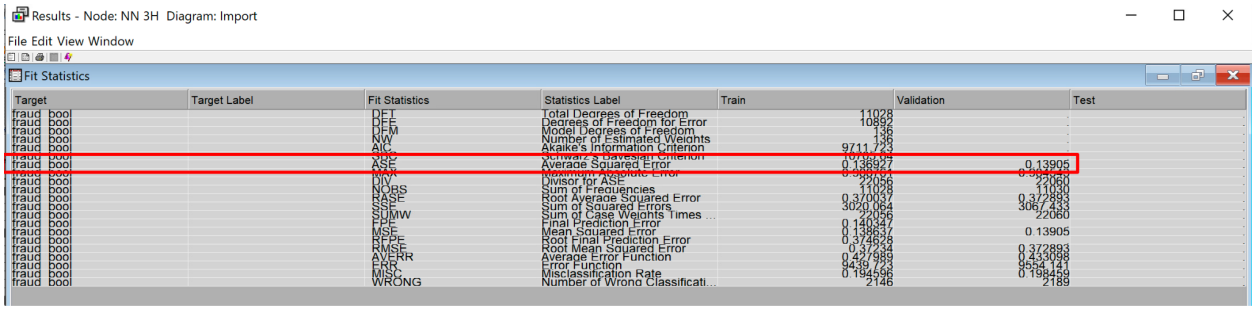
point at which efficiency began to suffer by linking neural networks with various hidden units and tweaking iterations, establishing the ideal balance between complexity and performance.



3 Hidden Unit Neural Networks

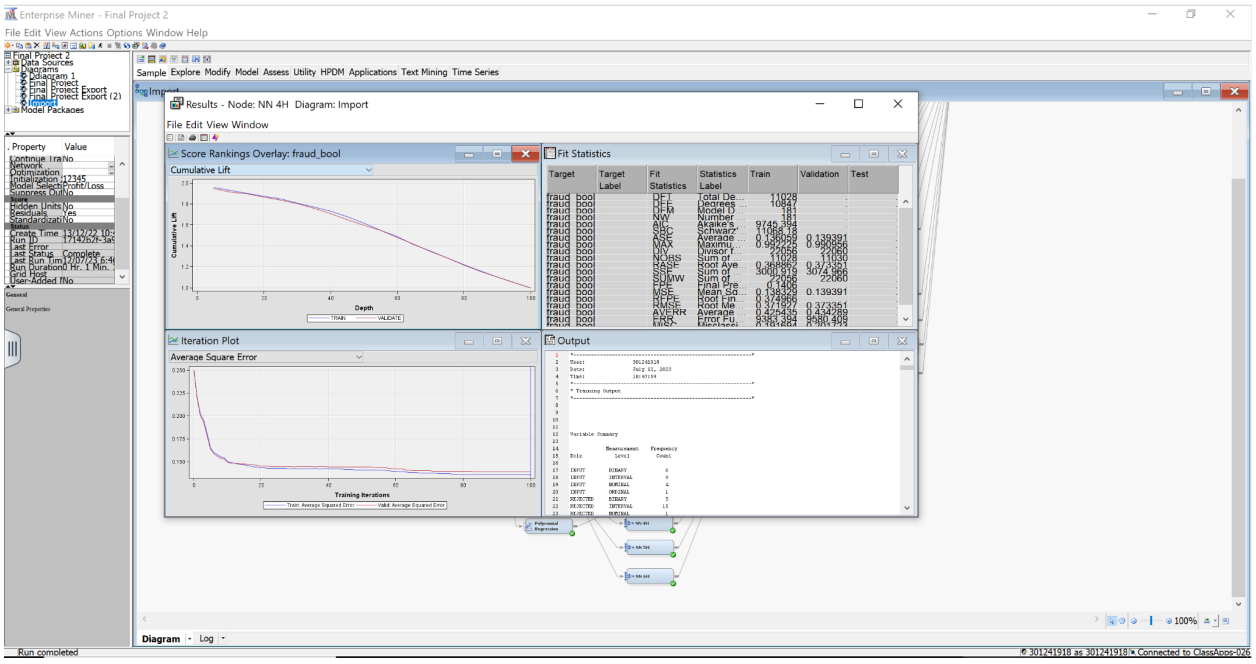


The average square error of a neural network with 3 hidden unit and 100 iterations is 0.13905, which is lowest in all Neural Networks.

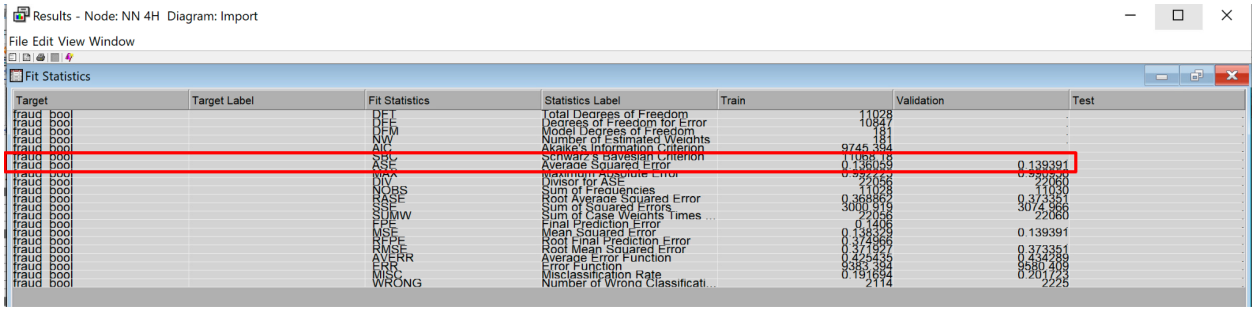


Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		DFT	Total Degrees of Freedom	11028		
fraud bool		DFE	Degrees of Freedom for Error	10892		
fraud bool		DFM	Model Degrees of Freedom	136		
fraud bool		NW	Number of Estimated Weights	136		
fraud bool		AIC	Akaike's Information Criterion	9711.723		
fraud bool		SBC	Schwarz's Bayesian Criterion	10101.05		
fraud bool		ASE	Average Squared Error	0.136927	0.13905	
fraud bool		MAX	Maximum Absolute Error	0.988361	0.930459	
fraud bool		DIV	Divisor for ASE	22056	22060	
fraud bool		NOBS	Sum of Frequencies	11028	11030	
fraud bool		RASE	Root Average Squared Error	0.370037	0.372893	
fraud bool		SSE	Sum of Squared Errors	3020.064	3067.433	
fraud bool		SUMW	Sum of Case Weights Times ...	22056	22060	
fraud bool		FPE	Final Prediction Error	0.140347		
fraud bool		MSE	Mean Squared Error	0.138637	0.13905	
fraud bool		RFPE	Root Final Prediction Error	0.374628		
fraud bool		RMSE	Root Mean Squared Error	0.37234	0.372893	
fraud bool		AVERR	Average Error Function	0.427989	0.433099	
fraud bool		ERR	Error Function	9439.723	9554.141	
fraud bool		MISC	Misclassification Rate	0.194596	0.198459	
fraud bool		WRONG	Number of Wrong Classificati...	2146	2189	

4 Hidden Unit Neural Networks

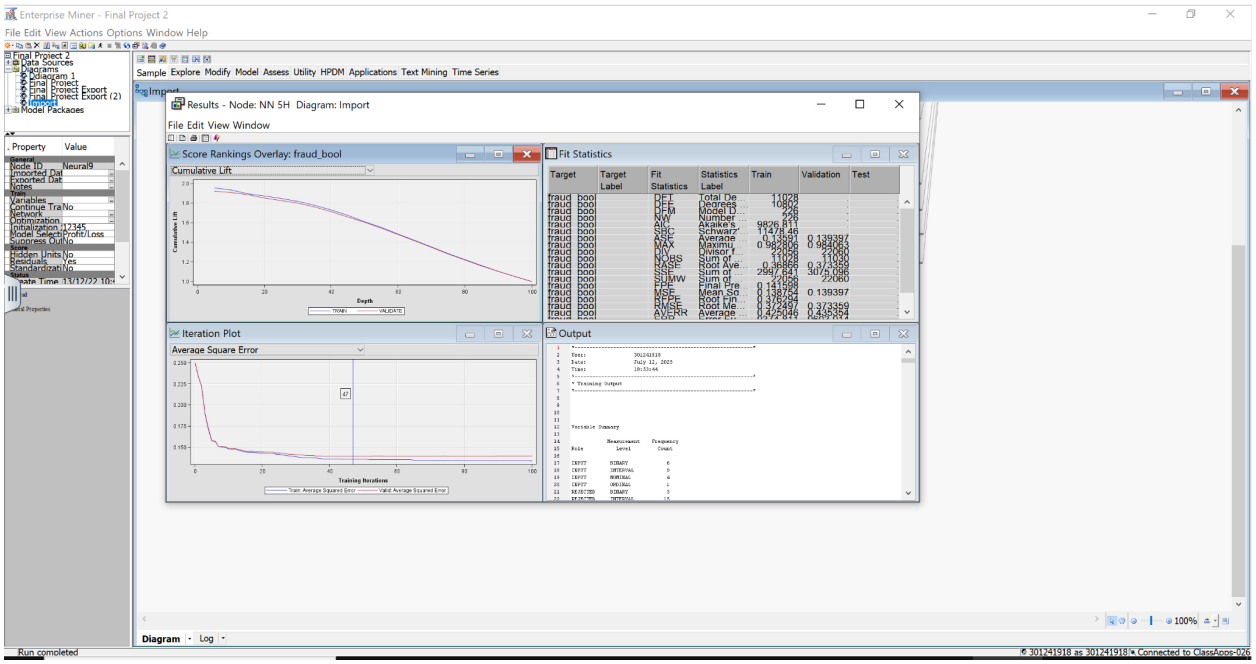


The average square error of a neural network with 4 hidden unit is 0.139391 with cut-off iterations at 100.

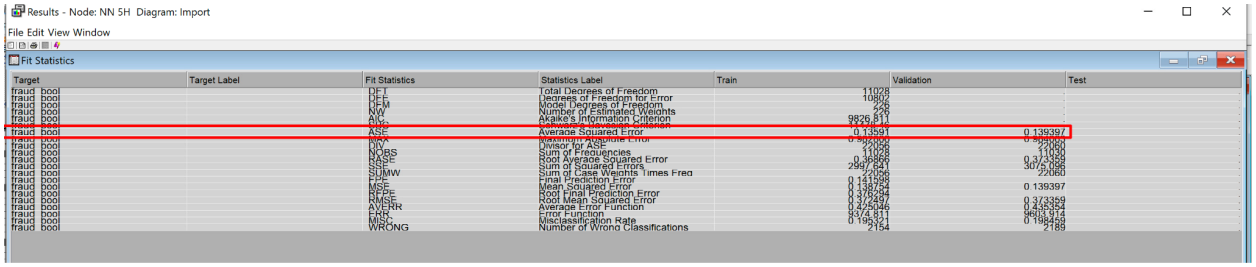


Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud bool		DFT	Total Degrees of Freedom	11028		
fraud bool		DFE	Degrees of Freedom for Error	10947		
fraud bool		DFM	Model Degrees of Freedom	181		
fraud bool		NW	Number of Estimated Weights	181		
fraud bool		AIC	Akaike's Information Criterion	9745.394		
fraud bool		SBC	Schwarz's Bayesian Criterion	11068.18		
fraud bool		ASE	Average Squared Error	0.136059	0.139391	
fraud bool		MAX	Maximum Absolute Error	0.992225	0.930459	
fraud bool		DIV	Divisor for ASE	22056	22060	
fraud bool		NOBS	Sum of Frequencies	11028	11030	
fraud bool		RASE	Root Average Squared Error	0.368867	0.373351	
fraud bool		SSE	Sum of Squared Errors	3000.919	3074.966	
fraud bool		SUMW	Sum of Case Weights Times ...	22056	22060	
fraud bool		FPE	Final Prediction Error	0.1406		
fraud bool		MSE	Mean Squared Error	0.138329	0.139391	
fraud bool		RFPE	Root Final Prediction Error	0.374966		
fraud bool		RMSE	Root Mean Squared Error	0.371927	0.373351	
fraud bool		AVERR	Average Error Function	0.429435	0.434289	
fraud bool		ERR	Error Function	9383.384	9680.409	
fraud bool		MISC	Misclassification Rate	0.191694	0.201743	
fraud bool		WRONG	Number of Wrong Classificati...	2114	2225	

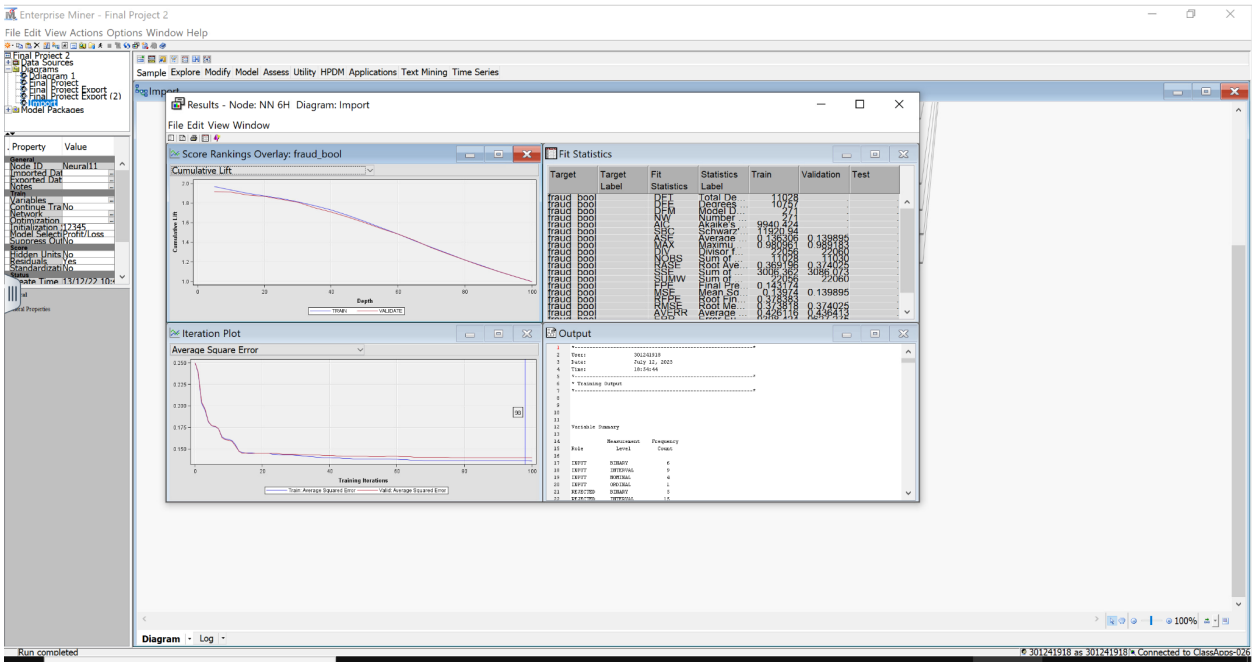
5 Hidden Unit Neural Networks



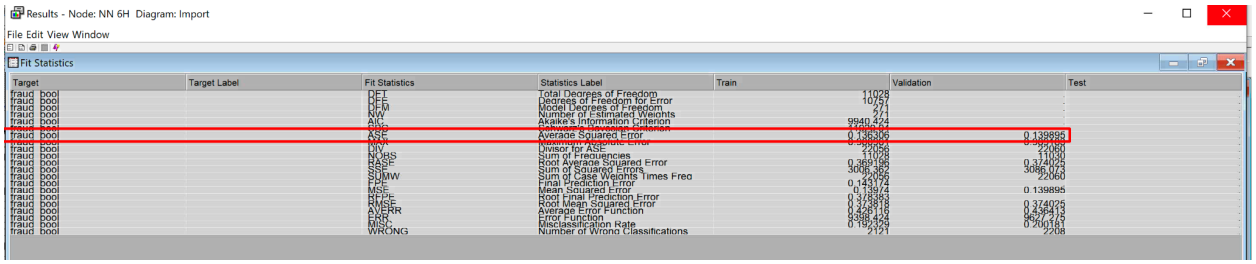
The average square error of a neural network with 5 hidden unit and 47 iterations is 0.139397



6 Hidden Unit Neural Networks



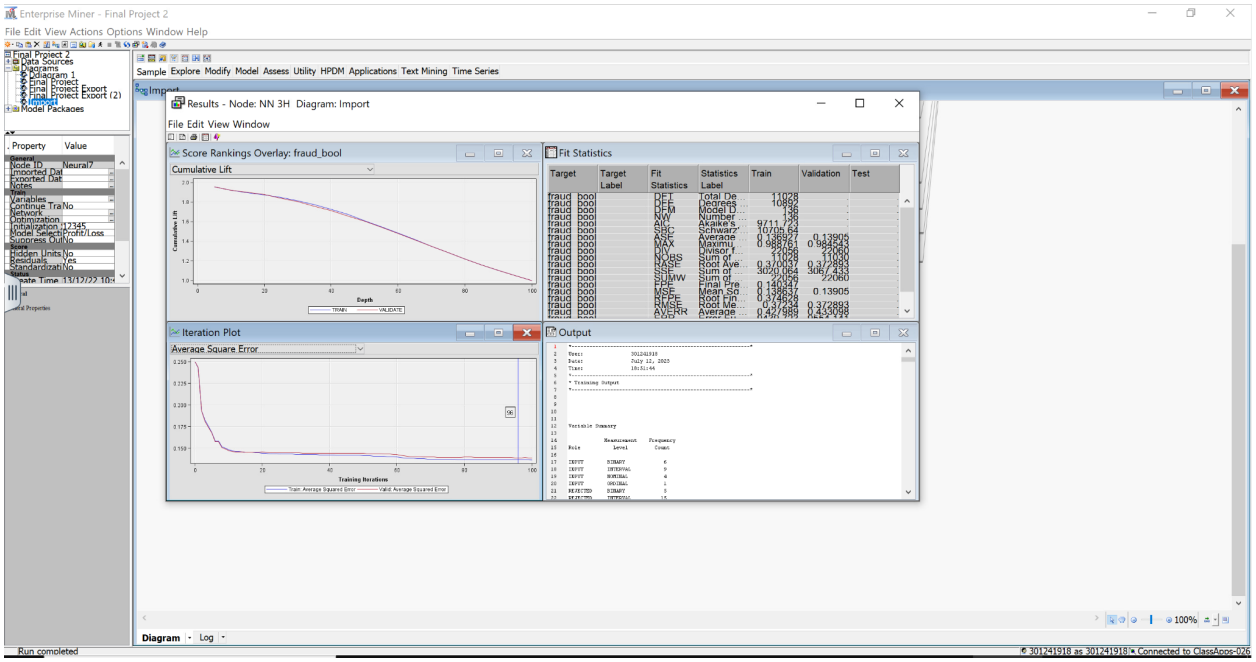
The average square error of a neural network with 6 hidden units is 0.139895 with cut-off iterations at 98. The Average Square Error is the highest, with 6 hidden units among all other neural networks.



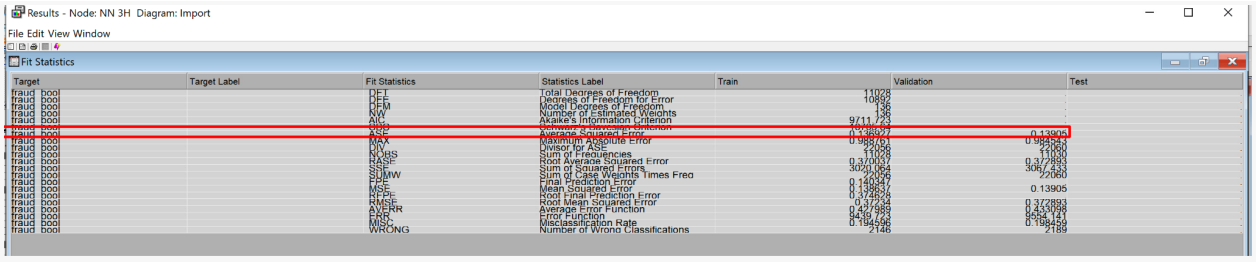
Stepwise Regression Neural Network:

We also explored the stepwise NN, to evaluate if the iterative process of adding and removing variables help the NN error scores or not.

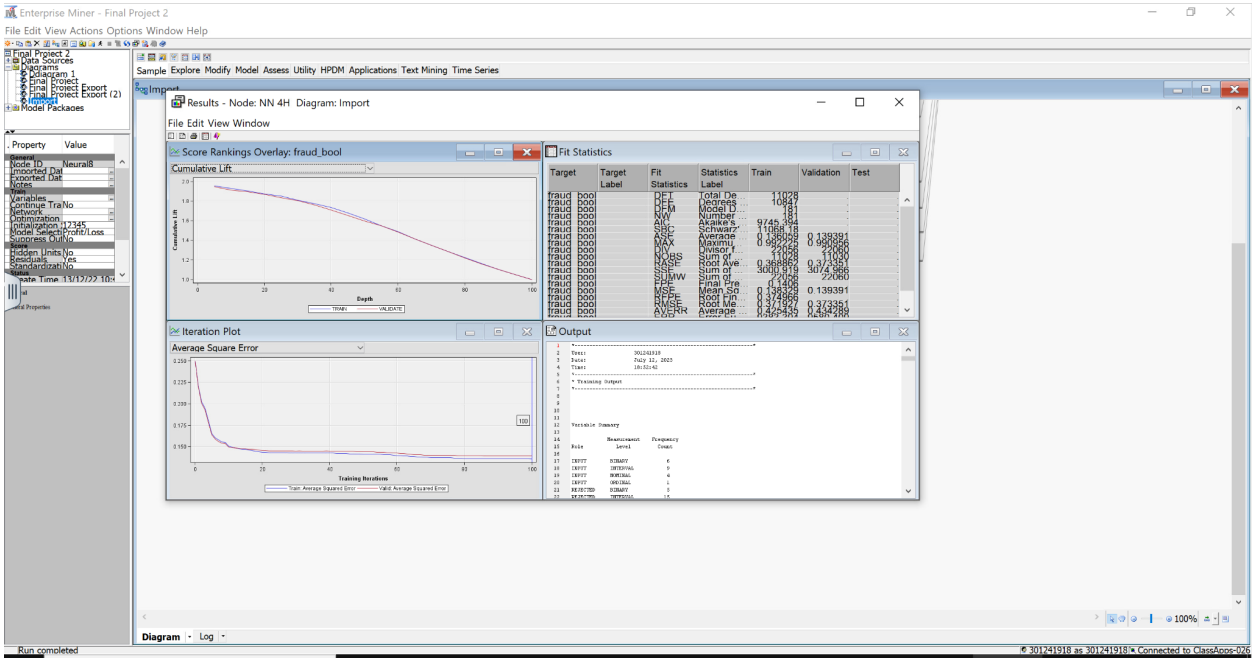
3 Hidden Unit Neural Networks



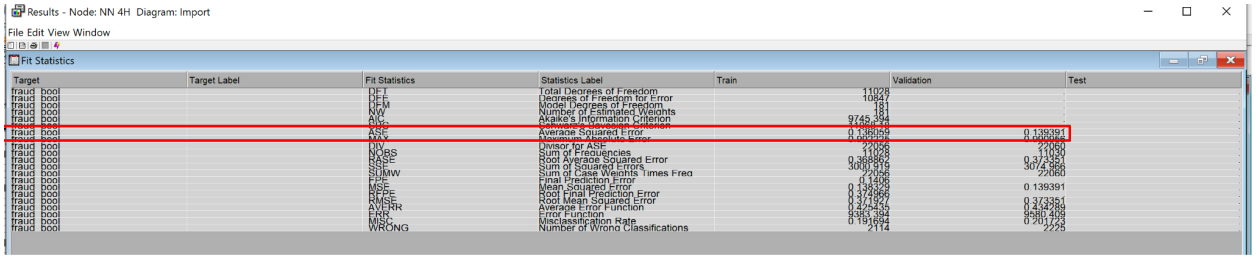
The average square error of a neural network with 3 hidden unit and 96 iterations is 0.13905, which is lowest in all Neural Networks.



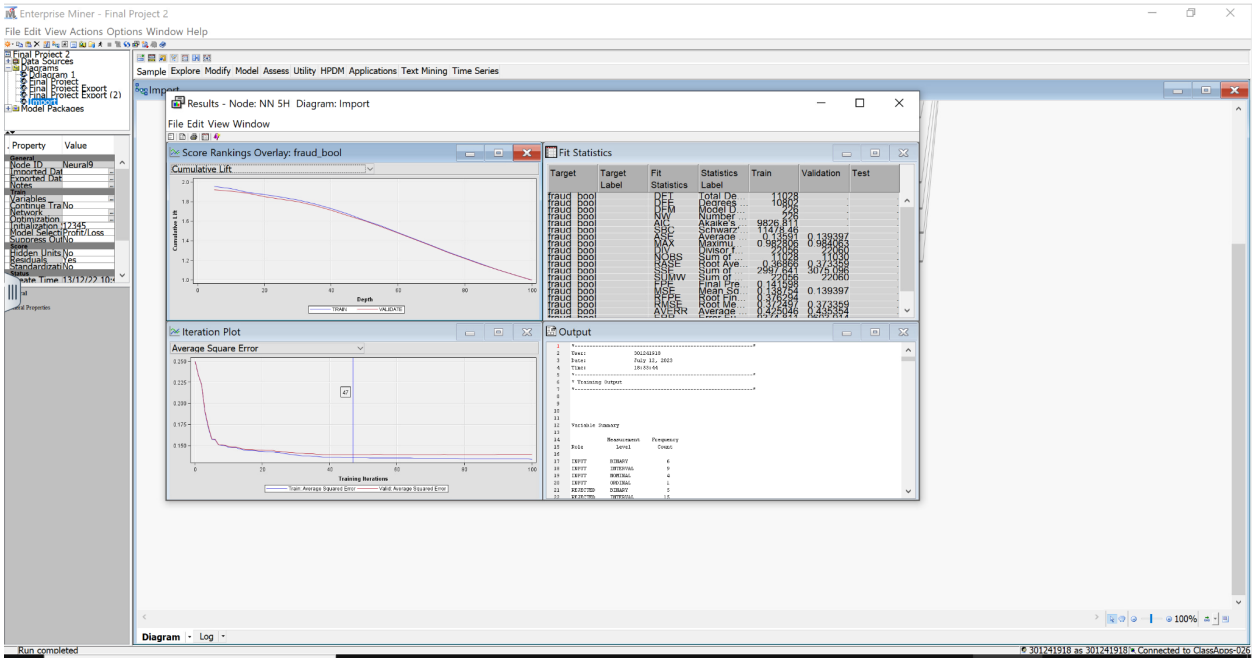
4 Hidden Unit Neural Networks



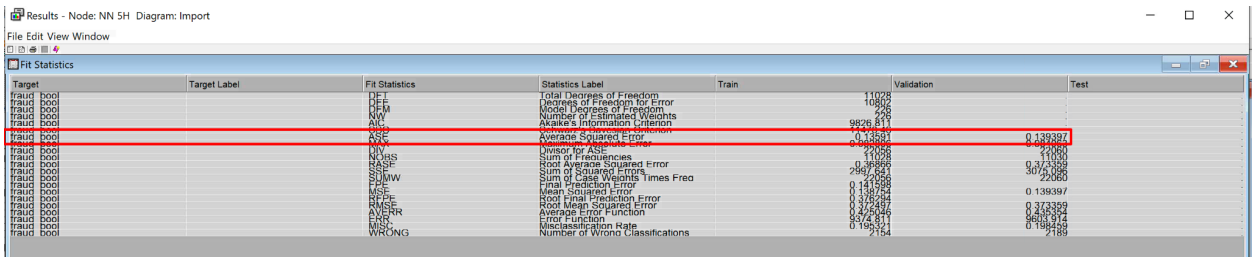
The average square error of a neural network with 4 hidden units is 0.139391 with cut-off iterations at 100.



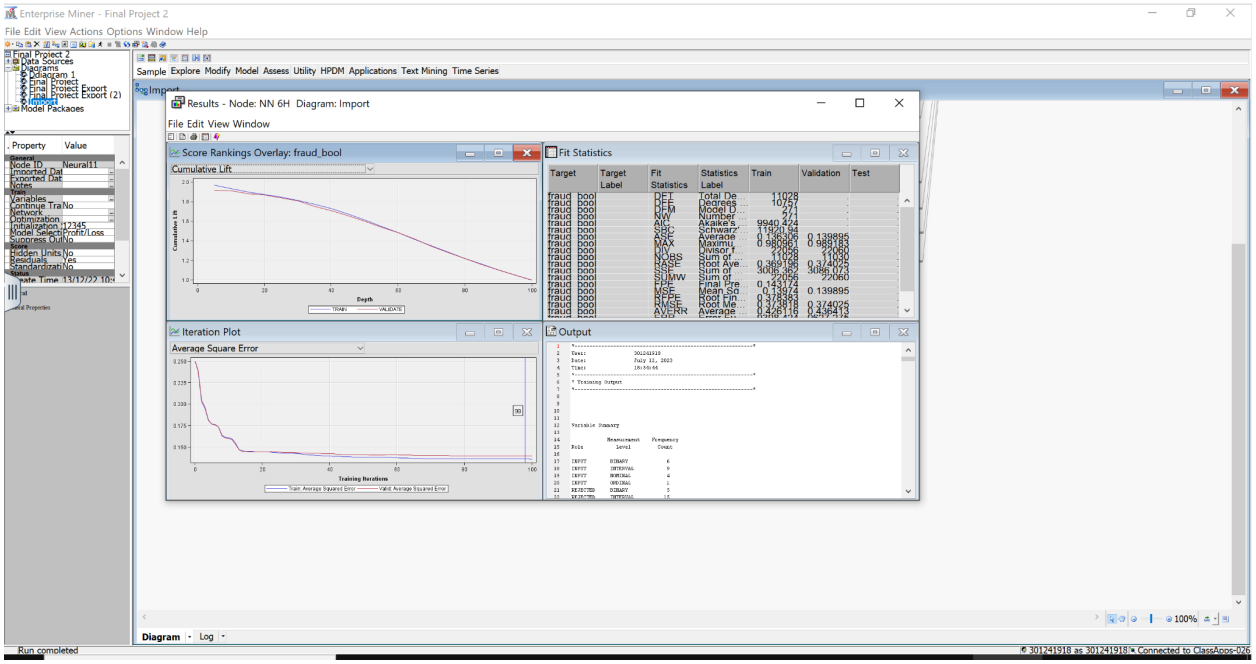
5 Hidden Unit Neural Networks



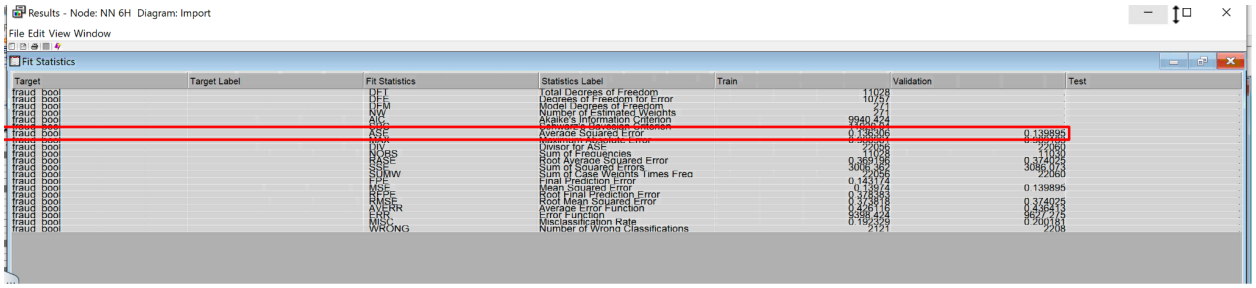
The average square error of a neural network with 5 hidden unit and 47 iterations is 0.139397



6 Hidden Unit Neural Networks



The average square error of a neural network with 6 hidden units is 0.139895 with cut-off iterations at 98. The Average Square Error is the highest, with 6 hidden units among all other neural networks.



Gradient Boosting

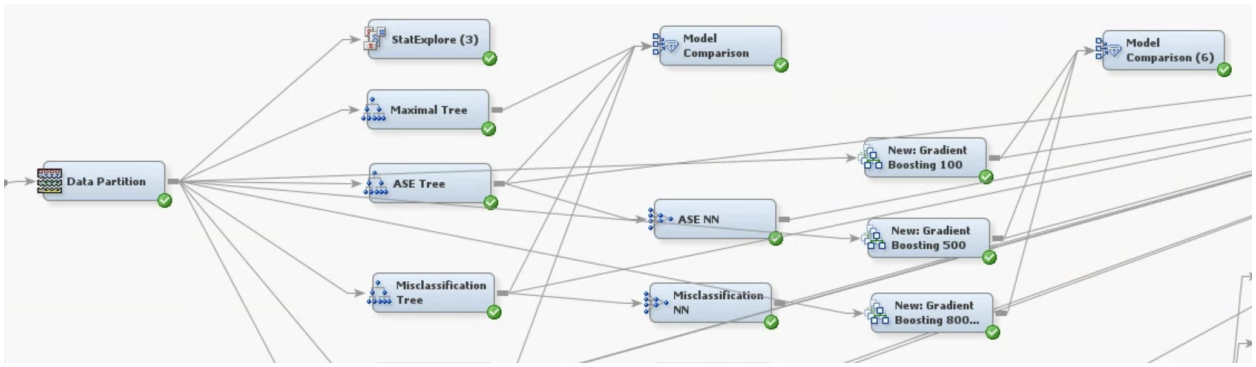
Gradient Boosting Models are one of the most popular ensemble techniques used in machine learning. Gradient Boosting works by combining many “weak learners” to create a stronger prediction model. Each weak learner focuses on a small part of the prediction, and looks to improve upon the mistakes of the previous weak learner.

For our project, we developed 2 gradient boosting models:

- Gradient Boosting 100

- Gradient Boosting 500
- Gradient Boosting 800

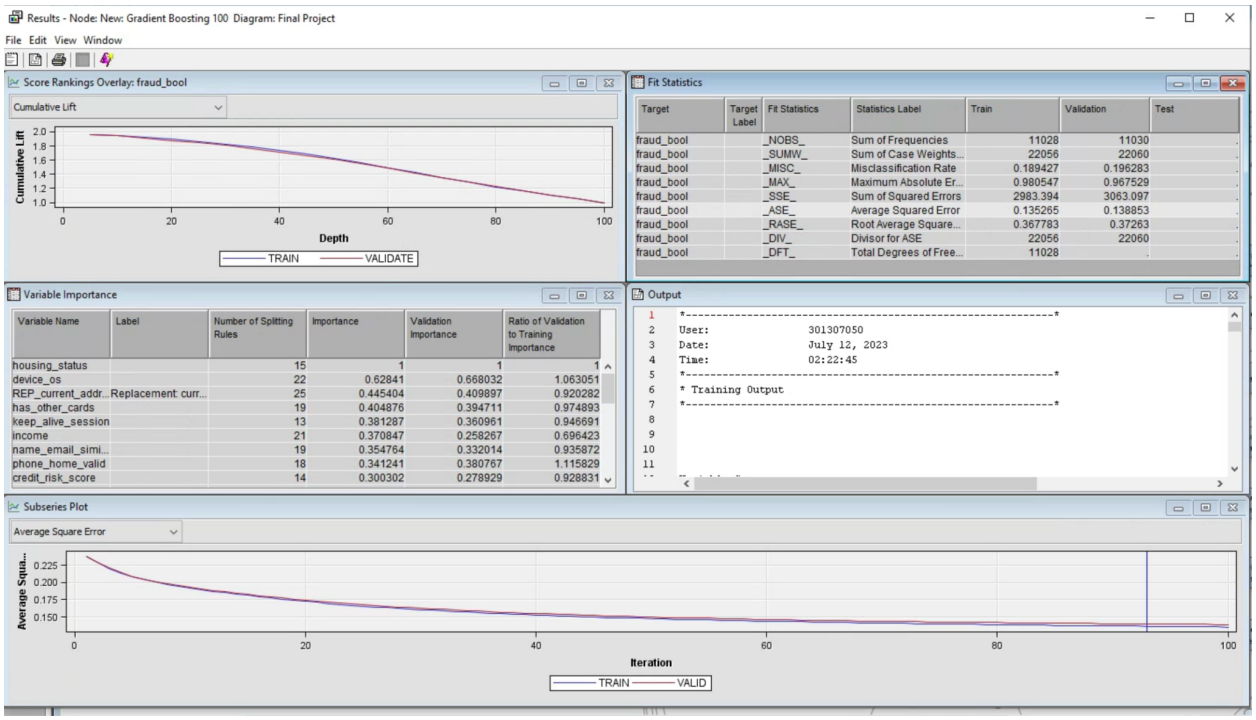
The screenshot of the two models are shown below:



Gradient Boosting 100

The first boosting model is the Gradient Boosting 100. This model was run for 100 iterations with the maximum branch and depth of each weak learner set to 2. The model achieved an Average Squared Error of **0.138853** and a misclassification rate of **0.196283.** The model achieved it's lowest ASE after 93 iterations.

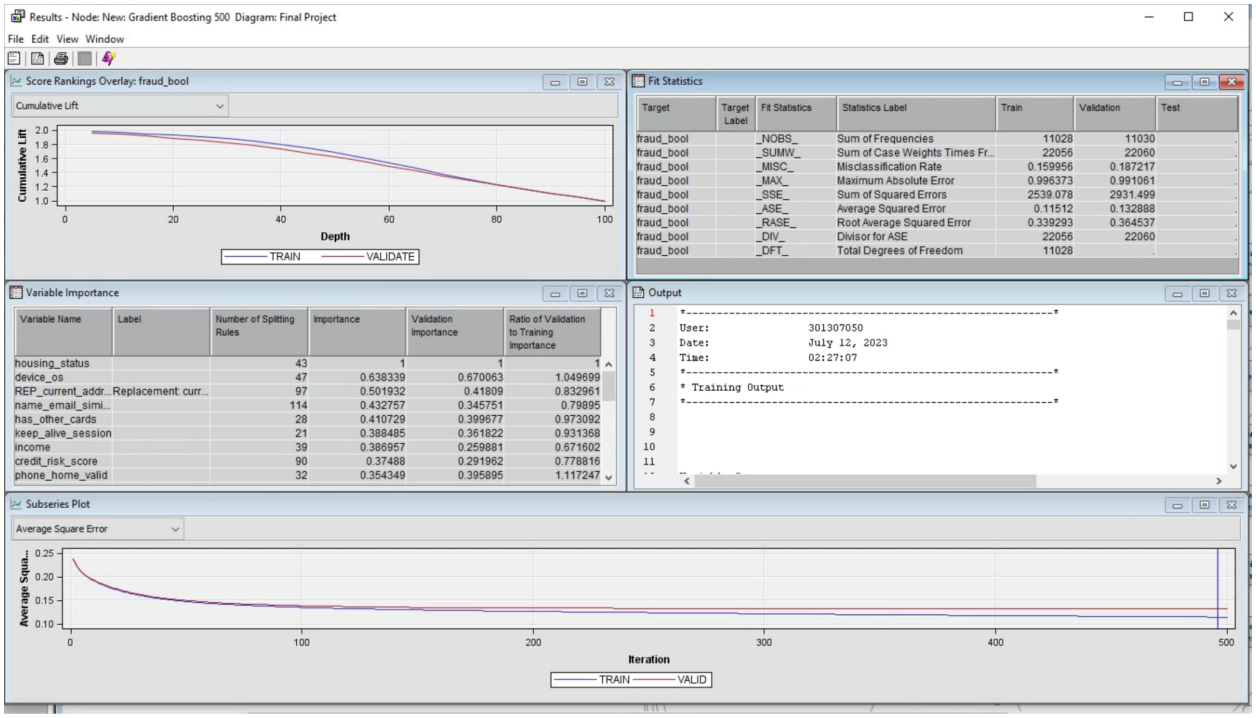
The results of the first boosting model is shown below:



Gradient Boosting 500

The second Gradient Boosting model was run for 500 iterations, to determine whether allowing the model to run for more iterations would lead to better training and predictions. All other hyper parameters were kept the same as the previous model. After 496 iterations, the Gradient Boosting 500 model achieved an Average Squared Error of **0.132888,** which is the lowest ASE among all the models developed. The misclassification rate was **0.187217.**

The results window of the Gradient Boosting 500 model is shown below:



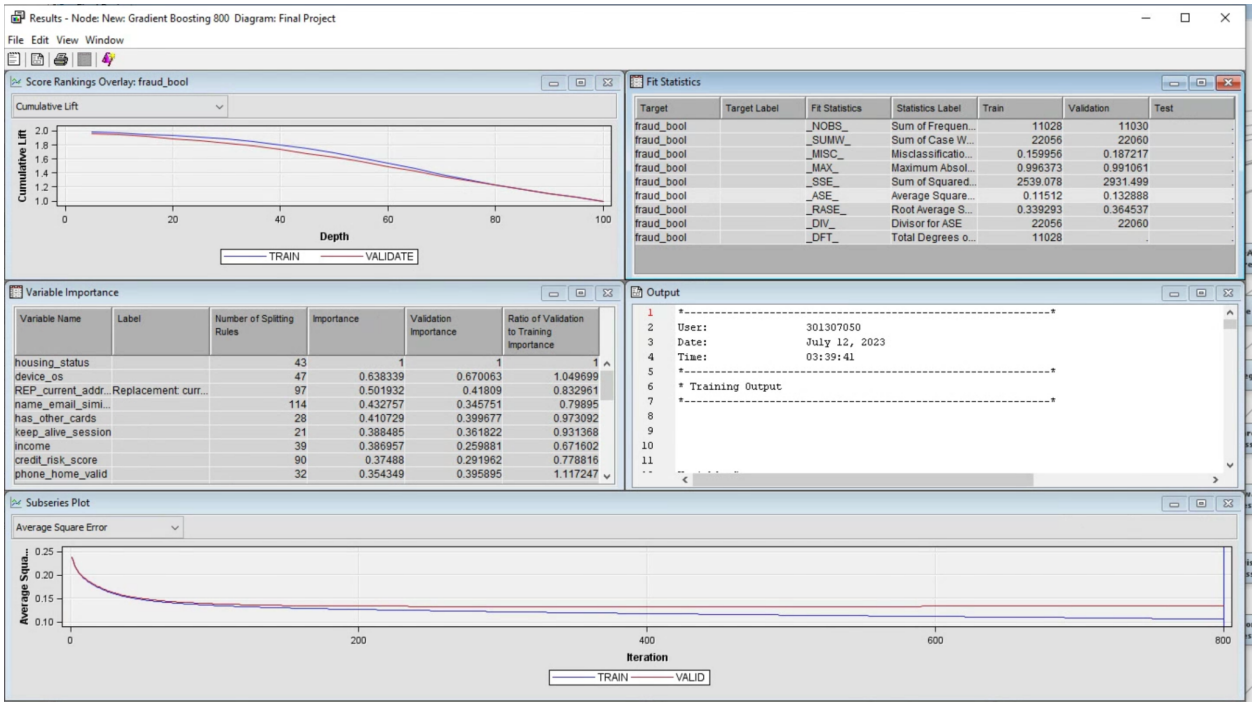
Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation
fraud_bool		_NOBS_	Sum of Frequencies	11028	11030
fraud_bool		_SUMW_	Sum of Case Weights Times Freq	22056	22060
fraud_bool		_MISC_	Misclassification Rate	0.159956	0.187217
fraud_bool		_MAX_	Maximum Absolute Error	0.996373	0.991061
fraud_bool		_SSE_	Sum of Squared Errors	2539.078	2931.499
fraud_bool		_ASE_	Average Squared Error	0.11512	0.132888
fraud_bool		_RASE_	Root Average Squared Error	0.339293	0.364537
fraud_bool		_DIV_	Divisor for ASE	22056	22060
fraud_bool		_DFT_	Total Degrees of Freedom	11028	

Gradient Boosting 800

To determine whether allowing the model to run for even more iterations would result in better predictions, we built the Gradient Boosting 800 model, which was run for 800 iterations, still with max depth and branches set to 2.

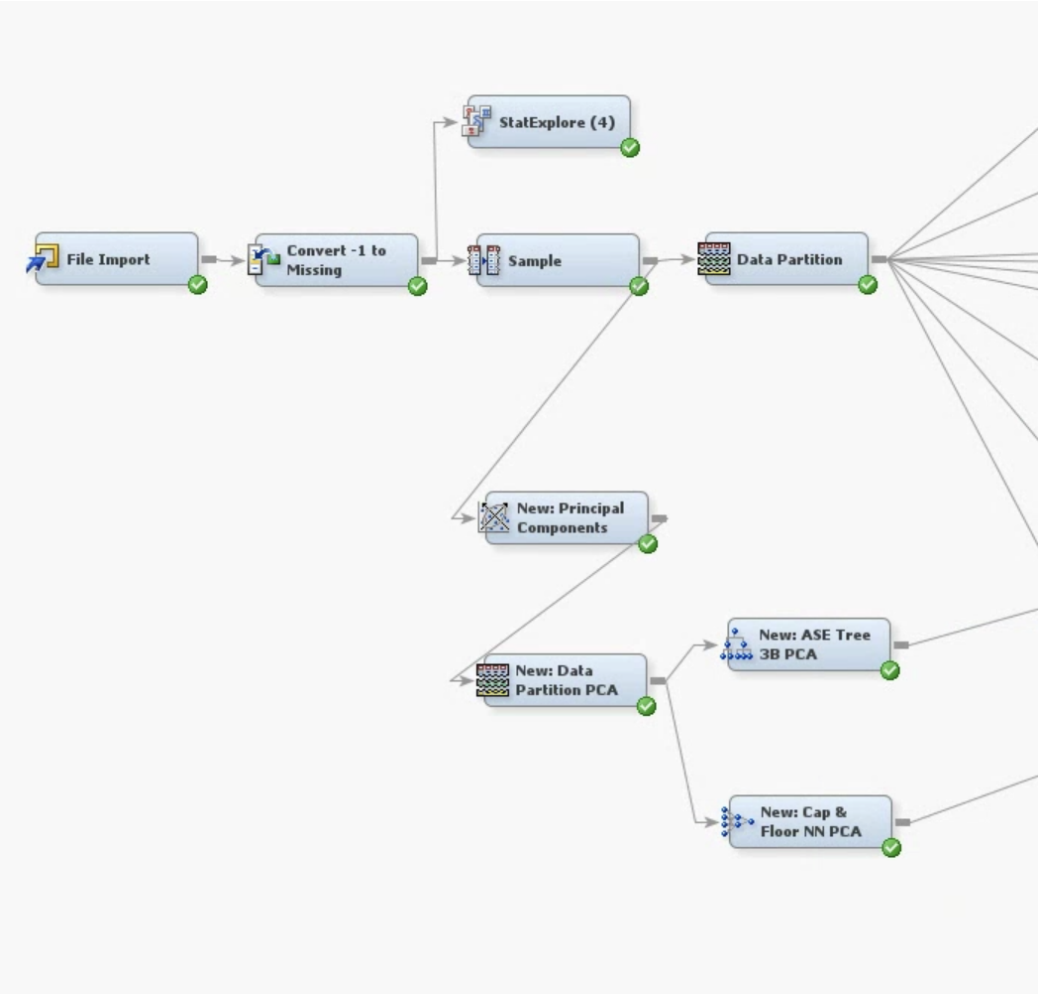
However, the validation ASE and misclassification rate remains the same at **0.132888** and **0.187217,** indicating that additional iterations do not improve the model further. The results window of the Gradient Boosting 500 model is shown below:



Principal Component Analysis

Principal Component Analysis is a technique to simplify complex data by combining the variables in a dataset into combined features that explain the largest variation in the data, called Principal Components. PCA reduces the dimensionality of the data while retaining as much information about the features as possible. This reduces the curse of dimensionality and simplifies the data for further modeling.

For our project, we conducted PCA analysis to determine whether the models built from the principal components would have higher predictive accuracy. We added a Principal Component node after the oversampling node, and then partitioned the data into training and validation sets for modeling. The most important variables included in PCA 1 were Housing Status, Credit Risk Score, and Proposed Credit Limit. Screenshot of the nodes are shown below:



From the Principal Components, we developed two models:

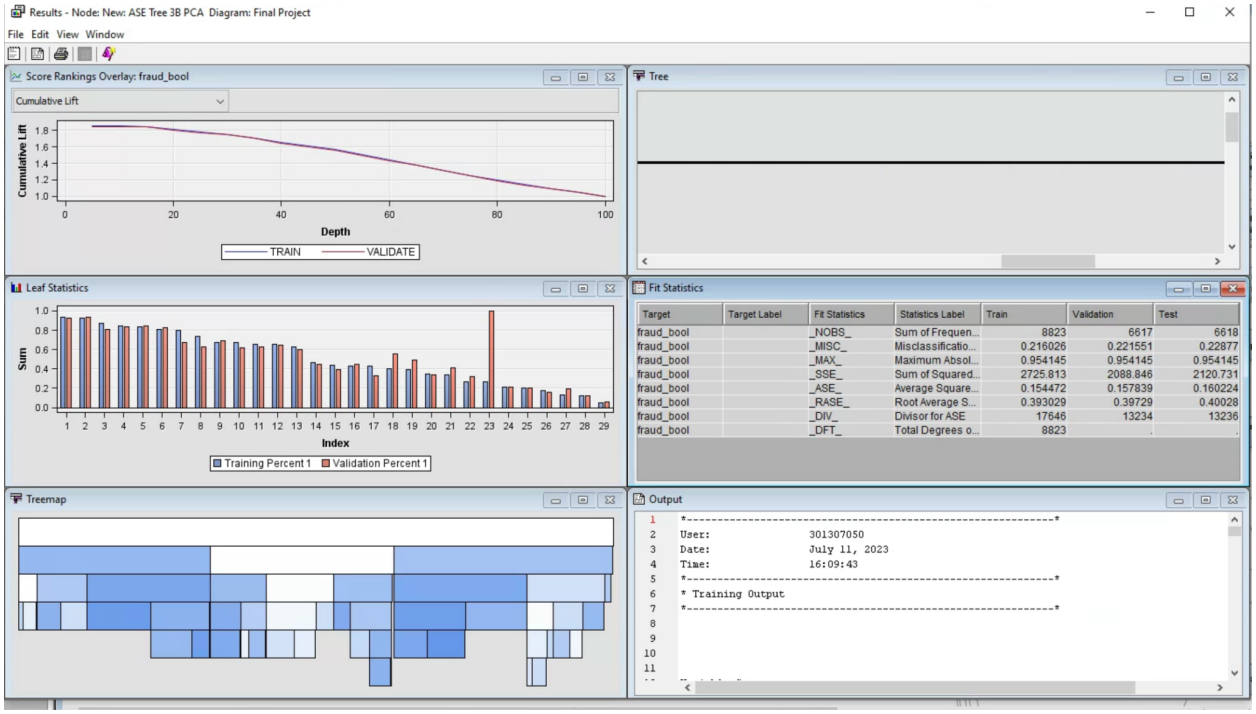
- ASE Tree 3B PCA
- Cap & Floor NN PCA

These were our two best models developed before conducting PCA, and we wanted to determine whether conducting PCA would improve these models.

ASE Tree 3B PCA

The first model - ASE Tree 3B PCA, achieved a validation ASE of **0.157839** and a misclassification rate of **0.221551.**

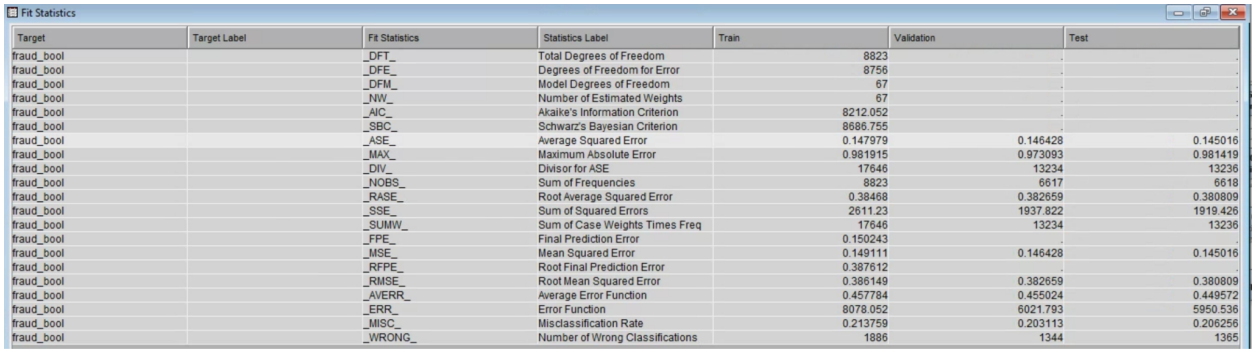
Results window shown below:



Cap & Floor NN PCA

The Cap & Floor Neural Network was identified as one of the best models we developed, and therefore we wanted to find out whether a Neural Network with the same settings would be improved by the Principal Component Analysis. We attached the Neural Network to the PCA data partition node.

The Neural Network had an ASE of **0.146428** and a misclassification rate of **0.203113.** Results are shown below.



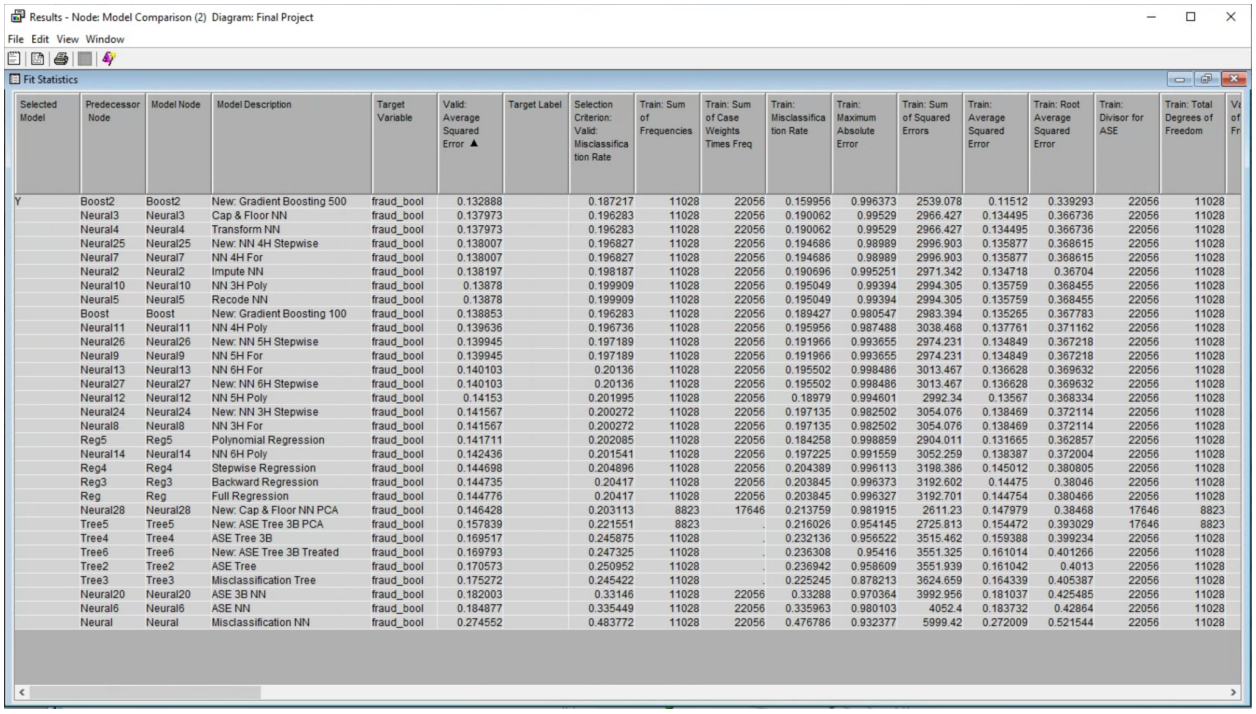
Fit Statistics

Target	Target Label	Fit Statistics	Statistics Label	Train	Validation	Test
fraud_bool		_DFT_	Total Degrees of Freedom	8823		
fraud_bool		_DFE_	Degrees of Freedom for Error	8756		
fraud_bool		_DFM_	Model Degrees of Freedom	67		
fraud_bool		_NW_	Number of Estimated Weights	67		
fraud_bool		_AIC_	Akaike's Information Criterion	8212.052		
fraud_bool		_SBC_	Schwarz's Bayesian Criterion	8686.755		
fraud_bool		_ASE_	Average Squared Error	0.147979	0.146428	0.145016
fraud_bool		_MAX_	Maximum Absolute Error	0.981915	0.973093	0.981419
fraud_bool		_DIV_	Divisor for ASE	17646	13234	13236
fraud_bool		_NOBS_	Sum of Frequencies	8823	6617	6618
fraud_bool		_RASE_	Root Average Squared Error	0.38468	0.382659	0.380809
fraud_bool		_SSE_	Sum of Squared Errors	2611.23	1937.822	1919.426
fraud_bool		_SUMW_	Sum of Case Weights Times Freq	17646	13234	13236
fraud_bool		_FPE_	Final Prediction Error	0.150243		
fraud_bool		_MSE_	Mean Squared Error	0.149111	0.146428	0.145016
fraud_bool		_RFPE_	Root Final Prediction Error	0.387612		
fraud_bool		_RMSE_	Root Mean Squared Error	0.386149	0.382659	0.380809
fraud_bool		_AVERR_	Average Error Function	0.457784	0.455024	0.449572
fraud_bool		_ERR_	Error Function	8078.052	6021.793	5950.536
fraud_bool		_MISC_	Misclassification Rate	0.213759	0.203113	0.206256
fraud_bool		_WRONG_	Number of Wrong Classifications	1886	1344	1365

Overall, the predictive performance of the models were not improved by conducting PCA analysis, and in some cases, the performance decreased.

Model Comparison

In order to devise the best model, we created 23+ different models which we ran throughout this entire project. The Model Comparison node in SAS Enterprise Miner helps us compare the statistics for all 23+ models in one panel. A screenshot of the summary statistics for each model is given below:

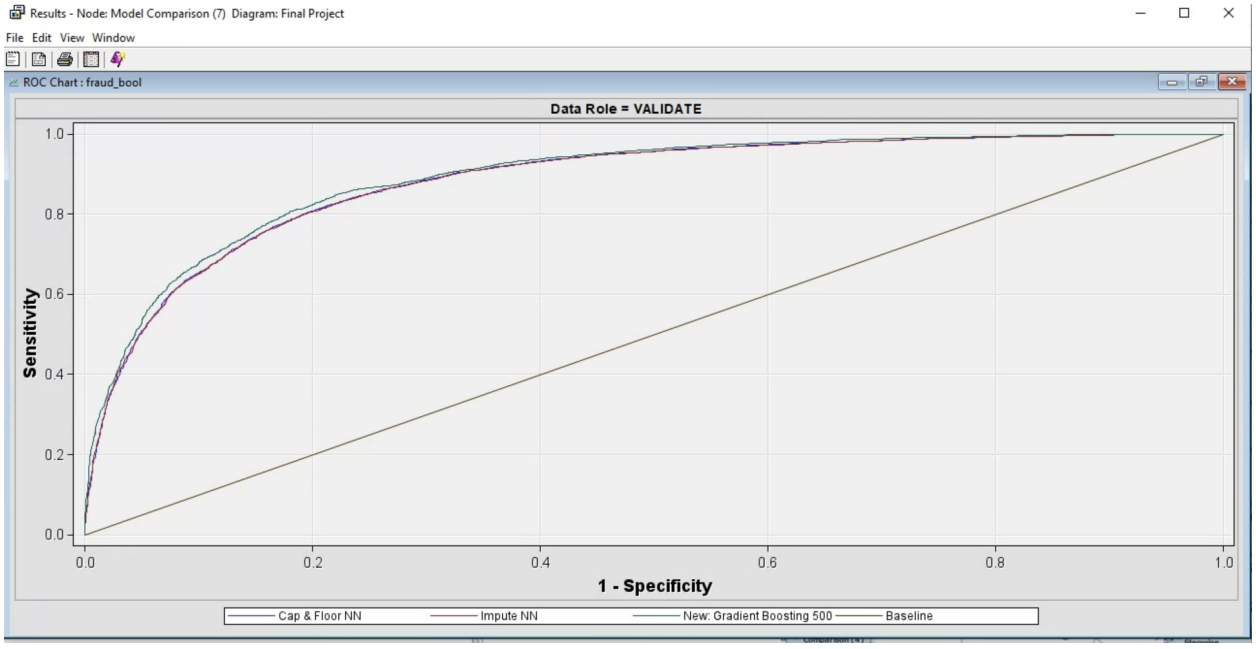


Selected Model	Predecessor Node	Model Node	Model Description	Target Variable	Valid: Average Squared Error ▲	Target Label	Selection Criterion: Valid: Misclassification Rate	Train: Sum of Frequencies	Train: Sum of Case Weights Times Freq	Train: Misclassification Rate	Train: Maximum Absolute Error	Train: Sum of Squared Errors	Train: Average Squared Error	Train: Root Average Squared Error	Train: Divisor for ASE	Train: Total Degrees of Freedom	Va of Fr
Y	Boost2	Boost2	New: Gradient Boosting 500	fraud_bool	0.132888		0.187217	11028	22056	0.159956	0.996373	2539.078	0.11512	0.339293	22056	11028	
	Neural3	Neural3	Cap & Floor NN	fraud_bool	0.137973		0.196283	11028	22056	0.190062	0.99529	2966.427	0.134495	0.366736	22056	11028	
	Neural4	Neural4	Transform NN	fraud_bool	0.137973		0.196283	11028	22056	0.190062	0.99529	2966.427	0.134495	0.366736	22056	11028	
	Neural25	Neural25	New: NN 4H Stepwise	fraud_bool	0.138007		0.196827	11028	22056	0.194686	0.98989	2996.903	0.135877	0.368615	22056	11028	
	Neural7	Neural7	NN 4H For	fraud_bool	0.138007		0.196827	11028	22056	0.194686	0.98989	2996.903	0.135877	0.368615	22056	11028	
	Neural2	Neural2	Impute NN	fraud_bool	0.138197		0.198187	11028	22056	0.190696	0.995251	2971.342	0.134718	0.36704	22056	11028	
	Neural10	Neural10	NN 3H Poly	fraud_bool	0.13878		0.199909	11028	22056	0.195049	0.99394	2994.305	0.135759	0.368455	22056	11028	
	Neural5	Neural5	Recode NN	fraud_bool	0.13878		0.199909	11028	22056	0.195049	0.99394	2994.305	0.135759	0.368455	22056	11028	
	Boost	Boost	New: Gradient Boosting 100	fraud_bool	0.138853		0.196283	11028	22056	0.189427	0.980547	2983.394	0.135265	0.367783	22056	11028	
	Neural11	Neural11	NN 4H Poly	fraud_bool	0.139636		0.196736	11028	22056	0.195956	0.987488	3038.468	0.137761	0.371162	22056	11028	
	Neural26	Neural26	New: NN 5H Stepwise	fraud_bool	0.139945		0.197189	11028	22056	0.191966	0.993655	2974.231	0.134849	0.367218	22056	11028	
	Neural9	Neural9	NN 5H For	fraud_bool	0.139945		0.197189	11028	22056	0.191966	0.993655	2974.231	0.134849	0.367218	22056	11028	
	Neural13	Neural13	NN 6H For	fraud_bool	0.140103		0.20136	11028	22056	0.195502	0.998486	3013.467	0.136628	0.369632	22056	11028	
	Neural27	Neural27	New: NN 6H Stepwise	fraud_bool	0.140103		0.20136	11028	22056	0.195502	0.998486	3013.467	0.136628	0.369632	22056	11028	
	Neural12	Neural12	NN 5H Poly	fraud_bool	0.14153		0.201995	11028	22056	0.18979	0.994601	2992.34	0.13567	0.368334	22056	11028	
	Neural24	Neural24	New: NN 3H Stepwise	fraud_bool	0.141567		0.200272	11028	22056	0.197135	0.982502	3054.076	0.138469	0.372114	22056	11028	
	Neural8	Neural8	NN 3H For	fraud_bool	0.141567		0.200272	11028	22056	0.197135	0.982502	3054.076	0.138469	0.372114	22056	11028	
	Reg5	Reg5	Polynomial Regression	fraud_bool	0.141711		0.202085	11028	22056	0.184258	0.998859	2904.011	0.131665	0.362857	22056	11028	
	Neural14	Neural14	NN 6H Poly	fraud_bool	0.142436		0.201541	11028	22056	0.197225	0.991559	3052.259	0.138387	0.372004	22056	11028	
	Reg4	Reg4	Stepwise Regression	fraud_bool	0.144698		0.204896	11028	22056	0.204389	0.996113	3198.386	0.145012	0.380805	22056	11028	
	Reg3	Reg3	Backward Regression	fraud_bool	0.144735		0.20417	11028	22056	0.203845	0.996373	3192.602	0.14475	0.38046	22056	11028	
	Reg	Reg	Full Regression	fraud_bool	0.144776		0.20417	11028	22056	0.203845	0.996327	3192.701	0.144754	0.380466	22056	11028	
	Neural28	Neural28	New: Cap & Floor NN PCA	fraud_bool	0.146428		0.203113	8823	17646	0.213759	0.981915	2611.23	0.147979	0.38468	17646	8823	
	Tree5	Tree5	New: ASE Tree 3B PCA	fraud_bool	0.157839		0.221551	8823		0.216026	0.954145	2725.813	0.154472	0.393029	17646	8823	
	Tree4	Tree4	ASE Tree 3B	fraud_bool	0.169517		0.245875	11028		0.232136	0.956522	3515.462	0.159388	0.399234	22056	11028	
	Tree6	Tree6	New: ASE Tree 3B Treated	fraud_bool	0.169793		0.247325	11028		0.236308	0.95416	3551.325	0.161014	0.401266	22056	11028	
	Tree2	Tree2	ASE Tree	fraud_bool	0.170573		0.250952	11028		0.236942	0.958609	3551.939	0.161042	0.4013	22056	11028	
	Tree3	Tree3	Misclassification Tree	fraud_bool	0.175272		0.245422	11028		0.225245	0.878213	3624.659	0.164339	0.405387	22056	11028	
	Neural20	Neural20	ASE 3B NN	fraud_bool	0.182003		0.33146	11028	22056	0.33288	0.970364	3992.956	0.181037	0.425485	22056	11028	
	Neural6	Neural6	ASE NN	fraud_bool	0.184877		0.335449	11028	22056	0.335963	0.980103	4052.4	0.183732	0.42864	22056	11028	
	Neural	Neural	Misclassification NN	fraud_bool	0.274552		0.483772	11028	22056	0.476786	0.932377	5999.42	0.272009	0.521544	22056	11028	

From the statistics we can come to the conclusion that Gradient Boosting 500 is the best model. It has the lowest Average Squared Error at **0.132888** and **0.187217** Misclassification Rate. Cap and Floor was the second modification in all the data modifications we have done. None of the skews were adjusted in this model. However it is important to note that the boosting models were developed before treating the data - cap & floor, adjusting skews, etc. Despite the lack of data adjustments, the Gradient Boosting 500 is the best model.

Using the ROC index and Gini coefficient from the screenshot below, we confirm that Gradient Boosting 500 is indeed the best model. The highest ROC and Gini are preferred. Cap and Floor NN have a ROC index of 0.883 and a Gini coefficient of 0.766. Gradient Boosting 500 has an ROC index of **0.892** and a Gini coefficient of **0.783.**.

Predecessor Node	Model Node	Model Description	Target Variable	Valid: Average Squared Error	Valid: Roc Index ▼
Boost2	Boost2	New: Gradient Boosting 500	fraud_bool	0.132888	0.892
Boost	Boost	New: Gradient Boosting 100	fraud_bool	0.138853	0.885
Neural3	Neural3	Cap & Floor NN	fraud_bool	0.137973	0.883
Neural4	Neural4	Transform NN	fraud_bool	0.137973	0.883
Neural25	Neural25	New: NN 4H Stepwise	fraud_bool	0.138007	0.883
Neural7	Neural7	NN 4H For	fraud_bool	0.138007	0.883
Neural2	Neural2	Impute NN	fraud_bool	0.138197	0.883
Neural10	Neural10	NN 3H Poly	fraud_bool	0.13878	0.882
Neural5	Neural5	Recode NN	fraud_bool	0.13878	0.882
Neural11	Neural11	NN 4H Poly	fraud_bool	0.139636	0.88
Neural26	Neural26	New: NN 5H Stepwise	fraud_bool	0.139945	0.88
Neural9	Neural9	NN 5H For	fraud_bool	0.139945	0.88
Neural13	Neural13	NN 6H For	fraud_bool	0.140103	0.88
Neural27	Neural27	New: NN 6H Stepwise	fraud_bool	0.140103	0.88
Neural24	Neural24	New: NN 3H Stepwise	fraud_bool	0.141567	0.877
Neural8	Neural8	NN 3H For	fraud_bool	0.141567	0.877
Neural12	Neural12	NN 5H Poly	fraud_bool	0.14153	0.877
Reg5	Reg5	Polynomial Regression	fraud_bool	0.141711	0.877
Neural14	Neural14	NN 6H Poly	fraud_bool	0.142436	0.875



As seen in the picture above, the Gradient Boosting 500 mode has the largest area under the ROC curve, compared to the other best models - the Cap & Floor and Impute Neural Networks, especially at lower levels of specificity. Also, Cap & Floor NN and Impute NN have the exact same ROC curve across all levels of specificity. It is difficult to make any distinction between the two. However, their error rates are different only by 0.000224 (0.138197-0.137973). Though negligible in most scenarios, in the case of modeling lower error rate gets higher priority.

Neural Networks(NN) have their own logic in deriving the best model, and one of the biggest disadvantages of these types of models is that the interpretation of data is next to impossible. However, we can analyze the fundamentals of this model to get an idea as to why we received the best model without significant modification of the dataset. Through all the adjustments made in our model, we were trying to fit our model. The more we tried to fit, the further we strayed from the truth of the dataset. Neural Network is a robust model mechanism that can work with all variables to create a relation. In this case, NN turned out to be the best model to use.

As a matter of fact, outside of the boosting models, all the other top models are NNs. The 2nd best is Impute NN which we discussed. The 3rd best model is a 3-hidden unit NN attached to a Forward regression with an ASE of 0.139412 and a Misclassification rate of 0.197915.

Recommendations and Key Findings

Models to Use

Upon evaluating the performance of all the models, we have determined that the Gradient Boosting 500 model is the most accurate at predicting fraudulent bank account applications. This was determined by evaluating average squared error, the ROC index, and Gini coefficient. We recommend that the bank implement the Gradient Boosting 500 model to identify and flag potentially fraudulent account applications.

Key Features

The following table outlines some of the selected key features that were identified to be important for predicting fraudulent applications by three models of varying types: 3 Branch Decision Tree, Forward Logistic Regression, and Cap and Floor Neural Network.

Decision Tree	**Logistic Regression Odds Ratios**	**Neural Network Weights**

Housing Status	Device Distinct Emails	Current Address Months Count
Device OS	Has Other Cards	Velocity_4w
Has Other Cards	Device OS	Device Distinct Emails
Keep Alive Session	Keep Alive Session	
	Housing Status	
	Payment Type	

Features to monitor

Based on our analysis, the key features that seem to be the most predictive of bank account fraud are housing status, device OS, whether the applicant has other cards, keep alive session, and payment type. Our models, including decision trees, logistic regressions, and neural networks, all identified these variables as predictors of fraud. Additionally, our neural network weights and decision tree splits suggest that current address months count, Velocity_4w, and Device Distinct Emails are also important in predicting fraud.

Applications that are most likely to be fraudulent are ones where the applicant does not have any other cards with the bank and has not been living in their current address for very long, with a current housing status of BA. Additional indicators are the applicant paying by payment type AC and choosing not to keep the browser session alive on logout. A high number of applications using different email addresses from the same device, and submitted at a time with a high velocity of applications are also more likely to be fraudulent.

The presence of these features increases the likelihood of an application being fraudulent. As such, banks should scrutinize such applications that contain any or all of these features during the account approval process in order to flag potentially fraudulent account applications for further investigation.

Finally, since the dataset contains anonymized values which cannot be interpreted without knowing their meaning, such as payment type, housing status, and employment status etc. It is recommended that the bank investigate the anonymized values to gain a clearer understanding of the features of fraudulent applications. Understanding the meaning of these variables and

how they can be used to identify fraudulent applications may help banks more effectively detect and prevent fraud.

Conclusion

In conclusion, our project aimed to identify key features of fraudulent bank account applications and to train machine learning models that can accurately predict fraudulent applications. After oversampling, data partition, and treating the missing and skewed data, we trained decision trees, logistic regressions, gradient boosting, and neural network models to predict fraudulent applications.

The result of our modeling showed that housing status, device OS, device distinct emails, and presence of other cards were key variables in predicting fraudulent applications. Based on our findings, we recommend that the bank consider the identified key features when evaluating new account applications, and use the Gradient Boosting 500 model for most accurate predictions. By implementing these recommendations, the bank will be better equipped to identify and prevent fraudulent account openings.

References

1. *5 types of bank account fraud – and how to prevent them*. SEON. (2022, December 7). Retrieved December 15, 2022, from https://seon.io/resources/bank-account-fraud/
2. *New account fraud*. OneSpan. (n.d.). Retrieved December 18, 2022, from https://www.onespan.com/topics/new-account-fraud

Appendix

Full Model Screenshot