

ITBW41 Predictive Analytics Project

Final Report

Healthcare Predictions

| Submitted by: | Shermaine Peh (213276Q)\*  Koh Shi Min (201357X)  Zhang Xiang (210896X)  Rawtbhik (210902G) |
| --- | --- |
| Module group: | ITBW41 - 01 |
| Tutor: | Mr Law Chee Yong |
| Report Last Amended: | 24th August 2022 |

**Table of Contents**

[**Executive Summary**](#_sn0nsekpssen) **2**

[**Data Understanding and Cleaning**](#_itlkekvvl2p1) **3**

[Stroke](#_hkncjd4dju3z) 3

[Data Understanding](#_qd7noq48w7va) 3

[Data Cleaning](#_hildi1hntmvc) 4

[Data Balancing, Partitioning & Labelling](#_pzme0uaq7kfy) 5

[Data Anonymization](#_fbhpdobmqjdv) 5

[Diabetes](#_2ljtobmxzsdh) 6

[Data Understanding](#_c3szcro9mdal) 6

[Data Cleaning](#_jiu8ff7bkjqi) 7

[**Modelling**](#_hepb4tschdbl) **9**

[Stroke](#_gjeoafunu3zy) 9

[Logistic Regression](#_g6812533wnya) 9

[Decision Tree](#_p18l9e7wnfza) 10

[Support Vector Machine](#_j70jiiesuz6x) 14

[Neural Network](#_aw5rcikinifn) 16

[Model Comparison](#_8vdtprdivgjq) 19

[Diabetes](#_2f1mrjlrjrdz) 20

[Logistic Regression](#_54569g4zbxop) 20

[Support Vector Machine](#_93tckwpz3q0u) 23

[Decision Tree](#_pjic1n1ktqb) 25

[Random Forest](#_nd24k9rz84jn) 27

[Model Comparison](#_xpduyi7wc0r3) 29

[**Conclusion**](#_atu53bol30qn) **30**

[Evaluating Objective](#_57wzxhs5yabb) 30

[**Appendix**](#_puypo4jrica5) **31**

[Stroke](#_67slacn7zbbt) 31

[Data Understanding & Preparation](#_v4nbbcdbyi4h) 31

[Modelling](#_z0ytil4wjt9i) 36

[Diabetes](#_459g0axo3p2c) 52

[Data Understanding & Preparation](#_juo4tu9334uy) 52

[Modelling](#_f9s6fl7b86sc) 57

# 

# ***Executive Summary***

What are common issues that require the help of AI and Machine Learning? How can Prediction help?

Upon deciding on our topic, our team discovered that the healthcare industry suffers from high attrition rates, hence face severe understaffing. Moreover, 1 in 201 patients face delayed treatment as a result of misdiagnosis or failure to seek help early enough. We, therefore, decided to focus on the healthcare industry before breaking it down into smaller factors.

Further understanding revealed that stroke and diabetes are two common health problems on the rise today. As a result, we divided our main topic into two categories, namely stroke and diabetes. Shermaine and Shi Min will take stroke prediction, while Zhang Xiang and Rawtbhik take diabetes.

So then, what is our goal in selecting this industry and our topics?

As a team, we have come to 3 common goals wanting to be able to develop predictive models of 80% accuracy in predicting stroke and diabetes, pinpoint factors associated with those health problems, and help predict health problems more accurately.

Following this, we'll take you through our predictive models curated, starting with data collection, understanding, cleaning, modelling, and evaluating them. In the end, we will then return to our goals to determine whether they have been met.

Last but not least, our team found that high blood pressure, obesity, and age are the leading causes of common health problems today. A person's risk of getting common health diseases increases with age. An individual with high blood pressure is more likely to suffer from health problems. People with a higher BMI are more likely to suffer from common diseases.

Overall, we recommend patients take note of the 3 main contributing factors and maintain a healthy lifestyle so that they don't succumb to these common health problems. Moreover, doctors can also advise and keep an eye out for patients with these worrying factors, so that health problems can be detected earlier.

# ***Data Understanding and Cleaning***

## Stroke

### Data Understanding

Aftering obtaining our dataset, we first did data understanding.

With the use of a[*bar chart*](#n8ni24jzm40g)*,* we compared the count of each target value. From this image, it shows that about **95%** of our raw data states that an individual does **not** have a stroke. With this result, our model is proven to be **highly unbalanced** and will be misled during the creation of the models as our models will only be accurate in predicting individuals that do not have a stroke. Hence, we will be balancing it.

Next, a [*bar chart*](#ugdn4i8yk9mv) also reveals that smoking\_status contains unknown smoking values. Based on the dataset explanation, "Unknown" in smoking\_status indicates that the patient's information is **not available**. Due to the **subjective** nature of smoking, smoking\_status that contains “Unkown” values will be removed.

Next, from [*this chart*](#y4tva1e0m45m), it reveals that the gender column also consists of one “Other” value. Similar to smoking\_status “Unknown”, as there was no further explanation for this value in the dataset, and there was only one occurrence of such value, we decided to remove it.

Next, we also see that the [*age column*](#87idud8wqq2h) contains **decimal** values. As the dataset **did not include a decimal age explanation**,it is **uncertain** whether it is a wrong value or if it signifies something else. Hence, we decided to remove it.

Following that, with a [*scatter plot*](#lzb6buxtdif5), we see that bmi columns contain 201 occurrences of “N/A” values. To ensure the dataset is not affected by the bmi outliers, a best practice would be to do an imputation on it using the average bmi from all rows.

Lastly, our [*dataset*](#z1qi59o7g12n) contains unique and explicit identifiers such as the id and age. These data can potentially link and reveal the personal information of an individual, hence, a best practice would be to anonymise them.

### Data Cleaning

Now for the data cleaning process, [here](#17jbalbz6g54) shows an overview of our data cleaning process.

A [best practice](#oc8rk4boh0td) before doing other cleanings would be to ensure the cleanliness and integrity of the dataset. To do so, performing clean and trim to remove any white and leading spaces in existing columns is one method. Hence, using the String Manipulation node in KNIME, all data types are converted to strings before using a function called "strip" to remove white and leading spaces.

Then, we removed dirty values such as “Others” and “Unknown” in gender and smoking\_status respectively using the “Row Filter” node.

Afterwards, the focus will be on the dirty values and outliers found from Data Understanding.

First, records that contain values such as “Unknown” and “Others” in smoking\_status and gender respectively has been removed using the “Row Filter” node.

Next, bmi outliers “N/A” values are converted using the “Rule Engine” node with this expression.

Afterwards, the bmi data type is then converted from string to decimal also known as double in KNIME for imputation.

Using the missing value node in KNIME, an imputation is then performed on the bmi null values using the average values of the entire bmi.

Next, the age column data type has been converted from decimal to integer as during the process of removal for smoking\_stauts “Unkown” all age outliers have been removed along with it.

#### 

#### Data Balancing, Partitioning & Labelling

The section below will cover data balancing, data partitioning, and data labelling.

With the use of [*KNIME’s SMOTE node*](#o0v1k4n5dcfq)*,* which uses the **Synthetic Minority Oversampling Technique.** The dataset has been oversampled by the **minority** class, whose **stroke is 1** so as to get **more values** for the **minority** classes. Overall the total row produced was **6726** rows from a previous of **3565, with** a **50:50** amount on the **target variable, stroke.**

After balancing, with the use of a [***“Partitioning”***](#3wksolui7kjb)node. The dataset has been **partitioned into 80% training and 20% testing**. Afterwards, the partitioned data is **labelled** into **0s and 1s**, using the **“constant value”** node, where **0 represents training data and 1 represents testing data**.

Finally, to **combine** the two labelled datasets into one, the **“Concatenate”** node was used. This resulted in us being able to get a **single dataset,** with the label as partitioning variable, to be inputted into SAS Viya.

#### Data Anonymization

After data transformation, balancing, partitioning, and labelling of the dataset. It is important to check for data protection issues, hence we perform [*data anonymisation.*](#ybno9twdt68f)

The first two “RowID” nodes update existing KNIME-generated row ids and create a column based on them. To ensure that there are **no duplicates** in the rowIDs, a **"Duplicate Row Filter"** node has been used**.** Lastly, the **id** column was **removed** from the dataset to be **saved into a CSV file.** At the same time, **a master table was created** without any columns except **id and rowID** for **future reference** and **linking** of individuals by **authorised personnel**.

## 

## Diabetes

### Data Understanding

Each data column in the dataset is understood through a **Codebook**. From the codebook we have **shortlisted 19 variables by logic, theory and through references in Kaggle** in predicting diabetes (Diabetes included: 20 Variables). During modelling, feature selection will be conducted to lookout for **multicollinearity issues and p-value** to remove unnecessary variables.

Our [target variable diabetes](#76yoyzzh2rsu) was differentiated into **6 categories** according to the [codebook](#76yoyzzh2rsu). It is encoded with values of 1, 2, 3, 4, 7 and 9. Since, our target for modelling would be a **binary result**, we will consider only encoded responses of 1 to 4, which give an understanding in knowing if surveyees have diabetes or do not have.

To add on, since we are conducting classification of **binary outcomes**, we have categorised values 1 to 4 into 2 sections, with diabetes or no diabetes. With that, values 1 and 2 were set into value “1” as “Yes” to having diabetes. Values 3 and 4 were set to value “ 0” as “No” to having no diabetes.

Noticing that the target variable [diabetes was unbalanced](#76yoyzzh2rsu) as there were more records of respondents without diabetes than respondents who have diabetes. The result of utilising an imbalanced dataset is that the model will be biassed towards the majority class only. To **avoid biases**, data balancing was performed in target variables before modelling.

In the dataset multiple variables contain input responses such as [“Don’t Know”, “Not Sure”, “Refused” and as well as BLANK](#kix.jva8w0ttncr7), which is represented by value 7 and 9. It is intended that rows that contain these values will be removed as they do not provide any valuable insights.

[BMI](#kix.g2udfsr8jvvw) were recorded in the **thousands (E.g., 1275 => 12.75)** as it is stated by the codebook that BMI values **“Has 2 implied decimal places”**. With this, we would need to divide the BMI values by **dividing 100** during cleaning to return a value that would be logical to comprehend the data.

[Outliers](#kix.g2udfsr8jvvw) were also found in the BMI feature, with values ranging from 12.75 to more than 90. Looking at the histogram, the range of values was widely spread out with fewer records left on the right side. As we know that machine learning algorithms are sensitive to the range and distribution of attribute values. Data outliers can spoil and mislead the training process resulting in **longer training times, less accurate models and ultimately poorer results**. Therefore, removal of the outlier is necessary during the cleaning phase.

In the codebook definition, the encoded value “1” can represent a “Yes” or “No” response, which might be complicated when trying to interpret modelling output. Therefore we decide to [synchronise](#kix.3v705w9nh9qy) all binary outcomes by changing value 1 and 2 to 1 and 0, where 1 represents a “Yes” and 0 represents a “No”.

### Data Cleaning

**Knime tools were used for this process.**

*Converting 7 and 9 to Missing Value plus the Removal of Missing Values*

Knowing that **values 7 & 9 would provide no valuable insight**, we decided to **convert** values that are greater than **value 6 to “?”** which is known as missing value in KNIME with the help of a rule engine. After converting 7 & 9 to missing values, we would then **remove all rows containing “missing Values”** in every column with the help of the “Missing Value” node.

*Balancing*

"Equal Size Sampling" node was used to **perform undersampling** in order to generate a balanced dataset as our dataset was large enough to deduce. If oversampling was done, our model would not perform well on synthetic generated data. We believe that having Real-world **human annotated data is a necessary** part of machine learning training data.

*Data Transformation*

For the BMI feature, we have to **divide these numbers by 100** to return the output back to 2 decimal places with the use of a “Math Formula” node to conduct calculation.

Furthermore, some of the variables in our dataset have values 1 that represent "Yes." However, some variables also have value 1 that represents "No." As a result, we decided to **set all the inputs to only 0s and 1s.**

*Outlier*

Since outliers were observed during understanding, we used a "Numeric Outliers" node to remove rows with BMI outliers. The outlier is calculated **using a standard Interquartile range multiplier** of 1.5, which sets the **lower bound BMI to 12.975** and the **upper bound BMI to 41.655**; any BMI value that falls outside of this range is considered an outlier and is removed.

*Partitioning*

We partitioned our dataset to training and testing sets with the help of the "Partitioning" node. To determine which rows of the dataset is used for training and testing in SAS Viya, we then used two nodes called "Constant Value Column" nodes to **create a new column called "Partition"** that would contain **values "0" and "1", which represent training and testing,** respectively.

# 

# ***Modelling***

For both Stroke, we used a total of **four** algorithms, namely Logistic Regression, Decision Tree, Support Vector Machine and Neural Network. As for diabetes, we used Logistic Regression, Decision Tree, Support Vector Machine and Random Forest.

## Stroke

### Logistic Regression

*This section is documented by Shi Min*

For Stroke prediction, Logistic Regression is used for feature selection. To select the features to be inputted into the model, we have made different comparisons to see which has the best accuracy. By default, all 11 input variables will be added to the logistic regression. The 11 default input variables are as follows:

1. Gender
2. Age
3. Hypertension
4. Heart\_disease
5. Ever\_married
6. Work\_type
7. Residence\_type
8. Avg\_gulcose\_level
9. Bmi
10. Smoking\_status
11. Stroke

First, from the [default model](#ydjwv5wgi3cz), it has provided a misclassification rate of 0.2140 and it has indicated that work\_type, residence\_type, and ever\_married are features that would not help explained the model.

Next, removing features that would not help in explaining the model, has provided a misclassification rate of 0.2169 with no features that are insignificant as shown [here](#h69cnngnf1lf).

Lastly, using a combination of forward and backward feature selection also known as [stepwise](#vekgpuivfs6j) feature selection, has provided a misclassification rate of 0.2184 with work\_type, residence\_type, ever\_married, bmi, gender, and heart\_disease as features being removed as shown below.

From the three Linear Regression models created, the decision chosen was to go with the first 2 models, the Default model, and the Significant Variables model, to test if there will be a difference between the two features inputted for models created.

Now into the details of the algorithm chosen.

### Decision Tree

*This section is documented by Shi Min*

For the Decision tree, the parameters tuned will be the maximum number of branches, maximum level, leaf size and the number of predictor bins are the parameters that will be tuned.

The maximum number of branches indicates how many criteria will be used to split a decision tree node. The maximum level will allow the decision tree to have a more in-depth analysis. Leaf size sets the minimum observations that are allowed in a leaf node while changing the predictor bin allows the specified number of bins used to categorize a predictor that is a measured variable as sometimes measure variables do not work well with non-linear models.

Next, four different models were created and a comparison was made based on the misclassification rate, receiver operating characteristic (ROC) curve, and the lift chart to select the best model.

The table below shows the model comparisons, with the default model, parameters tuned, and with significant variables.

| **Models** | **Parameters Tuned** | **Misclassification Rate** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| Default | – | 0.2103 | 0.5795 | 1.7506 |
| Default Tuned | 2 branches, 14 level max, 1 Leaf, predictor bin - 170 | 0.1813 | 0.6434 | 1.6058 |
| Significant Variable Tuned 1 | 3 branches, 13 level max, 1 Leaf, predictor bin - 165 | 0.1820 | 0.6360 | 1.7662 |
| Significant Variable Tuned 2 | 3 branches, 8 level max, 3 Leaf, predictor bin - 50 | 0.1939 | 0.6211 | 1.7710 |

By default, the standard parameters is 2 maximum branches, 6 maximum levels, leaf size of 6 and 50 predictor bins.

As the goal is to develop predictive models of 80% accuracy, a comparison among all four models’ misclassification rates can provide the trained model accuracy as it is an indication of the models trained error prediction. Hence, comparing the validation misclassification rate across all four models created, the lowest misclassification rate comes from the Default Tuned model which has 2 maximum branches, 14 maximum levels, leaf size of 1, and 170 predictor bins. With a misclassification rate of 0.1813, it indicates that the model has an 18.13% chance of wrongly predicting if a patient has a stroke or does not have a stroke. In addition, it also indicates that the Default Tuned model has an 81.87% accuracy.

However, when the training misclassification rate and validation misclassification rate are compared. The Default Tuned model has an 8% difference as compared to the other three models where on average is only a 1% to 2% difference as shown [*here*](#kj8g4ehgmpm2). When models can accurately predict one instance of a target variable as compared to the other instance, the model is deemed as overfitted. Hence to prevent a model from being overfitted, a best practice would be to keep the difference between training and validation misclassification rate below 10%.

As the Default Tuned model has a high percentage difference as compared to the others, it will not be selected as the best model. Following the Default Tuned model, Significant Variable Tuned 1 model with 3 maximum branches, 13 maximum levels, leaf size of 1, and 165 predictor bins is the next lowest misclassification rate. It has a misclassification rate of 0.1820, indicating that it has an 81.80% accuracy and will have an 18.20% chance of wrongly predicting if a patient has a stroke or does not have a stroke.

ROC curve indicates how well the model trained can correctly proportion the target variable, stroke. Hence, when looking at the ROC curve Significant Variable Tuned 1 model did slightly better as compared to the Significant Variable Tuned 2 and Default Tuned models trained. However, looking at the Lift, an indication of how well the model trained did compare to no model trained at all, Significant Variable Tuned 2 model did better as compared to the Significant Variable Tuned 1 and Default Tuned models. Hence, as the goal is to develop predictive models of 80% accuracy, Significant Variable Tuned 1 is the overall best model due to its low misclassification rate which provides an 81.80% accuracy.

Below sections will be an in-depth analysis of the best model.

Looking at the statistical summary of the [*best model*](#guxrz2imzuk8), it shows that the training misclassification rate and validation misclassification rate is estimated to be of 1% difference.

Hovering over each branch indicates the percent of the parent branch. The thickness of each branch indicates the number of occurrence for that branch. Hence, following the branch where most occurrences of the rule are split by, the final output is node 97. More details on the percentage of the parent branch after the avg\_gulcose\_level parent branch can be referred to [*here*](#3ksvlxgj0hee).

From [*this*](#41nkqy1omiy0) screen capture, node 97 can be derived as such, patients who are aged 66 to 79 with an average glucose level of 80.57 to 86.951, bmi of 21.334 to 34.4 and have no history of hypertension have a 70.15% chance of getting stroke and a 29.85% of not getting a stroke. With this information, doctors can be more aware of patient’s history of hypertension as well as their age, average glucose level, and bmi whose range falls between any of the ranges mentioned previously.

When switched over to the Variable Importance tab, it states that age is ranked the most important factor when the decision tree model is considering if a patient has a stroke or does not have a stroke, followed by average glucose level, bmi, hypertension, and heart disease. Looking at the variable importance statistical summary, it further shows all inputted variable importance as well as their standard deviation. From the [*variable importance*](#w723v846vbrh) statistical summary, gender and smoking\_status are of 0 importance indicating that the decision tree did not use these variables to make a prediction on the target variable stroke.

Changing Variable Importance to Leaf Statistic shows the purity of the leaf nodes. Purity in the leaf nodes indicates the probability of the target variable. From [*this*](#p18uwjjt9oaw) screen capture, there are 4 nodes in which the model predicts that the patients will have a 100% chance of getting a stroke, indicated in yellow. Looking at the [*first instance*](#vec4yfr31gut), node 34, shows that patients aged 48 to 63 with an average glucose level of 192.25 to 215.92 and bmi of 11.5 to 22.03 have a 100% chance of getting a stroke. In the [*second instance*](#yz9v6fhjcmcl), node 55, shows that patients aged 48 to 63 with an average glucose level of 104.461 to 105.176 and bmi of 25.6 to 30.908 have a 100% of getting a stroke. In the [*third instance*](#pabhhj3enodk), node 77, shows that patients aged 79 and above with an average glucose level of 58.792 to 87.867 and bmi of 33.9 to 34.4 have a 100% of getting a stroke. In the [*last instance*](#a92b21kikxbk), node 90, shows that patients aged 54 to 59 with an average glucose level of 102.716 to 110.409 and bmi of 32.997 to 36.4 also have a 100% of getting a stroke. With these informations, doctors can be more aware of patients if they happen to fall under the four nodes mentioned.

Looking at the [*model lift chart*](#pu52eo364rw) at the 20th percentile, it can be derived that the model can predict 1.7662 times better whether a patient will have a stroke or not as compared to having no models trained at all during the first 20% of the data passed in. This will result in doctors being able to identify patients with strokes quickly and aid them when necessary.

Lastly, looking at the ROC curve, shows that the training and validation ROC are quite close to each other. This indicates that the model is not too far apart and overfitted. To obtain the correct prediction accuracy, the focus will be on the validation ROC curve. This [*screen capture*](#c0xj9vhfigt4), states that the separation (KS(Youden)) is at 0.6360 indicating that the model has a 0.6360 chance of correctly proportioning patients who have a stroke and patients who do not have a stroke.

### 

### Support Vector Machine

*This section is documented by Shermaine*

A **support vector machine** is used to perform classification by constructing a set of **hyperplanes** that maximizes the margin between two classes.

For this algorithm, **four** models were created, and the **kernel and penalty** parameters were tuned.

Using the **misclassification, ROC and Lift** as a comparison, I then compared all 4 models to find the best model for the support vector machine as shown below.

| **Models** | **Parameters Tuned** | **Misclassification** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| Default | **Kernel**: Linear,  **Penalty Value**: 1 | 0.2162 | 0.5721 | 1.6716 / 2.02 |
| Default Tuned with Kernel | **Kernel**: Quadratic  **Penalty Value**: 1 | 0.2006 | 0.6077 | 1.7459 / 2.02 |
| Default Tuned with Kernel & Penalty | **Kernel**: Cubic,  **Penalty Value**: 86.5 | 0.1783 | 0.6434 | 1.7608 / 2.02 |
| Tuned with Selected Variables | **Kernel**: Cubic,  **Penalty Value**: 77.7 | 0.1991 | 0.6033 | 1.7162 / 2.02 |

* The **kernel** function is used for **spatial classification** and generally transforming data points to a high-dimensional feature space, which computes **decision boundaries** in terms of **similarity** measures. The **strongest** kernel here is **Cubic**, which was used in my best model.
* The **penalty value** then **balances model complexity** and training error by influencing the **decision boundary margin size**. A high penalty results in a decision boundary with a smaller margin, which **minimizes the number of misclassified** data.

For tuning, I first tuned the **kernel**, until I reached the **Cubic** Kernel, providing me with the **lowest misclassification.** Following that, I then tuned the **penalty value** and finally for the last tuning, to **reduce the features** to remove the **insignificant** values as mentioned earlier here. However, despite tuning, the reduced features did not provide a better model than the model with all features. Overall, the **best model** produced was the model tuned with a **cubic kernel** and a **penalty value** of 86.5, providing us with a total of **82.2% accuracy.**

After selecting the best model, below provides a more in-depth explanation of the model.

From [*this graph*](#7nf9tgsic2ly), looking at the misclassifications highlighted, we can see that the validation misclassification is 0.1783. To ensure that it is **not overfitted**, we compared it to the training misclassification rate of 0.1632, which proves **little difference.**

Then, looking at the **relevance important plot**, we can also see that **gender and residence type** are **not as significant** as the rest in predicting stroke.

Next, let's take a look at the [*misclassification plot*](#bsnvv32qlc7f), with the values below. Here, we can use the values to calculate our

* Accuracy (TP+TN/ Total) = 82.2 %
* Sensitivity (TP/ TP + FN) = 87.4%
* Specificity (TN/ TN + FP) = 80%
* Precision (TP/ TP + FP) = 79.1%

Moving on will be to the [*lift chart*](#dootrbbh4v8i), where a **higher lift value provides a better model.** Here, our model tells us that at 20%, we can derive that our model can predict **1.76 times better whether a patient will have a stroke or not** as compared to a randomly selected patient, and **0.26 times less accurate than a perfect model.**

Finally, looking at the [*ROC curve*](#8i1iov4di8wi), which **summarizes the true positive rate and false positive rate** across **thresholds** or **cutoffs** in the data. The 45-degree line represents the performance of the naïve model, and the vertical dashed line corresponds to an **optimal threshold** in the data. In this model, the KS(Youden) is shown to be 0.6672, and with the ROC curve further up and to the left than the other models, **maximizing the true positive rate and minimizing the false positive rate.**

Overall, the Support Vector Machine can tell us that it is **relatively good in predicting stroke**, and also **meets our target** of 80% accuracy. Additionally, we can also see from this model, that gender and residence\_type are the less significant variables, while age is the most important factor, followed by avg\_glucose, hypertension, BMI, heart\_disease and so on.

### Neural Network

*This section is documented by Shermaine*

A neural network model **recognizes hidden patterns and correlations in data, clustering and classifying it.**

For this algorithm, **four** models were created, and the **hidden layer, L2, neurons, and optimisation method** parameters were tuned.

Using the **misclassification, ROC and Lift** as a comparison, I then compared all 4 models to find the best model for the neural network as shown below.

| **Models** | **Parameters Tuned** | **Misclassification** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| Default | **Hidden Layers**: 1, **L2**: 1  **Neuron**: 10,  **Optimisation method**: LBFGS | 0.2140 | 0.5795 | 1.6790 / 2.02 |
| Default Tuned with Hidden Layers, Neuron, Optimisation | **Hidden Layers**: 2, **L2**: 0.01 **Neurons**: 20,  **Optimisation method**: SGD | 0.4354 | 0.5988 | 1.7979 / 2.02 |
| Default Tuned with Hidden Layers, L2, Neurons | **L2**: 0.00046  **Neurons**: 20, 30 | 0.1679 | 0.6672 | 1.7682 / 2.02 |
| Tuned with Selected Variables | **Hidden Layers**: 2, **L2**: 0.000046 **Neurons**: 10, 14 | 0.1991 | 0.6033 | 1.7608 / 2.02 |

* The number of hidden layers represents an **arbitrary decision boundary to accuracy**, whereby adding **hidden layers** helps to get the **best decision boundary.**
* Weight values start at **random perturbations** and are iterated away from zero in the model fitting process, so relatively **small neuron size** indicates that these components have a relatively **small influence** on determining predictions and may indicate an **overly complex signal architecture.**
* When training with high neurons and hidden layers, our model is **prone to overfitting**, hence **regularization** makes slight modifications to the learning algorithm such that the model **generalizes better**, improving the model’s performance on the unseen data as well. However, **high regularisation** will cause the weight matrices to be **nearly equal to zero,** resulting in a much **simpler linear network and slight underfitting** of the training data. Hence, we need to **lower** the L2 to optimize the value of the regularization coefficient in order to obtain a **well-fitted model as such.**
* **Optimizers** are algorithms or methods used to **change the attributes** of the neural network such as weights and learning rate to **reduce the losses**. Optimizers are used to solve optimization problems by **minimizing** the function.

For tuning, I first tuned the **hidden layer**, whereby **2** provided me with the best misclassification rate. When tuned to **3**, the model was **overfitted.** Following that, I then tuned the **optimisation** method to SGD, which, however, gave me a **higher misclassification.** Hence, I decided to **keep** my optimisation method to the **default** one for my next tuning. For the next tune, I then tuned the **regularisation** and adjusted the **number of neurons** further, until the best model was provided. Finally, for the last tuning, I **reduce the features** to remove the insignificant values as mentioned earlier here. However, despite tuning, the reduced features did not provide a better model than the model with all features.

Overall, the **best** model produced was the model tuned with **2 hidden layers, 0.000046 L2, 20 neurons for one hidden layer and 30 neurons for another, and stagnation auto-stop,** providing us with a total of **83.2%** accuracy.

After selecting the best model, below provides a more in-depth explanation of the model.

From [*this graph*](#vlds7ota5833), looking at the misclassifications highlighted, we can see that the validation misclassification is 0.1679, while the training misclassification rate is 0.1468, proving little difference and that the model is **not overfitted.**

Then, looking at the [relevance important plot](#t4e0dx88w7fx), we see that **gender and residence type** are also **not as significant** as the rest in predicting stroke.

Next, let's take a look at the [*misclassification plot*](#ib26v4r83pz1), with the values below. Here, we can use the values to calculate our

* Accuracy (TP+TN/ Total) = 83.2 %
* Sensitivity (TP/ TP + FN) = 86.3%
* Specificity (TN/ TN + FP) = 80.1%
* Precision (TP/ TP + FP) = 81.3%

Moving on will be to the [*lift chart*](#uzvqsm8pu1zc), it tells us that at **20%,** we can derive that our model can predict **1.77 times better** whether a patient will have a stroke or not as compared to a randomly selected patient, and **0.24 times less accurate** than a perfect model. This will result in doctors being able to identify patients who have strokes and aid them **more efficiently.**

Finally, looking at the [*ROC curve*](#o1plv4da9iz) tells us that the training and validation ROC are quite close to each other. To see how well our Neural Network can predict, we can see that the KS(Youden) is at 0.6672, indicating that the model has a **0.6672 chance of correctly proportioning patients who have stroke and patients who do not.**

Overall, the Neural Network can tell us that it is **relatively good in predicting stroke,** and also **meets our target** of **80% accuracy**. Additionally, we can also see from this model, that gender and residence\_type, similar to Support Vector Machine, are the less significant variables. On the other hand, we see that age is the most important, followed by avg\_glucose\_level, hypertension, heart\_disease and so on, as shown from the relative importance plot.

### Model Comparison

*This section is documented by Shermaine*

For the final model, we **compared all models** to select the best model for stroke prediction.

When comparing all 3 of our models put together in the [*model comparison*](#e4yyyws71ytb), we can conclude that the **neural network model is our best model.**

Additionally, from the [*misclassification plot*](#s1gv8yvmcyf), it also holds the **lowest** validation misclassification rate of 0.168, with an **accuracy of 83.2%** compared to the Support Vector Machine and Decision Tree models with 82.2% and 81.8% accuracy respectively.

Moving on to the [*lift chart*](#pbbre3gba0dm), where the **lift** is plotted on the vertical axis and percentile on the horizontal axis, we can also see that the **neural network** performed at the **highest percentage** as compared to the other 2 models.

Then, looking at the [*ROC curve*](#ibxfzxwfi89o), we can see that the **neural network** model is also **furthest up** to the **left** and performing the best, followed by the support vector machine and lastly, the decision tree model.

Finally, to [*conclude*](#698du8gbkpje) the stroke prediction, while we see that our **best** model was trained with **all 11 input variables,** demonstrating the **importance and the necessity of all input variables** to produce the **most accurate model,** we, however, see that the **four most consistent** factors that affect stroke are actually **age, average glucose, hypertension, and BMI.**

Hence, we **recommend** patients take note of these **main contributing factors** so they **don't succumb** to stroke. Moreover, **doctors** can also advise and **keep an eye** out for patients with these worrying factors, so stroke can be **detected earlier.**

## 

## Diabetes

### Logistic Regression

*This section is documented by Rawtbhik*

Variables/Features that will be used:

1. Diabetes - Target Variable
2. Blood Pressure
3. High Cholesterol
4. BMI
5. Smoke Frequency
6. Smoking Status (Exclude from modelling due to multicollinearity)
7. Stroke
8. Coronary Heart disease
9. Arthritis (p-value > 0.05)
10. Kidney Disease
11. Consumption of Fruits (p-value > 0.05)
12. Consumption of Vegetables
13. Alcohol Consumption
14. Difficulty in Walking
15. Physical Activiteness
16. Difficulty in concentration/remembering
17. Gender
18. Age group 5YR
19. Depressive Disorder (p-value > 0.05)

We have tested our data with the Logistic Regression to find if there are any multicollinearity issues and Insignificant variables. Through this process, we found that in our dataset the “Smoke Frequency” and “Smoking Status” features are **highly correlated which causes both variables to have p-value of more than 0.05**, where either one of them would be removed. With that we decide to choose “Smoke Frequency” as it **tells us more about how often patients smoke**, and frequency of people smoking affects health conditions more.

Additionally, “Depressive Disorder”, “Arthritis” and “Consumption of Fruit” also **have p-values that are more than 0.05, we will remove it when we test our models.**

*This section is documented by Rawtbhik*

Here are the several tunings that have been done for logistic regression, and I will be sharing on its Misclassification Rate, ROC and Lift.

|  | **Parameters Tuned** | **Misclassification Rate** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| **Default** | - | 0.2952 | 0.4165 | 1.625 |
| **Tune 1** | Variable Selection P< 0.05 | 0.2923 | 0.4157 | 1.6282 |
| **Tune 2** | Forward/backward selection (SBC), Prediction Cutoff =0.52 | 0.2938 | 0.4124 | 1.626 |

For Default, which basically means **having all the variables in the model.** Its **misclassification rate is 0.2952, has ROC of 0.4165 and 1.625 for the lift** at the 20th percentile which would be implied to the rest of the tunings that I would further share on.

For the first tune, is the **exclusion of variables that have a p-value of more than 0.05** since real-life companies and industries would set p-value of 0.05 for feature selection. As stated **misclassification rate is 0.2923, has a ROC of 0.4157 and Lift of 1.628.**

For the second tune, **variable selection method** was done from none to forward then backwards and added a prediction cutoff of 0.52. It produced a **misclassification rate of 0.2938, ROC of 0.4124 and lift of 1.626.**

Now isn't it obvious which is the best performing one? It sure is the first tuning that I have done with an accuracy of 70.77%.

Now, allow me to share with you in detail and in depth my best model for logistic regression.

As you can see [here](#7zovlden26ta), all variables that have been chosen to be the classification have **p-value of less than 0.05**. These variables are important factors affecting Diabetes.

[Lift chart](#on0d3xy24h53) help us to interpret how well our model has been trained by comparing it to the model that has not been trained. Hence, looking at the lift chart at the 20 percentile, tells us that the logistic regression model can predict whether a person is likely to come down with diabetes or not by **1.6282 times better than the model that has not been trained**.

For the ROC [curve](#6lnh6btodr1z). To know how well the logistic regression model can predict, we will focus on the validation of the ROC curve. From this screen shot, we can see that the **separation (KS(Youden)) is at 0.4157**. The validation performs better than the training data.

### 

### Support Vector Machine

*This section is documented by Zhang Xiang*

In the support vector machine (SVM) it creates a **decision boundary** to classify two classes. To prevent overfitting, the choice of hyperplane can be made to separate the data. By finding the maximum margin hyperplane, the minimum distance will be maximised from the hyperplane to the closest training point. With that mentioned, parameters for SVM in SAS Viya need to be tuned to be kernel and penalty values to fix overfitting or improve model performance.

**Penalty value** influences the decision boundary margin size. With a high penalty assigned, it results in a decision boundary with a smaller margin, which minimises the number of misclassified training data, it works the other way round if penalty value is low. Hence, increasing penalty value while testing the model is needed, so as to minimise misclassification and improve accuracy.

**Kernel** function generally transforms data points to a high-dimensional feature space, which computes decision boundaries in terms of similarity measures in a high-dimensional feature space. In the modelling process to obtain the highest accuracy, the kernel will be tuned with “Linear”, “Quadratic” and “Cubic”.

|  | **Parameters Tuned** | **Misclassification Rate** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| Default | Kernel: Linear,  Penalty Value: 1 | 0.3039 | 0.3967 | 1.6244 |
| Tune 1 | Kernel: Quadratic,  Penalty Value: 2 | 0.3022 | 0.3990 | 1.6125 |
| Tune 2 | Kernel:Quadratic,  Penalty Value: 88 | 0.3016 | 0.3997 | 1.6116 |
| Variable Selection p< 0.05 | Kernel: Quadratic,  Penalty Value: 2 | 0.3015 | 0.4020 | 1.6090 |

First row in the table shows the default model without any tuning and the others are with parameters tuned.

With the purpose of kernel and penalty value stated above, **parameters were tuned one by one** to **avoid overfitting** and **find patterns** when tuning is made, then select the most accurate model. During tuning, the **cubic kernel was not able to execute and generate** the model. Kernels tuned to quadratic return a better result than linear, as shown above the remaining tuning is quadratic. Tuning of penalty value was done incrementally, with the result observed.

The best model for the support vector machine was with the **3 insignificant variables excluded**, with **Kernel tuned to Quadratic** and **Penalty Value of 2**, it returned a best result of **69.85% accuracy** without overfitting. Overfitting was checked by comparing misclassification rate value of training dataset with validation dataset. When the differences between them are low, it means that model is not overfitted, whereas if more than 10% differences it would be considered as overfitted. Training data misclassification rate found was 0.2991, as compared to the validation misclassification rate is 0.3015, which is **0.0024 different** only.

In the best model [relative importance plot](#gwnd3osulbq3), high blood pressure is the most important variable in identifying diabetes, followed by cholesterol, age and BMI.

The [ROC curve](#7qta2zo0bq59), when the curve is **closer to the top left** which means the specificity and sensitivity is 100%, the closer it is the **better the performance**. Additionally, performance can be observed by looking at the **area under the curve (AUC)**. In the Image it shows that ROC training data is performing better than the validation as it is closer to the top left and larger AUC.

The [Misclassification plot](#mbi8rm5xx6d7) shows true positive rates of people with diabetes are **75.41% in validation with 2971 records** at the default prediction cutoff value of 0.5. It performs better and more accurately in identifying diabetic patients than those without.

### 

### Decision Tree

*This section is documented by Zhang Xiang*

In the decision tree hyperparameters that need to be tuned were branches, level max, leaf size and predictor bin to achieve the best model.

The number of **branches** indicates how many splits will occur in a decision tree node, where a choice must be made. Each branch represents one of the possible alternatives or courses of action available at that point.

**Maximum level** refers to the maximum depth of a decision tree. The deeper the tree, the more splits it has, and it captures more information about the data. With maximum levels going too deep it might cause the overfitting towards the model.

**Leaf size** is the number of Examples in its subset. Each tree generated in a way that every leaf has at least the minimal leaf size number of examples.

**Predictor bin** specifies the number of bins used to categorise a predictor that is a measure variable. Binning is basically a form of quantization where mapping a set of numbers with continuous values into smaller, more manageable “bins”.

|  | **Parameters Tuned** | **Misclassification Rate** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| Default | Branches: 2,  Level max: 6,  Predictor Bin: 50 | 0.3082 | 0.3835 | 1.5220 |
| Tune 1 | Branches: 2,  Level max: 9,  Predictor Bin: 50 | 0.3034 | 0.3957 | 1.5751 |
| Tune 2 | Branches: 3,  Level max: 13,  Predictor Bin: 82 | 0.3070 | 0.3860 | 1.5431 |
| Variable Selection p< 0.05  Tune 1 | Branches: 2,  Level max: 13,  Predictor Bin: 82 | 0.3046 | 0.3909 | 1.5269 |
| Variable Selection p< 0.05  Tune 2 | Branches: 2,  Level max: 13,  Predictor Bin: 138 | 0.3022 | 0.3959 | 1.5037 |

As shown in tune 1, with branches set to 3, its performance in accuracy drops. Definitely testing of model branches in the variable selection of p value less than 0.05 was also conducted before, branches of 2 is decided as the final parameter tuned. Adding on to that, level max and predictor bin was **tested incrementally**, **finding the best input for each parameter**.

The first row in the table is the default decision tree with the 3 insignificant variables included and next few are with tuning. The best model for the decision tree was with the **3 insignificant variables excluded**, with **2 branches**, **level max 13**, **leaf size 5** and **predictor bin 138**, resulting in **69.78% accuracy**.

As shown in the [decision tree](#9ayqcu37gofw) it was splitted into 2, as branches were assigned with 2. In decision tree [node 3 to 6](#cr7b33guj073), It shows that groups that **do not have high blood pressure** have **fewer** records of diabetic patients than those with **high blood pressure**, and it was also identified that with a **high BMI of more than 27.5** there is a **higher chance of getting diabetes**. Looking at the group with High Blood Pressure, it is obvious that in the histogram on the left with a BMI more than or equal to 28.5 have 72.17% of people with diabetes.

Next look at the [Leaf Diagram](#eqr6r6n6rzfq), which enables a quick comparison between the leaf purity. In Node 182, it contains the highest percentage of diabetic responders at 100%. Information derived shows that the majority of diabetic responders are in the **age group from 7 or 8** which represent the age range from **50-54 or 55-59** and have a past **history record of having coronary heart disease**, **no** **difficulty in walking**, **no alcohol consumption**, **no high blood pressure** and have **BMI more than 33.25**.

[Lift Chart](#xbvqtnbqwvtr) at 20% was taken into account, in the chart it shows that it has 1.5 times more positive labels than average, which means with the decision tree it can predict approximately 1.5 times better in identifying diabetic patients.

### Random Forest

*This section is documented by Rawtbhik*

A Random Forest basically **grows multiple decision trees** which are merged together for a more accurate prediction. The logic behind the Random Forest model is that multiple individual decision trees **perform much better as a group** than they do alone. When using Random Forest for classification, each tree gives a “vote.” The forest chooses the classification with the majority of the “votes.” For Random Forest, the things that will be tuned are the **number of trees, bootstrap, predictor split nodes, predictor bins, leaf size and maximum level**. I have 3 different tunings to share.

|  | **Parameters Tuned** | **Misclassification Rate** | **ROC** | **Lift** |
| --- | --- | --- | --- | --- |
| **Default** | **No.of trees** =100, **Bootstrap**=0.6  **Predictor split nodes:**5 **Predictor bins**:50  **Leaf Size:**1  **Max Level** :6 | 0.3088 | 0.3845 | 1.604 |
| **Tune 1**  **(Best Model)** | **No.of trees** =100, **Bootstrap**=0.5  **Predictor split nodes:**7  **Predictor bins**:80  **Leaf Size**:20  **Max Level** :30 | 0.2956 | 0.4114 | 1.6179 |
| **Tune 2** | **No.of trees** =100, **Bootstrap**=0.7  **Predictor split nodes:**7, **Max Level** :30,  **Leaf Size**:20  **Predictor bins**:80 | 0.2967 | 0.4099 | 1.6048 |
| **Tune 3** | **No.of trees** =100, **Bootstrap**=0.8  **Predictor split nodes:**7, **Max Level** :30,  **Leaf Size**:20  **Predictor bins**:80 | 0.2990 | 0.4053 | 1.6161 |

For the default model which has a misclassification rate of 0.3094, ROC of 0.3845 and a lift of 1.7506 at 20 percentile.

First tuning that has **100 number of trees, decreased bootstrap from 0.5 to 0.6, increased predictor node split from 5 to 7, 80 predictor bins, leaf size of 20 and maximum level being 30** would produce **misclassification rate of 0.2964, ROC at 0.4071 and lift of 1.549**.

Second tuning that has **100 numbers of trees, bootstrap from 0.7, predictor node split of 7, 80 predictor bins, leaf size of 20 and maximum level being 30** would produce **misclassification rate of 0.2967, ROC at 0.4099 and lift of 1.6048.**

Third tuning that has **100 numbers of trees, bootstrap from 0.8, predictor node split of 7, 80 predictor bins, leaf size of 20 and maximum level being 30** would produce **misclassification rate of 0.2990, ROC at 0.4053 and lift of 1.6161.**

To share more about how the tunings are done with the various considerations that took place:

* **Maximum Level is** one of the most important hyperparameters when it comes to increasing the accuracy of the model. As we increase the depth of the tree, the model accuracy increases.
* **Bootstrap** actually randomly performs row sampling and feature sampling from the dataset to form sample datasets for every model. To achieve an accuracy that increases, it is made possible by using bootstrap but only upto a certain limit and then it will start to gradually decrease the performance of the model as overfitting takes place.
* If we have a very low value of the **predictor split node** then, in this case, our tree will continue to grow and start overfitting. By increasing the value of the predictor split node, it would decrease the total number of splits thus limiting the number of parameters in the model which results in reducing the occurrence of overfitting in the model. However, the value should not be kept very large that a number of parameters drop extremely causing the model to underfit. Therefore, value 5 was assigned to avoid overfitting and underfitting.
* **Leaf Size:** The size of a leaf is the number of Examples in its subset. The trees of the random forest are generated in such a way that every leaf has at least the minimal leaf size number of Examples.
* **Predictor Bin:** Specifies the number of bins for the measure predictors.

I will talk more about the **best performing model for random forest** which is tuning 1.

As you can see [here](#f6h8i7aktlni), we have **high blood pressure, bmi and age group being the top 3** most important factors affecting diabetes.

The 20th percentile of the [lift chart](#ottx3aedmnny), tells us that the random forest model has the ability to predict whether a person is likely to come down with diabetes or not by **1.5896 times better than the model that has not been trained.**

By viewing the [ROC](#k7m92ina8rcm) curve of Validation, we can visualise how well the random forest model predicts. From this screen shot, we can see that the **separation (KS(Youden)) is at 0.4059**. As shown here, training performs better than the validation data.

### Model Comparison

*This section is documented by Zhang Xiang and Rawtbhik*

Now when comparing our 4 models, we can see that the logistic regression model is the best performing one with an **accuracy of 70.77%** With the help of the validation’s ROC, we are able to see that the **logistic regression is the outermost curve**, indicating that it has the **highest specificity and sensitivity** compared to the rest. In common, we found that the [top 4 factors](#kix.xfoecgcbb9rm) that affect a person having diabetes or not are **high blood pressure, age, bmi and high cholesterol**.

# 

# **Conclusion**

To conclude, when comparing all our models, which include diabetes and stroke, we find that High blood pressure, BMI and Age group are the main factors that cause common health problems.

Hence we recommend these factors to patients so that they can take note of these factors and will not fall victim to getting common health illnesses. Additionally, doctors may also use these factors as a guide to advise patients and detect problems earlier.

The challenge we went through as a team is finding the appropriate and suitable dataset to meet our business objective. We had a hard time going through feature selection to make sure that the correct variables are chosen. The interpretation, finding of reasons and commonality behind our models were challenging too. However, despite that, our team managed to meet our previously stated objective.

## Evaluating Objective

Previously, we stated 3 main objectives, and we now evaluate our models to see if it meets our objective as follows

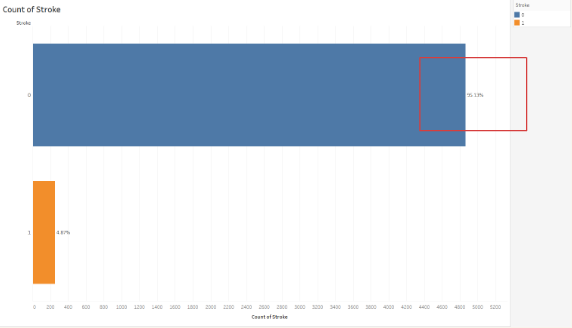
1. To train at least one model of more than 80% accuracy.
   1. Our Decision Tree, Support Vector Machine and Neural Network models for stroke all met the target of >80% accuracy.
2. Pinpoint factors associated with those health problems.
   1. We have also identified the top 3 factors mentioned above, namely high blood pressure, BMI and Age.
3. Help predict health problems more accurately.
   1. Our team believes that as a whole, we have helped predict health problems more accurately, as all of our models reach at least 69% accuracy, with the ability to pinpoint the main factors associated with the target health problems.

Therefore, we would conclude that our team has met the overall objective we have set for this project and recommend the above-mentioned factors to both patients and doctors for early prevention and detection.

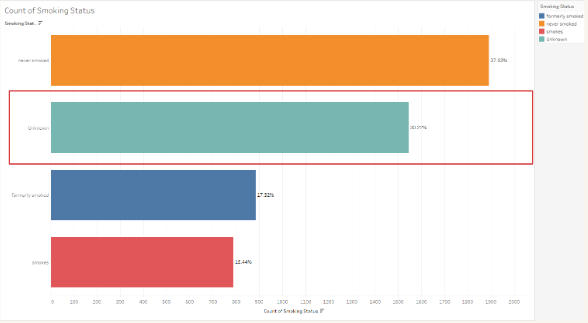
# **Appendix**

## Stroke

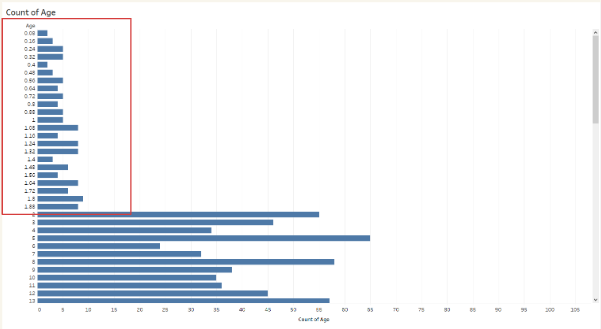
### Data Understanding & Preparation



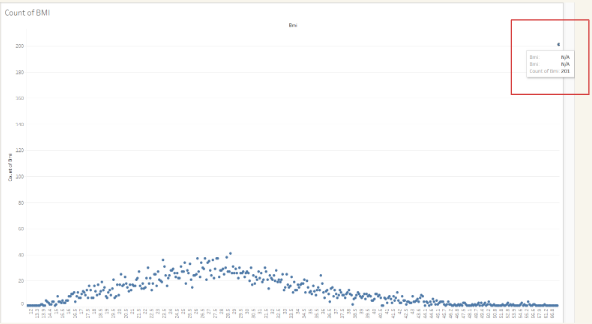
Stroke Data Understanding, Unbalanceness



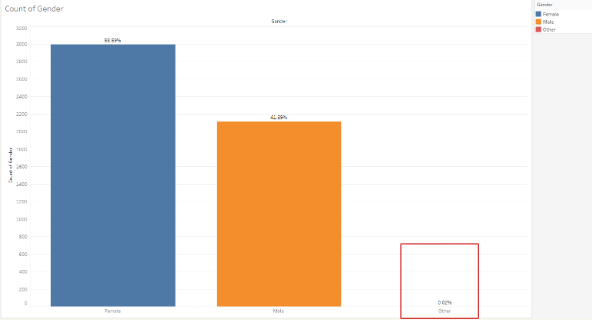
Stroke Data Understanding, Smoking Unknown



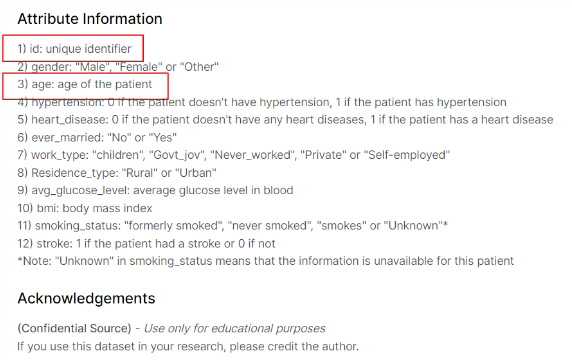
Stroke Data Understanding, Age Decimal



Stroke Data Understanding, BMI Anomaly

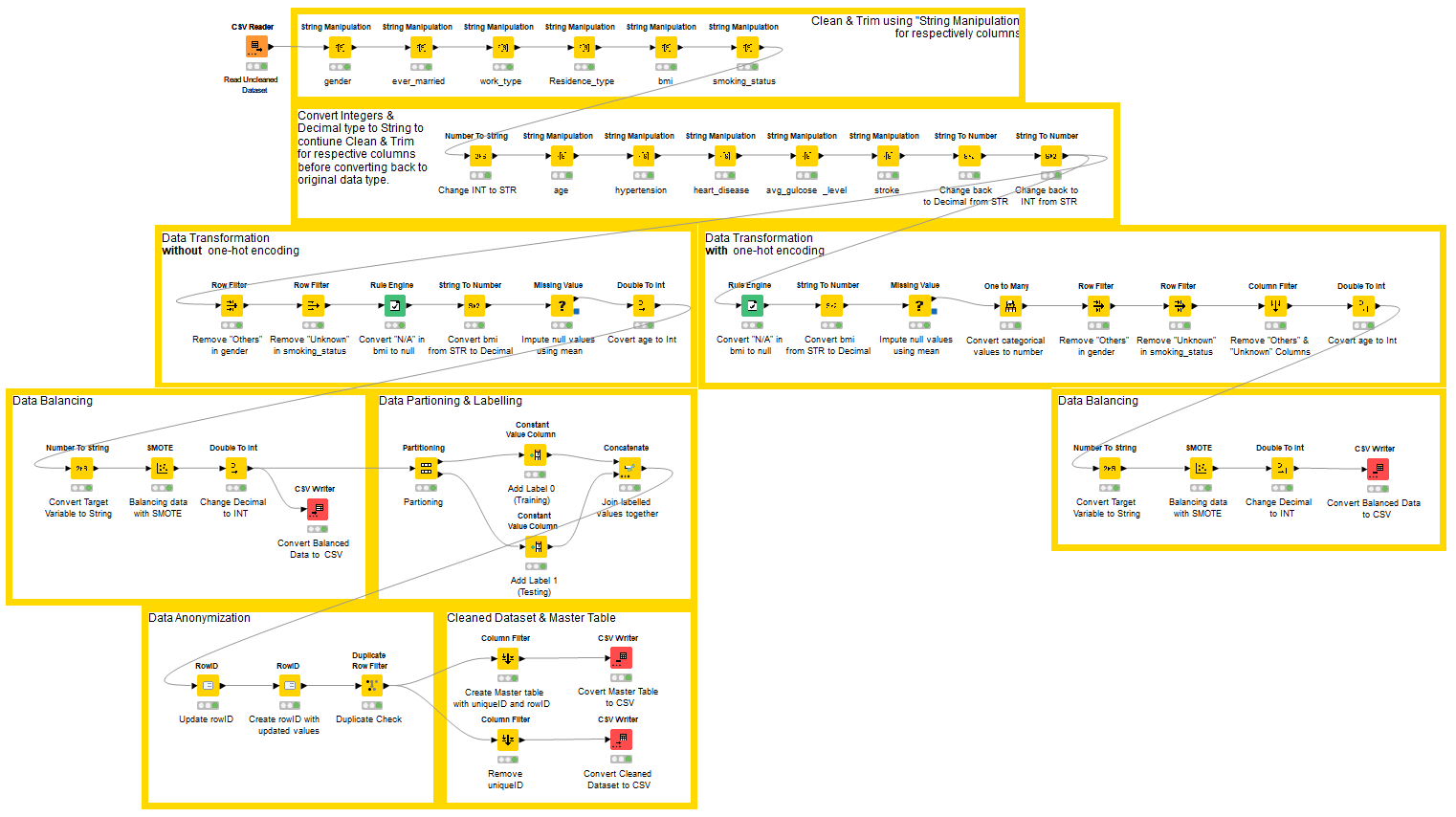


Stroke Data Understanding, Gender “Others”

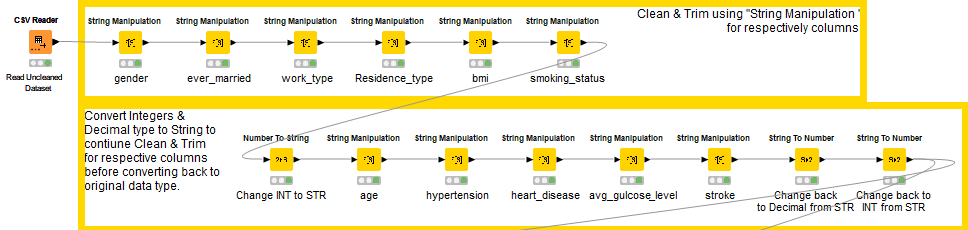


Stroke Data Understanding, Data Privacy

**Data Cleaning**

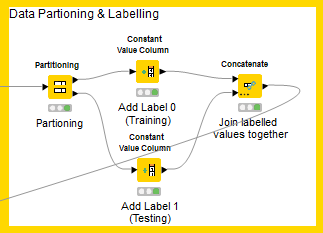


Stroke Data Cleaning, Overview

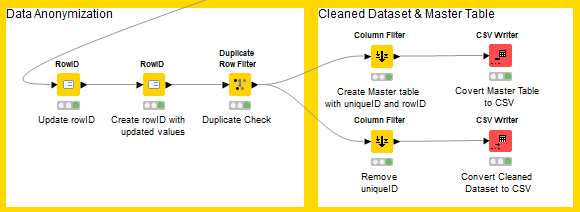
Stroke Data Cleaning, Main



Stroke Data Cleaning, Balancing



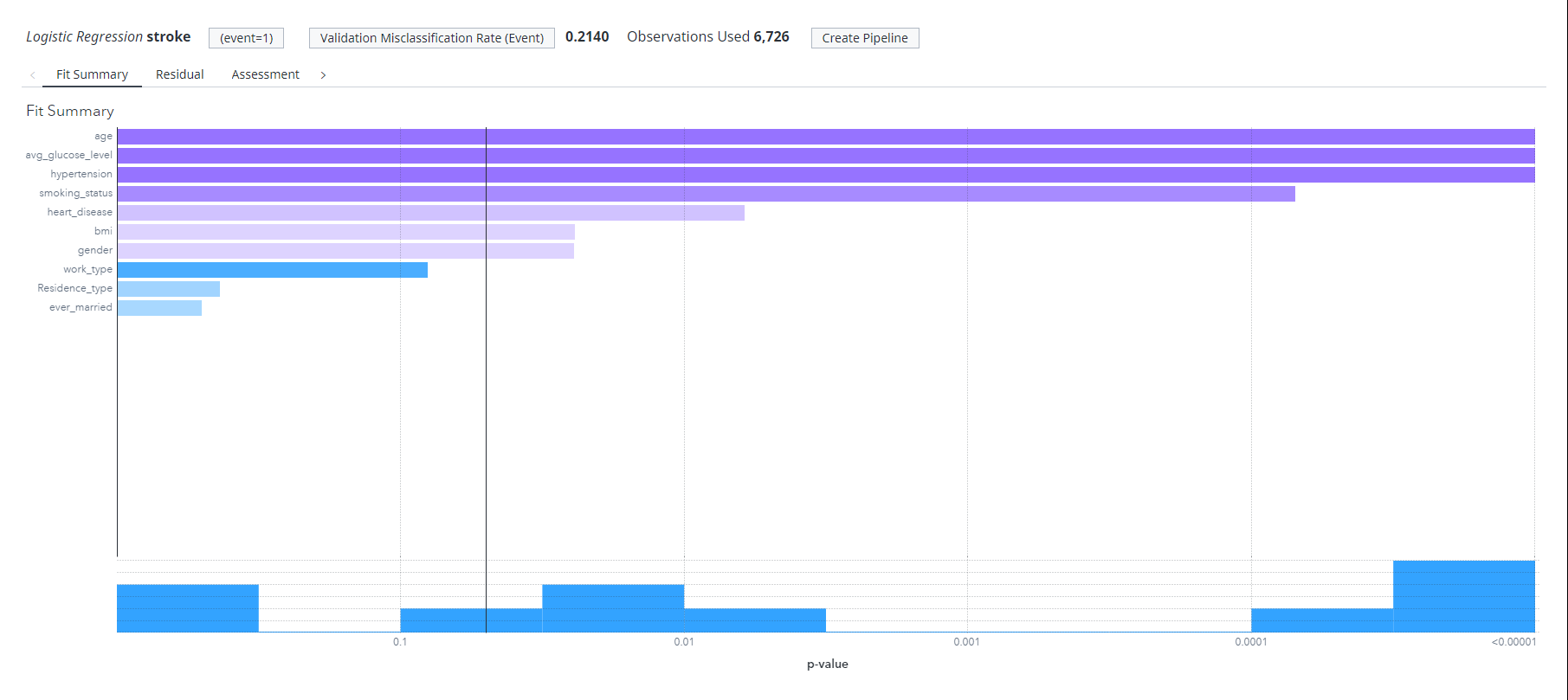
Stroke Data Cleaning, Partitioning



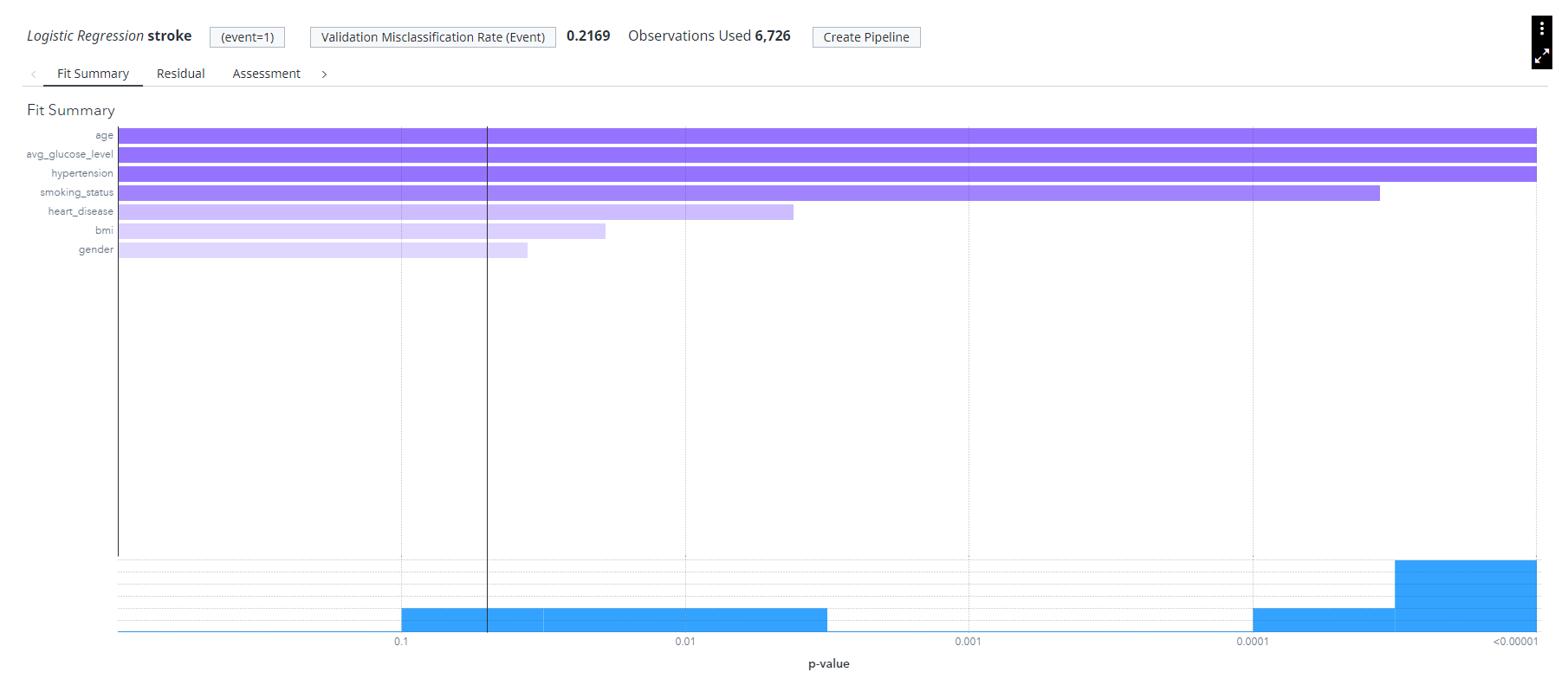
Stroke Data Cleaning, Anonymisation

### **Modelling**

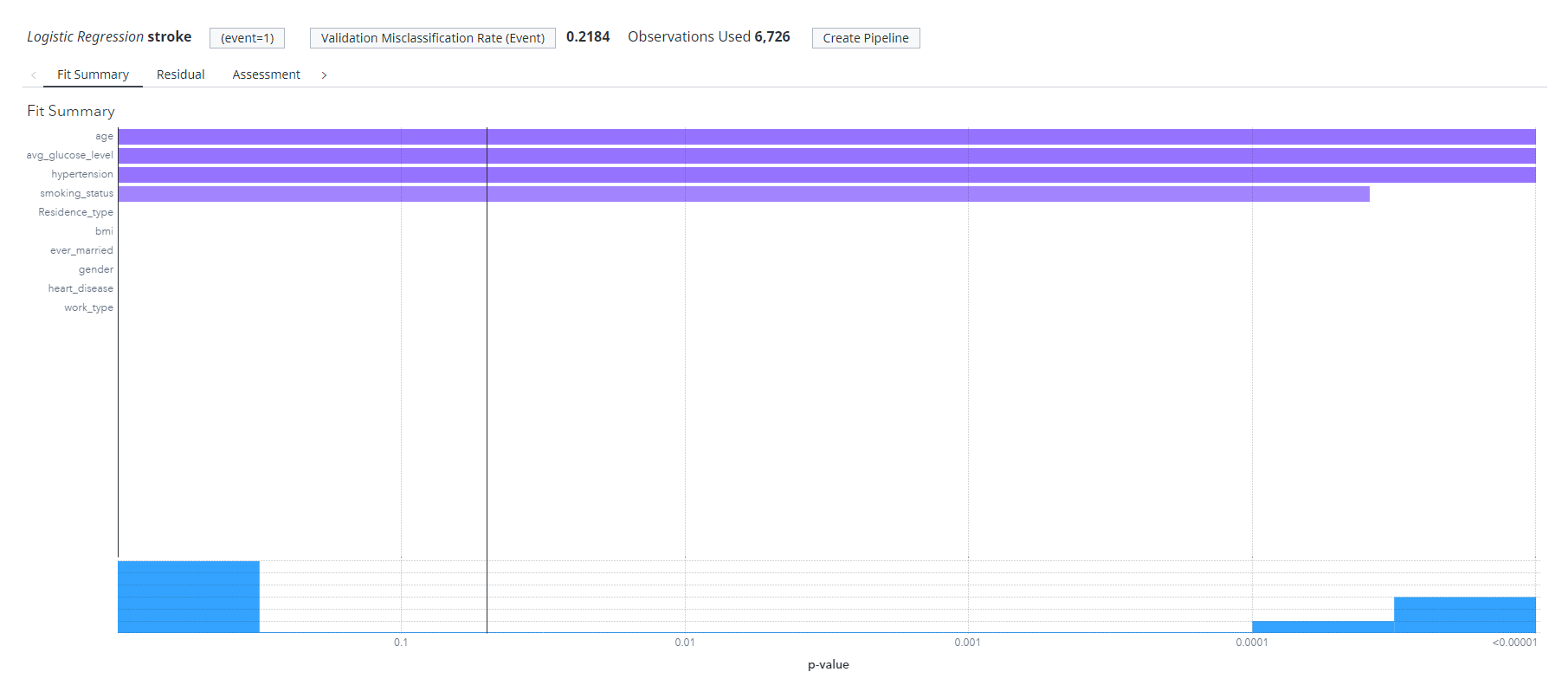
**Linear Regression**



*Logistic Regression - Default Model Variable Importance*



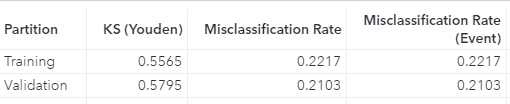
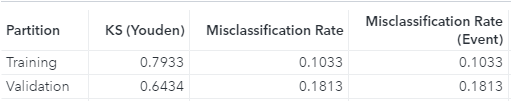
*Logistic Regression - Significant Variables Model Variable Importance*



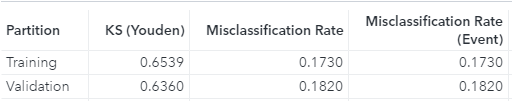
*Logistic Regression - Stepwise Model Variable Importance*

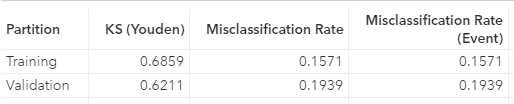
**Decision Tree**

Comparison of four different model training and validation misclassification rate:

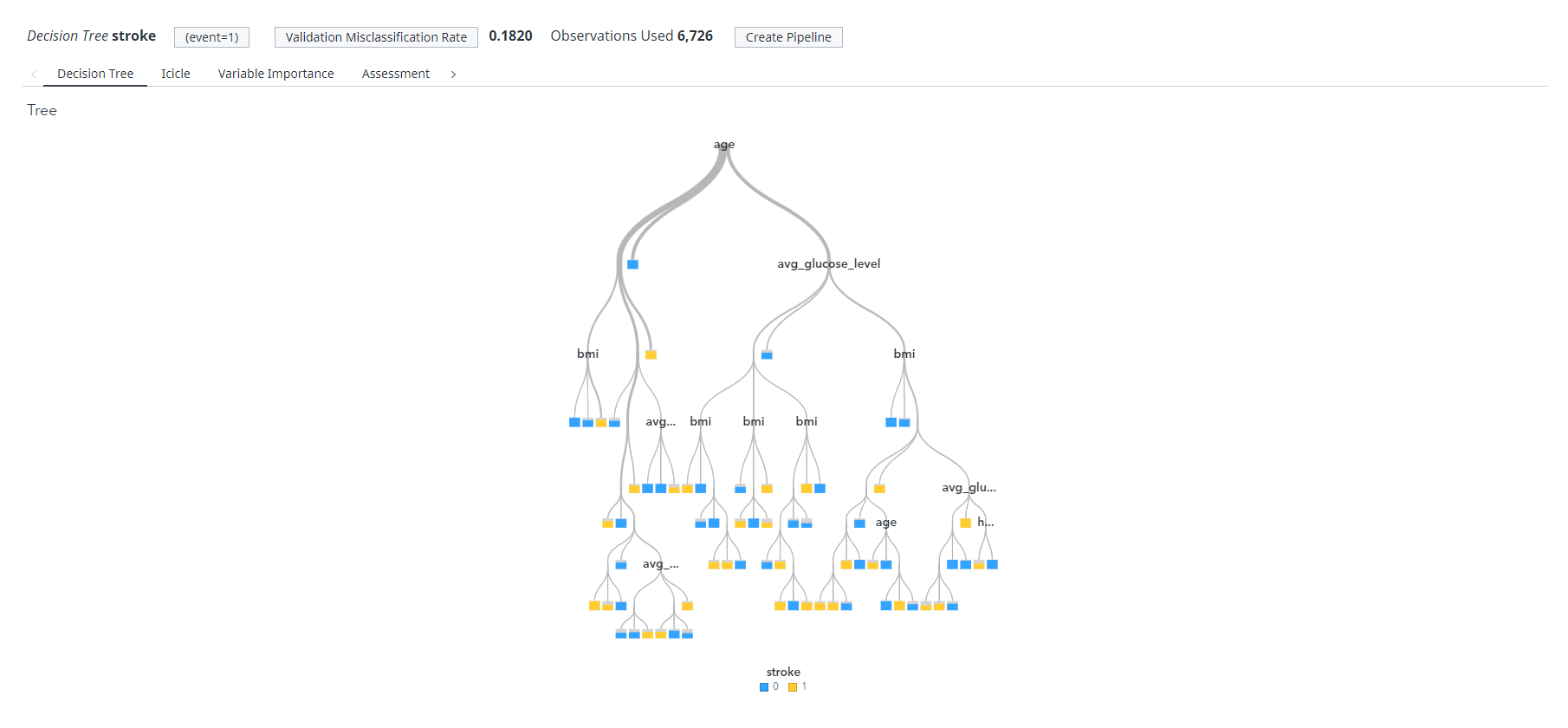
*Decision Tree - Default Misclassification Rate Decision Tree - Default Tuned Misclassification Rate*



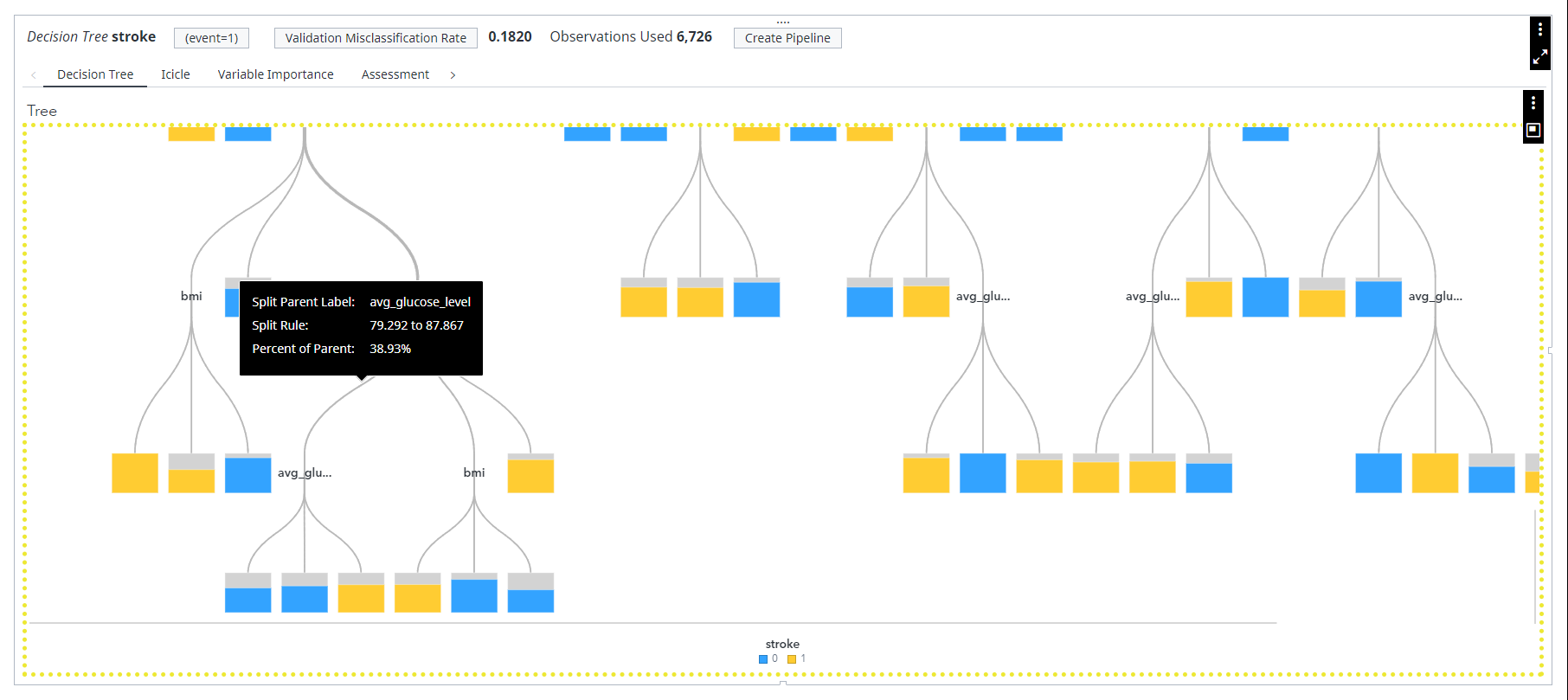
*Decision Tree - Significant Variable Tuned 1 Misclassification Rate* 

*Decision Tree - Significant Variable Tuned 2 Misclassification Rate*

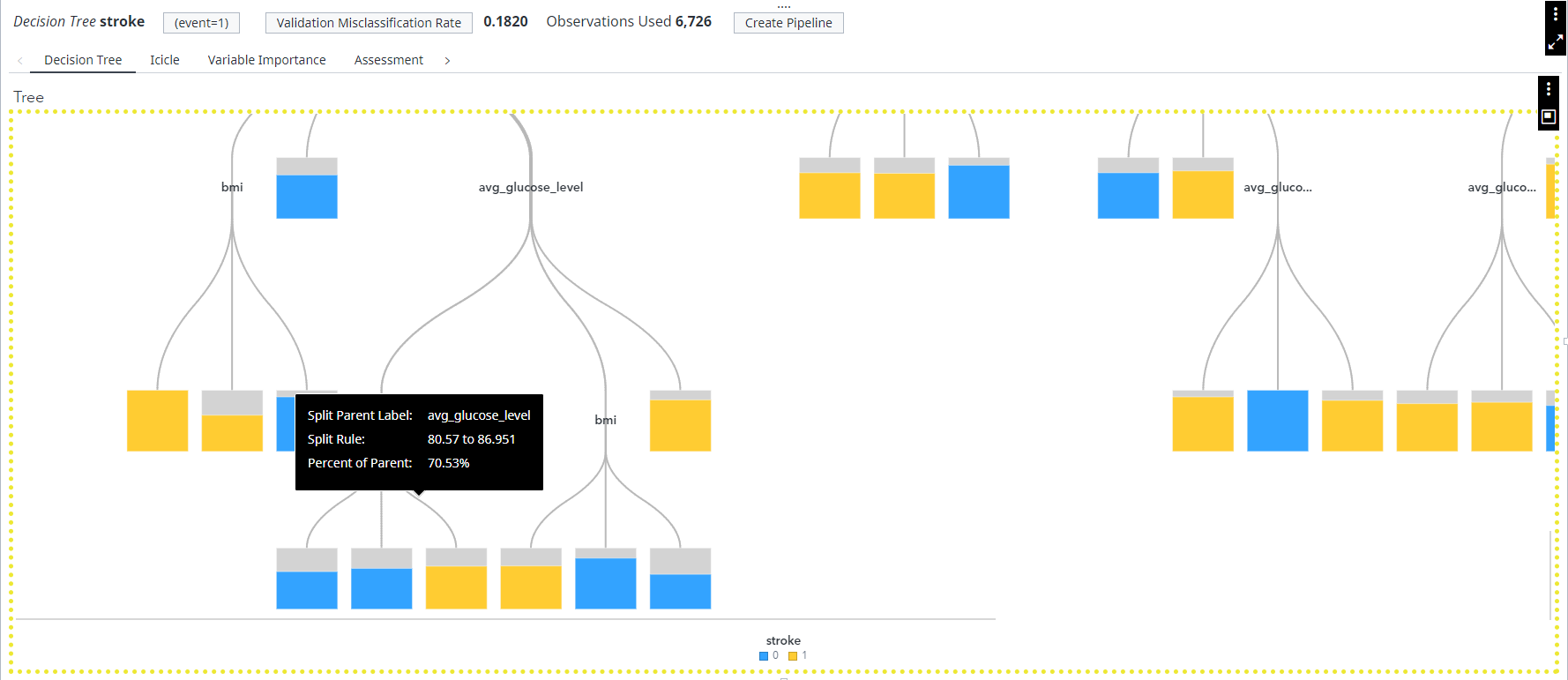
Most occurrence of rule split flow (following the thickness line):



*Decision Tree - Tree Overview*

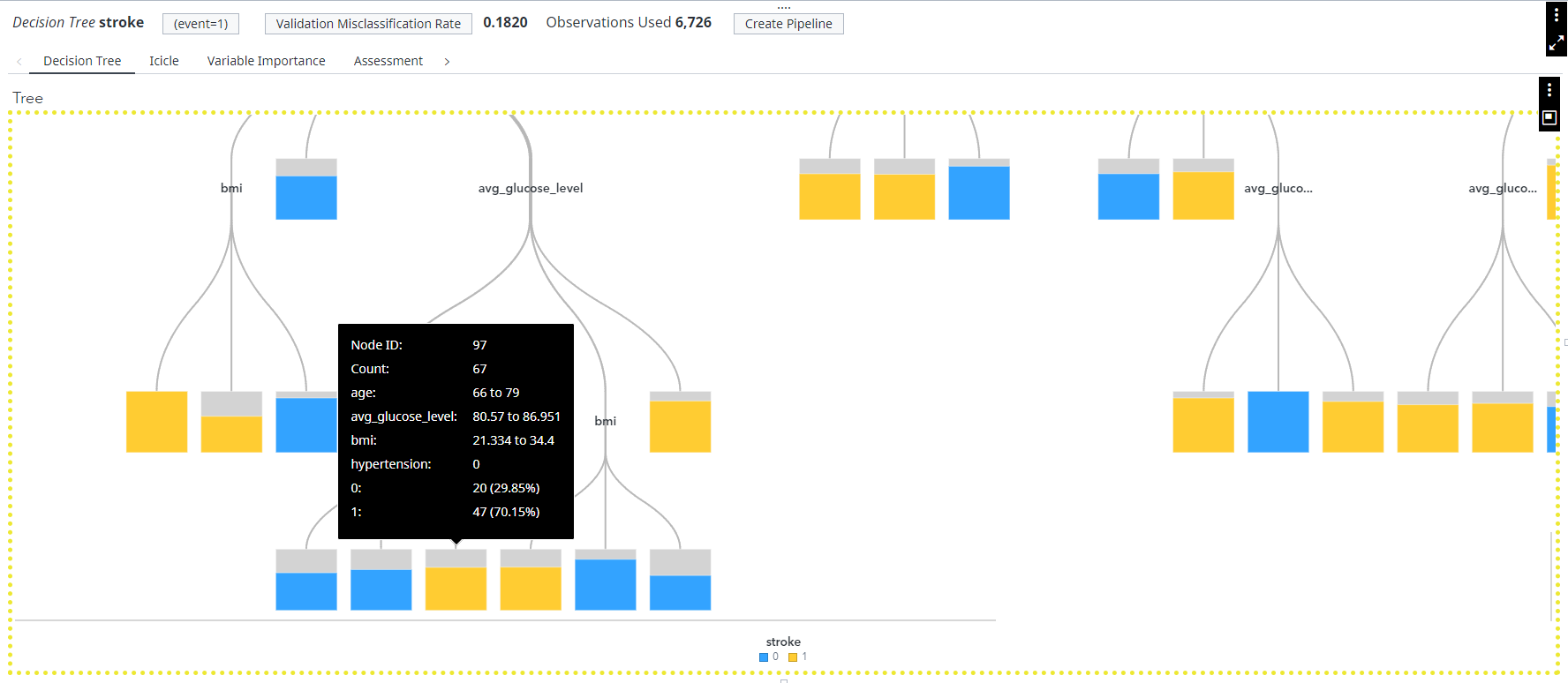


*Decision Tree - Tree (avg\_gulcose\_level node percentage of parent)*

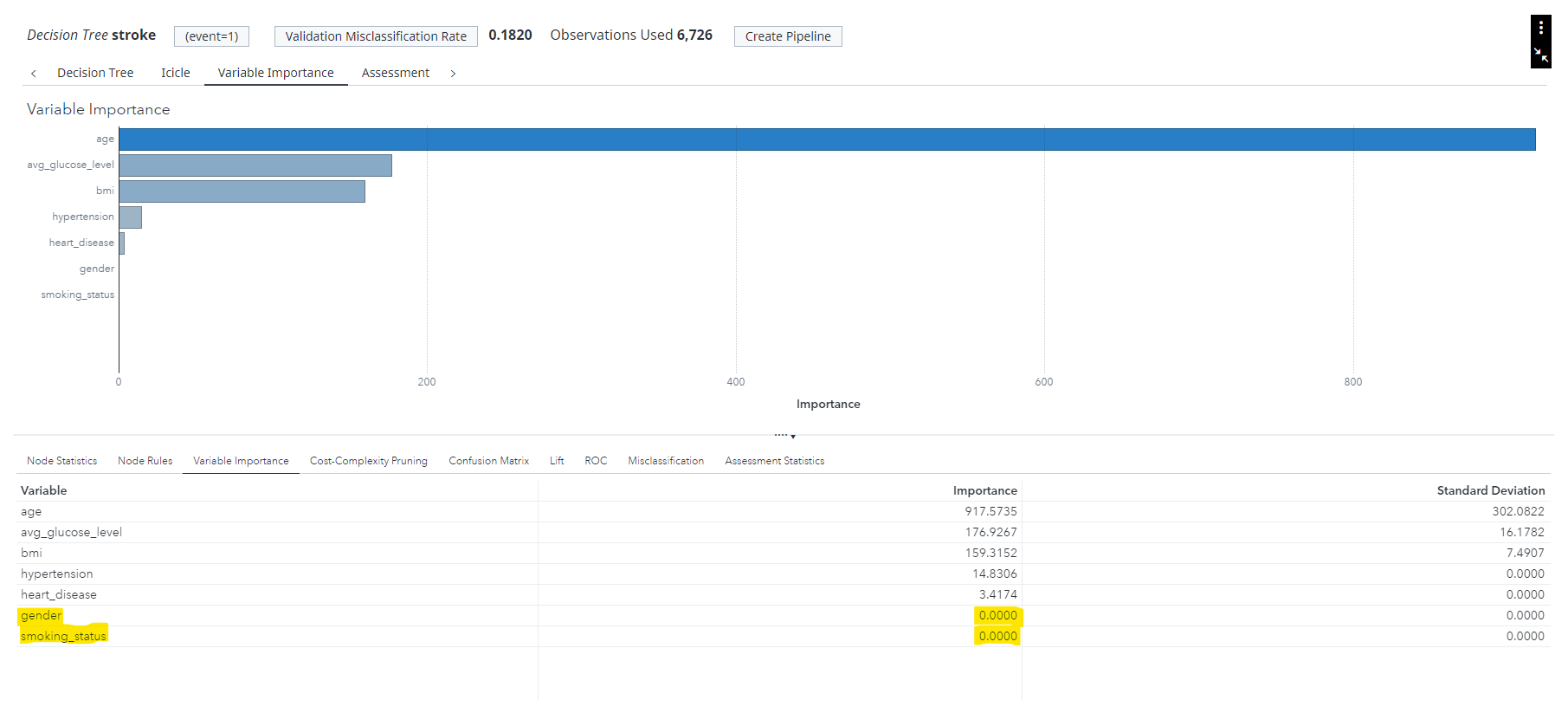


*Decision Tree - Tree (avg\_gulcose\_level node percentage of parent)*

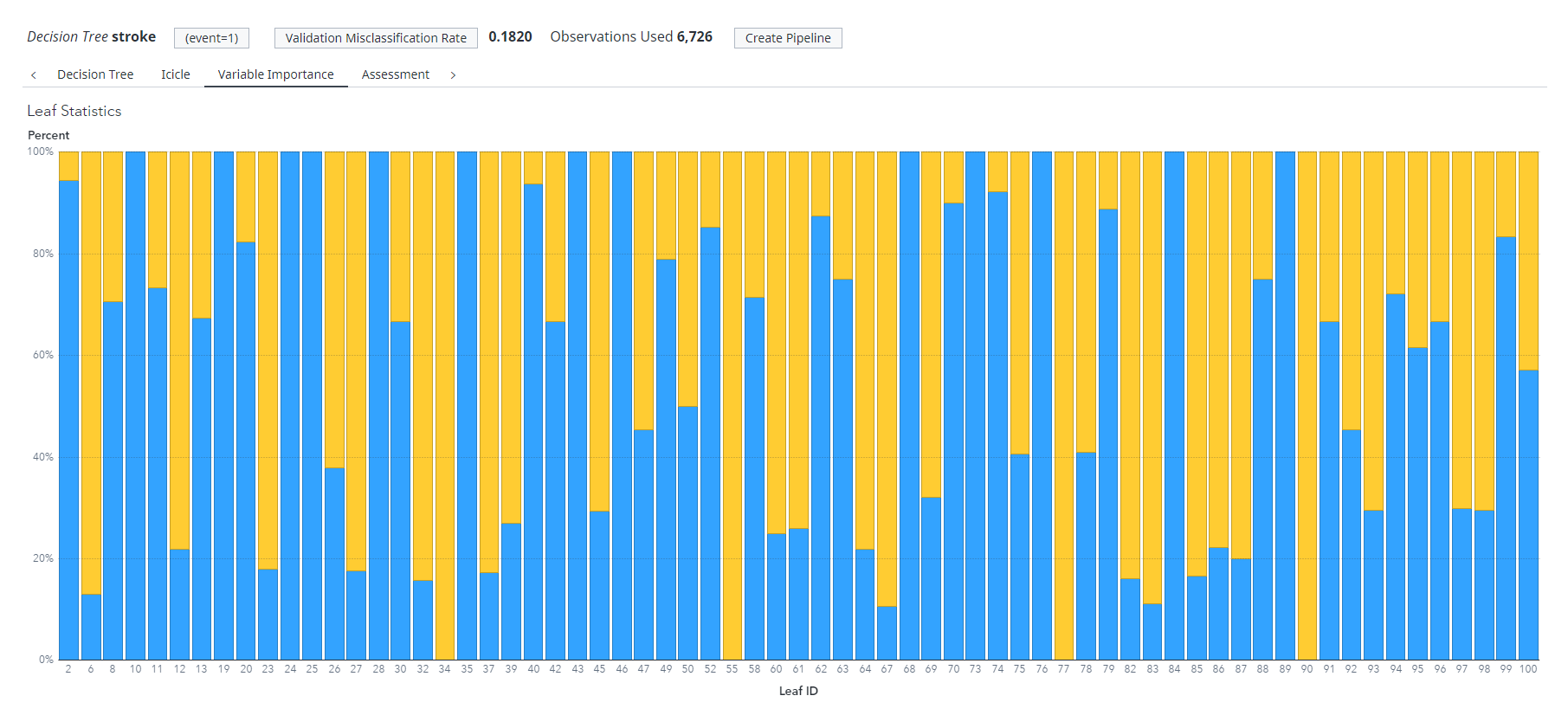
Best Model Screen Captures:



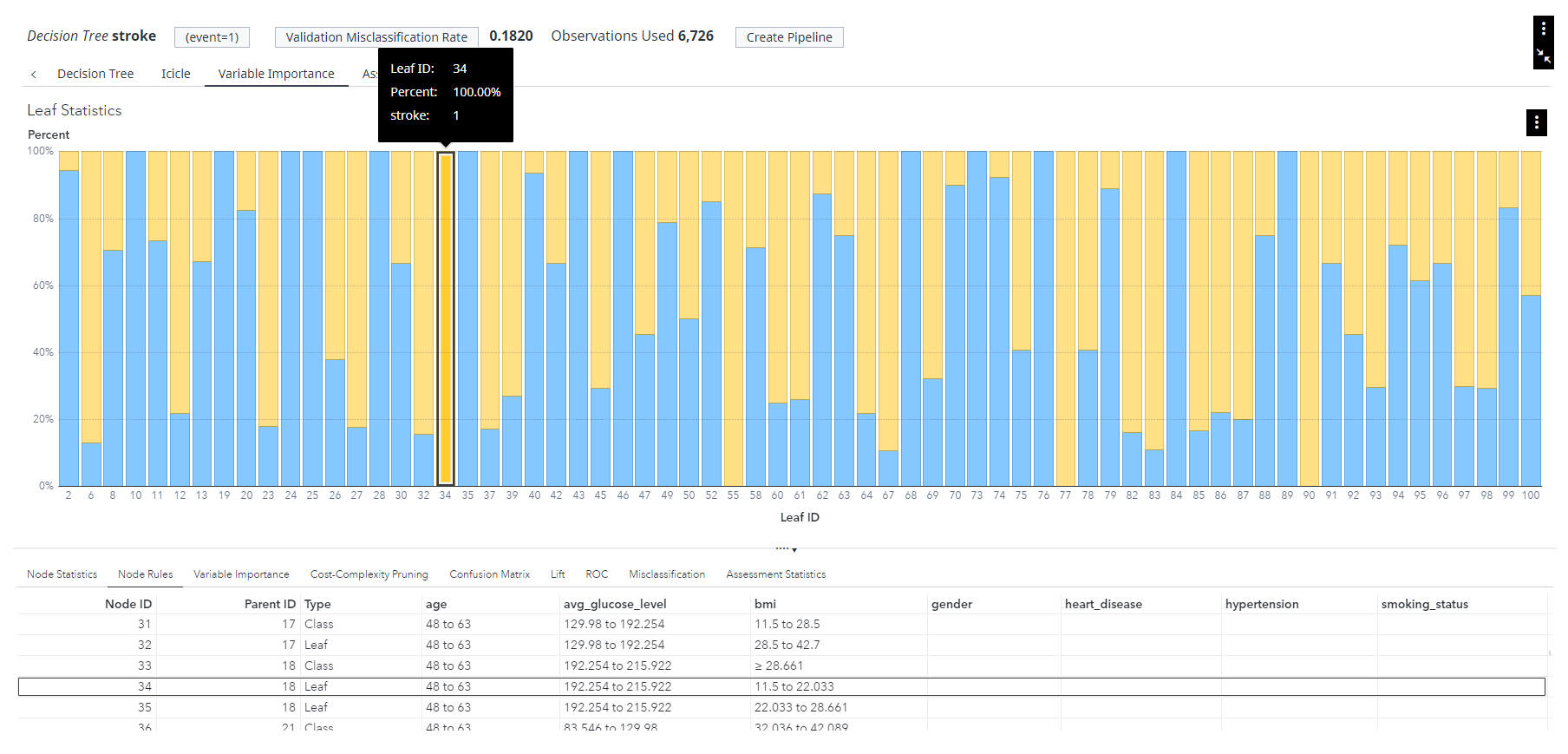
*Decision Tree - Node 97 Details*



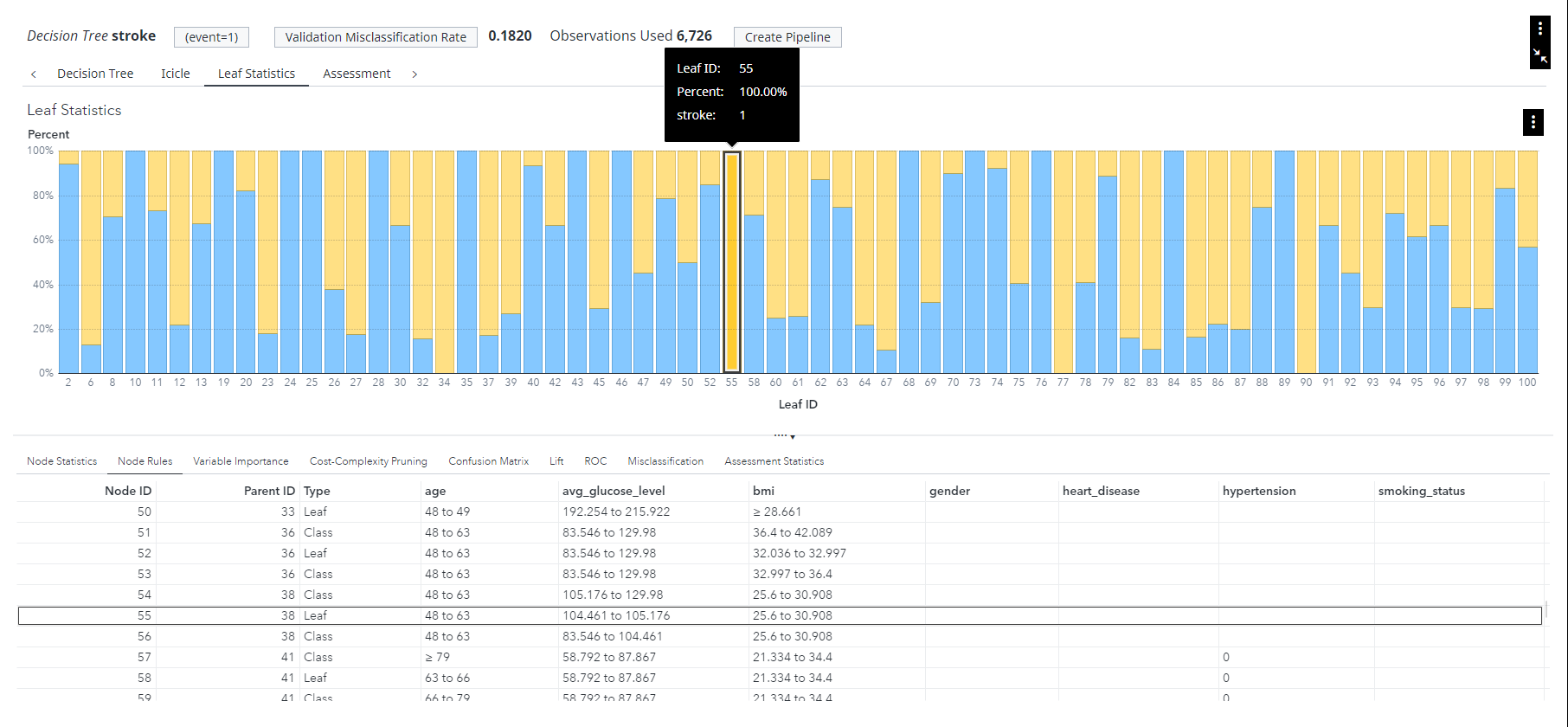
*Decision Tree - Variable Importance*



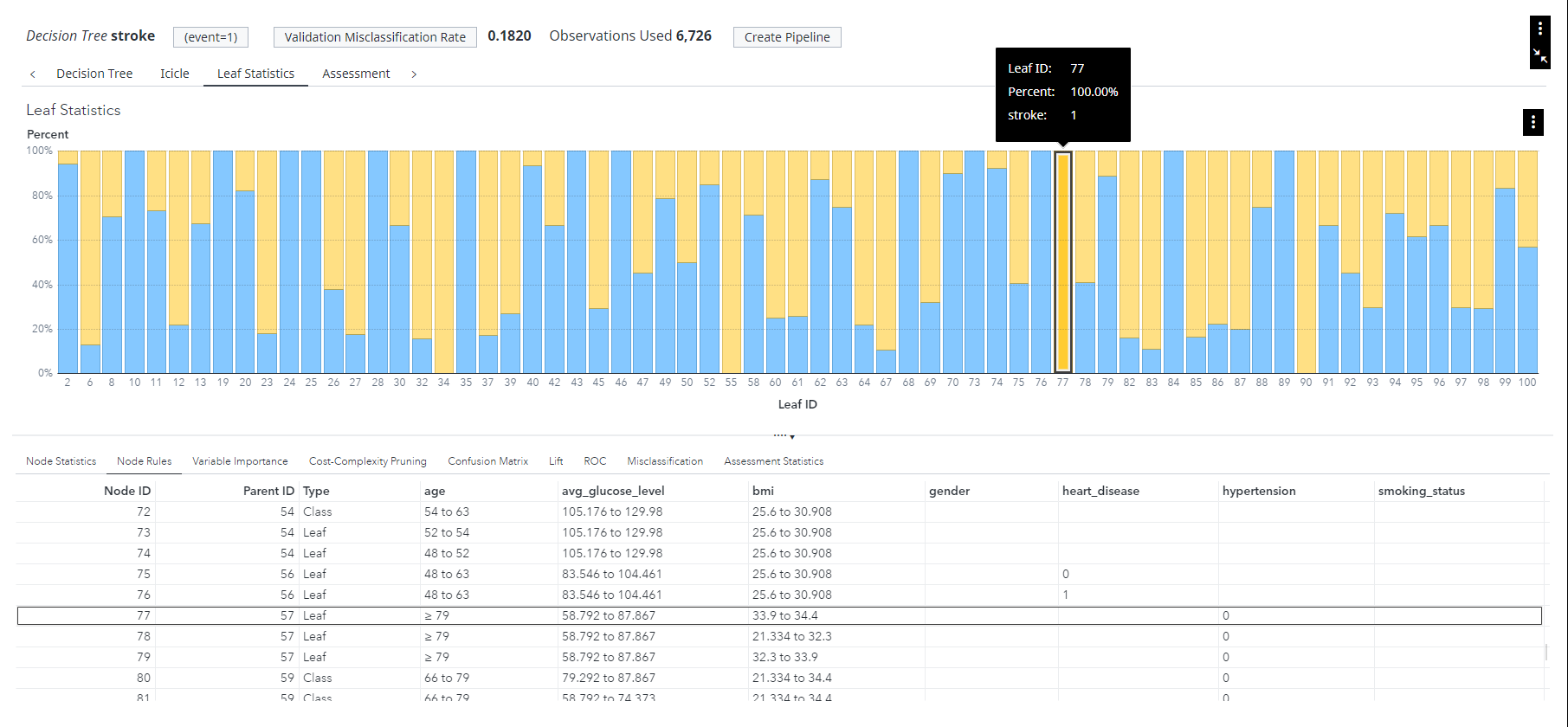
*Decision Tree - Leaf Statistic Overview*



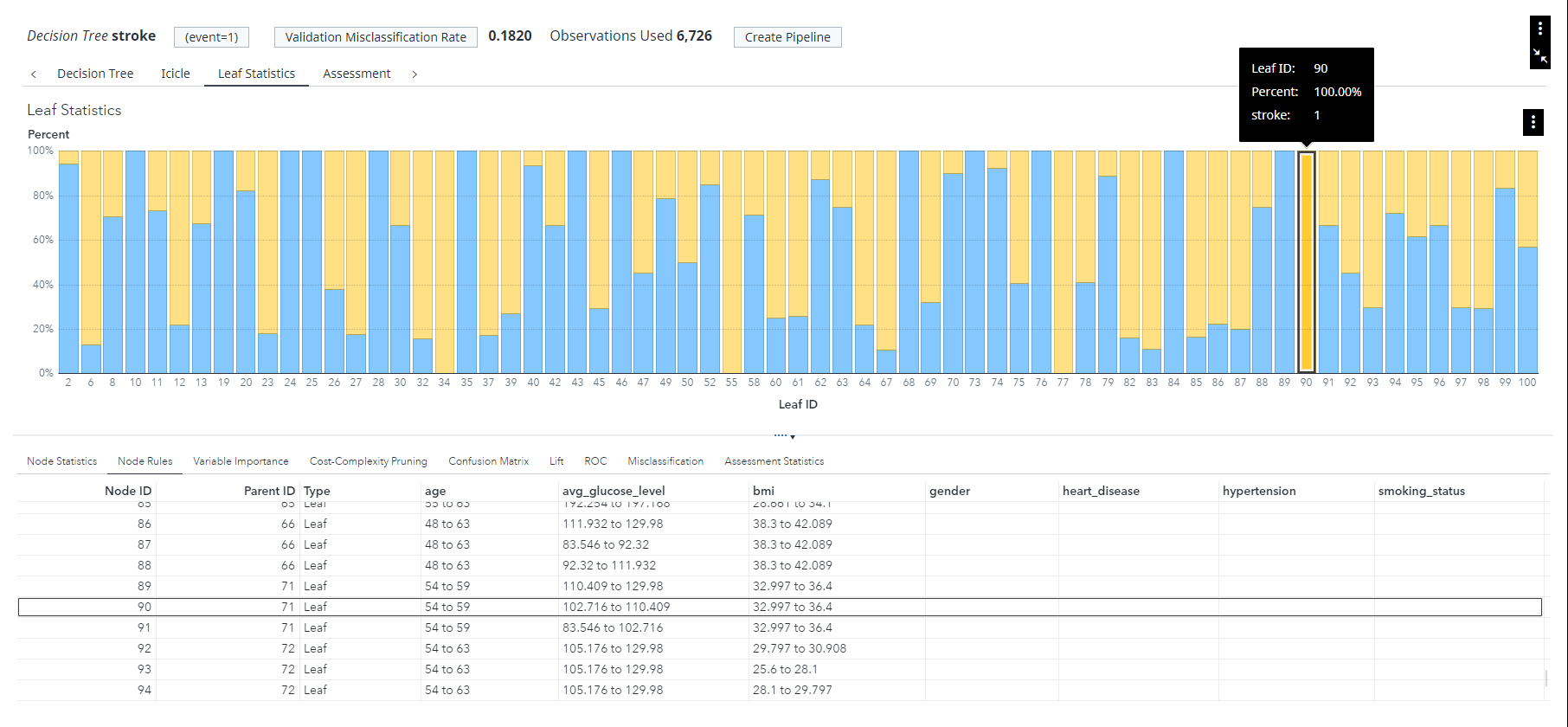
*Decision Tree - Leaf Statistic (Node 34)*



*Decision Tree - Leaf Statistic (Node 55)*



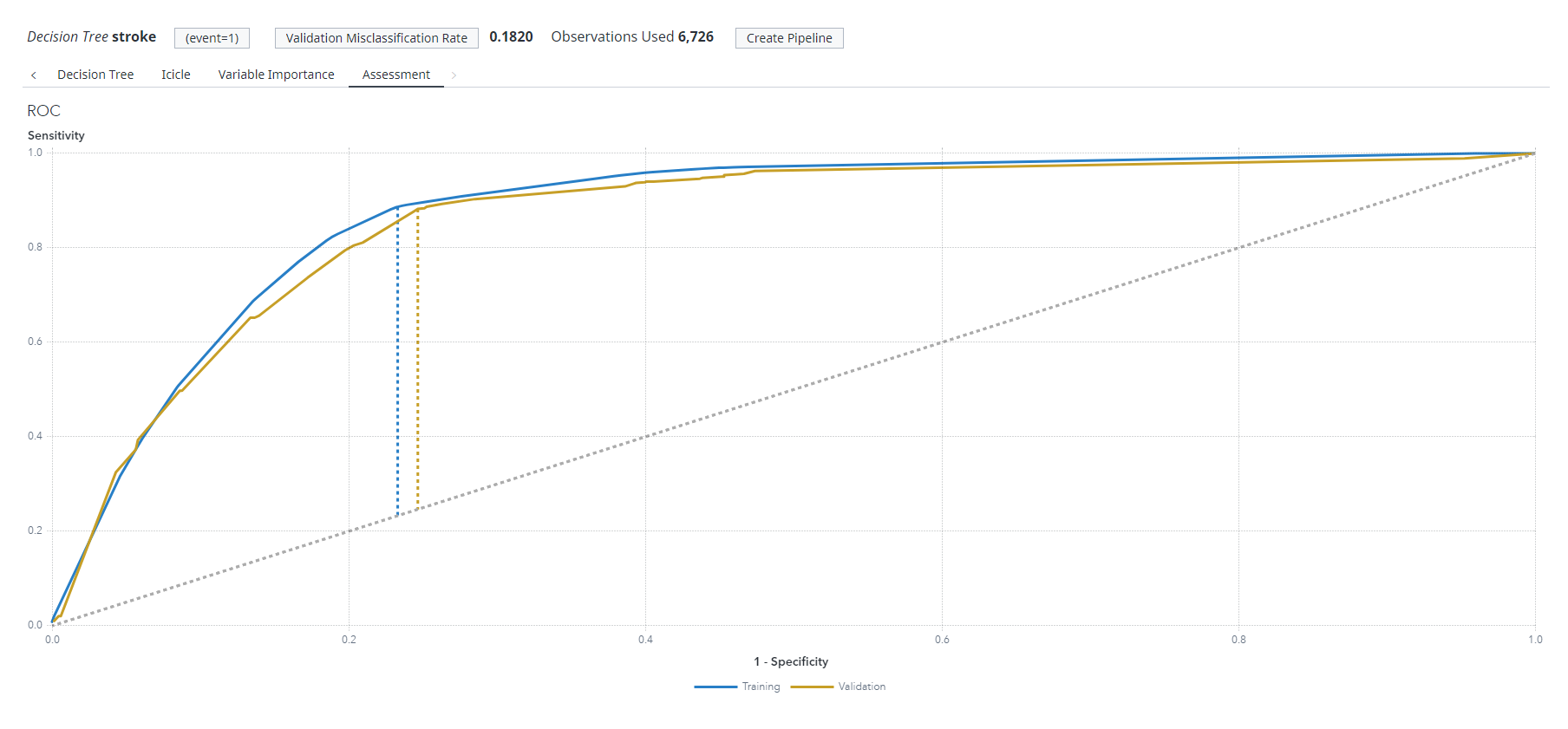
*Decision Tree - Leaf Statistic (Node 77)*



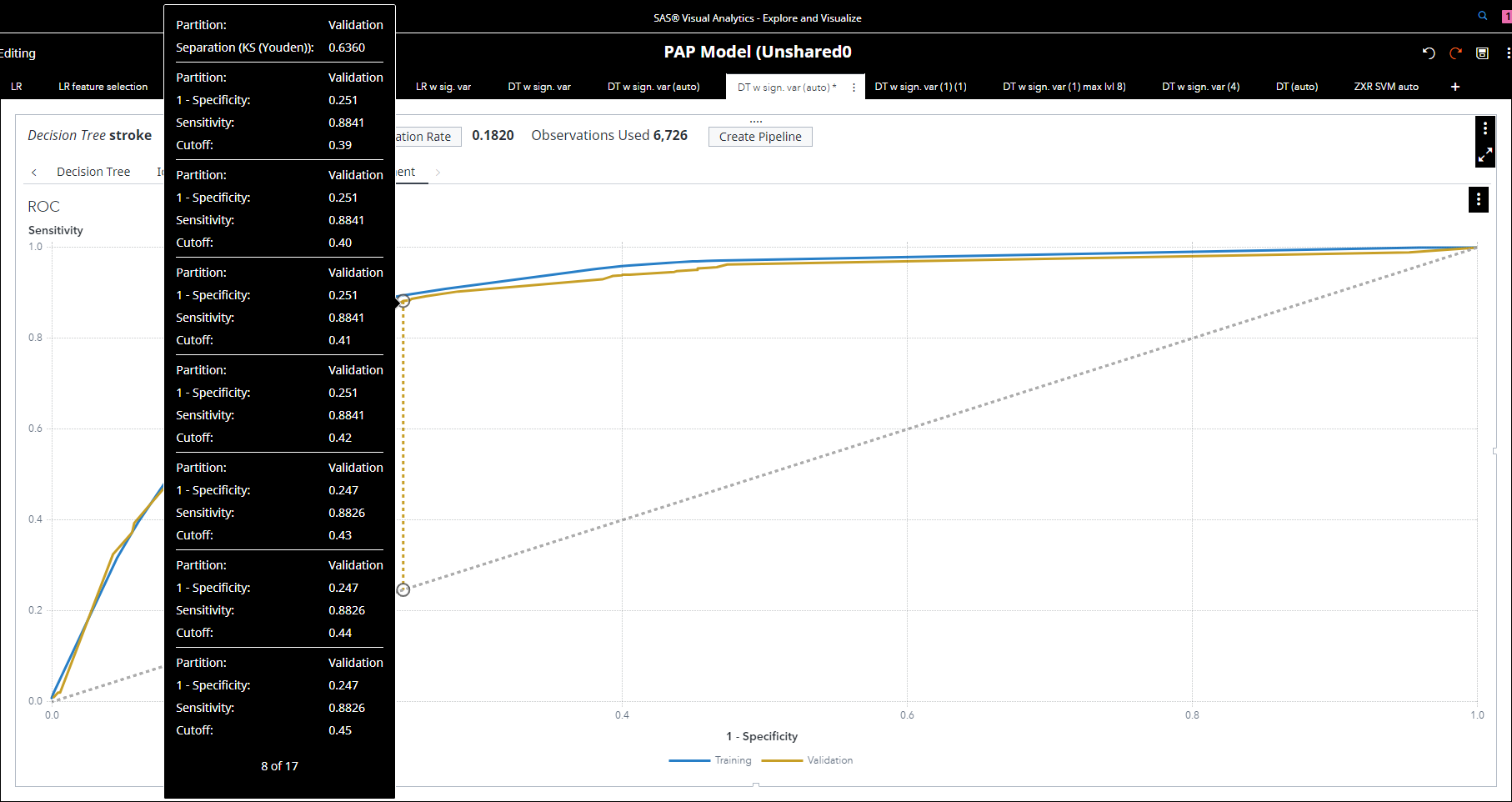
*Decision Tree - Leaf Statistic (Node 90)*



*Decision Tree - Lift*



*Decision Tree - ROC Curve Overvew*

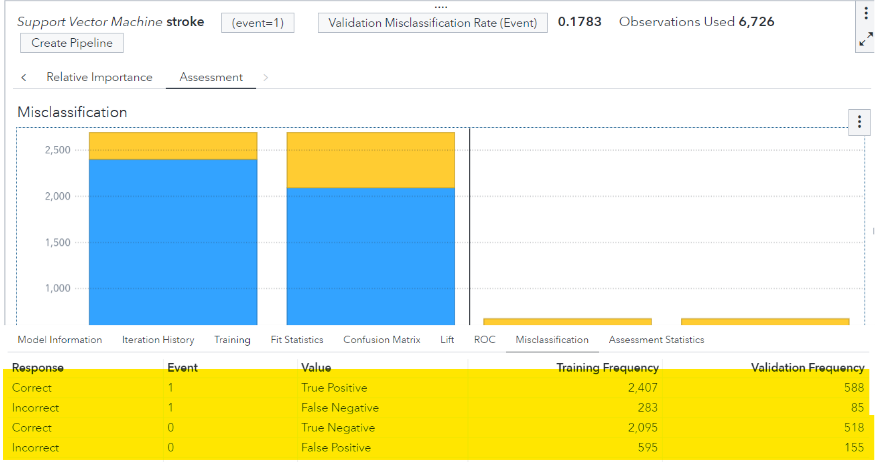


*Decision Tree - ROC Curve Details*

**Support Vector Machine**



Stroke SVM, Relative Importance Plot



Stroke SVM, Misclassification Plot

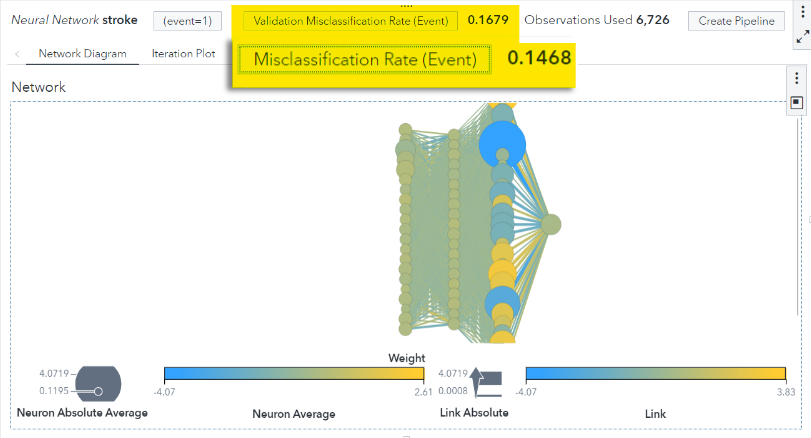


Stroke SVM, Lift Chart

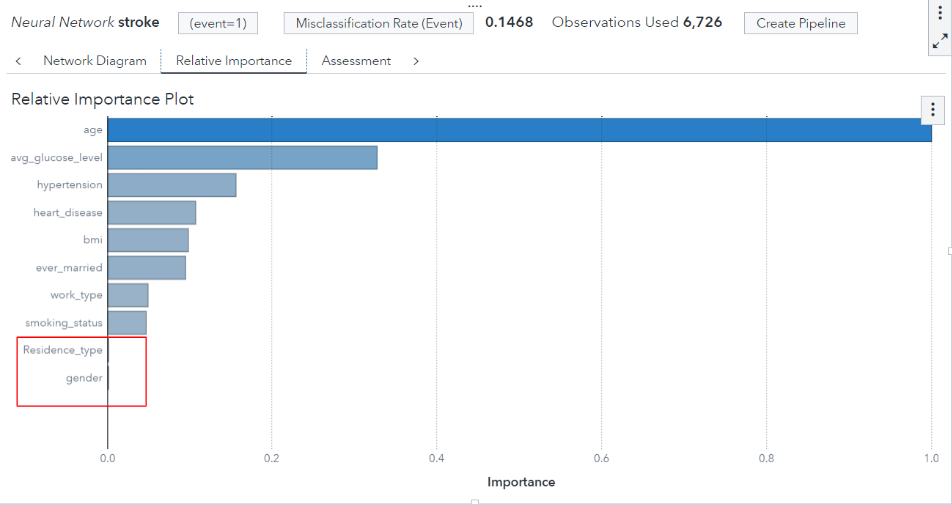


Stroke SVM, ROC Curve

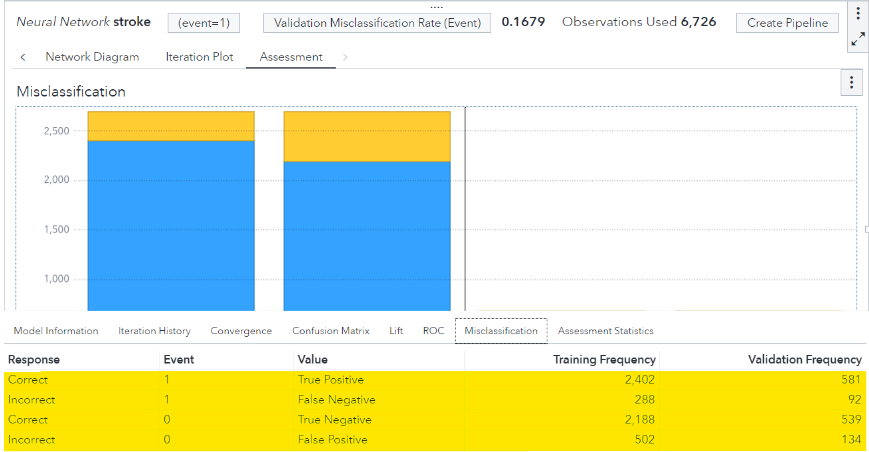
**Neural Network**



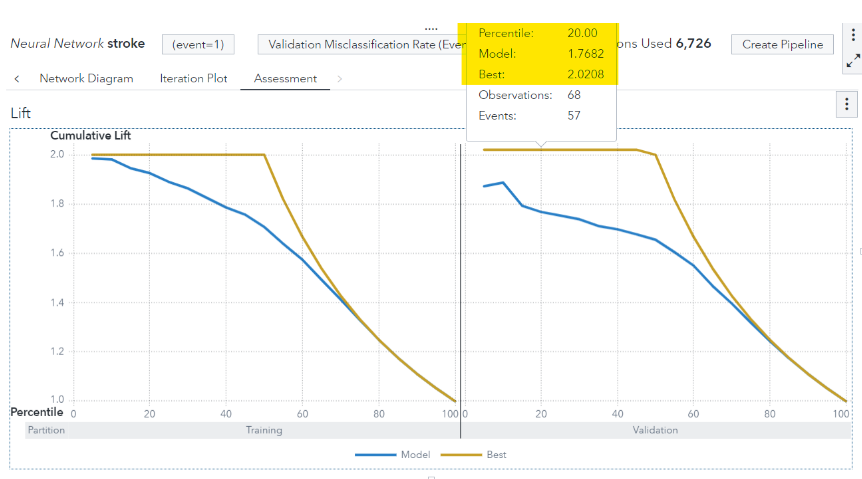
Stroke NN, Overview



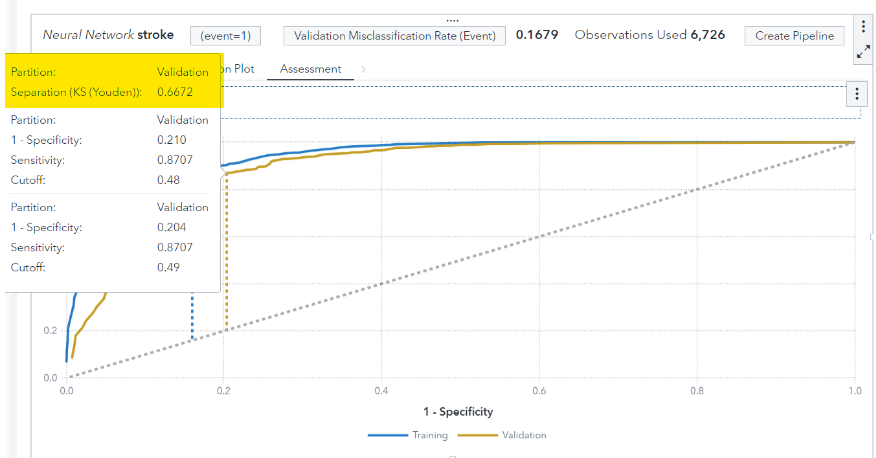
Stroke NN, Relative Importance Plot



Stroke NN, Misclassification Plot

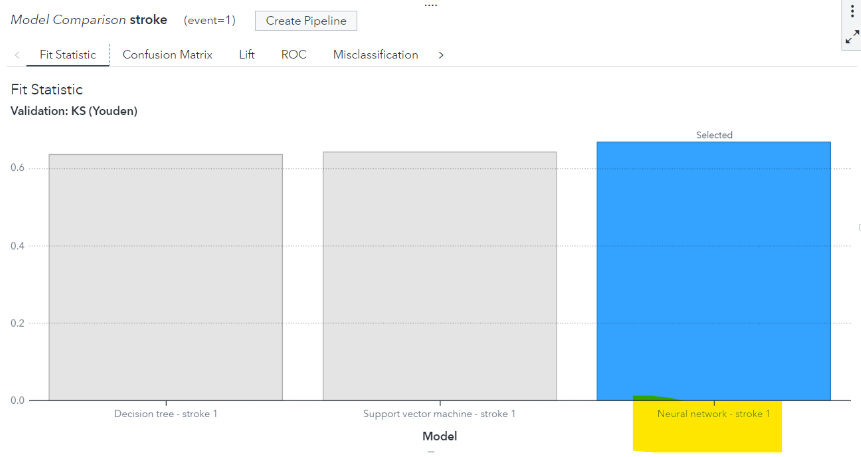


Stroke NN, Lift Chart

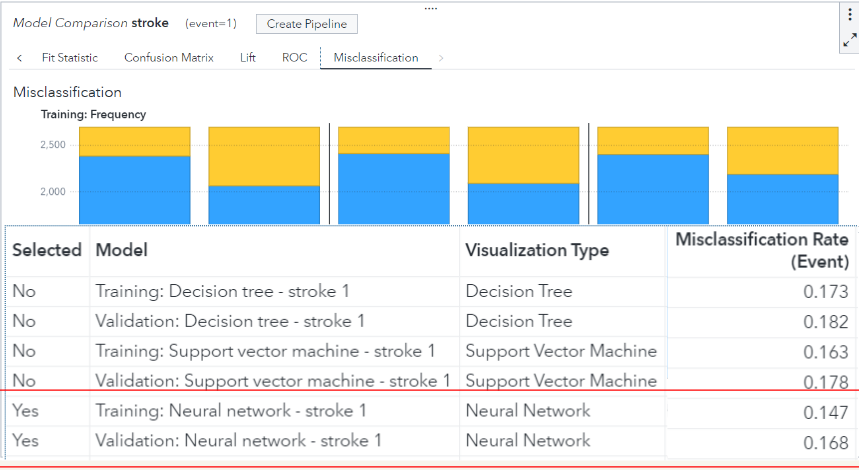


Stroke NN, ROC Curve

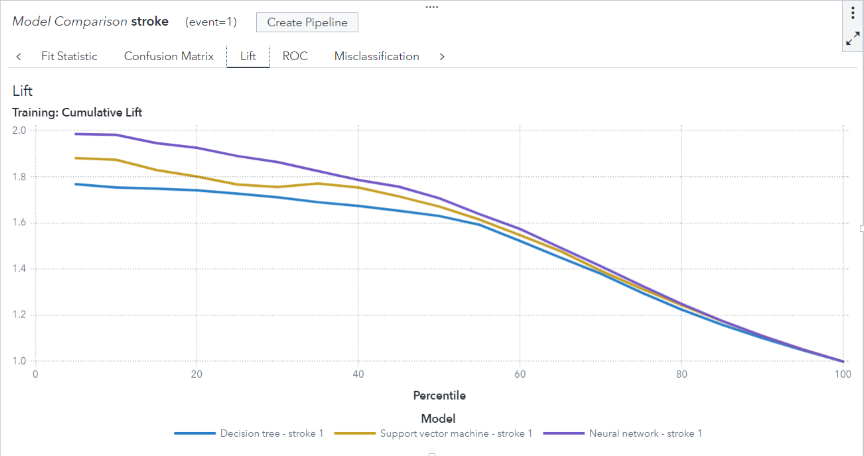
**Model Comparison**



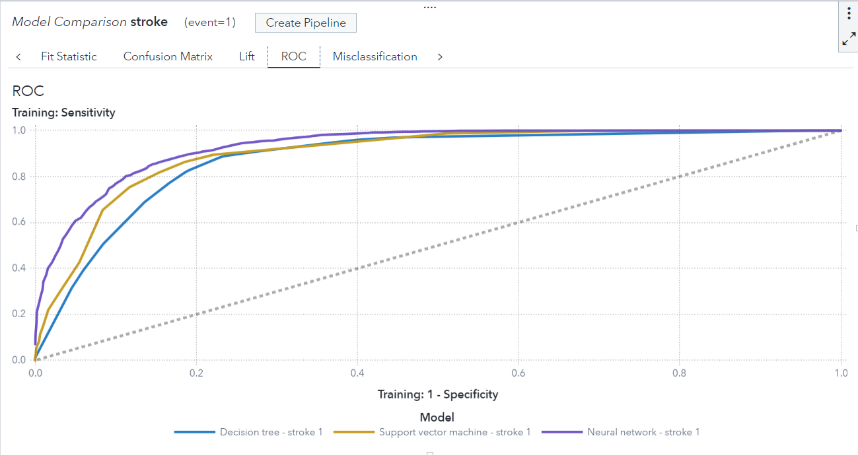
Stroke Model Comparison, Overview



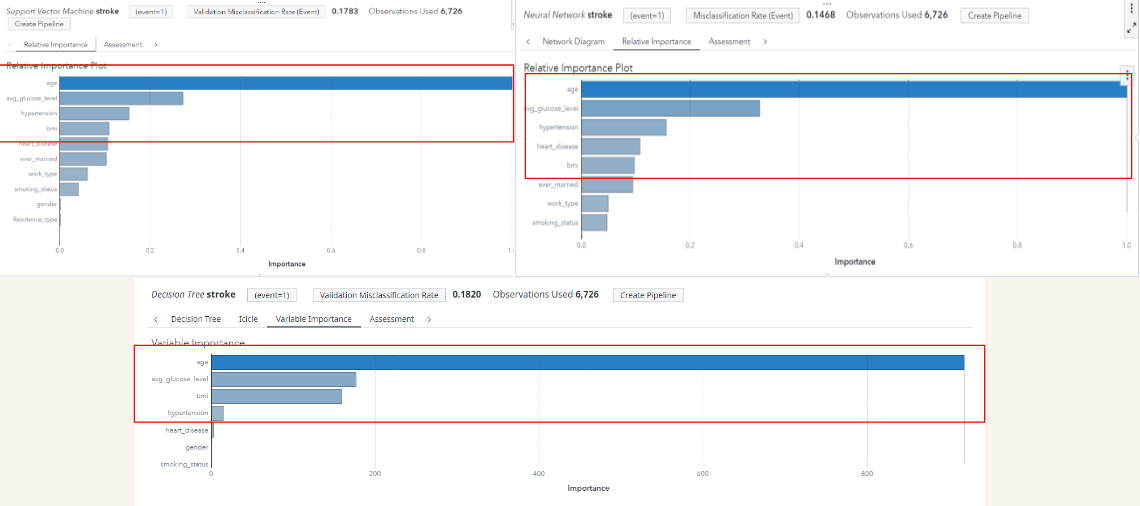
Stroke Model Comparison, Misclassification Plot



Stroke Model Comparison, Lift Chart



Stroke Model Comparison, ROC Curve



Stroke Model Comparison, Recommendation

## Diabetes

### Data Understanding & Preparation

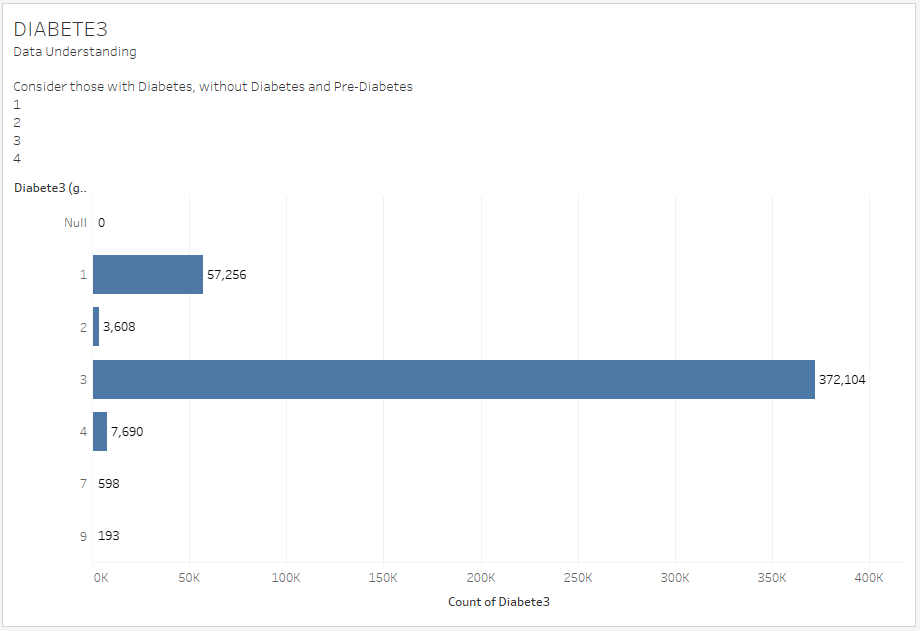


Fig. - Diabetes Data Understanding, Target Variable

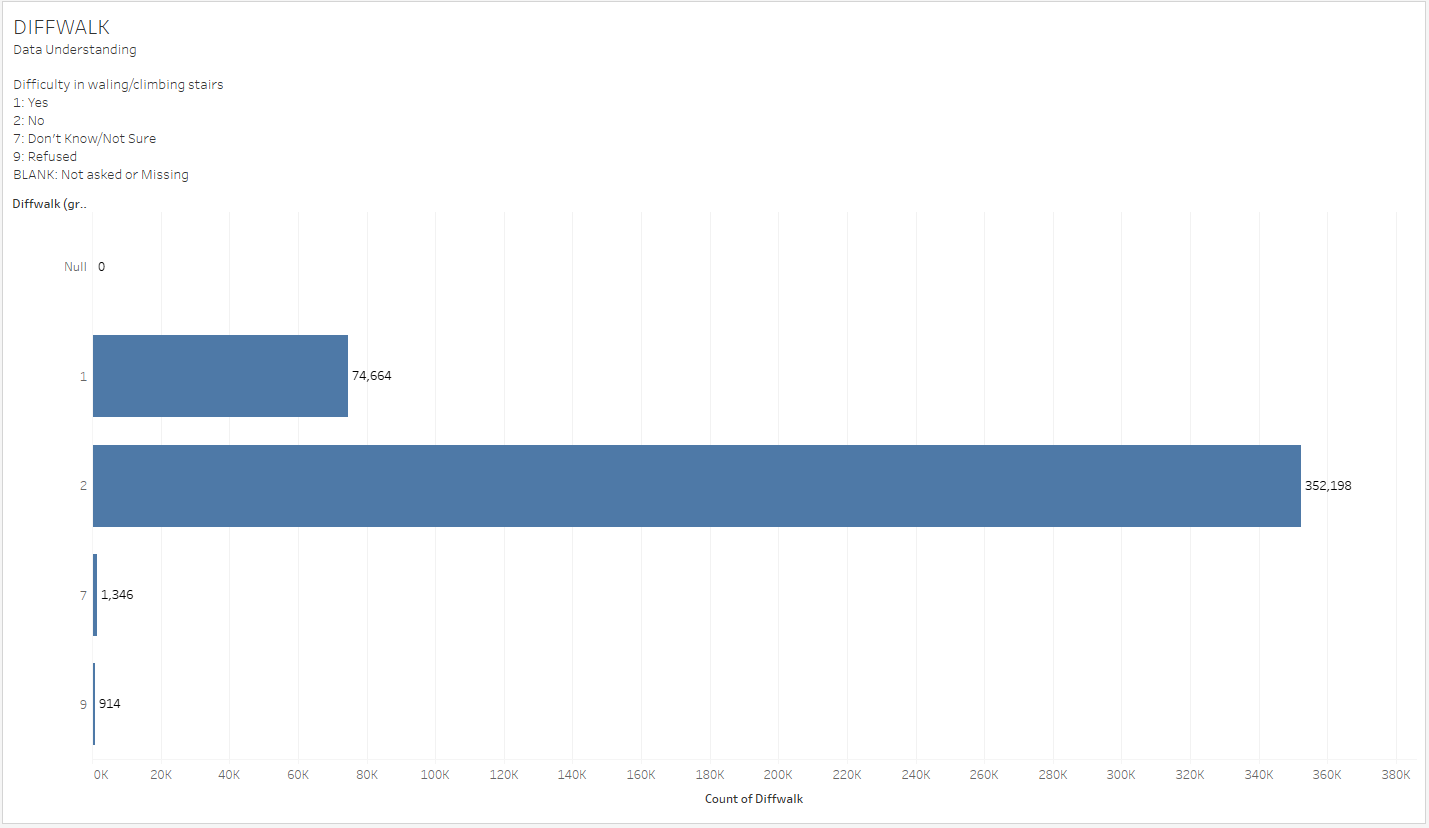


Fig. - Diabetes Data Understanding, Unnecessary Response/Missing Value

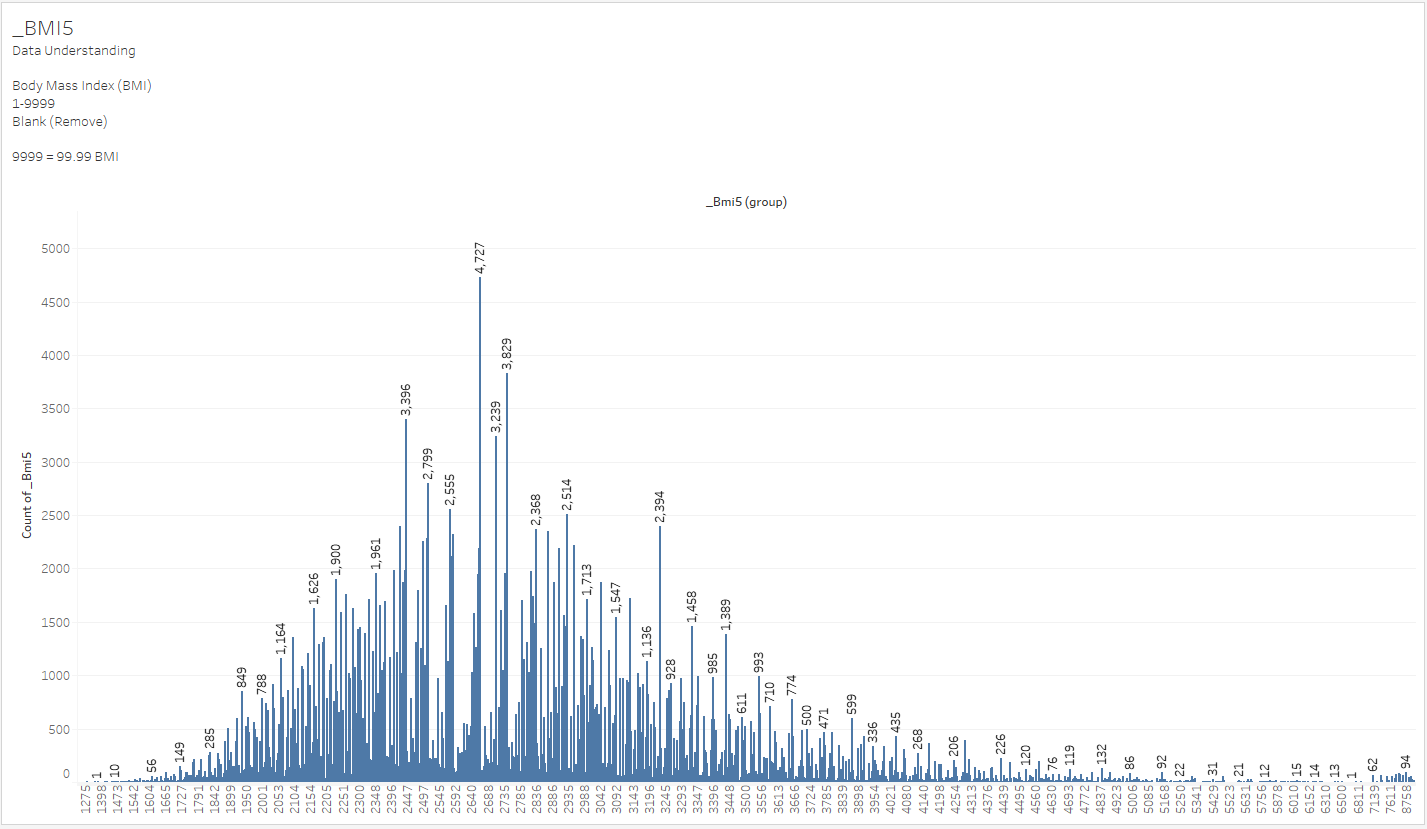
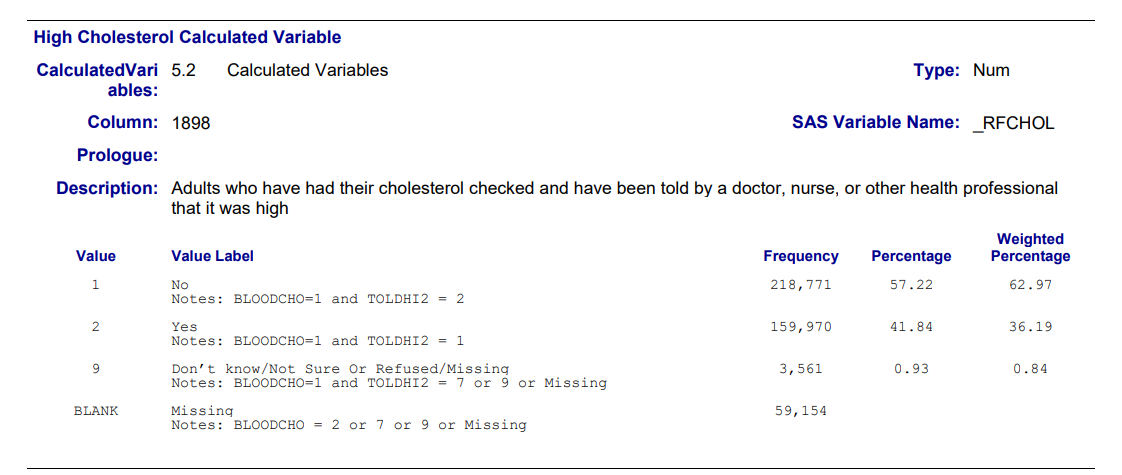


Fig. - Diabetes Data Understanding, BMI Outliers



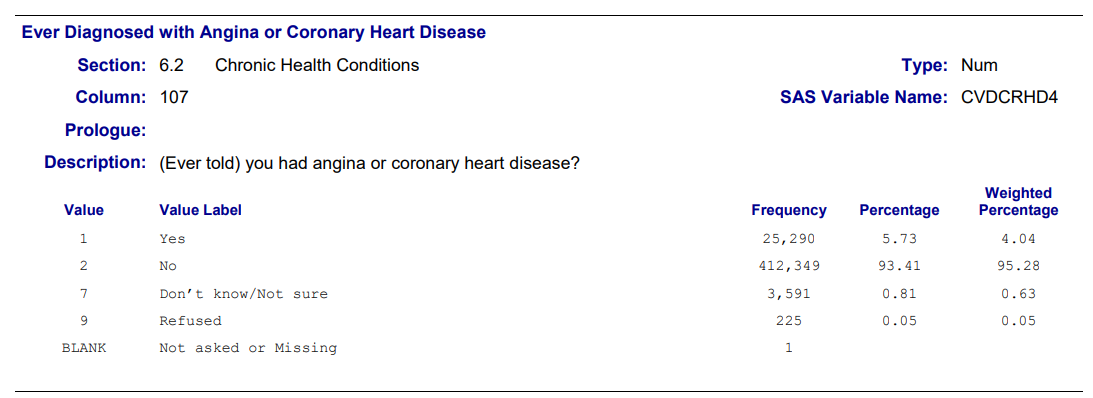
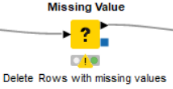
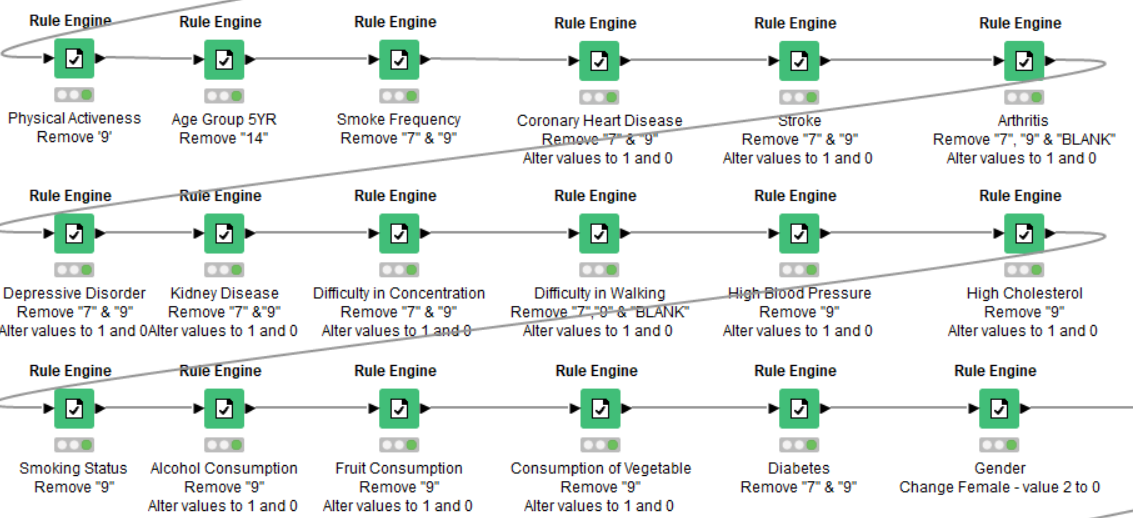


Fig. - Diabetes Data Understanding, Synchronise Codebook Encoded Response

****

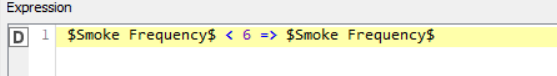


Fig. - Diabetes Data Cleaning, Data Transformation + Missing Value Removal

### 

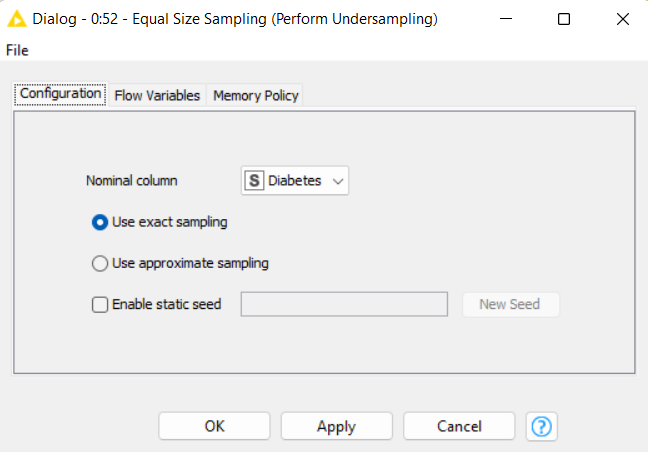
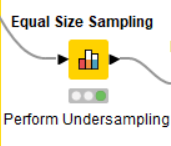


Fig. - Diabetes Data Cleaning, Perform Undersampling

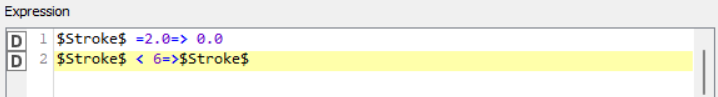
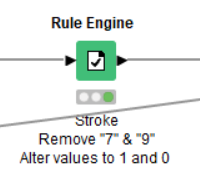
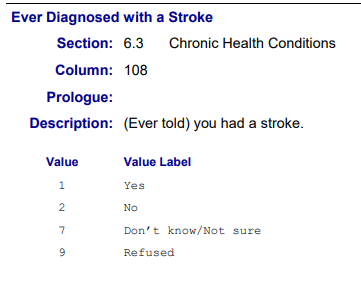


Fig. - Diabetes Data Cleaning, Synchronise Encoded Response

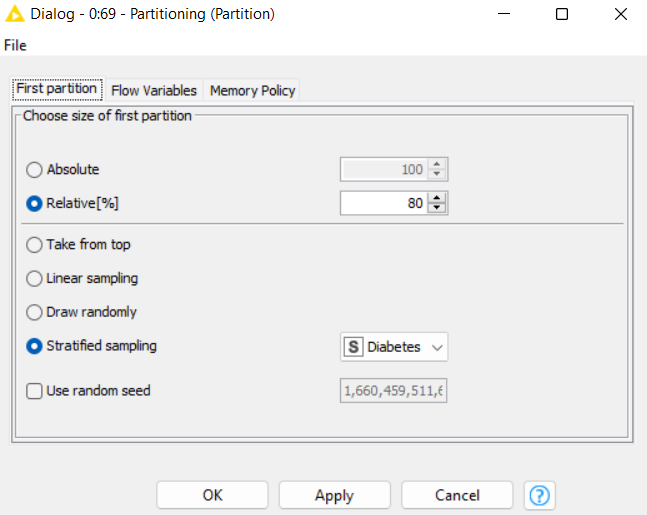
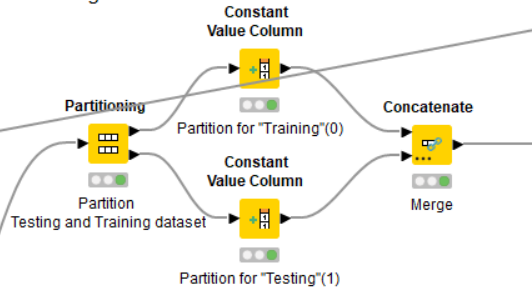


Fig. - Diabetes Data Cleaning, Partitioning

### 

### **Modelling**

**Logistic Regression**

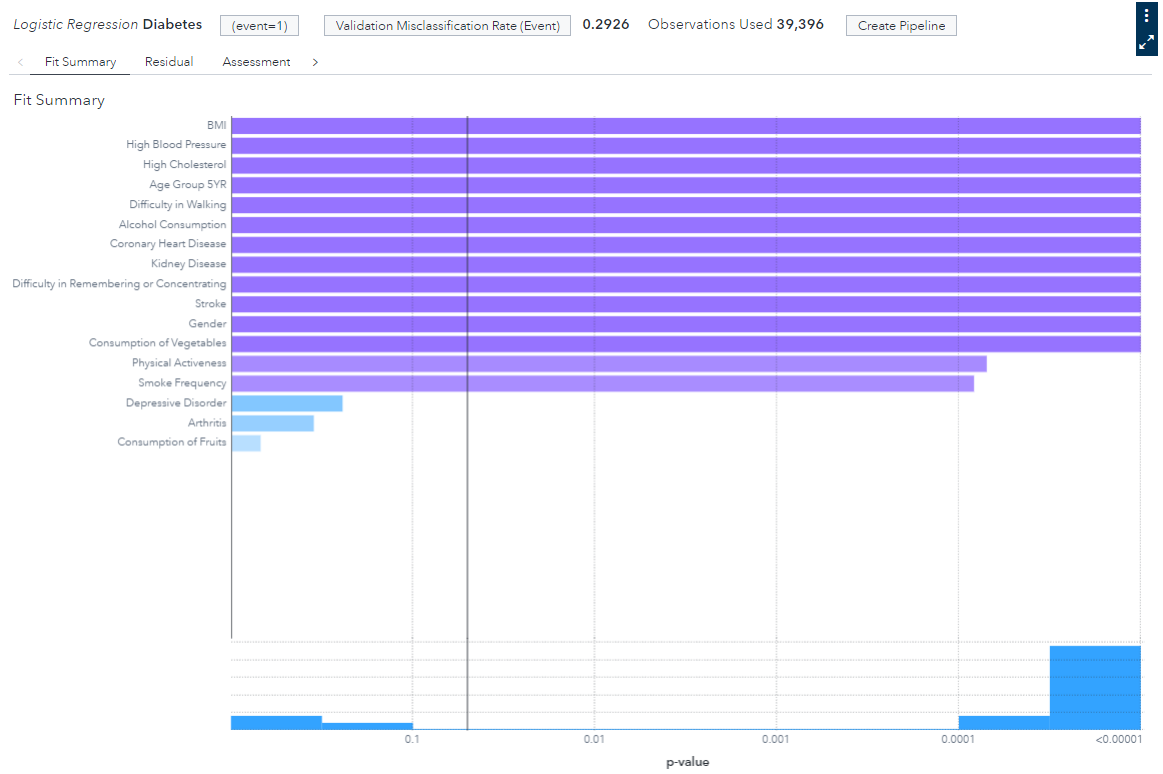


Fig. - Diabetes Logistic Regression, Feature Selection



Fig. - Diabetes Logistic Regression, Variable Importance



Fig. -Diabetes Logistic Regression, Lift



Fig. - Diabetes Logistic Regression, ROC Curve

**Random Forest**

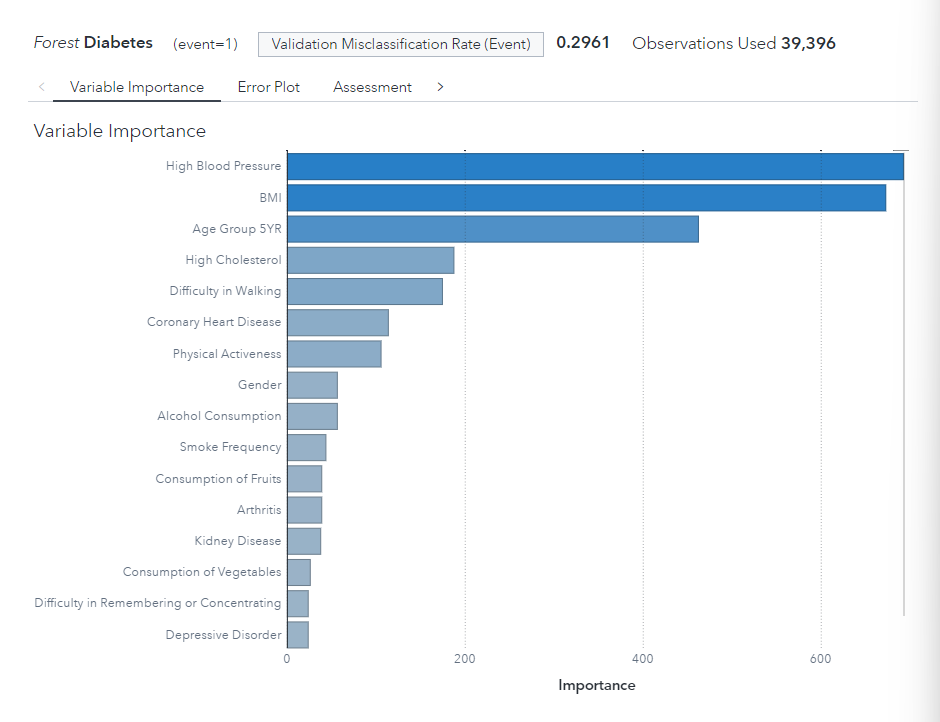


Fig. -Diabetes Random Forest, Variable Importance



Fig. - Diabetes Random Forest, Lif Chart

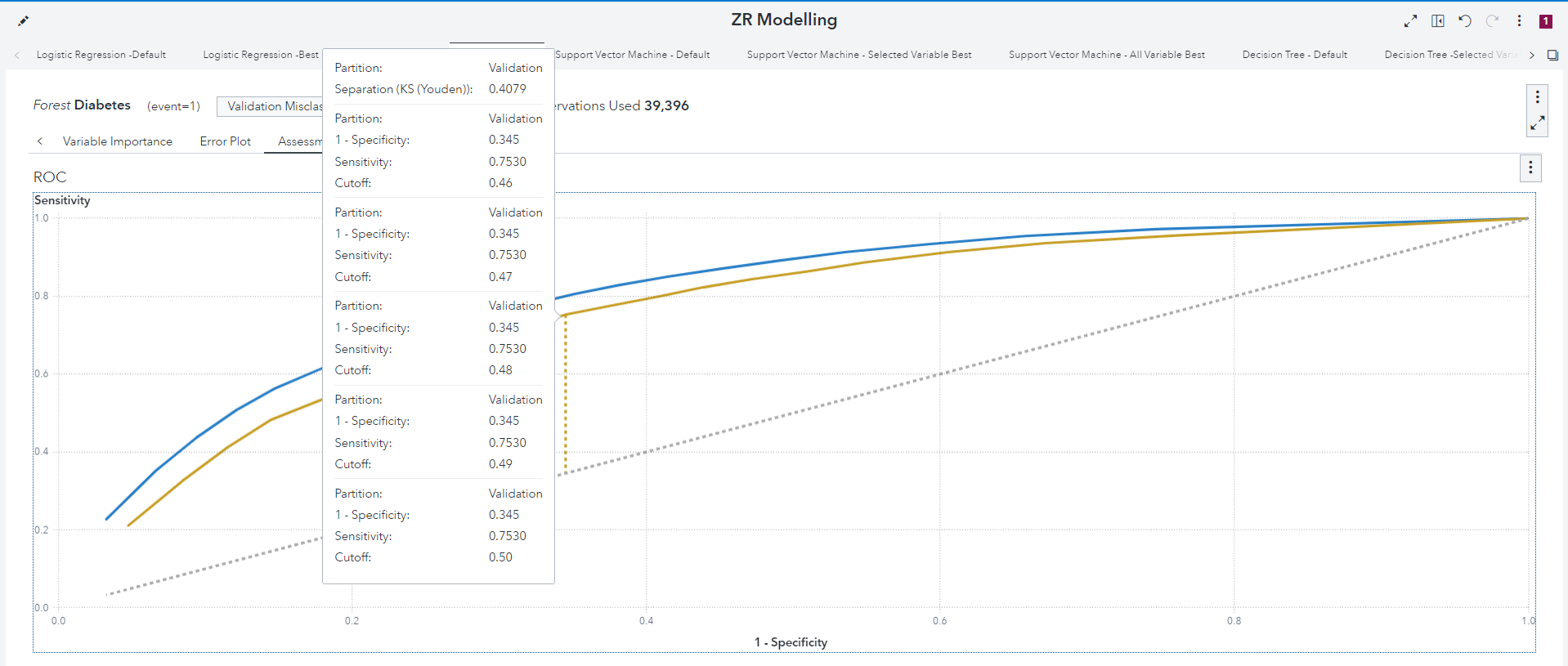


Fig. -Diabetes Random Forest, ROC Curve

**Support Vector Machine**

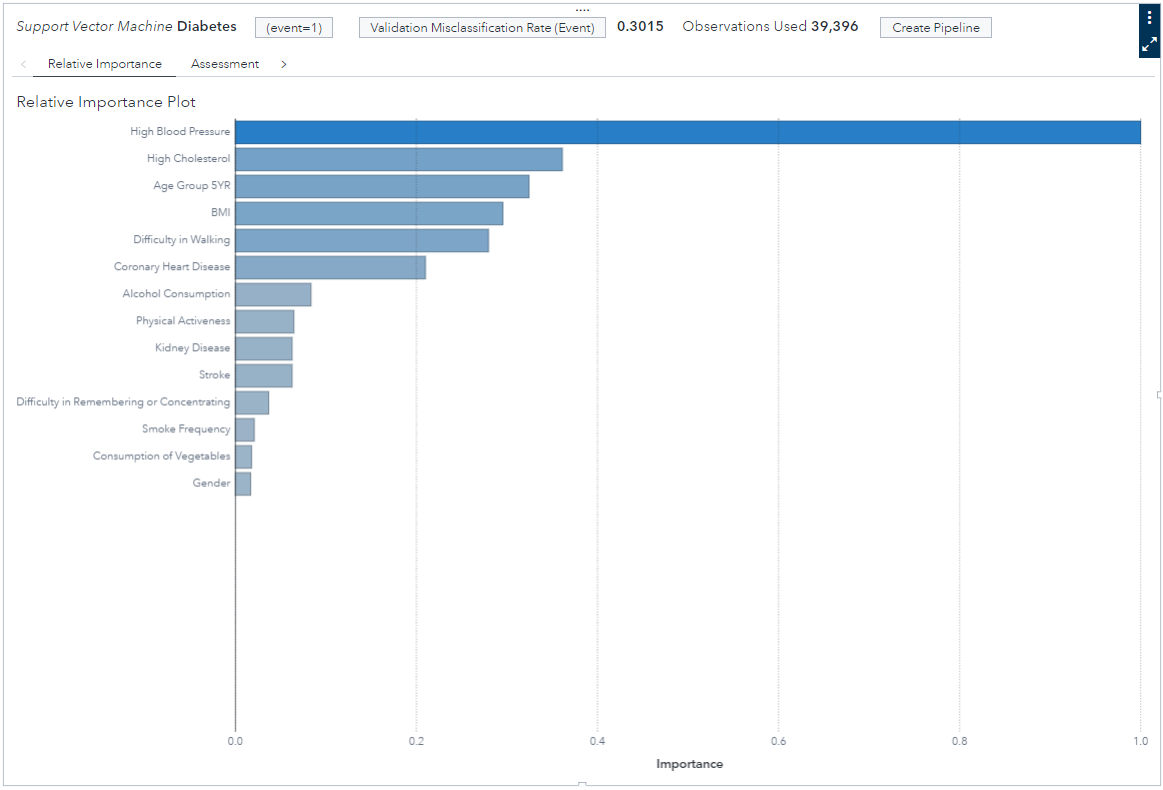


Fig. - Diabetes Support vector Machine, Variable Importance

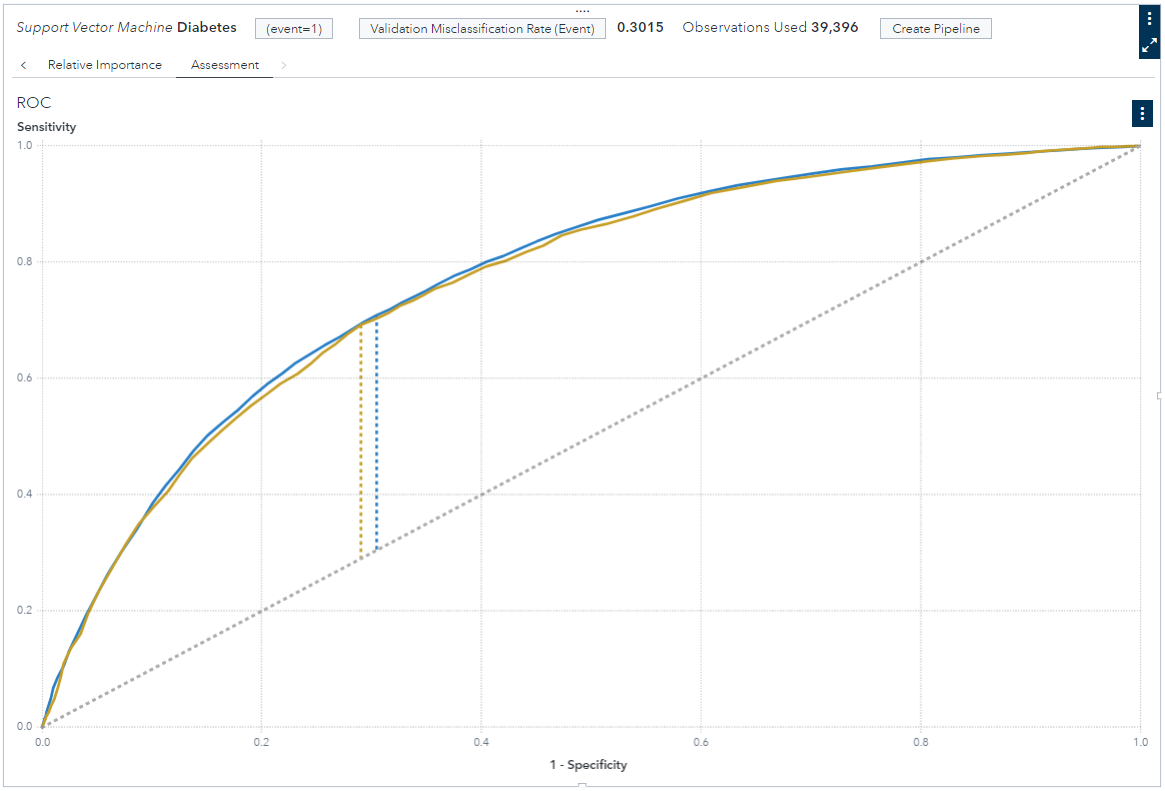


Fig. - Diabetes Support vector Machine, ROC Curve



Fig. - Diabetes Support vector Machine, Lift Plot

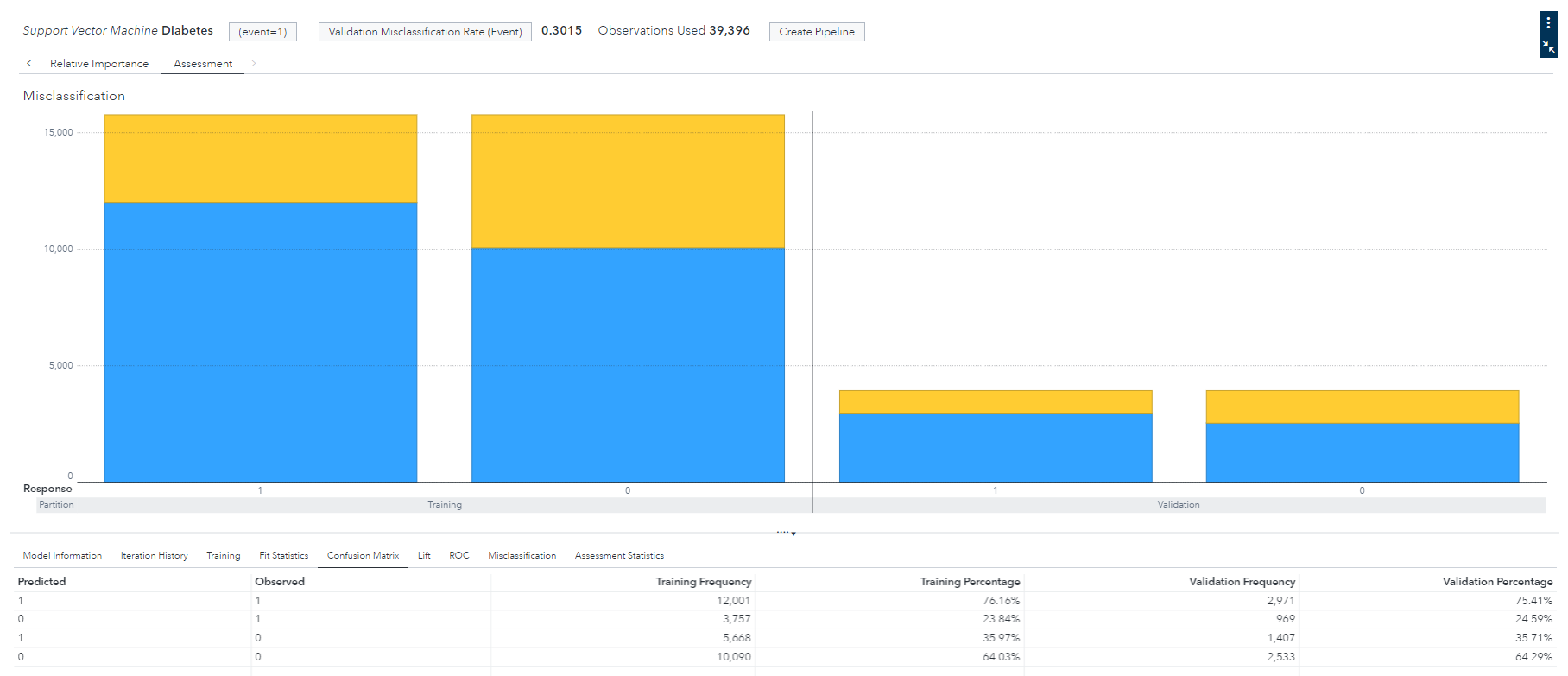


Fig.- Diabetes Support vector Machine, Misclassification Plot

**Decision Tree**

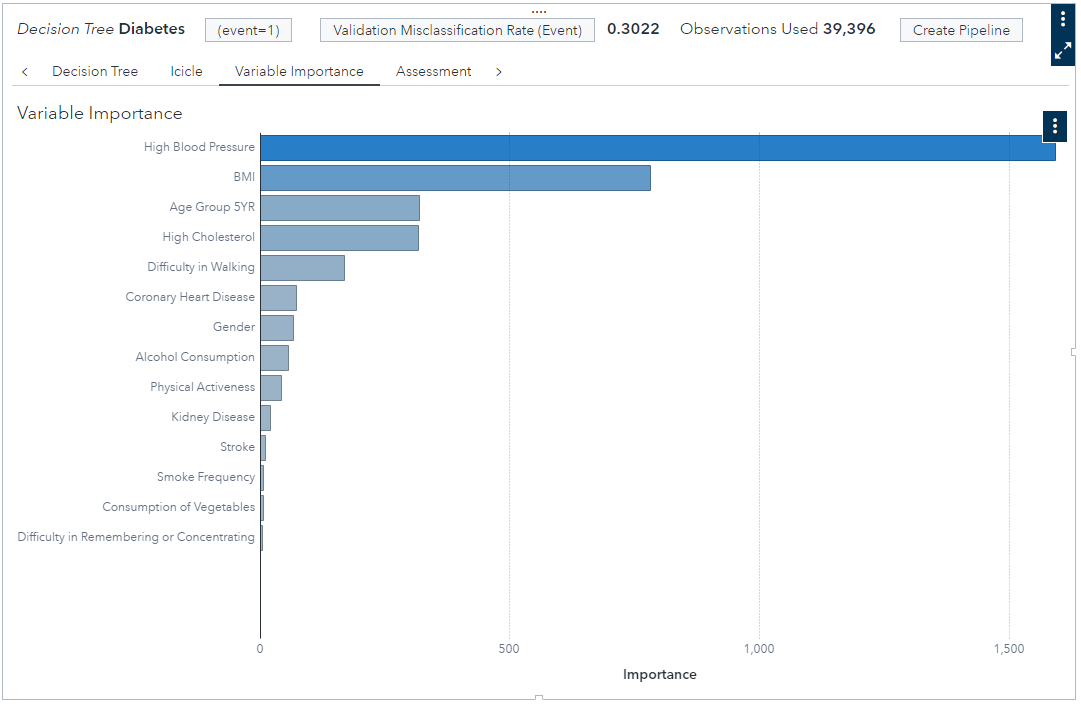


Fig. - Diabetes Decision Tree, Variable Importance

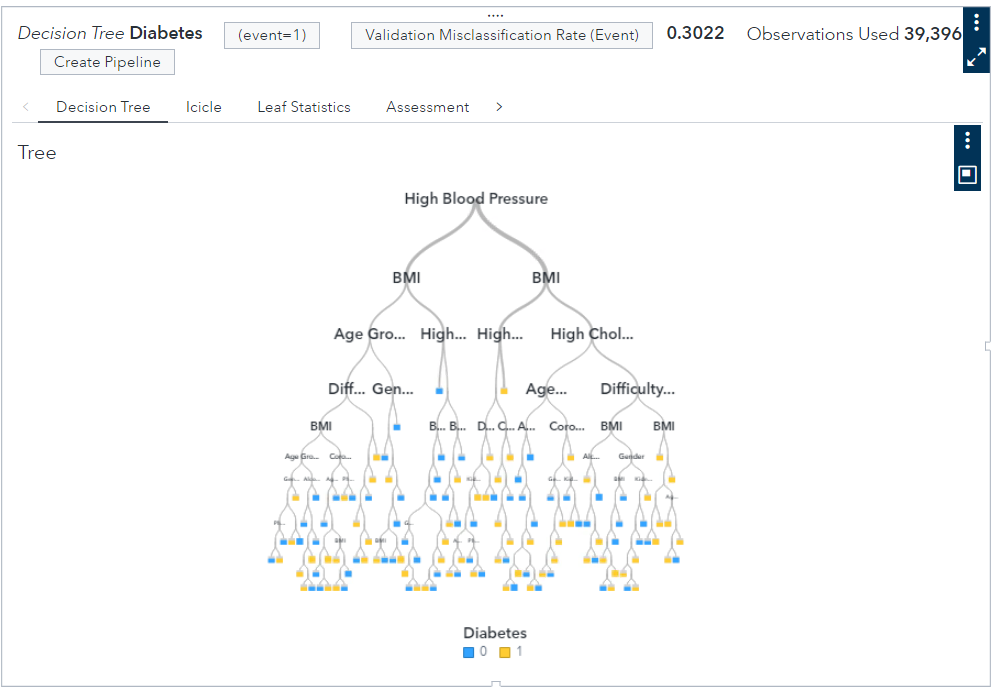


Fig. - Diabetes Decision Tree, Decision Tree Diagram

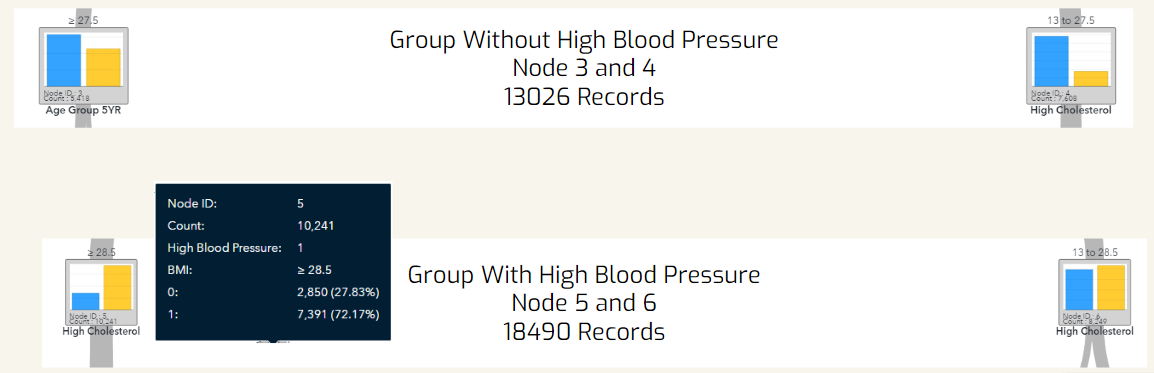
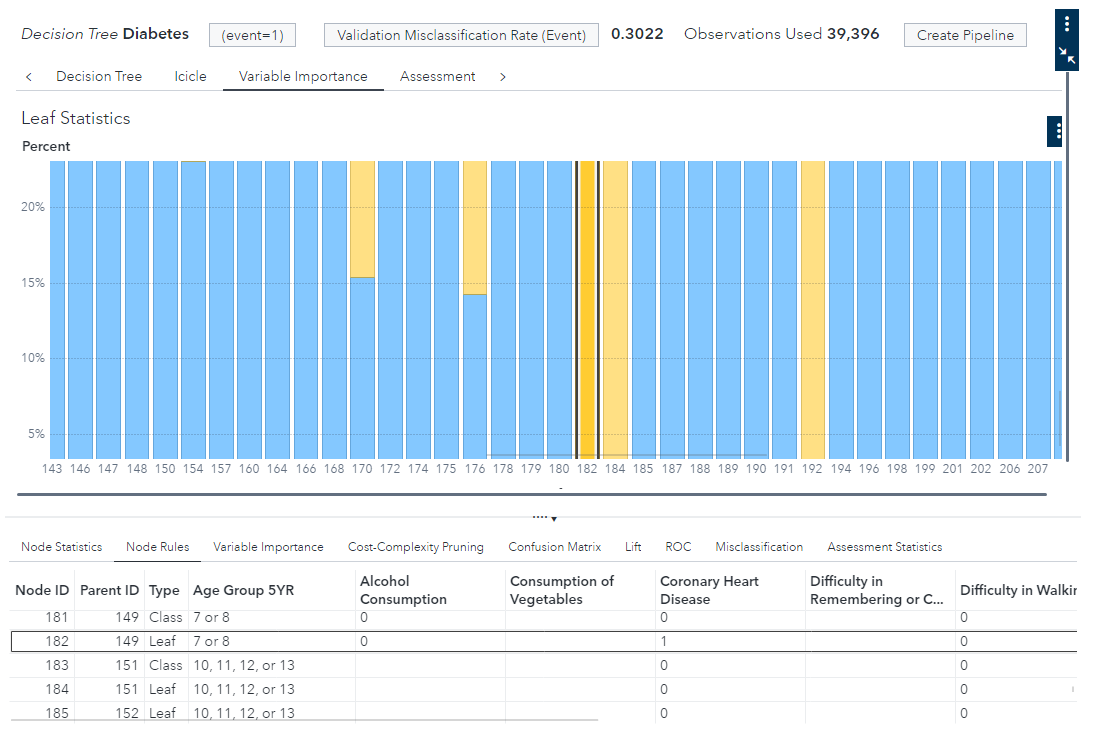
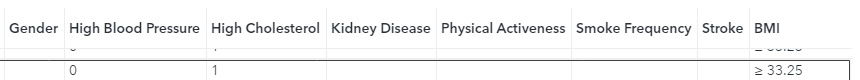


Fig. - Diabetes Decision Tree, Decision Tree node 3-6





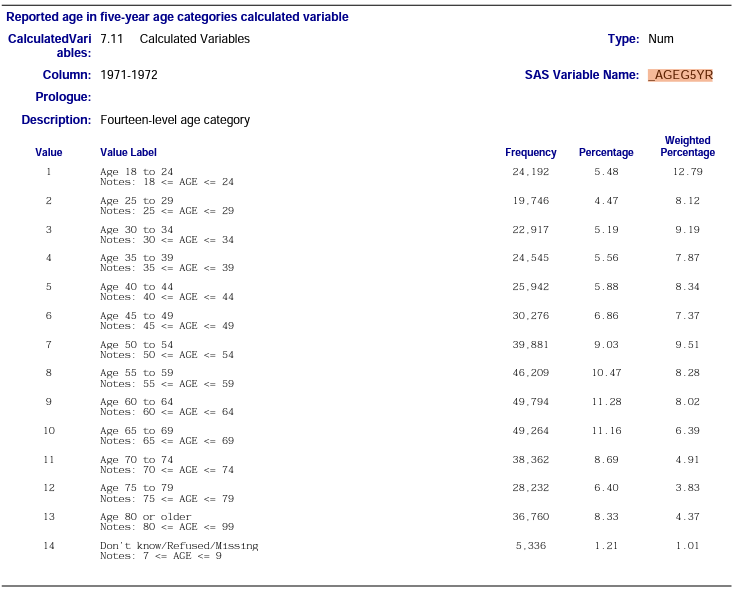


Fig. - Diabetes Decision Tree, Leaf Statistic

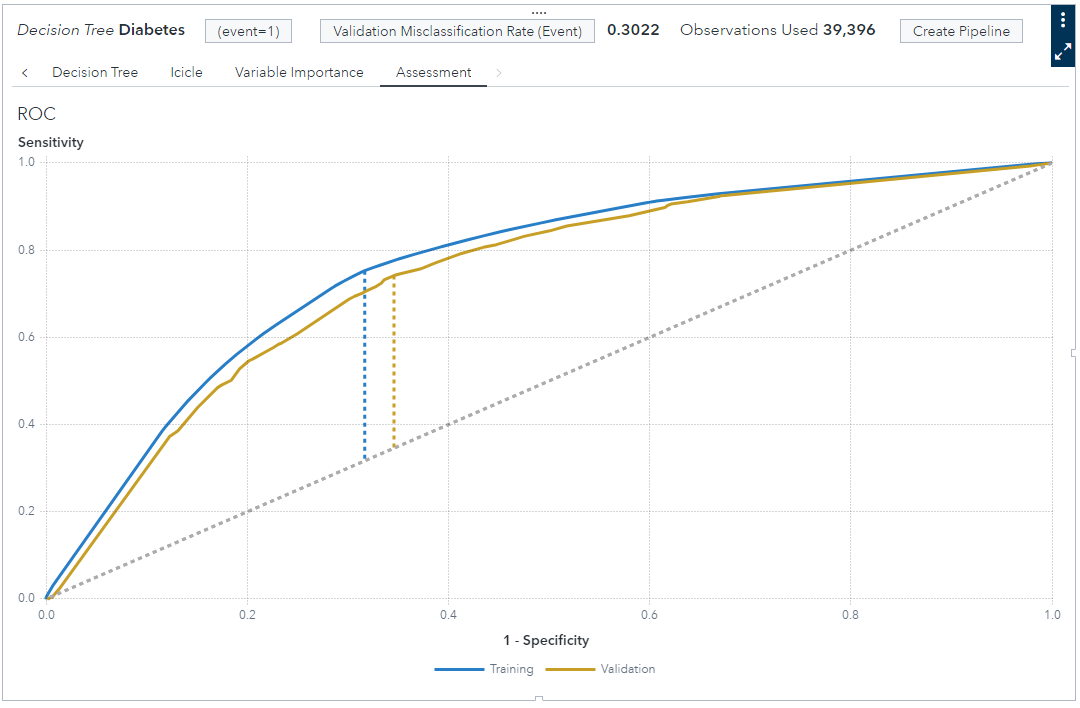


Fig. - Diabetes Decision Tree, ROC Curve



Fig. - Diabetes Decision Tree, Lift Plot

**Model Comparison**

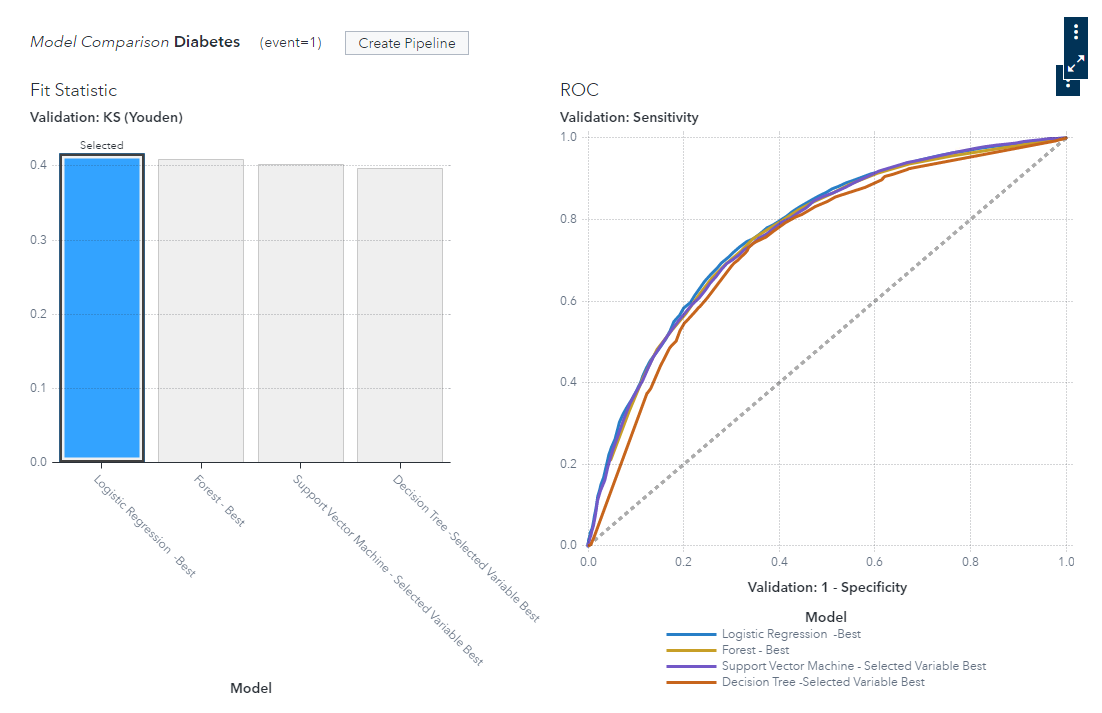
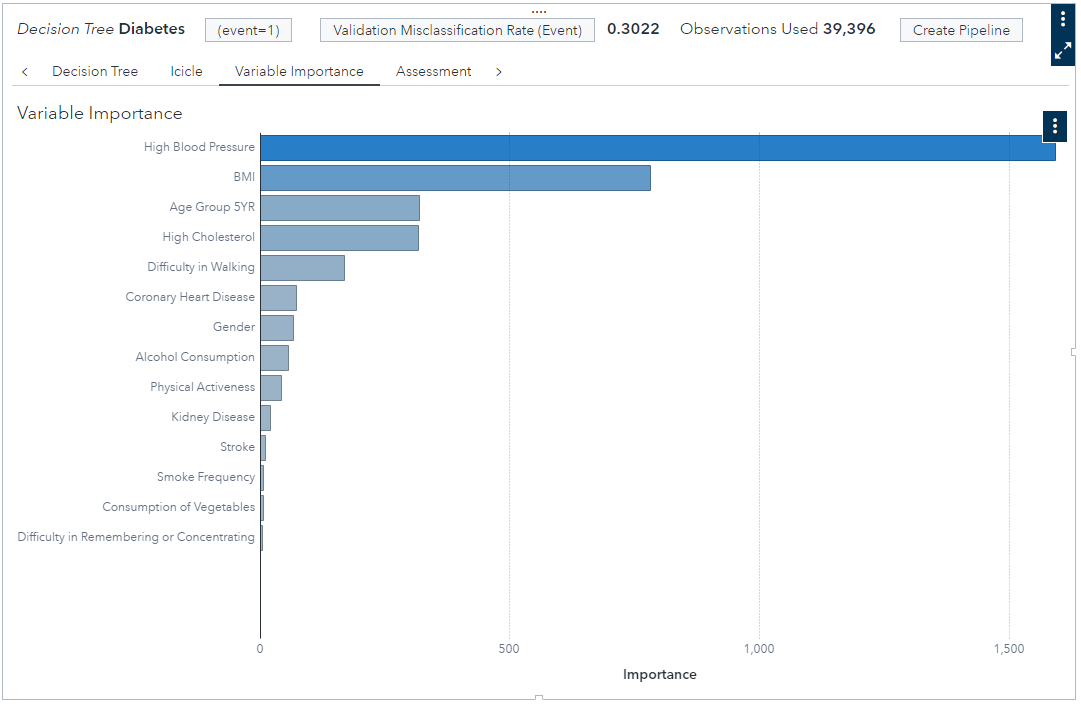
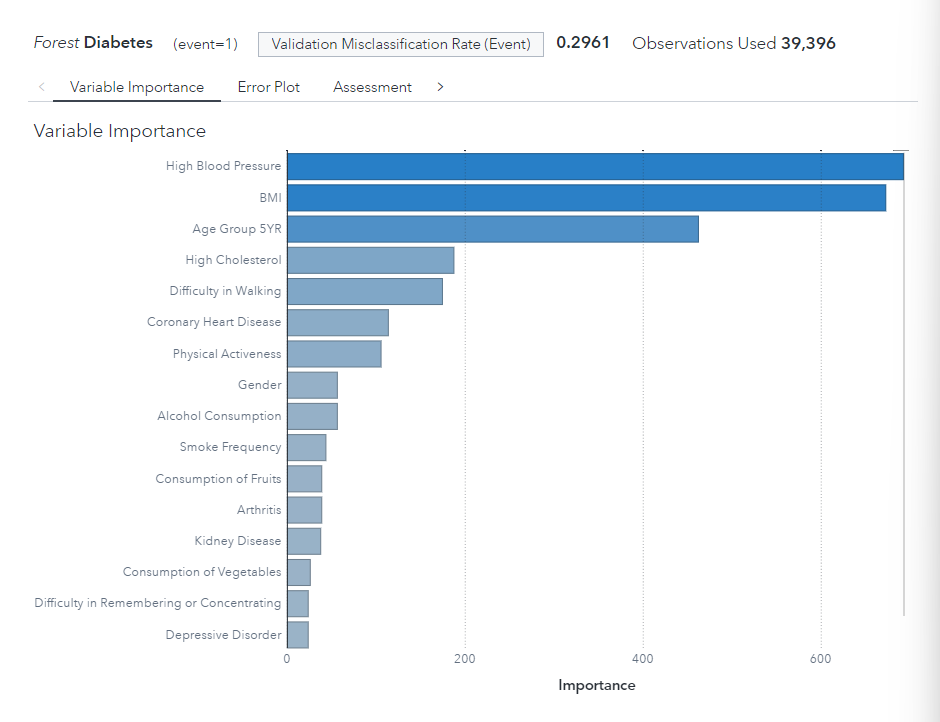


Fig. - Diabetes Model Comparison





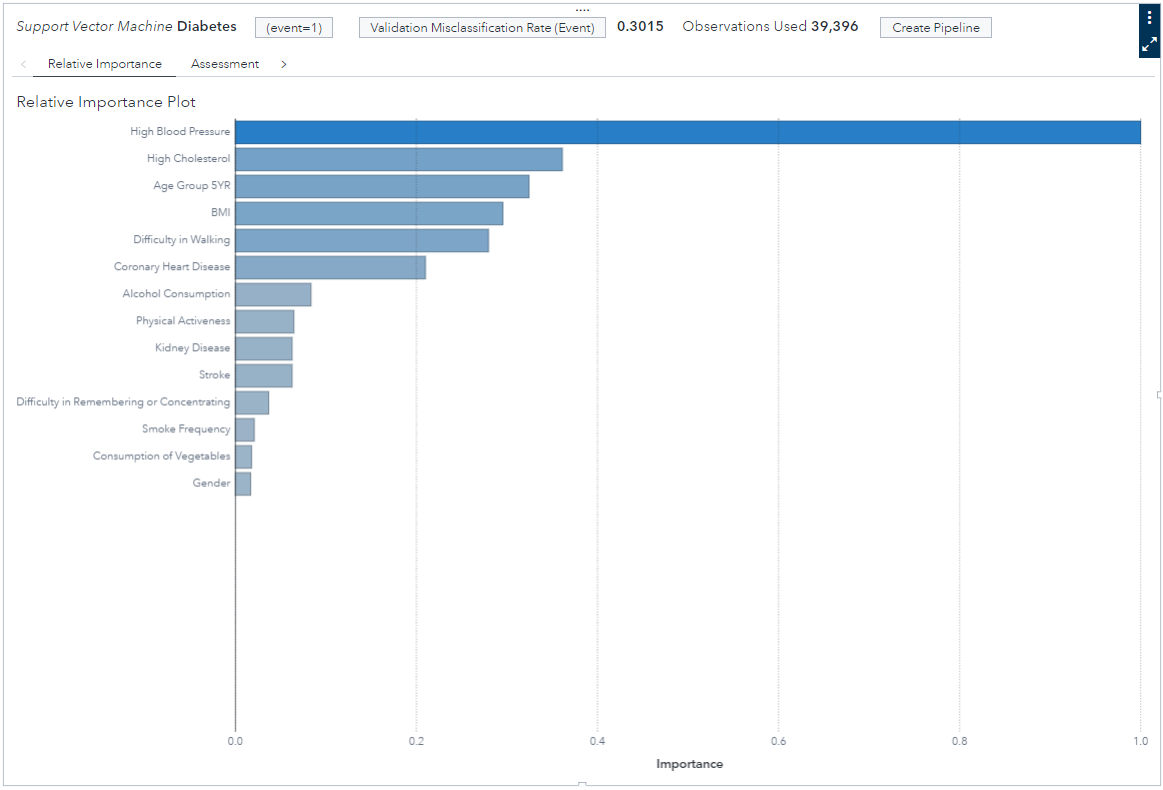


Fig. - Diabetes Model Comparison - Variable Importance